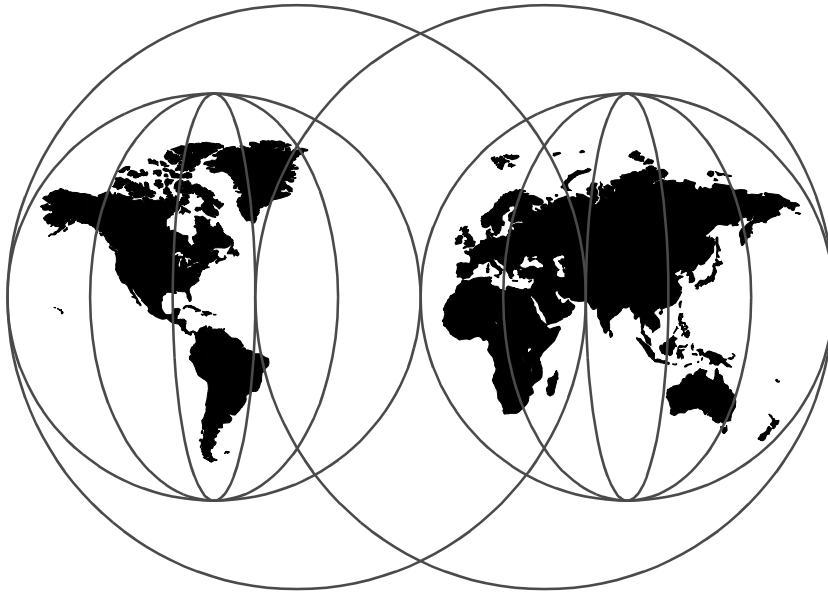


Inside the RS/6000 SP

*Marcelo R. Barrios, Mohammad Arif Kalem, Yoshimichi Kosuge, Philippe Lamarche
Belinda Schmolke, George Sohos, Juan Jose Vasquez, Tim Wherle*



International Technical Support Organization

<http://www.redbooks.ibm.com>

SG24-5145-00



International Technical Support Organization

Inside the RS/6000 SP

July 1998

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix A, "Special Notices" on page 379.

First Edition (July 1998)

This edition applies to Version 2 Release 4 of IBM Parallel System Support Programs for AIX (5765-529) for use with the AIX Version 4 Operating System.

Comments may be addressed to:

IBM Corporation, International Technical Support Organization
Dept. HYJ Mail Station P099
522 South Road
Poughkeepsie, New York 12601-5400

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 1998. All rights reserved

Note to U.S Government Users – Documentation related to restricted rights – Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

Figuresxi
Tables	xvii
Preface	xix
The Team That Wrote This Redbook	xix
Comments Welcome	xxi
<hr/>	
Part 1. The Big Picture	1
Chapter 1. History and Design Philosophy	3
1.1 Ultra-Large Computing Problems	3
1.2 Origins of the SP	4
1.3 High-Level System Characteristics	6
1.3.1 Scalability	6
1.3.2 Use of Mainstream Architectures and Technologies	7
1.3.3 Flexibility	8
1.3.4 Manageability	8
Chapter 2. System Architectures	9
2.1 Way of Categorizing Computers	9
2.2 Single Instruction, Single Data Uniprocessor	10
2.2.1 Everyday Examples	12
2.2.2 Limitations of SISD Architecture	12
2.3 Parallelism	13
2.4 Single Instruction, Multiple Data (SIMD) Machines	14
2.4.1 Everyday Examples of SIMD Machines	14
2.4.2 Limitations of the SIMD Architecture	14
2.5 Multiple Instruction, Single Data Machines	15
2.6 Multiple Instruction, Multiple Data Machines	15
2.6.1 Shared Memory MIMD Machines	15
2.6.2 Shared Nothing MIMD Machines	18
<hr/>	
Part 2. System Implementation	23
Chapter 3. Hardware Components	25
3.1 Frames	25
3.1.1 Power Supplies	26
3.1.2 Hardware Control and Supervision	27
3.2 Nodes	27
3.2.1 Internal Nodes	27

3.2.2	Extension Nodes	33
3.3	Control Workstation	35
3.3.1	High Availability Control Workstation	36
3.3.2	Supported Control Workstations	36
3.4	High-Performance Communication Network	38
3.4.1	Hardware Concepts	38
3.4.2	SP Switch Network	42
3.4.3	SP Switch Products	45
3.5	Peripheral Devices	47
3.6	Configuration Rules	48
3.6.1	Hardware Components	48
3.6.2	Basic Configuration Rules	49
3.6.3	Short Frame Configurations	50
3.6.4	Tall Frame Configurations	53
3.6.5	Numbering Rules	73
Chapter 4.	Software Components	81
4.1	System Data Repository (SDR)	82
4.1.1	Introduction	82
4.1.2	SDR Data Model	82
4.1.3	User Interface to SDR	84
4.1.4	SDR Daemons	84
4.1.5	Client/Server Communication	85
4.1.6	Locking	86
4.1.7	Manipulating SDR Data	87
4.1.8	Useful SDR Commands	88
4.1.9	The SDR Log File	89
4.1.10	Backup and Restore of SDR	89
4.2	The Hardware Control Subsystem	89
4.2.1	What Can Be Monitored and Controlled?	89
4.2.2	How It Works	90
4.2.3	User Interfaces	92
4.3	Overview of Switch Software	93
4.3.1	Introduction to the Switch	93
4.3.2	Switch Operating Principles	94
4.3.3	Primary Node Behavior	95
4.3.4	Secondary Nodes Behavior	96
4.3.5	Switch Topology File	96
4.3.6	Routing in SP Switch	101
4.3.7	Switch Initialization	104
4.3.8	Switch Clocks	105
4.3.9	Switch Log Files	105
4.3.10	The out.top File	106

4.3.11	Managing the Switch	107
4.4	Time Service	110
4.4.1	Why Network Time Protocol?	110
4.4.2	NTP Architecture	110
4.4.3	NTP Modes	111
4.5	High Availability Infrastructure (HAI)	112
4.5.1	"In the Beginning ..."	113
4.5.2	HAI Packaging	114
4.5.3	HAI Architecture Overview - "The Picture"	114
4.5.4	Topology Services	115
4.5.5	Group Services	120
4.5.6	Event Management	127
4.5.7	Putting It All Together - A Simple Example of Exploiting HAI	136
4.6	SP Security	141
4.6.1	General Security Concepts	141
4.6.2	AIX Security	144
4.6.3	Kerberos	144
4.6.4	SP Security Management	152
4.6.5	Andrew File System (AFS)	163
4.6.6	Sysctl	165
4.7	Parallel I/O	172
4.7.1	Network File System	172
4.7.2	Andrew File System and Distributed File System	172
4.7.3	Virtual Shared Disk	172
4.7.4	Hashed Shared Disk	175
4.7.5	Recoverable Virtual Shared Disk	176
4.7.6	Parallel Input/Output File System	178
4.7.7	General Parallel File System	180
4.8	File Collection	187
4.8.1	Introduction	187
4.8.2	Characteristics of File Collections	188
4.8.3	Predefined File Collections	188
4.8.4	File Collection Organization	189
4.8.5	Directory and Master Files	190
4.8.6	Managing the File Collections	192
4.8.7	Basic File Collection Maintenance	192
4.8.8	Building a File Collection	194
4.8.9	Installing a File Collection	195
4.8.10	Refusing Files in a File Collection	196
4.8.11	Removing a File Collection	196
4.9	Managing AIX Automounter	196
4.9.1	The Other Automounter	199
4.9.2	The AIX Automounter	200

4.9.3 Differences between AIX and BSD Automounter	201
4.10 Job Management	202
4.10.1 AIX Job Management	202
4.10.2 The LoadLeveler Approach	203
4.10.3 Resource Manager	204
4.11 Parallel Environment (PE)	204
4.11.1 POE Operating Environment	205
4.11.2 MPI/MPL Message Passing Libraries	206
4.11.3 LAPI (Low-Level API)	207
4.11.4 Visualization Tool (VT)	208
4.11.5 Parallel Debuggers	209
4.11.6 Xprofiler Profile Analyzer	209
4.11.7 User Space and IP Protocols	210
Chapter 5. System Management	211
5.1 SP Monitors	213
5.1.1 SP Perspectives	213
5.2 Installation	218
5.2.1 Installation Process	218
5.2.2 Software Upgrade	223
5.2.3 Coexistence	226
5.3 SP User Management	227
5.3.1 Configuring SP User Management	228
5.3.2 AIX Users vs. SP Users	229
5.3.3 SP User Management (SPUM)	230
5.3.4 Restricting User Access to the CWS	231
5.3.5 SP Access Control (SPAC)	232
5.4 Problem Management	234
5.4.1 AIX, BSD and PSSP-specific Error Logging	234
5.4.2 Problem Management (PMAN) Subsystem	236
5.4.3 Perspectives	239
5.4.4 IBM Support Tools	240
5.5 System Partitioning	240
5.5.1 Overview	241
5.5.2 Considerations for Partitioning	241
5.5.3 Partitioning Justification	242
5.5.4 Partitioning the SP System	243
5.5.5 Repartitioning or Modifying an Existing System Partition	244
5.5.6 Syspar Directory Structure	245
5.5.7 Managing System Partitions	246
5.6 Performance Management	247
5.6.1 General SP Performance Considerations	248
5.6.2 Performance Management Tools	250

5.6.3 Other Performance Management Aides	254
5.7 Accounting	256
5.7.1 How Standard AIX and SP Accounting Differ	256
5.7.2 Tools and Considerations	257
5.8 Managing Print Service	258
5.9 Managing Resource Workload	258
5.9.1 Managing Resource Manager	258
5.9.2 Managing LoadLeveler	262
5.10 SP Backup and Recovery	268
5.10.1 mksysb and savevg	269
5.10.2 Using mksysb and savevg	271
5.10.3 SP-Specific Data That Needs to Be Backed Up Frequently	274
<hr/>	
Part 3. Solutions	277
Chapter 6. Enterprise System Management	279
6.1 ADSTAR Distributed Storage Manager	280
6.2 High Availability Cluster Multiprocessing (HACMP)	282
6.2.1 HACMP for AIX	282
6.2.2 HACMP Enhanced Scalability (HACMP ES)	285
6.2.3 Summary of HACMP Products	287
6.3 Tivoli	288
6.3.1 SP Implementation	288
Chapter 7. Cross-Industry Applications	291
7.1 Enterprise Resource Planning (ERP)	292
7.1.1 SAP R/3	292
7.1.2 SAP R/3 Design	293
7.1.3 RS/6000 SP Implementation	294
7.1.4 SAP R/3 Examples on SP	294
7.2 Lotus Notes	295
7.2.1 Lotus Domino	296
7.2.2 RS/6000 SP Implementation	297
7.2.3 Lotus Domino Examples on SP	298
7.3 Decision Support Systems (DSS)	300
7.3.1 Why a Parallel Approach is Key for DSS	303
7.3.2 Why Use the SP for DSS	304
7.3.3 Examples:	305
7.4 Internet/Web Servers	308
7.4.1 Scalability	308
7.4.2 Applications	311
7.4.3 Internet Service Provider (ISP)	311
7.4.4 Content Hosting	311

7.4.5	Mega Web Site Management	312
7.4.6	Other Well-Known Applications	313
Chapter 8. Commercial Applications 315		
8.1	OLTP Mission-Critical Applications	316
8.1.1	Distributed System	316
8.1.2	2-Tier and 3-Tier Concepts	317
8.1.3	OLTP Solution Components	319
8.1.4	Why Use the SP for OLTP	320
8.1.5	Transaction Processing Monitors: CICS, Encina and Tuxedo	321
8.1.6	Mission-Critical Power Solutions Examples	326
8.2	Databases: DB2, Oracle, Informix and Sybase	330
8.2.1	DB2	334
8.2.2	Oracle	340
8.2.3	Informix and Sybase	351
8.2.4	Informix	352
8.2.5	Sybase	353
8.3	Server Consolidation	355
8.3.1	The Costs of Distributed Computing	355
8.3.2	Cost of Computing Study	355
8.3.3	Total Cost of Ownership	356
8.3.4	Scalability	356
8.3.5	Reliability and Availability	356
8.3.6	Utilization	357
8.3.7	Flexibility	357
8.3.8	A Server Consolidation Example	358
Chapter 9. Scientific and Technical 361		
9.1	Research Applications	362
9.1.1	RS/6000 SP Applications to Research	363
9.1.2	Research Solutions Examples	363
9.2	Process Applications	366
9.2.1	RS/6000 SP Implementations to Process Applications	367
9.2.2	Process Solutions Examples	367
9.3	Petroleum Research Applications	368
9.3.1	Seismic Processing Applications	368
9.3.2	Reservoir Modeling Applications	369
9.3.3	SP Implementations to Reservoir Modeling Applications	371
9.3.4	Seismic Research Examples	372
9.4	Engineering Applications	373
9.4.1	Mechanical Engineering	373
9.4.2	Electronic Design and Analysis	376
9.4.3	RS/6000 SP Implementations	377

9.4.4 Engineering Application Examples	377
Appendix A. Special Notices	379
Appendix B. Related Publications	383
B.1 International Technical Support Organization Publications	383
B.2 Redbooks on CD-ROMs	383
B.3 Other Publications.	384
How to Get ITSO Redbooks	385
How IBM Employees Can Get ITSO Redbooks	385
How Customers Can Get ITSO Redbooks	386
IBM Redbook Order Form	387
List of Abbreviations.	389
ITSO Redbook Evaluation	399

x Inside the RS/6000 SP

Figures

1. von Neumann Architecture	11
2. Schematic View of a Shared Memory MIMD Machine.	16
3. IBM's MIMD Shared Memory Implementation in RS/6000 Model J40.	17
4. Relationship between Extra Processors and Performance Gain.	18
5. Conventional Shared Nothing MIMD Machine.	18
6. IBM's First Enhancement of the Shared Nothing MIMD Machine	19
7. Further IBM Enhancements to the Shared Nothing MIMD Machine	21
8. The Relationship between the Different Types of Nodes.	27
9. RS/6000 SP Switch Router Offers Infinite Possibilities	33
10. Control Workstation Connectivity and Functionality.	36
11. SP Switch Link and SP Switch Port.	39
12. SP Switch Chip	40
13. SP Switch Board	41
14. SP Switch Adapter.	41
15. SP Switch System	42
16. 16-Node SP System	43
17. 32-Node SP System	44
18. SP System with Intermediate Switch Board.	45
19. Minimum Switched Configuration	51
20. Maximum Switched Configuration Using Thin Nodes	52
21. Maximum Switched Configuration Using Wide Nodes.	52
22. Maximum Switched Configuration Using High Nodes	53
23. Maximum Switched Configuration Using Mixed Nodes	53
24. Minimum SP Switch-8 Configuration	55
25. Maximum SP Switch-8 Configuration Using Thin Nodes.	56
26. Maximum SP Switch-8 Configuration Using Wide Nodes	56
27. Maximum SP Switch-8 Configuration Using High Nodes	57
28. Maximum SP Switch-8 Configuration Using Mixed Nodes	57
29. Minimum Single Stage SP Switch Configuration	58
30. Minimum Single Stage SP Switch Configuration with SEF	59
31. Maximum Single Stage SP Switch Configuration with SEFs (1)	59
32. Maximum Single Stage SP Switch Configuration with SEFs (2)	60
33. Maximum Single Stage SP Switch Configuration with SEFs (3)	60
34. Minimum Single Stage SP Switch Configuration with NEF	61
35. Maximum Single Stage SP Switch Configuration with NEF (1)	62
36. Maximum Single Stage SP Switch Configuration with NEF (2)	62
37. Minimum Single Stage SP Switch Configuration with SEF and NEF.	63
38. Maximum Single-Stage SP Switch Configuration with SEFs and NEFs (1)	64
39. Maximum Single-Stage SP Switch Configuration with SEFs and NEFs (2)	65
40. Minimum Two-Stage SP Switch Configuration	66

41. Minimum Two-Stage SP Switch Configuration with SEFs	67
42. Maximum Two-Stage SP Switch Configuration with SEFs (1).	68
43. Maximum Two-Stage SP Switch Configuration with SEFs (2).	68
44. Maximum Two-Stage SP Switch Configuration with SEFs (3).	69
45. Minimum Two-Stage SP Switch Configuration with NEF.	70
46. Minimum Two-Stage SP Switch Configuration with SEFs and NEF	71
47. Maximum Two-Stage SP Switch Configuration with SEFs and NEFs (1) .	72
48. Maximum Two-Stage SP Switch Configuration with SEFs and NEFs (2) .	73
49. Frame, Slot, and Node Numbers	74
50. Base Configuration With no Nonswitched Expansion Frame	76
51. Configuration With One Nonswitched Expansion Frame.	76
52. Configuration With Two Nonswitched Expansion Frames.	77
53. Configuration With Three Nonswitched Expansion Frames	78
54. SP Switch-8 and HiPS LC-8 Configuration	79
55. RS/6000 SP System Software Architecture.	81
56. SDR Data Model	83
57. SDR Directory Structure - Sample.	83
58. Example: How Nodes Communicate with the SDR	86
59. SP System Monitor	90
60. Fault Service Architecture	94
61. Sample Topology File	98
62. Nomenclature Used in Topology Files.	99
63. Example: How a Topology File Maps Physically on the Switch.	100
64. Structure of the Data Packet	101
65. Service Packet.	102
66. Example: Routing in the SP Switch	103
67. Example of out.top File Output	107
68. NTP Hierarchy	111
69. HAI Infrastructure - "The Picture".	114
70. TS and GS Interfaces	116
71. TS Process Flow	119
72. Group Services Structure.	120
73. GS Internals	123
74. "Group Awareness" of GS Daemons.	124
75. EM Design	128
76. EM Client and Peer Communication	129
77. haemd Joining Group Services	132
78. SP Resource Monitors.	134
79. Topology Table for Scenario	138
80. Security Perimeter	142
81. Authentication and Authorization.	143
82. Entities Involved in a Third-Party Authentication System	145
83. Client Logs into AS.	146

84. Auhtentication Server Ticket-Granting Ticket	147
85. Complete Ticket-Granting Ticket Message from the AS to AC.	147
86. AC Receives TG Ticket and Session Key from KAS.	148
87. Authenticator and Ticket-Granting Ticket	149
88. SSX Server Packet Ticket, Sent from AS to AC	150
89. Service Packet Ticket Sent from AC to SSX.	150
90. Service Provider Decypher Process	151
91. Sysctl Architecture.	166
92. VSD Architecture	173
93. VSD State Transitions	175
94. HSD Architecture and Function	176
95. RVSD Function	177
96. RVSD Subsystems and HAI	178
97. PIOFS Design	179
98. GPFS Structure	182
99. GPFS Software Architecture	183
100.GPFS Performance vs. NFS and PIOFS	187
101.SP Default File Collection Hierarchy	190
102./var/sysman/sup Files and Directories	190
103.The World before NIS and NFS Automounter.	197
104.The World after NIS and NFS Automounter	198
105.BSD Automounter and AIX Automounter Coexistence	200
106.LoadLeveler Divides Work Among Nodes.	203
107.Interaction of LoadLeveler and Resource Manager	204
108.POE Architecture.	206
109.Communication Protocol Layers	208
110.SP Perspectives	214
111.Hardware Perspective.	215
112.Event Perspective	216
113.System Monitor GUI (spmon)	217
114.Node Network Boot	221
115.PMAN Design	236
116.PMAN Daemons	238
117.Example of an 8_8 Configuration	242
118.Directory Structure for Topology Files.	245
119.PTX/6000 Functional Overview.	252
120.PTPE Architecture.	253
121.PTPE Monitoring Hierarchy.	254
122.Delegation of Different Workload to Various Nodes	259
123.Resource Manager Running on the Nodes in a Partition	262
124.Interactions of LoadLeveler Daemons for Serial Batch Jobs	264
125.Interaction of LoadLeveler with Resource Manager	265
126.Contents of an mksysb Backup Image	270

127.Backing Up a Simple System Using mksysb and savevg	272
128.Restoring a Simple System Using mksysb and restvg	273
129.Scalable Backup Policy	274
130.Is SP a solution?	279
131.Solution Domains	280
132.ADSM Structure	280
133.Generic HACMP Cluster	282
134.HACWS Cluster.	285
135.HACMP ES Relationship to HAI Components	286
136.High Availability Family	288
137.Tivoli and SP Integration	290
138.Cross-Industry Applications.	291
139.Conceptual Model of SAP R/3 Structure	293
140.Enterprise-Wide Domino Configuration	297
141.Logical model of Domino on the SP	298
142.Business Intelligence Framework	301
143.Three DSS Axes/Trends	304
144.Scalability with DSS using DB2 PE	305
145.Commercial Applications.	315
146.3-Tier Architecture.	318
147.2-Tier and 3-Tier Architecture	319
148.OLTP Scheme with SP	320
149.Typical SP OLTP 3-Tier Environment	321
150.Typical SP DCE OLTP 3-Tier Environment.	322
151.Function Shipping	332
152.Data Shipping	333
153.Four Physical Nodes Using DB2 and the SP switch	335
154.Partitioning Map	337
155.RS/6000 SP Configuration with HACMP.	337
156.Oracle Parallel Server	341
157.Events During a “ping”.	343
158.Events During a False “ping”.	344
159.Query Execution Time.	351
160.Speedup Numbers	351
161.SP As a Flexible Server	358
162.Three-Dimensional Turbulence Calculation Done at LLNL.	365
163.Parallel Programing Tools Offered for the RS/6000 SP	366
164.Biological/Chemistry/Process Parallel SP Applications.	368
165.Upscaled vs Full Model Using Landmark’s Parallel-VIP	371
166.Petroleum Exploration and Production Parallel SP Applications.	372
167.A Sample View of CATweb	375
168.Mechanical Engineering Parallel SP Applications.	376
169.Electronic Design and Analysis Parallel SP Applications	377

170.Speedup on RS/6000 SP using Parallel LS-DYNA3D 378

Tables

1. Flynn's Taxonomy	10
2. Expansion Slots Available on the Most Current Nodes	28
3. Internal Disk Available on the Most Current Nodes	29
4. Memory Options Available on the Nodes	29
5. Number and Kind of Processor on the Nodes	30
6. Minimum Level of PSSP and AIX that is Allowed on Each Node	32
7. Supported Control Workstations	37
8. List of sysctl Scripts included in the /etc/sysctl.conf File	171
9. Supported Change (Migration or Upgrade) Paths under PSSP 2.4.	224
10. PSSP and AIX Levels	226
11. Different LoadLeveler Daemons Run on Different Nodes	263
12. Node Allocation	306
13. Parallel Query Processing - Architecture Comparison	334

Preface

The Scalable Parallel (SP) system is one of the most successful products within the RS/6000 family. Its reputation is broadly known through multiple events, such as the chess matches with Kasparov, and the hosting of mega Web sites as the Atlanta and Nagano Olympics. However, many people know Deep Blue only through these events. But what exactly is Deep Blue? What kind of software does it run? How it works?

This redbook provides answers to all those questions and more. It provides a comprehensive and detailed look at the different components that make up this giant. It also includes the history and evolution about the SP and how all started in the late 1980s.

The Team That Wrote This Redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization Poughkeepsie Center.

Marcelo R. Barrios is a project leader at the International Technical Support Organization, Poughkeepsie Center. He has been with IBM for five years working in different areas related to RS/6000. Currently he focuses on RS/6000 SP technology by writing redbooks and teaching IBM classes worldwide.

Mohammad Arif Kalem is an RS/6000 and SP Product Specialist at Saudi Business Machines (SBM), Saudi Arabia. He joined IBM Pakistan in 1990 and worked in AIX support before joining SBM in 1996. During the last two years at SBM, he has focused on RS/6000 SP products for his large customers in the petroleum industry. Arif has an MBA (MIS) degree from the Institute of Business Administration, Karachi, Pakistan.

Yoshimichi Kosuge is an Advisory RS/6000 SP Project Leader at the International Technical Support Organization (ITSO), Poughkeepsie Center. He is responsible for RS/6000 SP System Management, HACMP ES, and Cluster Technology. Before joining the ITSO, he worked at IBM Japan as an Advisory I/T Specialist for RS/6000 and AIX.

Philippe Lamarche is an IT Specialist in AIX presales in West Region Europe since 1995. He holds degrees in Electronics and Physics from French universities. He joined IBM in 1982 as a professional designer for IBM microelectronics at the Essonnes Laboratory, France. He has been involved

in different SP and applications sizings with French customers. His areas of expertise include UNIX graphic workstations and MCAD products.

Belinda Schmolke is an RS/6000 and SP evangelist in South Africa. She has 11 years of experience in the UNIX field. She holds a bachelors degree in Computer Science and Engineering from the Massachusetts Institute of Technology.

George Sohoh has only recently joined IBM as a Development Staff member at the Scalable PowerParallel Lab in Poughkeepsie, NY. He has extensive experience developing and tuning supercomputer applications. George holds a B.Sc. in Mathematics from the University of Patras and a Ph.D. in Applied Mathematics from the University of Arizona. Before joining IBM, he worked as a scientist and consultant for the environmental engineering industry.

Juan Jose Vasquez is an SE working for OFISER Ltd., an IBM Chile business partner, since 1992. He holds a Electrical Engineering degree from Universidad de Santiago. He has been involved with AIX machines since 1992. He also teaches AIX courses in IBM Chile. His areas of interest cover HACMP and SP systems.

Tim Wherle has held many titles during his nine years with IBM that all essentially mean Systems Engineer. He currently works for IBM Canada in Ottawa, marketing mid-range servers primarily to the Canadian federal government. Tim's areas of expertise are SP technical marketing, SP implementation. He also has broad experience with the full RS/6000 server family and the AS/400.

Thanks to the following people for their invaluable contributions to this project:

Stephen Atkins
Peter Badovinatz
Stephanie Beals
Bob Bartfai
Mike Browne
Bill Carlson
Endy Chiakpo
Mike Coffey
Rich Davis
Mike Ferrell
Rich Ferri
Todd M. C Li
Ron Linton
Ed Merenda
Bill Mihaltse

Jamshed Mirza
Brian O'leary
Mike Schmidt
Paul Silverstein
Steve Tovcimak
IBM Poughkeepsie Center

Comments Welcome

Your comments are important to us!

We want our redbooks to be as helpful as possible. Please send us your comments about this or other redbooks in one of the following ways:

- Fax the evaluation form found in "ITSO Redbook Evaluation" on page 399 to the fax number shown on the form.
- Use the electronic evaluation form found on the Redbooks Web sites:
For Internet users <http://www.redbooks.ibm.com>
For IBM Intranet users <http://w3.itso.ibm.com>
- Send us a note at the following address:

redbook@us.ibm.com

Part 1. The Big Picture

This part of the book puts the RS/6000 SP in context. It describes the marketplace requirements that prompted IBM to develop a supercomputer, and the history of the SP's development. We describe the high-level system characteristics that have made the SP so appealing in the marketplace, and allowed IBM to continually build on its success. We also define the SP's place in the broad categorization of computer architectures.

Chapter 1. History and Design Philosophy

Since its introduction to the marketplace four and a half years ago, the Scalable Powerparallel (SP) supercomputer has built up an impressive resume. The SP has

- Been installed in over 70% of the US Fortune 500 companies, as well as scientific and technical institutions worldwide.
- Beaten humanity's chess champion, Gary Kasparov.
- Expanded from 0 to approximately 28% of IBM's total UNIX-based system revenue.
- Been selected by the US Department of Energy for the Accelerated Strategic Computing Initiative (ASCI) project, which will result in the creation of the most powerful computer in the world.
- Repeatedly set world records for Internet Web-serving capability, most recently at the Nagano Winter Olympics.

In this chapter we discuss the types of computing problems that prompted IBM to create the SP. We review the history of the SP's development, and present its key design points.

1.1 Ultra-Large Computing Problems

The development pace and raw capabilities of computer systems over the last forty years astonish most of us; however, the progress of computer technology is no match for our ability to generate new, bigger, more complex problems for computers to solve. The body of human knowledge and creation continues to grow exponentially, as does the planet-wide infrastructure - the Internet - to share it all. The globalization of world economies has forced businesses to seek any competitive advantage, such as extracting business intelligence from the vast data reservoir of their operational transactions. Consider the data processing requirements of the following problems:

- Accurately simulate the explosion of a nuclear weapon
- Rapidly analyze 3D seismic surveys, each involving 1 to 10 terabytes of data, for a resource exploration client
- Integrate all core business transactions - financial, materials management, inventory, sales, and so forth - of a global enterprise comprising tens of thousands of users
- Analyze 200 million chess board positions *each second* to be competitive with human grandmasters

Talk about job security for a well-designed ultra-large computer.

Traditionally, *supercomputers* have solved these types of problems. A supercomputer is typically described as having an immensely powerful, expensive, esoteric, complex, and delicate "personality". It is usually a single, monolithic system, managed by graduate-level scientists and researchers, and somewhat of an elite tool for academia, government, and perhaps some large corporations.

Today, clustering of computers is emerging as a technological solution for these ultra-large scale problems. Although the SP is central to IBM's mid-range clustering development, it was conceived in the supercomputer era.

1.2 Origins of the SP

In the late 1980s, IBM set out to build a supercomputer for large, technical customers. The High Performance Supercomputer Systems Development Laboratory (HPSSDL) was formed within IBM's Large Systems Division in Kingston and Poughkeepsie, New York. HPSSDL intended to create a supercomputer with a more familiar personality, one based on widespread, non-exotic technology. Not surprisingly, IBM's ES/9000 mainframe vector processor architecture initially provided the basis for development. This architecture eventually proved to be too limiting. Implementation aspects such as guaranteed instruction execution order, special interrupt handling, and a unique floating point representation (incompatible with the emerging IEEE-based standard) restricted the speed and interoperability of the design.

In 1990 IBM's advanced workstation division in Austin, Texas introduced the RISCSystem/6000 (RS/6000) family of UNIX-based workstations and servers. These early RS/6000 machines boasted stellar floating point performance for their time, owing to the strength of the Performance Optimization with Enhanced RISC (POWER) CPU architecture. The fact that they ran UNIX was of great interest to HPSSDL, as UNIX was entrenched in their target market - large scientific and technical customers. HPSSDL, which was at an impasse with mainframe processor technology, experimented with off-the-shelf RS/6000 machines by adding ESCON adapters and interconnecting them with an ESCON Director. The RS/6000 machines were repackaged as nodes and placed in frames. Only five of the large, sheet metal drawers for the nodes could be placed in one frame. The drawers were judged to be too small to contain a person, so they were nicknamed "dog coffins."

As HPSSDL was creating dog coffins, an IBM research group in Yorktown, New York was working on a high-speed switch code-named Vulcan. Yet another group in Yorktown was trying to solve both the problem of deploying these hot new RS/6000 workstations to the desktops of IBM workers, and the system management headaches that come with LAN administration. This group developed a frame that could house 16 RS/6000 machines, as well as management software called Agora to create a true "LAN-in-a-can".

In December 1991, these independent efforts began to come together. HPSSDL was reorganized and renamed to HPSSL (the "Development" part of the name was dropped) under the leadership of Irving Wladawsky-Berger. (Mr. Wladawsky-Berger went on to become head of the RS/6000 Division, and currently is the General Manager of the Internet Division.) HPSSL's mission was to ship a product in 12 months. Designing a system from scratch was out of the question given the time constraint, so a task force was created, which selected the necessary system components from available IBM technology. The RS/6000 Model 370 furnished the node technology. The Yorktown LAN consolidators provided their knowledge and experience in packaging the nodes. The Vulcan switch, now code-named Envoy (the origin of the "E" commands for the switch), was chosen over the ESCON interconnect, which was experiencing problems with excessive latency. Work from the ESCON interconnect experiment formed the basis for the first iteration of the Vulcan switch software. The total product was introduced to the marketplace as the SP1.

In September, 1993, Argonne National Laboratories in Argonne, Illinois received shipment of the first SP1, a 128-node system. Cornell University in Ithaca, New York, bought a 512-node system shortly thereafter. Next came the petroleum industry. By the end of the year, 72 systems had been installed around the world, all with scientific and technical customers.

Initially, IBM had no intention of positioning the SP1 in the commercial marketplace, but commercial customers started knocking on IBM's door. In the early 1990's, the death of the mainframe was accepted as conventional wisdom. Therefore, many large commercial enterprises were looking for alternatives to the mainframe in order to deploy new applications. IBM formed an application solutions group for the SP1, which, among other things, ported a parallel version of Oracle's database to the SP1. In 1994, SP development absorbed personnel from the discontinued AIX/ESA product, who bolstered the system's manageability and helped spawn the Parallel System Support Program (PSSP, described in Part 2, "System Implementation"). By the end of 1994, the commercial segment accounted for 40% of all installed SPs. By the end of 1996, the share climbed to 70%.

The SP2 was announced in 1994. It incorporated new node types from Austin and a faster switch, code-named Trailblazer (the origin of the tb2 and tb3 nomenclature of the switch adapters). The SP2 had moved out from under the umbrella of the Large Systems Division and was its own little enterprise within IBM. SP2 sales were strong: 352 systems were installed by the end of 1994, and 1,023 by the end of 1995.

In 1996, the SP2 was renamed to simply the SP and formally became a product of the RS/6000 Division. It represents the high-end of the RS/6000 family. IBM secured a number of large SP contracts, of particular note the ASCI project of the US Department of Energy. These contracts, coupled with the broad marketplace acceptance of the product, have fuelled SP development. In 1996, IBM introduced a faster version of the Trailblazer switch, more than doubling the bandwidth of its predecessor, new nodes, including Symmetric Multiprocessor (SMP) versions, and more robust and functional PSSP software. As of the end of 1997, there were over 3770 SP systems installed throughout the world. From 1994 to 1997, the SP install base has been growing at an annual rate of 169%.

1.3 High-Level System Characteristics

The marketplace acceptance of the SP rests on the underlying design philosophy of the machine. From the product's inception, IBM strived to create a supercomputer that had the following high-level system characteristics:

- Scalability to ultra-large sizes
- Use of mainstream architectures and technologies
- Flexibility
- Manageability

We describe these characteristics in this section.

1.3.1 Scalability

The SP efficiently scales in all aspects, including:

- Hardware and software components, to deliver predictable increments of performance and capacity
- Network bandwidth, both inside and outside the machine
- Systems management, to preserve the investment in tools, processes, and skills as the system grows

Importantly, the SP can scale both up and down. It can be subdivided, either logically or physically, into smaller SPs. In addition, scaling is a consistent

process regardless of the initial size of an SP implementation. A customer with a one-node SP can grow to 512 nodes the same way a customer with a 400-node system does. The simple erector set style of expanding the system also suits many customers' procurement environments. A researcher or department with some funds could easily add a node or two or upgrade the memory in an existing node, without incurring a large upgrade or box swap cost.

1.3.2 Use of Mainstream Architectures and Technologies

1.3.2.1 Architectures

Unlike many other supercomputers, the SP does not implement exotic, special-purpose architectures. It supports the key programming models in both the technical and commercial areas. Enablers, tools, and skills to program the SP are widely available.

1.3.2.2 Technologies

The SP relies heavily on mainstream hardware and software components. This bolsters the SP product business case by minimizing development costs and integration efforts.

The system runs Advanced Interactive Executive (AIX), IBM's standards-compliant implementation of UNIX, and as a result inherits a portfolio of over 10,000 off-the-shelf applications and enablers. The hardware building blocks of the system are simply RS/6000 machines. The SP attaches the same disk and tape subsystems as any RS/6000 machine, and is managed with the same enterprise-level tools, such as Tivoli and Adstar Distributed Storage Manager (ADSM).

1.3.2.3 High-End Technology Flows into Volume Marketplace

The SP's affinity with the mainstream RS/6000-AIX marketplace allows innovations from the ultra-large SP implementations to flow into the mainstream volume marketplace. A prime example of this is the U.S. Department of Energy ASCI project: switch, node and software technology that IBM has committed to deliver in the ASCI contract will be available to *all* SP customers a year or two later.

1.3.2.4 Marketplace Volumes Fuel High-End Development

The marketplace success of SP technology, developed to meet the requirements of the high-end customers, generates the required revenue to sustain high-end development.

1.3.3 Flexibility

The SP is a multipurpose platform addressing a broad range of customer requirements. Not only can it help solve problems across a range of industries, but a single SP can also be used for a variety of applications within a single organization. A customer can easily configure an SP to support diverse workloads such as serial batch jobs, parallel jobs, interactive applications, communication gateways and routing, and serving of all kinds.

1.3.4 Manageability

The SP is production-worthy. It has good reliability, availability and serviceability, allowing it to host mission-critical applications. It has a single point-of-control and is managed with consistent tools, processes and skills regardless of system size.

Chapter 2. System Architectures

In the computer world, the architecture is constantly changing to keep pace with continually evolving technologies and increasing demands by the users.

In the beginning, people were content with one processor running at thousands of clock cycles per second, but soon that was not sufficient and the clock speed was increased, and increased. This is what is happening in the Intel world. The first Intel processor was released in November 1971. The 4004 ran at a clock speed of 108KHz. The latest Intel processor reported on the Intel home page is the Pentium II processor, which runs at a clock speed of 300 MHz, an improvement of much more than 250,000 percent. IBM has recently announced the first prototype of the giga-processor, which has one thousand million clock cycles per second.

Although the speed of conventional computers has been increasing rapidly since their invention, they are still not fast enough to compute the answers to massive mathematical models or to give reasonable response time when retrieving useful collections of data from databases. The requirements are getting more and more complex and the amount of data being manipulated is getting larger and larger. Increasing the clock speed further is getting increasingly more complex and costly because of the physical limitations of the elements used to make the chips. But the clock speed is just one of the many things that must be manipulated to increase the amount of work done per unit of time.

2.1 Way of Categorizing Computers

A method of classifying computers was devised in the late 1960's by Michael Flynn. Flynn was/is a professor of electrical engineering at Stanford University in California. Flynn's taxonomy has become widely accepted. It describes computers by how streams of instructions interact with streams of data, as follows:

- There can be single or multiple instruction streams.
- There can be single or multiple data streams.

This gives rise to four types of computers. Table 1 on page 10 shows this classification.

Table 1. Flynn's Taxonomy

		No. of Data Streams	
		Single	Multiple
Number of Instruction Streams	Single	SISD	SIMD
	Multiple	SIMD	MIMD

2.2 Single Instruction, Single Data Uniprocessor

In 1945, mathematician John von Neumann from the Institute of Advanced Studies at Princeton University described how a computer was made up of three elements:

- A memory where instructions and data were stored
- A processing element where the instructions were executed
- A pathway, or bus, to move data or instructions between the memory and the processing element

This is the Single Instruction stream, Single Data (SISD) stream model as defined by Flynn. The first computers designed in the late 1940s using von Neumann's ideas had a memory element (RAM) of 1000 words, were the size of a grand piano, used vacuum tubes and were very costly to maintain. They eventually evolved into the first commercially available computers in the mid 1950s. In the late 1950s the transistor was invented to replace the bulky vacuum tubes, and soon after, the integrated circuit followed. The integrated circuit sped things up considerably as the electrons had much less distance to travel and cost much less to maintain and manufacture. In 1971, Intel released the first microprocessor. Since the first microprocessor, many advances have been made to make the SISD uniprocessor computer work faster and process more data per unit of time.

In the von Neumann machine, the instructions direct the processing element to perform different operations on the data. A clock is used to synchronize operations, and there has to be a control unit to direct the flow of data and to decode the instructions. This is shown in Figure 1 on page 11.

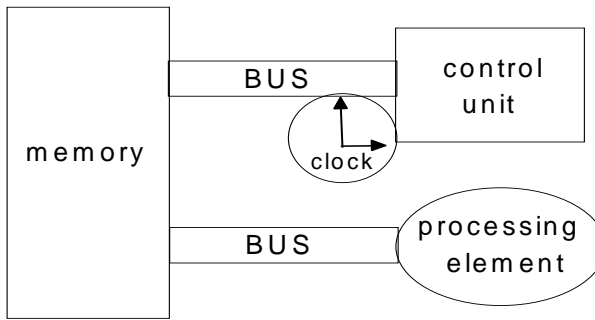


Figure 1. von Neumann Architecture

Instructions are the basic work elements of a computer. In the beginning, the following were true:

- There was a wide array of different instructions which were able to do both complex and simple actions. A complex instruction example is: "copy this block of memory from here to there". A simple instruction example is: "change the sign of this number".
- Each instruction was a different length depending on its complexity.
- The chips had to be elaborate, and there was associated microcode to understand the sophisticated (long) instructions.

This meant that the clock cycle was determined by the length of time that it took to execute each instruction. These computers did not make efficient use of the processing complex because:

- The control unit had a large number of operations to search to find each specific operation to execute. The average search time was equal to the time it takes to find the middle operation in the long list of operations. So if the choice of operations to execute were shortened, then the search time would be reduced.
- Since each instruction was of different length, the control unit did not necessarily know when to load the next instruction. So sometimes the processing unit ended up waiting a clock tick or two at the end of the execution of a long instruction. If all instructions were of equal length, then the processing complex would always know when to start loading the next instruction.

And, because of the Complex Instruction set, the chips had to be rather detailed and elaborate, and some of the longer instructions were even microcoded into the chip.

To counter the inefficiencies of this Complex Instruction Set, the Reduced Instruction Set Computer - RISC was developed. This processor was much faster and more efficient than the CISC processor because:

- Each instruction was the same size, so the control unit always loaded the next instruction at the right time.
- There were fewer instructions to search through to find the next one to execute, so the search time was reduced.
- Because the instructions were simpler, the chip could also be simpler. The instructions could be wired into the chip, not microcoded, which improves efficiency and speed.

In keeping with the spirit of making the processors faster and more efficient, IBM developed the POWER Architecture. POWER stands for Performance Optimized With Enhanced RISC. This means that instead of having one big processing element, they divided up the processing element into several parts, each one dedicated to a specific set of instructions. If the control unit were able to organize the operations properly, it would be able to pass multiple instructions simultaneously to the processing element, thereby achieving superior performance and efficiency.

But still the speed of these machines was limited by the speed of the clock cycle.

2.2.1 Everyday Examples

The whole of the RS/6000 family of computers was uniprocessor until the first model J30 was released. The original uniprocessors with the POWER architecture were the 2xx, 3xx and 5xx models.

2.2.2 Limitations of SISD Architecture

Since the clock cycle determines the speed of the computer, decreasing the time would make the computer faster. To achieve a faster clock cycle:

- The signal path must be made smaller.
- The transistors must be made faster.

Unfortunately, both seem to have reached a physical limit.

To make the signal path smaller would be very costly, and the RS/6000 range was designed to be cost effective. To increase the number of transistors per unit area would dramatically increase the heat generated per unit area, and the extra cooling systems would be the costly elements. With each new generation of chips, the transistor density doubles.

Engineers are working on lower-power chips which would reduce the heat generated. The goal is to have the power density remain constant while increasing the number of transistors per unit area.

To make the transistors faster is also costly. To achieve this, engineers are working on reducing the resistance in the wires that carry the current. Aluminum is the element that is used in 1998, but recently a chip with copper wires has been developed. In addition to reducing the resistance of the wires, the developers are also working on improving the insulation between the wires.

Given, however, that the clock cycle cannot be made equal to zero, the sequential mode of operations imposed by the von Neumann model destroys any further hope of increasing the speed of the computer. This is known as the von Neumann Bottleneck. To break this bottleneck, a non-serial architecture must be developed. Enter parallelism.

2.3 Parallelism

One way to avoid the von Neumann Bottleneck is to perform actions concurrently. An action involves instructions and data. There are three possibilities for parallelism:

- Process the data elements in parallel. Copies of the same instruction are used to process different data elements concurrently. This would be an SIMD machine.
- Process the instructions in parallel. Several instructions cooperate to process the same data elements simultaneously. This is an MISD machine.
- Do both at the same time. Several instructions processing different data elements execute concurrently. This is an MIMD machine.

There are inherent difficulties and limitations with parallel architectures. These problems range from the difficulty of programming to the cost of the machines. The first two of these parallel options were not considered by the IBM developers due to the original design goals to use off-the-shelf hardware (to benefit from the economy of scale of using normal everyday machines), and these specialized architectures would have to be made specially for a customer at great cost to that customer.

2.4 Single Instruction, Multiple Data (SIMD) Machines

In SIMD machines, many processing elements execute the same instruction on different data elements. The use for this kind of machine is limited, even though, under the right circumstances, it is very fast. IBM does not specifically manufacture any, but the RS/6000 SP could be used as an SIMD machine if the specific programming tools were created to feed each node the same instruction to be executed on different data elements in parallel.

An SIMD machine has the following components:

- One front-end machine that runs the compiled code
- An bank of processors with minimal memory that do simple operations on data which they receive from the front-end machine

The front-end machine looks for any arrays of data that can be processed in parallel, with the same instruction running on different processors, acting on different data elements of the array, and sends the parallel work to groups of processors.

This architecture is amazingly efficient when the data sets are huge arrays that have to have the same operation applied to the whole or part of the array. SIMD machines are also well suited to processing neural nets.

2.4.1 Everyday Examples of SIMD Machines

The most common example of this type of architecture is the Connection Machine, which is manufactured by the Thinking Machines company in Cambridge, Massachusetts.

This machine is black with a panel of red LEDs on the outside. The LEDs go on and off to indicate which processor or group of processors is working. It is intriguing to watch the LED patterns, which change with each diagnostic program that is running. With relatively simple changes to the code that is used to manipulate the various arrays, one can achieve greater efficiency, and more of the LEDs go on simultaneously.

2.4.2 Limitations of the SIMD Architecture

This machine is relatively expensive to produce, because the banks of processors are not standard "off the shelf" components. While the number of customers who need to process exclusively large arrays of data is growing, there are not enough to justify a production plant exclusively devoted to manufacturing machines of this architecture, and so these machines are generally custom-made at great expense.

2.5 Multiple Instruction, Single Data Machines

Although this machine is theoretically possible, it is not manufactured commercially. It has been built in research laboratories, but there were no customers willing to invest in this architecture because it does not specifically address any of the commercial or mainstream technical applications.

2.6 Multiple Instruction, Multiple Data Machines

Multiple Instruction, Multiple Data Machines are the machines that are commonly known as multiprocessors, even though the SIMD machine is also technically a multiprocessor. In MIMD computers, parallelism is achieved by having multiple different instructions executing at the same time on different data elements. Obviously, additional control elements are required to get the right instructions and the right data to the right processor at the same time.

Memory is where the data and the instructions are stored, and so each processing unit needs to access the memory somehow. MIMD computers are divided into two types with respect to how the memory is handled:

1. Shared Memory, accessed by all processing units
2. Shared Nothing, where each processing unit has its own memory, disk, and communications

2.6.1 Shared Memory MIMD Machines

With the Shared Memory MIMD machine, the original von Neumann machine has been extended to include what is known as a cache. This is local memory attached to each specific processor. This cache allows the processor to have faster access to its own instructions and data, but it still accesses the main memory through the memory bus. This parallel machine architecture is shown in Figure 2 on page 16.

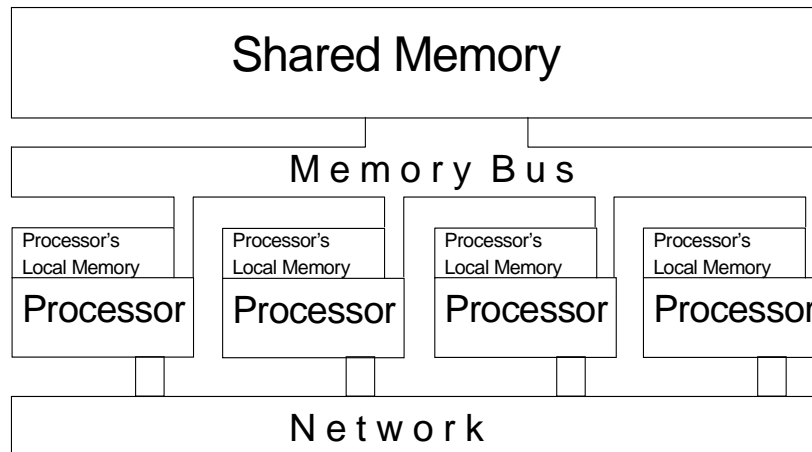


Figure 2. Schematic View of a Shared Memory MIMD Machine

2.6.1.1 Everyday Examples of Shared Memory MIMD Machines

1997 IBM examples of Shared Memory MIMD machines are the RS/6000 models J40, J50, R40 and R50. The IBM implementation of the Shared Memory MIMD machines has taken the von Neumann original machine a step further.

Instead of having only one level of cache on each processor, the J40s have two levels of cache, which gives easier access to the instructions retrieved from main memory.

Instead of using a memory bus to link the various processors to the main memory, IBM developed a memory cross bar, which has better throughput than a conventional memory bus. The conventional bus only allows one (albeit very fast) transaction at a time, but the cross bar allows up to four transactions at the same time, at the same fast speed.

As you see in Figure 3 on page 17, level 1 cache is per processor, and the level 2 cache is per processor card. Two adjacent processors share the same level 2 cache. Models J40, J50, R40 and R50 can have either 2, 4, 6 or 8 processors per machine because the processors come in pairs on cards with shared level 2 cache.

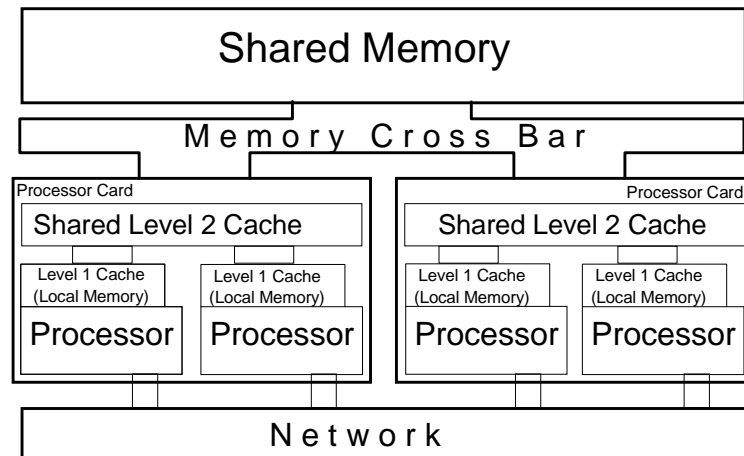


Figure 3. IBM's MIMD Shared Memory Implementation in RS/6000 Model J40

A 1998 example of a Shared Memory MIMD machine is the RS/6000 model F50, which allows 1, 2, 3 or 4 processors. This allows the users more flexibility - instead of being forced to buy pairs of processors, they can get odd numbers of processors. The disadvantage here is that if one had three processors and wanted to go up to four processors, one would have to replace one processor with a two-processor card.

2.6.1.2 Limitations of Shared Memory MIMD Machines

With the shared memory extended version of the von Neumann machine, the bottleneck is the memory bus. Every instruction or data element that is used by any of the processors has to go through the same bus. When the number of processors using the same memory is small, the performance increase of doubling the number of processors is almost linear, but as more and more processors are added, the performance gain is less and less. This is shown schematically in Figure 4 on page 18. Some people dispute the numbers along the bottom axis (number of processors), but, whatever scale is used, the idea is the same: performance gain by adding Shared Memory MIMD processors is not linear.

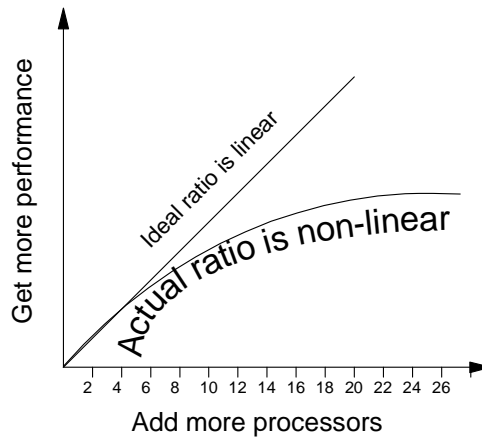


Figure 4. Relationship between Extra Processors and Performance Gain

IBM tried to address this problem by using the memory cross bar instead of the conventional memory bus. IBM also implemented the level 1 and level 2 caches with a view to reducing accesses to the main memory. These enhancements have merely pushed the bottleneck a bit further out.

2.6.2 Shared Nothing MIMD Machines

With Shared Nothing MIMD machines, each processor has its own memory. These machines are basically independent computers sharing nothing except a network, as shown in Figure 5 on page 18. This arrangement is commonly referred to as "the nodes" of a parallel machine.

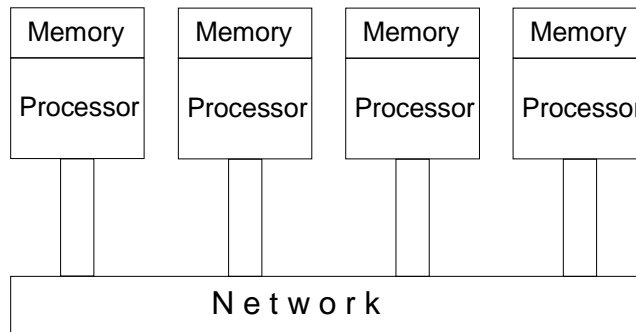


Figure 5. Conventional Shared Nothing MIMD Machine

In order for the machines to work together effectively when each one is doing different instructions on different data independently of the other machines, there has to be a good message passing protocol. For parallelism to be effective with these completely independent nodes, the message passing protocol has to be very fast and absolutely trustworthy. IBM developed a high-speed switch to enable the nodes to talk to each other over multiple redundant paths. To support this hardware, IBM uses a software message passing protocol called Message Passing Interface API. This Application Programming Interface is distributed free of charge to developers who want their previously nonparallel programs to take full advantage of the parallelism allowed by the high-speed switch. The nodes are also connected by a conventional (slow) Ethernet network for system management purposes. The increased bandwidth and speed of the high-speed switch network is shown in Figure 6.

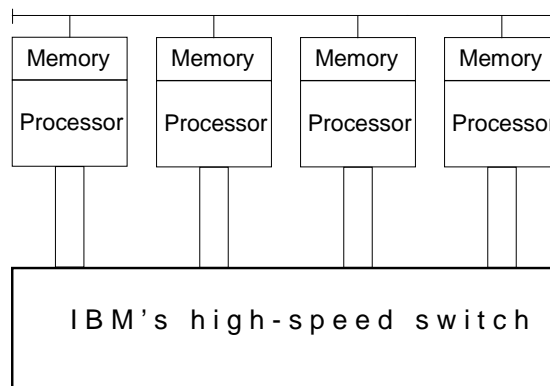


Figure 6. IBM's First Enhancement of the Shared Nothing MIMD Machine

IBM's implementation of this architecture is designed to be scalable and cost effective. That is, the customer should be able to connect as many individual machines as his application requires without paying a premium.

IBM has achieved this cost effectiveness by grouping the individual nodes in frames with one high-speed switch per frame. The customer has therefore only to buy one high-speed switch per frame. The high-speed switch is the only component in IBM's implementation of the Shared Nothing MIMD architecture that is not standard, commercially available RS/6000 technology.

The scalability of the communications is achieved by the use of the high-speed switch, which allows near linear scalability of the message passing protocol that is used by the various processors when they are doing parallel jobs. There is a detailed description of the high-speed switch topology and software in Chapter 3.4, "High-Performance Communication

Network” on page 38 and in Chapter 4.3, “Overview of Switch Software” on page 93.

System management is a big issue when many machines are connected in such a way. The objective is to minimize the time spent logging into each different node to do its specific system administration. If all the nodes could be controlled from one central place, then performance, security, storage, errors and all the other system management issues could be addressed more effectively. IBM has addressed this issue with the introduction of the Control Workstation and the Parallel System Support Programs, which is the software that one interfaces with on the Control Workstation to manage the various nodes.

Some of the RS/6000 nodes have LEDs that indicate the status of the hardware during the bootup process, and also if there are any serious hardware problems. Some of the RS/6000 nodes have keys that have three positions to indicate to the hardware whether the machine is to boot up in normal, secure or service mode. It would be inconvenient to have to get up from the Control Workstation to go and look at the various LEDs and move the keys, so the system management software from the Control Workstation manages the keys electronically and displays the status of the LEDs in a window on the Control Workstation screen.

In order to have this hardware control system management, there has to be a dedicated serial link from each node to the Control Workstation. This means that hardware system management information does not compete for communication bandwidth with the regular system management information that is moving between the nodes on the Ethernet. IBM has implemented this system management communication with the Control Workstation through the use of serial cables. Instead of having one serial cable from each node to the Control Workstation, there is one serial cable from each frame to the Control Workstation. There must therefore be a frame supervisor card in each frame that allows the Control Workstation to talk to all the nodes in that particular frame. These enhancements to the conventional Shared Nothing MIMD architecture are shown in Figure 7 on page 21.

There are further enhancements to the conventional Shared Nothing MIMD model.

Instead of just having one processor/memory complex as each node, one could have any of the computing models that have been discussed so far, particularly:

1. SIMD uniprocessor machines

2. Shared Memory MIMD machines

This choice gives rise to a discussion of the different types of nodes that one could put in the frame of the machine. This is covered in detail in Chapter 3.2, "Nodes" on page 27.

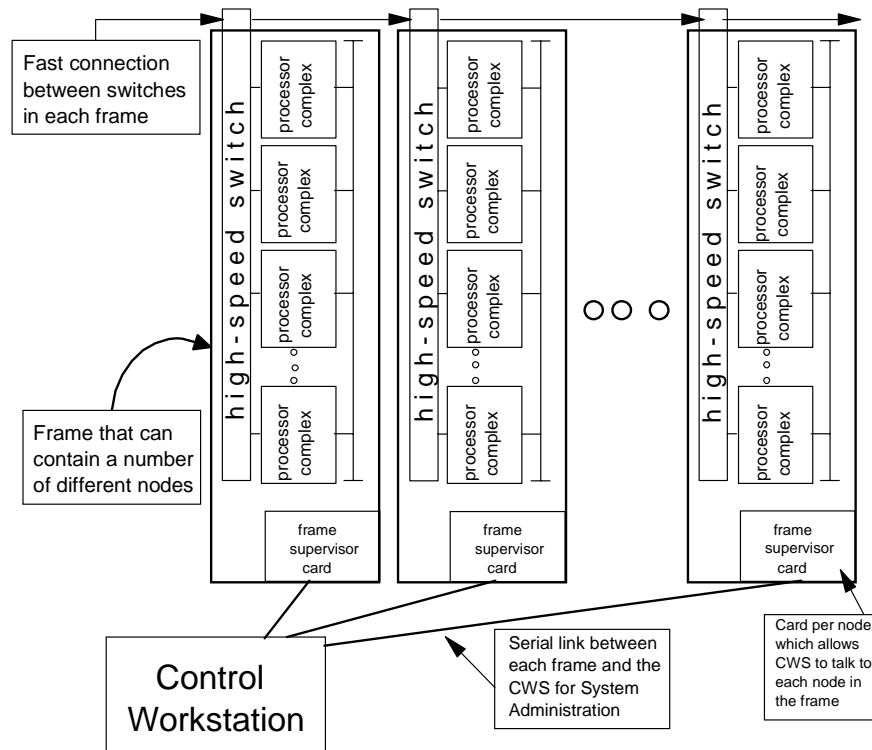


Figure 7. Further IBM Enhancements to the Shared Nothing MIMD Machine

There are many issues associated with the system management of such a large system. The Control Workstation and the frame supervisor cards are just part of the system management of this system; the details are presented in Chapter 4, "Software Components" on page 81 and in Chapter 3, "Hardware Components" on page 25.

Scientific programming of these particular large systems is still finding better ways to solve larger and more complex Grand Problems, and programmers are inventing new disciplines to make use of this parallel environment. There are two levels of parallelism:

- Many individual nodes are able to work together on the same problem (interprocess communication facilitated by the high-speed switch).
- Multiple processors within each node work together on the same part of the same problem. There is a section on the scientific uses of this system; see Chapter 9.1, “Research Applications” on page 362 and Chapter 9.4, “Engineering Applications” on page 373.

For the commercial use of this system, the SPMD approach is evolving. With this Single Program, Multiple Data model, the programmer only writes one program that is loaded onto various nodes of the Shared Nothing MIMD computer. The program is written so that each node processes a different part of a massive data set. Special instructions are then included to synchronize the execution in the different nodes, to distribute the data sets, and especially to gather the results in one place. This special way of programming is discussed in detail in *9076 SPx Base Software*, GG24-4351.

Part 2. System Implementation

This second part provides detailed information about the implementation of the RS/6000 SP hardware and software, as well as the system management facilities included in the basic PSSP software. Finally, the last chapter of this part provides a description of high level and cross-platform system management tools that are commonly used in RS/6000 SP Environments.

Chapter 3. Hardware Components

This chapter provides information about the hardware components of the IBM RS/6000 Scalable POWERparallel System (RS/6000 SP).

The basic components of the RS/6000 SP are:

- Frames
- Nodes
- Control Workstation (CWS)
- Switches
- Peripheral Devices

This chapter also provides you with configuration rules for creating or expanding your SP.

3.1 Frames

SP processor nodes are mounted in either a tall or short frame. The frame spaces that nodes fit into are called drawers. A tall frame has eight drawers, while a short frame has four drawers. Each drawer is further divided into two slots. One slot can hold one thin node. A wide node occupies one drawer (two slots) and a high node occupies two drawers (four slots). An internal power supply is included with each frame. Frames get equipped with the optional processor nodes and switches that you order.

Strictly speaking, there are three types of frames:

- Tall frames
- Short frames
- Switch frames

The first two types are used to host nodes, and they are usually called just frames. The third one is used to host switches or Intermediate Switch Boards (ISB), which are described later in this chapter. This special type of frame can host up to eight switch boards.

After the first SP was made commercially available some years ago, there have been a number of model and frame configurations. Each configuration was based on the frame type and the kind of node installed in the first slot. This led to an increasing number of possible prepackaged configurations when more nodes became available.

The introduction of a new frame in 1998 is the first intent to simplify the way frames and the nodes inside are configured. This new frame replaces the old tall frame. The most noticeable difference between the new and old tall frame is the reduction in height. Another physical difference is the footprint.

Before this new frame offering, the frame and the first node in the frame were tied together, forming a model. Each new node becoming available was potentially installable in the first slot of a frame, so a new model was born. With the new offering, IBM simplified the SP frame options by decoupling the imbedded node from the frame offering. Therefore, when you order a frame, all you receive is a frame with the power supply units and a power cord. All nodes, switches, and other auxiliary equipment are ordered separately.

All new designs are completely compatible with all valid SP configurations using older equipment. Also, all new nodes can be installed in any existing SP frame provided that the required power supply upgrades have been implemented in that frame.

The reason for this is that the new nodes introduced with PSSP 2.4 have a higher power consumption, hence there is a higher power requirement for the frame. But before going into node requirements, let us describe some of the frame components.

Note

From here on, the reference to a tall frame applies to the old and the new tall frames, if no explicit mention is made.

3.1.1 Power Supplies

Tall frames come equipped with redundant (N+1) power supplies; if one power supply fails, another takes over. Redundant power is an option on short frames. These power supplies are self-regulating units. Power units with the N+1 feature are designed for concurrent maintenance; if a power unit fails, it can be removed and repaired without interrupting running processes on the nodes.

A tall frame has four power supplies. In a fully populated frame, the frame can operate with only three power supplies (N+1). Short frames come with two power supplies and a third optional one for N+1 support.

The power consumption depends on the number of nodes installed in the frame. For details refer to *Planning Volume 1, Hardware and Physical Environment*, GA22-7280.

3.1.2 Hardware Control and Supervision

Each frame (tall and short) has a supervisor card. This supervisor card connects to the Control Workstation via a serial link.

3.2 Nodes

The basic RS/6000 SP building block is the processor node. Each node has a processor or processors, memory, disk drives and Micro Channel or PCI expansion slots for I/O and connectivity. There are three types of nodes: thin, wide, and high, and they may be mixed in a system and are housed in a frame. Depending on the type of node selected, a tall RS/6000 SP frame can contain up to 16 nodes. These frames can be interconnected to form a system with up to 128 nodes (512 or 1024 by special request).

Also, nodes can be classified in - those which are inside the RS/6000 SP frame, and those which are not. The latter are known as extension nodes and will be discussed after the thin, wide and high nodes. The relationship between these various nodes is shown in Figure 8.

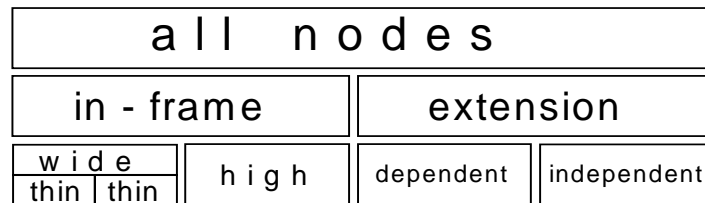


Figure 8. The Relationship between the Different Types of Nodes

3.2.1 Internal Nodes

There are five most current nodes - two thin ones, two wide ones and a high one.

The high node is functionally equivalent to the IBM RS/6000 R50. There can be 2, 4, 6 or 8 604e processors running at 200 MHz. Both the wide node and the thin node come in two flavors: uniprocessor and SMP.

The uniprocessor wide node is functionally equivalent to an IBM RS/6000 7013 - 595 deskside system with an IBM RS/6000 POWER2 processor operating at 135 MHz. The SMP wide node contains either two or four 332 MHz PowerPC processors. These nodes are functionally equivalent to an IBM RS/6000 7025-H50, which uses the PCI bus architecture.

The uniprocessor thin node contains one IBM RS/6000 POWER2 processor running at 160 MHz. These nodes are functionally equivalent to the IBM RS/6000 7012-397 using Micro Channel bus architecture. The SMP thin nodes contain either two or four 332 MHz PowerPC processors per node. These nodes are functionally equivalent to an IBM RS/6000 7025-H50 workstation, which uses the PCI bus architecture. Thin nodes are ordered singly but must be installed in pairs.

The various nodes will now be compared with respect to the various attributes that are used to distinguish between them.

Expansion Slots

Table 2 shows the number of PCI and MCA slots available on the different nodes. If the customer had existing PCI devices that he wanted to integrate into the new system, then there would only be two node choices.

Table 2. Expansion Slots Available on the Most Current Nodes

	Uni. Thin	SMP Thin	Uni. Wide	SMP Wide	SMP High
MCA Slots	Four (3)	-	Eight (1)	-	Sixteen (2)
PCI Slots	-	Two	-	Ten	-

(1) The uniprocessor wide node may have eight MCA slots, but one of them has to be used for the required Ethernet adapter. If the system has a switch, then another slot will be occupied by the required switch adapter card.

(2) The SMP high node may have 16 MCA slots, but a required SCSI-2 adapter uses one of them, a required Ethernet adapter uses another, and if the system uses a high-speed switch, then the switch adapter card occupies another of the MCA slots. This means that there are effectively 13 MCA slots available.

(3) The Uniprocessor thin nodes may have four MCA slots, but if the system is configured with a high-speed switch, one of the MCA slots is occupied by the required switch adapter card.

Internal Disk

Table 3 on page 29 shows the internal disk available on the nodes. Normally the first disk contains the AIX operating system, and customer data is

external to the node to provide for high availability in case of individual node failure.

Table 3. Internal Disk Available on the Most Current Nodes

	Uni. Thin	SMP Thin	Uni. Wide	SMP Wide	SMP High
Min Internal Disk	4.5 GB	4.5 GB	4.5 GB	4.5 GB	
Max Internal Disk	18.2 GB	18.2 GB	36.4 GB	36.4 GB	6.6 GB/ 18.0 GB (1)

(1) The SMP high node has a maximum of 6.6 GB internal disk if the PowerPC 604 processors are used, and a maximum of 18.0 GB if the PowerPC 604e processors are user.

Memory

There are very different memory cards available for use in the different nodes in the RS/6000 SP system. Certain combinations of cards are not possible in specific nodes. It is important to consider the upgrade options when making the initial memory configuration. Table 4 shows the memory slots and memory available for each of the different nodes.

Table 4. Memory Options Available on the Nodes

	Uni. Thin	SMP Thin	Uni. Wide	SMP Wide	SMP High
Min Memory	64 MB	256 MB	64 MB	256 MB	256 MB
Max Memory	1 GB	3 GB (1)	2 GB	3 GB	4 GB
Memory Slots	4	2	8	2	4

(1) The SMP thin nodes have two memory cards and support up to 3GB of memory. Memory is supplied by 128 MB DIMMS that must be mounted in pairs (256 MB increments). The memory cards are not required to be configured symmetrically. Each card has the capacity to mount 2 GB of DIMMS, however, only 3GB are addressable per node.

Number and Kind of Processor

Some applications are likely to be good contenders for running on the SMP nodes, as long as the particular application has been designed and written to exploit an SMP architecture. In the commercial world, with typical database applications, this is very often the case - but not always.

The floating point performance and price performance of the POWER2 Super Chip (P2SC) nodes is superior to the PowerPC 604 node performance for scientific/technical applications. The thin uniprocessor and wide uniprocessor nodes are the obvious choice for these nodes. In practice, a technical or scientific environment might well have a requirement to run particular commercial type workloads, such as file servers or communication gateways, and these workloads (which are not floating point workloads) may be run on other nodes.

For commercial parallel applications (applications written to exploit more than one node in the system) there is even more to think about - whether the application exploits SMP nodes, and how the various nodes can be mixed in such an application. Table 5 shows the different kinds of processors that are in each of the nodes.

Table 5. Number and Kind of Processor on the Nodes

	Uni. Thin	SMP Thin	Uni. Wide	SMP Wide	SMP High
Kind	POWER2	PowerPC (604lx)	POWER2	PowerPC (604lx)	PowerPC 604e
Number	1	2 or 4	1	2 or 4	2, 4, 6 or 8
Speed	160MHz	332MHz	135MHZ	332MHz	200MHz

3.2.1.1 Which Node is Best?

Hopefully, the tables showing the various capacities of the different nodes have shown that size is not the only criterion upon which to base a decision. There are other more subtle criteria.

It is common for RS/6000 SP professionals to consider, now that SMP nodes exist, that everything will run better on these SMP nodes, and uniprocessor nodes will not be needed any more. This is most definitely not true. There are many factors to be considered when selecting nodes, and it will be shown that in some cases the uniprocessor nodes are still the best choice.

Some of the factors to consider when choosing which kind of node to purchase are straightforward:

- Does it have PCI slots?
- Does it have enough MCA slots?
- Can it supply enough memory?

These decisions are related to capacity and the choices are binary - either the node has enough slots or it does not, either the node has capacity for enough memory or it does not.

When determining the required capacity, however, take care to consider all the adapters that will be required, both at installation and for future expansion. Remember that for highly available solutions, it is often necessary to have additional adapters for redundancy.

In the commercial world, it is common to have only the AIX operating system and related logical volumes on the internal disk. It is usual to store critical customer data on external disks, which can be accessed by more than one node. If a node fails, its internal disk cannot be accessed by any other node, so twin-tailing is a better idea for high data availability, but it requires the use of more adapter slots.

The other decisions made when choosing a node are more difficult and complex:

- What are the high availability requirements?
- What are the software requirements?

High Availability

If the goal is high availability, it may not make sense to have one large and powerful node, because this single node would be a single point of failure. It would not be cost effective to provide an additional equal node purely for availability purposes. There is a better option: using a number of less powerful nodes in combination so that if one fails, the rest will carry on.

Redundancy can be provided by using RAID or mirrored disks, alternate networks, additional adapters, and spare or additional nodes.

System Management

One would think that having less nodes would make system management easier. The practicality of this actually depends on what applications are being used. For certain parallel applications and scientific/technical applications, where system management is less of a complex task, having less nodes is not so important.

When the RS/6000 SP is being used for server consolidation, then fewer nodes will definitely mean reduced management overhead.

Uniprocessor systems are generally not good at running multiple applications together on the same system. If it is necessary to run multiple applications on any uniprocessor or SMP because they are sharing resources, they will

always be in contention and it will be impossible to prevent the impact of one application on another.

Some customers buy one SP system, but have different partitions belonging to different departments in the company. The departments are completely separate and the partitions enable the computing facilities related to each department to remain completely separate. There are, however, certain defined node placement rules which are based on Switch Chip Boundaries. The switch and partitioning are discussed in Chapter 3.4, “High-Performance Communication Network” on page 38 and in Chapter 5.5, “System Partitioning” on page 240, respectively.

Software Requirements

The minimum version of PSSP which will run on each type of node is shown in Table 6 on page 32. The application the customer is using may require specific versions of AIX. Not all the versions of AIX run on all the nodes, so this too must be considered when nodes are being chosen.

Table 6. Minimum Level of PSSP and AIX that is Allowed on Each Node

	Uni. Thin	SMP Thin	Uni. Wide	SMP Wide	SMP High
Min. PSSP Level	2.2	2.4	2.2	2.4	2.2
Minimum AIX Level	AIX 4.1.5	AIX 4.3	AIX 4.1.5	AIX 4.3	AIX 4.1.5

Performance Considerations

There needs to be a differentiation between serial and parallel applications. If the customer’s application is a single serial application, it will not, by definition, utilize more than one node in the system.

It is common to run numerous serial applications on the same system, either as client/server applications, or as Server Consolidation. Server Consolidation is where a number of different applications are run on discrete nodes, usually to achieve reduced management and support costs. In this scenario, a database system may be chosen, but the serial or classical version will be used. Refer to “Server Consolidation” on page 355 for more information about this topic.

Such applications are likely to benefit from running on a high node, as long as they were designed to exploit the SMP architecture.

If the application is a parallel application, it is important to consider how the database can handle nodes of varying power within the total configuration.

3.2.2 Extension Nodes

In addition to the internal thin, wide and high nodes, there are extension nodes. Extension nodes extend the RS/6000 SP system's capabilities but are not housed in the RS/6000 SP frame. There are two kinds of extension nodes - dependent and independent. The dependent ones need another node inside the RS/6000 SP to provide them with certain PSSP functions, while the independent nodes connect to the RS/6000 SP switch fabric. They are completely independent and could function equally well with or without being connected to the RS/6000 SP switch.

3.2.2.1 Dependent Node

A specific type of dependent node is the RS/6000 SP Switch Router, which is a high-performance I/O gateway for the RS/6000 SP system and provides the fastest available means of communication between the RS/6000 SP system and the outside world, or among multiple RS/6000 SP systems.

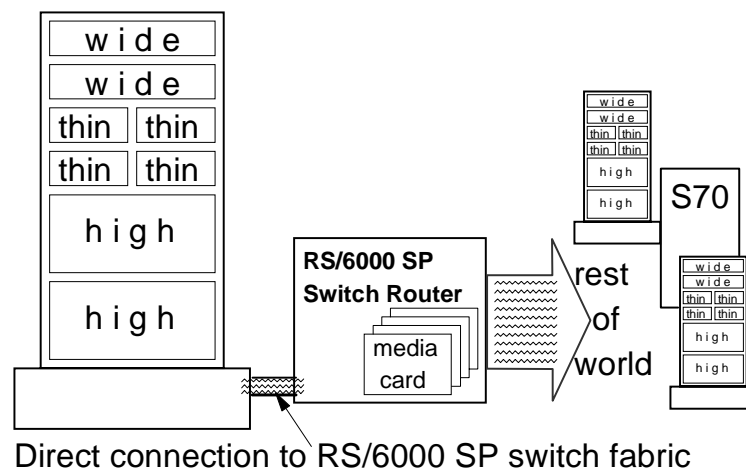


Figure 9. RS/6000 SP Switch Router Offers Infinite Possibilities

This RS/6000 SP gateway combines the Ascend GRF with the IBM RS/6000 SP Switch Router Adapter to enable direct network attachment to the RS/6000 SP Switch. It has four slots for media cards, which provide network attachment, and it comes with an RS/6000 SP Switch Router Adapter in one slot to provide attachment to an RS/6000 SP Switch port. The remaining slots can be used by other media cards to provide attachment to external networks. These slots can also be used by additional RS/6000 SP Switch Router Adapters to provide scalable, incremental steps of RS/6000 SP I/O performance for a single RS/6000 SP system, alternate paths to an RS/6000 SP system, and interconnection for multiple RS/6000 SP partitions and

systems. Each media card has its own IP engine, enabling RS/6000 SP I/O to scale nearly linearly with the number of cards.

Although an RS/6000 SP node can be equipped with a variety of network adapters and used to make network connections for the whole system, the RS/6000 SP Switch Router with the Switch Router Adapter and optional network media cards offers many advantages when connecting the RS/6000 SP to external networks, as follows:

- It offers performance to RS/6000 SP system users who need faster I/O than that provided by node I/O adapters. This includes research organizations, universities, and others relying on leading-edge technical computing applications that need to quickly move large amounts of data between the RS/6000 SP system and external networks; and commercial computing installations that need high-speed access to data servers, such as the new RS/6000 S70.
- It offers availability. A range of features designed for use in high-availability environments can improve the system up-time: dynamic network configuration bypasses failed paths using standard IP protocols; media cards are hot-swappable and can be removed or added while the system is running; power supplies are redundant, load-balancing, and hot-swappable; the RS/6000 SP Switch Router can be accessed and rebooted remotely; and multiple RS/6000 SP Switch Routers can be connected to the RS/6000 SP system to provide alternate paths in case of failure of one. Support for industry-standard features allows administrators to monitor attached networks with existing management tools. For example, the RS/6000 SP Switch Router works with standard SNMP management packages.
- It offers price performance. The RS/6000 SP Switch Router can cut communication costs of the RS/6000 SP system and offer better price/performance than dedicated SP I/O nodes. Since the I/O gateway role can now be performed by the RS/6000 SP Switch Router, the cost of additional processor nodes and I/O adapters can be avoided. In many cases, this lowers the overall cost of an I/O gateway. Also, since RS/6000 SP I/O nodes typically connect to an external hub or router, this cost can be eliminated. And, in installations where there are multiple RS/6000 SP systems or partitions, a single RS/6000 SP Switch Router can act as a common gateway, reducing the gateway cost for each additional SP system or partition to just the cost of an RS/6000 SP Switch Router Adapter. This is especially useful where multiple RS/6000 SP systems or partitions need to interconnect.

3.2.2.2 Independent Nodes

Imagine: "A 64-bit SMP system that delivers the performance, scalability and reliability for today's critical e-business applications. This system excels in online transaction processing (OLTP) applications and supports many popular commercial applications." A machine like this could be an example of an independent node.

Imagine: "A machine that provides the capacity and scalability for linking mission critical applications to your corporate intranet for the exploitation of evolving e-business technologies". A machine like this could be an example of an independent node.

We have the technology, we just need to use it.

3.3 Control Workstation

The beauty of an RS/6000 SP is that it has a single point of control. This is the Control Workstation. The Control Workstation is a separate RS/6000 with a color graphics monitor which serves as point-of-control for managing and maintaining the RS/6000 SP nodes.

The workstation connects to the frame via a 15m IBM RS232 line which is supplied with each frame, and via another form of network: Ethernet. A system administrator can log into the Control Workstation from any workstation on the network to perform system management, monitoring and control tasks. These two different types of connection are necessary because the serial link is used for hardware related control and monitoring communications and the Ethernet is used for system management commands, which are AIX based. The serial connection is also used for hardware-related communications even before AIX has started up on the node.

The Control Workstation also acts as a boot/install server for other servers in the RS/6000 SP system. In addition, the Control Workstation can be set up as an authentication server using Kerberos. It can be the Kerberos primary server, with the master database and the administration service as well as the ticket granting service. Or it can be set up as a Kerberos secondary server, with a backup database, or just the ticket-granting service. The functionality and connectivity of a Control Workstation is shown in Figure 10.

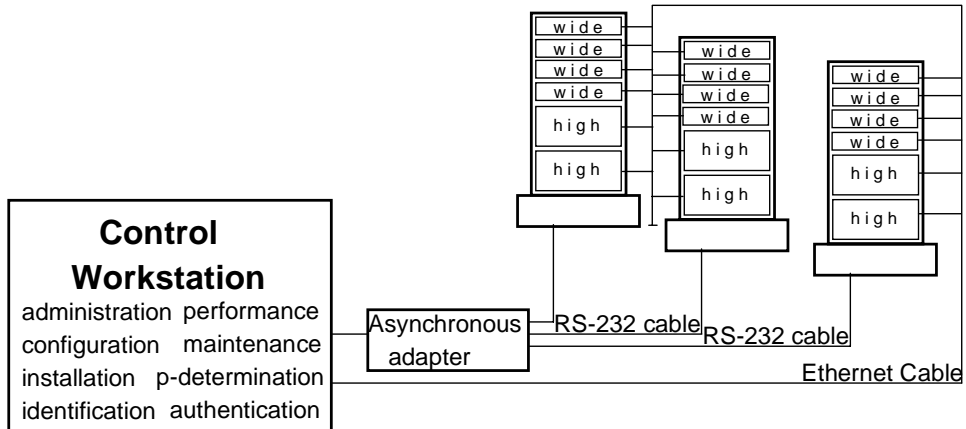


Figure 10. Control Workstation Connectivity and Functionality

Some of the newer control workstations, such as the 7025-F50 and the 7025-F40, offer a support processor as either a standard or an optional feature. The support processor is a standard processor that handles system start-up with some system monitoring functions. When this option is installed with a modem on the first serial port, remote system restarts can be performed on SPs located in unmanned locations.

3.3.1 High Availability Control Workstation

An high availability option is available for control workstations that will enable a second Control Workstation to be connected. This option eliminates the Control Workstation as the single point of failure. When the primary control workstation becomes unavailable, either through a planned event or a hardware or software failure, the RS/6000 SP high availability component detects the loss and shifts that component's workload to the backup system.

To provide this extra availability and to eliminate the Control Workstation as a single point of failure, additional software and hardware are needed.

3.3.2 Supported Control Workstations

The Control Workstation connects to the RS/6000 SP system both through the Ethernet and through the 15m RS-232 cable. System management commands and monitoring go through the Ethernet, and low-level hardware management information goes through the RS-232 cable. Table 7 on page 37

shows the various models of RS/6000 that are supported for use as a Control Workstation.

Table 7. Supported Control Workstations

Micro Channel Control Workstation	PCI Control Workstation
RS/6000 7013 Model 5xx	RS/6000 7025 Model F30 (5)
RS/6000 7012 Model 3xx	RS/6000 7024 Model E20 (6)
RS/6000 7012 Model Gxx	RS/6000 7024 Model E30 (6)
RS/6000 7030 Model 3xx (1)	RS/6000 7025 Models F40 and F50 (7)
RS/6000 7015 Model 9xx (2)	RS/6000 7026 Model H10 (7)
RS/6000 7013 Model Jxx (1)	RS/6000 7043 Models 140 and 240 (7,8)
RS/6000 7011 Model 25x (3)	
RS/6000 7009 Models C10 and C20 (4)	
RS/6000 7006 Model 41T/W (9)	
RS/6000 7006 Model 42T/W (9)	

(1) This requires a 7010 150 X-station and display. Other models and manufacturers that meet or exceed this model can be used. An ASCII terminal is required as the console.

(2) RS/6000 7015 Model 9xx and Rxx in either the 7015-99x or 7015-R00 rack. This requires a 7010 150 X-station and display. Other models and manufacturers that meet or exceed this model can be used. An ASCII terminal is required as the console.

(3) RS/6000 7011 Model 25x. This is not recommended, as total system performance is slow.

(4) RS/6000 7009 Model C10 and C20. This is not recommended beyond one frame due to slow TTY response.

(5) On new systems, either the 8-port or the 128-port PCI bus asynchronous adapter should be used for frame controller connections. The use of the support processor option is highly recommended. If this option is used, the frames must be connected to a serial port on an asynchronous adapter and not to the serial port on the Control Workstation planar board.

(6) Needs to run PSSP 2.2 or later.

(7) The native RS-232 ports on the system planar board cannot be used as TTY ports for the hardware controller interface. The 8-port asynchronous adapter, PCI bus or the 128-port Asynchronous Controller are the only RS-232 adapters that are supported. These adapters require AIX 4.2.1 or AIX 4.3 on the Control Workstation. Use of the support processor option is highly recommended.

(8) The 7043 can only be used on RS/6000 SP systems with up to four frames or expansion frames. This limitation applies to the number of frames, not to the number of nodes.

(9) If the High Availability Control Workstation feature is never going to be configured, this RS/6000 may be used as a Control Workstation.

3.4 High-Performance Communication Network

The key element of the communication subsystem is the high-performance communication network, which in the SP system is called an SP Switch. The basic responsibility of the SP Switch is to exchange many messages simultaneously between processor nodes to support the demands of the user's parallel applications.

The hardware component that supports this communication network consists of two basic components: the SP Switch board and the SP Switch adapter. There is one SP Switch adapter per node and generally one SP Switch board per frame. This setup provides connections to other nodes. Also, the SP system allows SP Switch boards-only frame that provides switch-to-switch connections and greatly increases scalability.

If you are interested in how to connect SP Switch adapters and SP Switch boards, refer to "Configuration Rules" on page 48 for more details.

3.4.1 Hardware Concepts

This section discusses hardware concepts of the switch used in the SP system.

3.4.1.1 SP Switch Link

An SP Switch link connects two network devices. It contains two channels carrying packets in opposite directions. Each channel includes 8 bits Data, 1 bit Data Valid, and 1 bit Token signal.

3.4.1.2 SP Switch Ports

An SP Switch port is part of a network device and is connected to other SP Switch ports through the SP Switch link. The SP Switch port includes two ports (input and output ports) for full duplex communication.

Figure 11 on page 39 shows the relationship between the SP Switch link and the SP Switch port.

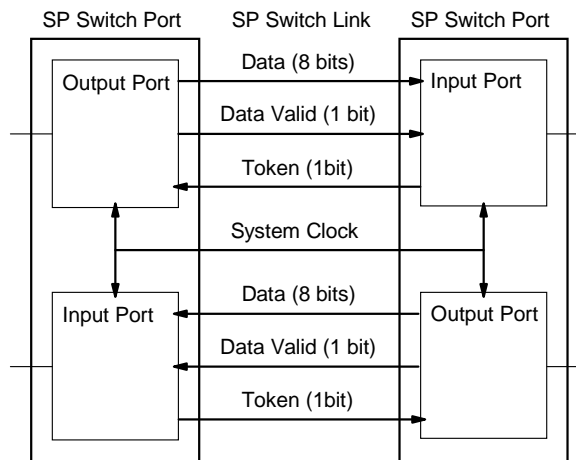


Figure 11. SP Switch Link and SP Switch Port

3.4.1.3 SP Switch Chip

An SP Switch chip contains eight SP Switch ports and a crossbar that allows packets to pass directly from port to port. These crossbar paths allow packets to pass through the SP Switch with low latency. As soon as an input port decodes the routing information carried by an incoming packet, it sends a crossbar request to the appropriate output port.

Figure 12 on page 40 shows the SP Switch ports, the crossbar, and the SP Switch chip.

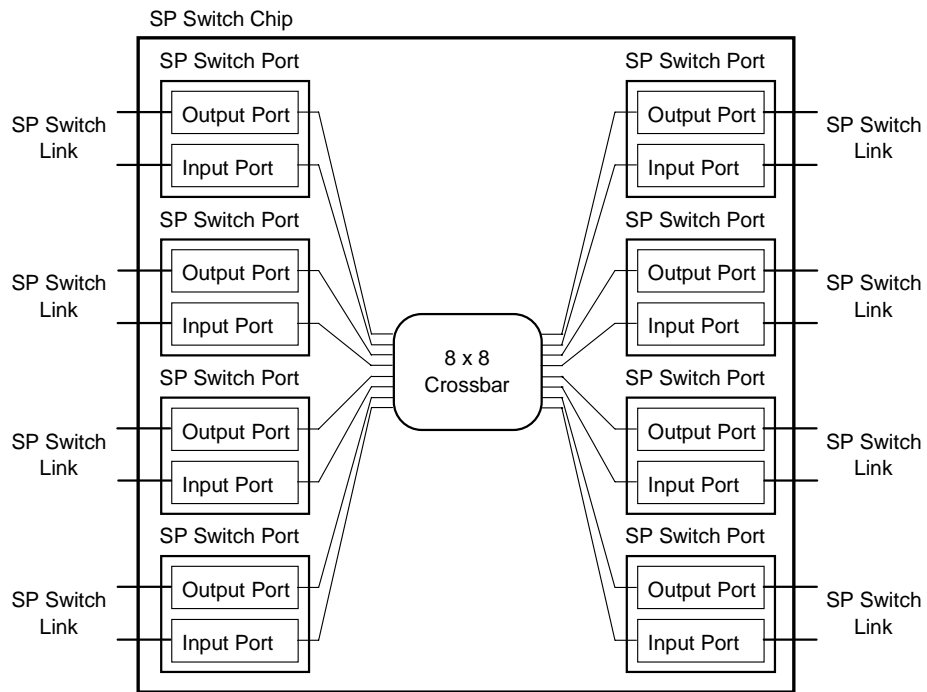


Figure 12. SP Switch Chip

3.4.1.4 SP Switch Board

An SP Switch board contains eight SP Switch chips that provide connection points for each of the nodes to the SP Switch network, as well as for each of the SP Switch boards to the other SP Switch boards. There are 32 SP Switch ports in total. Of these, 16 could be connected to nodes, and the other 16 to other SP Switch boards. The SP Switch board is mounted in the SP Frame.

Figure 13 on page 41 shows the SP Switch board.

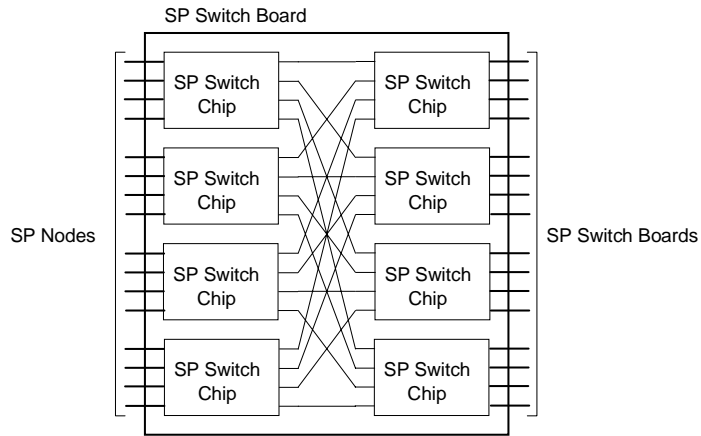


Figure 13. SP Switch Board

3.4.1.5 SP Switch Adapter

Another network device that uses an SP Switch port is the SP Switch adapter. An SP Switch adapter includes one SP Switch port that is connected to an SP Switch board. The SP Switch adapter is installed on an SP node.

There are two kinds of SP Switch adapters. One is designed for the nodes that have a Micro Channel Architecture (MCA) bus. Another SP Switch adapter is designed for nodes that have both a PCI bus and an MX bus.

Figure 14 on page 41 shows an SP Switch adapter.

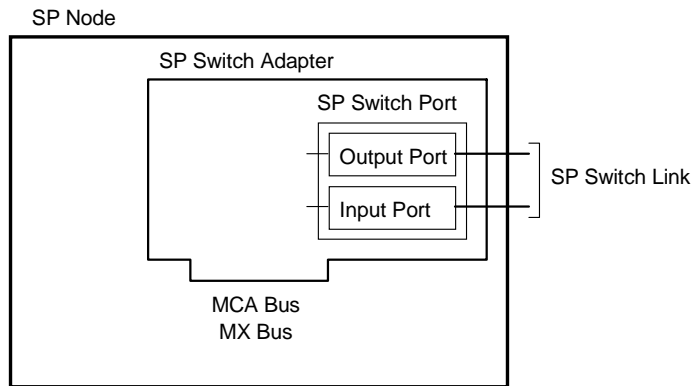


Figure 14. SP Switch Adapter

3.4.1.6 SP Switch System

Figure 15 on page 42 shows the SP Switch system. In one SP frame, there are 16 nodes equipped with SP Switch adapters, and one SP Switch board. Sixteen SP Switch adapters are connected to 16 of 32 SP Switch ports in the SP Switch board. Another 16 SP Switch ports are available for other SP Switch boards.

Figure 15 on page 42 also shows a sample of an SP configuration, which is formed by one tall model frame equipped with the SP Switch and 16 thin nodes.

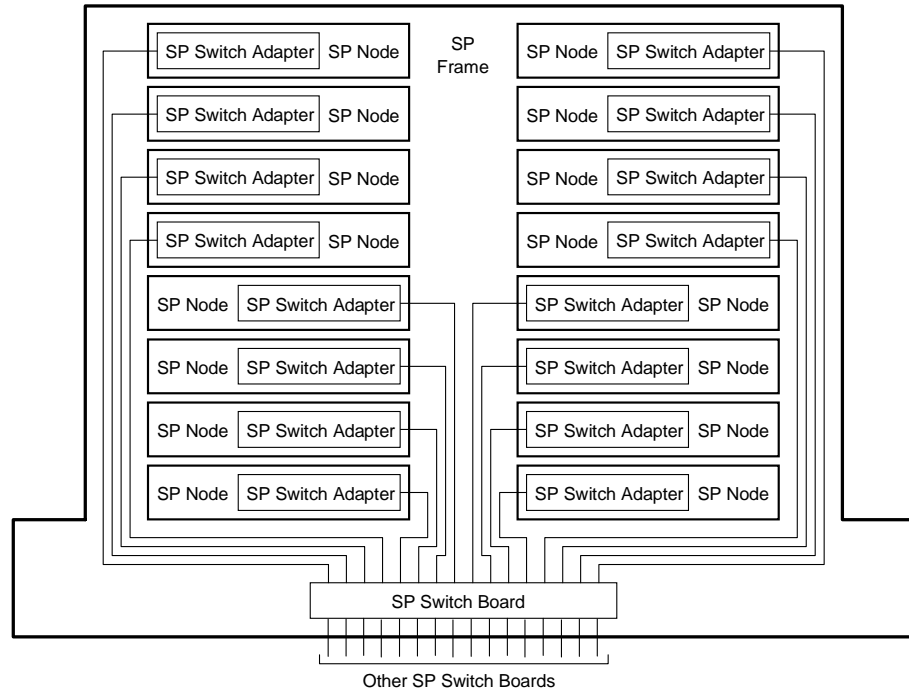


Figure 15. SP Switch System

3.4.2 SP Switch Network

The goal of the SP Switch network is low-latency, high-bandwidth, and fault-tolerant communication.

SP nodes are grouped into 16 nodes that are connected to one side of the SP Switch boards. Figure 16 on page 43 shows a 16 nodes SP system containing one SP Switch board. This SP Switch board is called a node

switch board (NSB). Figure 16 on page 43 also illustrates possible shortest-path routes for packets sent from node A to two destinations. Node A can communicate with node B by traversing a single SP Switch chip, and with node C by traversing three SP Switch chips.

The 16 unused SP Switch ports on the right side of the node switch board are used for creating larger networks. There are two ways to create larger networks:

1. For an SP system containing up to 80 nodes, these SP Switch ports connect directly to the SP Switch ports on the right side of other node switch boards.
2. For an SP system containing more than 80 nodes, these SP Switch ports connect to additional stages of switch boards. These SP Switch boards are called as intermediate switch boards (ISB).

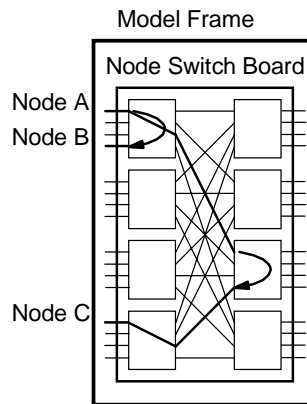


Figure 16. 16-Node SP System

The strategy for creating an SP system of up to 80 nodes is shown in Figure 17 on page 44. The direct connection (16 links) between two NSBs forms a 32-node SP system. Example routes from node A to node B, C, and D are shown. Just as for a 16-node SP system, packets traverse one or three SP Switch chips when the source and destination pair is attached to the same node switch boards. When the pair is attached to different node switch boards, the shortest path routes contain four SP Switch chips. For any pair of nodes connected to separate SP Switch boards in a 32-node SP system, there are four potential paths, providing a high level of redundancy.

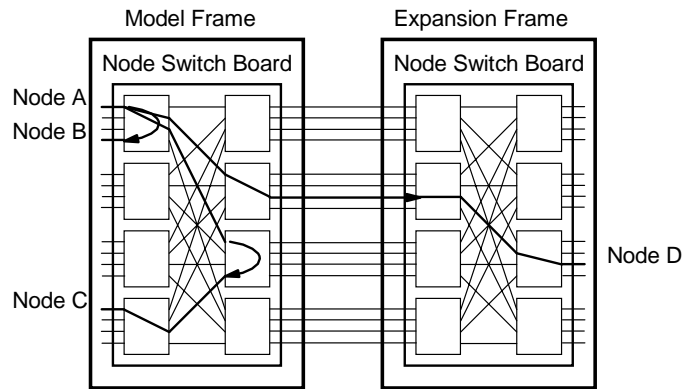


Figure 17. 32-Node SP System

The strategy for creating an SP system of more than 80 nodes is shown in Figure 18 on page 45. Additional SP Switch board (ISBs) are cascaded to the node switch boards, effectively adding more stages to the network to scale the aggregate bandwidth. This figure also displays routes from node A to node B and C. Destinations within the same 16-way group are reached in one or three hops as for smaller networks. Destinations not in this group, but on the Expansion Frame (A) side of the network, are reached in five SP Switch chips. It takes six SP Switch chips to reach nodes on the opposite side, the Expansion Frame (B).

This figure shows only a concept of an SP Switch network using intermediate switch boards (ISBs). Actual connections may be different.

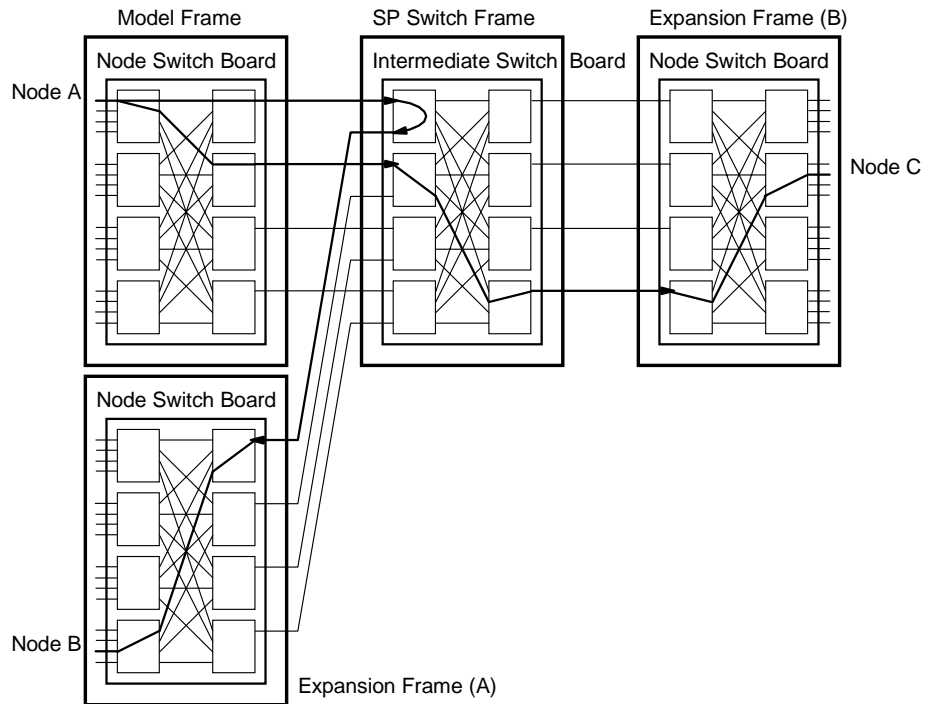


Figure 18. SP System with Intermediate Switch Board

3.4.3 SP Switch Products

There are two kinds of switches as product. So far, this section has explained the SP switch (F/C 4011). The other switch is the High Performance Switch (F/C 4010). Both provide the message passing network that connects all nodes, with a minimum of four paths between any pair of nodes.

3.4.3.1 SP Switch

The SP Switch provides low-latency, high-bandwidth communication between nodes. The SP Switch offers the following improvements over the High Performance Switch:

- Higher availability
- Fault isolation
- Concurrent maintenance of nodes
- Improved switch chip bandwidth

The required SP Switch Adapter or SP Switch MX Adapter connects each SP node to the SP Switch board.

3.4.3.2 High Performance Switch

The High Performance Switch (HiPS) is not compatible with the SP Switch (SPS). You cannot mix High Performance Switches with SP Switches in a system or in separate system partitions. To take advantage of the SP Switch's performance, you must upgrade all High Performance Switches to SP Switches.

The High Performance Switch is being phased out and is not available for new systems; however, they will still be available for existing systems that are already equipped with High Performance Switches.

3.4.3.3 SP Switch-8 and High Performance LC-8 Switch

Eight port switches are a low cost alternative to the full size sixteen port switches. Both the 8-port SP Switch-8 (SPS-8) (F/C 4008) and the 8-port High Performance Switch (HiPS LC-8) (F/C 4007) provide switch functions for an 8-node SP system. The SP Switch-8 is compatible with high nodes while the HiPS-LC8 is not.

The SP Switch-8 is the only low cost switch available for new systems; however, the High Performance LC-8 is still available for existing systems.

The SP Switch-8 has two active switch chip entry points. Therefore, your ability to create system partitions is restricted with this switch. With the maximum eight nodes attached to the switch, you have two possible system configurations:

1. A single partition containing all 8 nodes
2. Two system partitions containing four nodes each

The High Performance LC-8 only has one switch chip. Therefore, only a single partition is possible.

3.4.3.4 Switch Adapter

If you plan to use a switch in your SP system you will need a switch adapter to connect each RS/6000 SP node to the switch subsystem. SP Switches use either the SP Switch Adapter (F/C 4020) or the SP Switch MX Adapter (F/C 4022). The SP Switch Adapter is used in MCA type nodes while the SP Switch MX Adapter is used in PCI type nodes. One switch adapter is needed for each node in the SP system. The HiPS Adapter-2 (F/C 4018) is used in systems where the HiPS or HiPS-LC8 is installed. This adapter is only available for existing HiPS-equipped SP systems.

3.5 Peripheral Devices

The attachment of peripheral devices, such as disk subsystems, tape drives, CD-ROMs, and printers, is very straightforward on the SP. There are no SP-specific peripherals; since the SP uses mainstream RS/6000 node technology, it simply inherits the array of peripheral devices available to the RS/6000 family. The SP's shared-nothing architecture gives rise to two key concepts when attaching peripherals:

1. Each node has I/O slots. Think of each node as a stand-alone machine when attaching peripherals: it can attach virtually any peripheral available to the RS/6000 family, such as SCSI and SSA disk subsystems, MagStar tape drives, and so on. Peripheral device attachment is very flexible, as each node can have its own mix of peripherals, or none at all.
2. From an overall system viewpoint, as nodes are added, I/O slots are added, thus the scalability of I/O device attachment is tremendous. A 512-node high node system would have several thousand I/O slots!

Each node must have internal disks to hold its copy of the operating system. Since multiple internal disks can be installed in a node, and software loading is provided by the CWS or boot/install server across the SP administrative Ethernet, it is common to have nodes without any peripheral devices.

When you attach a disk subsystem to one node, is it automatically visible to all the other nodes? The answer is no, but the SP provides a number of techniques and products to allow access to a disk subsystem from other nodes. This is discussed in 4.7, "Parallel I/O" on page 172.

There are some general considerations for peripheral device attachment:

- Since the nodes are housed in frames, cable lengths from the I/O adapter in a node slot to the actual peripheral must be long enough.
- Devices such as CD-ROMs and bootable tape drives may be attached directly to SP nodes, but IBM does not support their use as installation devices. Nodes must be network-installed by the CWS or a boot/install server.
- Many node types do not have serial ports. High nodes have two free serial ports for general use.
- Graphics adapters for attachment of displays are not supported.

3.6 Configuration Rules

The RS/6000 SP system has extremely wide scalability. For standard configuration, the RS/6000 SP system mounts up to 128 processor nodes. This section provides you with information on how you can expand your SP system and what kind of configuration fits your requirement. This section also provides you with a variety of sample configurations. Also many configuration rules are given. You may use these configuration rules as a check list when you configure your SP system.

3.6.1 Hardware Components

Before starting with configuration rules, let us introduce some hardware components that you can use when you configure your SP system.

3.6.1.1 Nodes

You can use the following node types to configure your system:

- PCI Bus SMP Thin Node (F/C 2050)
- PCI Bus SMP Wide Node (F/C 2051)
- MCA Bus SMP High Node (F/C 2009, F/C 2006)
- MCA Bus Wide Node (F/C 2007)
- MCA Bus Thin Node (F/C 2022, F/C 2008)

Node Considerations

All PCI bus nodes can be installed in any existing SP frame provided that the required power supply upgrades have been implemented in that frame.

All PCI bus nodes are not compatible with the High Performance Switch (HiPS), including HiPS LC-8. If they are going to be placed into your SP system configured with a switch, that switch must be either an SP Switch or an SP Switch-8.

3.6.1.2 Frames

You can use the following frame types to configure your system:

- Short model frame (F/C 1500 or F/C 1500 + Switch)
- Short expansion frame (F/C 1500)
- Tall model frame (F/C 1550 or F/C 1550 + Switch)
- Nonswitched tall expansion frame (F/C 1550)
- Switched tall expansion frame (F/C 1550 + Switch)

- SP Switch frame (F/C 2031 or F/C 2030)

Frame Considerations

There are two generations of tall frames. All new frame designs are completely compatible with all valid SP configurations using older equipment. From the viewpoint of configuration rules, they can be treated completely the same. So you do not need to pay to attention whether you use old tall frames or new tall frames.

3.6.1.3 Switches

You can use the following type of SP Switch adapters boards when configure your system:

- SP Switch (F/C 4011)
- SP Switch-8 (F/C 4008)
- High Performance Switch (HiPS) (F/C 4010)
- High Performance Switch LC-8 (HiPS-LC8) (F/C 4007)

Switch Considerations

There are two generations of switches. The High Performance Switch (HiPS) and HiPS LC-8 are old switches. SP Switch and SP Switch-8 are the newest and currently supported switches. HiPS is available only for expanding existing SP systems using HiPS. For this reason, this section explains only the case of SP Switch and SP Switch-8. If you keep using HiPS or HiPS LC-8, you can read SP Switch as HiPS and SP Switch-8 as HiPS LC-8, but you have to pay attention to the following rule:

Configuration Rule 1

HiPS and HiPS LC-8 are not compatible with PCI bus nodes, while SP Switch and SP Switch-8 are.

If you keep using HiPS LC-8, you have to pay attention to one more rule:

Configuration Rule 2

HiPS LC-8 is not compatible with high nodes, while other switches are.

3.6.2 Basic Configuration Rules

There are three simple but important configuration rules:

Configuration Rule 3

You cannot install both tall frames and short frames for a single SP

Configuration Rule 4

You have to choose one of SP Switch, SP Switch-8, HiPS, or HiPS LC-8 for a single SP system.

Configuration Rule 5

Thin nodes must be installed in pairs, in the two slots of a single drawer, and the processor type must be the same for both nodes.

Because of Configuration Rule 3, the rest of this section is separated into two major parts. The first part provides the configuration rules for using short frames, and the second part provides the rules for using tall frames.

3.6.3 Short Frame Configurations

When you install short frames on your SP system, two configurations are available:

1. An SP system with no switch

This configuration is for an SP system that does not require high speed communication between nodes, and also does not require more than eight nodes.

2. An SP system with SP Switch-8

This configuration is for an SP system that requires high speed communication between nodes, but does not require more than eight nodes.

If your SP system requires more than eight processor nodes, you have to install tall frames. Please refer to “Tall Frame Configurations” on page 53.

Let’s look through these two configurations one by one.

3.6.3.1 Nonswitched Configuration

This configuration does not have a switch and mounts 1-8 nodes. Even though this configuration does not have a switch, configuration rules are completely the same as the configuration rules of a switched configuration. Refer to “Switched Configuration” on page 51.

3.6.3.2 Switched Configuration

This configuration has a switch and mounts 1-8 nodes.

There are two configuration rules when you install a short frame with a switch.

Configuration Rule 6

You have to use SP Switch-8 instead of SP Switch if you install a short frame.

Configuration Rule 7

You can install only a single SP Switch-8 in your SP system.

This is the reason why you cannot mount more than eight processor nodes on your SP system for this configuration.

A minimum switched configuration is formed by the following components shown in Figure 19 on page 51:

- One short model frame equipped with an SP Switch-8
- With your choice of two thin nodes, one wide node, or one high node

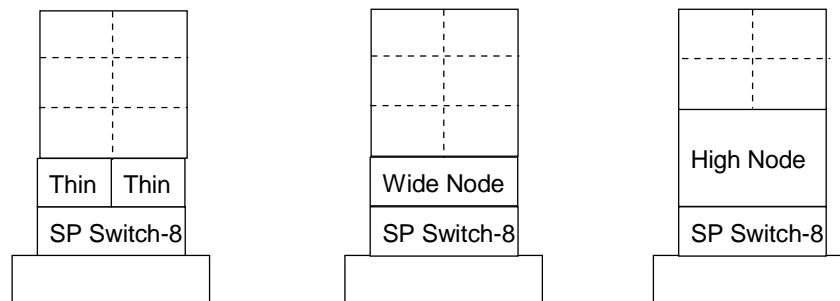


Figure 19. Minimum Switched Configuration

It does not make sense to use SP Switch-8 for an SP system that has only a single node, but it is possible, and you can reduce the installation time for adding nodes.

You can expand a minimum configuration to up to eight nodes. Depending on the number of nodes in your SP system, up to three short expansion frames may be added. Nodes in the short expansion frames share unused switch ports in the short model frame. Please refer to “The Switch Port Numbering Rule” on page 75 for other rules.

Maximum configuration examples are shown in Figure 20 on page 52 through Figure 23 on page 53. Figure 20 on page 52 shows the case of using only thin nodes to expand your SP system.

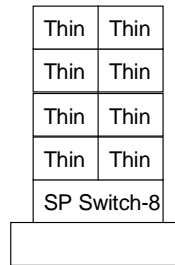


Figure 20. Maximum Switched Configuration Using Thin Nodes

Figure 21 on page 52 shows the case of using only wide nodes to expand your SP system.

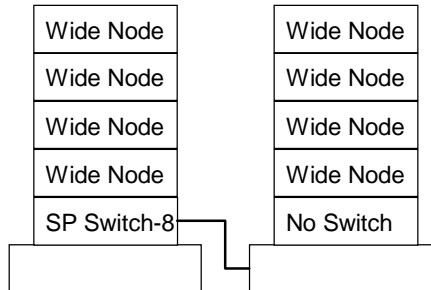


Figure 21. Maximum Switched Configuration Using Wide Nodes

Figure 22 on page 53 shows the case of using only high nodes to expand your SP system.

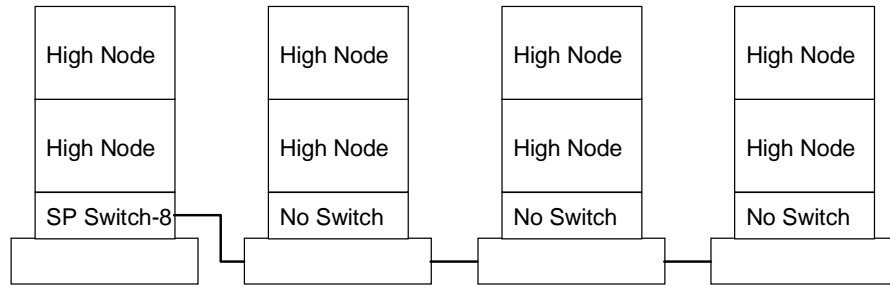


Figure 22. Maximum Switched Configuration Using High Nodes

Figure 23 on page 53 shows the case of using mixed nodes to expand your SP system.

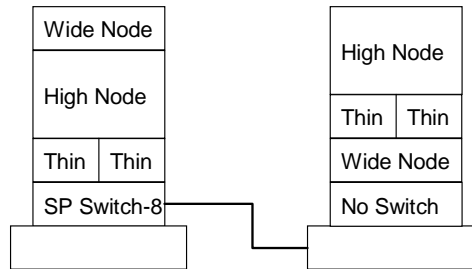


Figure 23. Maximum Switched Configuration Using Mixed Nodes

3.6.4 Tall Frame Configurations

When you install tall frames on your SP system, four configurations are available:

1. An SP system with no switch

This configuration is for an SP system that does not require high speed communication between nodes, and also does not require more than 80 nodes.
2. An SP system with SP Switch-8

This configuration is for an SP system that requires high speed communication between nodes, but does not require more than eight nodes.
3. An SP system with SP Switch

This configuration is for an SP system that requires high speed communication between nodes, but does not require more than 80 nodes.

4. An SP system with SP Switch and SP Switch frame

This configuration is for an SP system that requires high speed communication between nodes, but not require more than 128 nodes.

If your SP system requires more than 128 nodes, SP systems with more than 128 nodes are available. Consult your IBM representative for more information.

Let us look at these four configurations.

3.6.4.1 Nonswitched Configuration

This configuration does not have a switch and mounts 1-80 nodes. Even though this configuration does not have a switch, configuration rules are completely the same as the configuration rules for a single-stage SP Switch configuration. Refer to "Single Stage SP Switch Configuration" on page 57.

The only difference is that the nonswitched configuration does not have an SP Switch in the model frame and expansion frame, while the single-stage SP Switch configuration does.

3.6.4.2 SP Switch-8 Configuration

This configuration has a switch and mounts 1-8 nodes.

A minimum SP Switch-8 configuration is formed by the following components (shown in Figure 24 on page 55):

- One tall model frame equipped with an SP Switch-8
- With your choice of two thin nodes, one wide node, or one high node

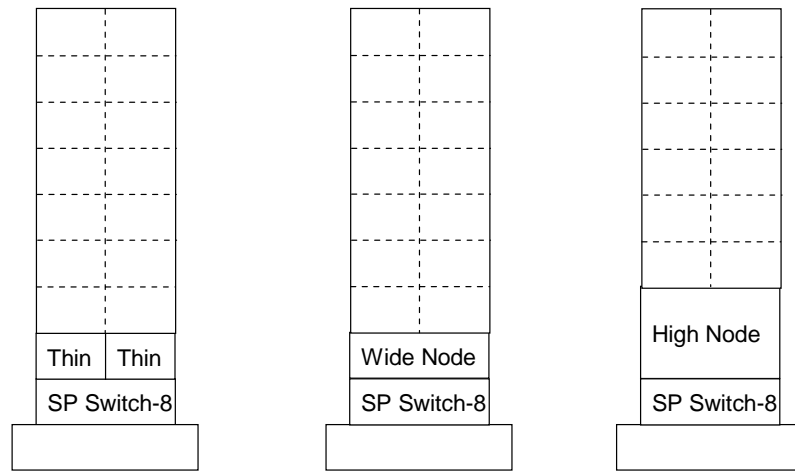


Figure 24. Minimum SP Switch-8 Configuration

It does not make sense to use the SP Switch-8 for an SP system that has only a single node, but it is possible, and you can reduce the installation time for adding nodes.

You can expand these minimum configurations to up to eight nodes. Depending on the number of nodes in your SP system, one nonswitched tall expansion frame may be added. Nodes in the nonswitched tall expansion frame share unused switch ports in the tall model frame. When you add nonswitched tall expansion frames to your SP system, pay attention to the following rule:

Configuration Rule 8

A nonswitched expansion frame is supported in this configuration only if the model frame is filled before the total node count of eight is reached, if you use a tall model frame equipped with SP Switch-8.

So, if you have available slots in the tall model frame, you cannot use nonswitched tall expansion frames.

Maximum configuration examples are shown in Figure 25 on page 56 through Figure 28 on page 57.

Figure 25 on page 56 shows the case of using only thin nodes to expand your SP system.

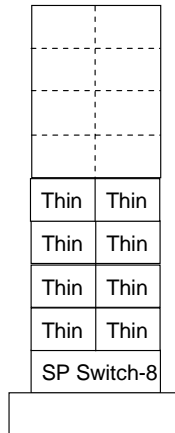


Figure 25. Maximum SP Switch-8 Configuration Using Thin Nodes

Figure 26 on page 56 shows the case of using only wide nodes to expand your SP system.

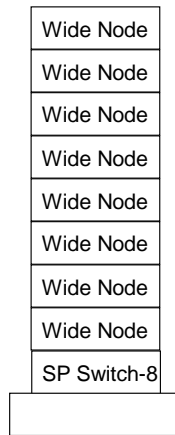


Figure 26. Maximum SP Switch-8 Configuration Using Wide Nodes

Figure 27 on page 57 shows the case of using only high nodes to expand your SP system

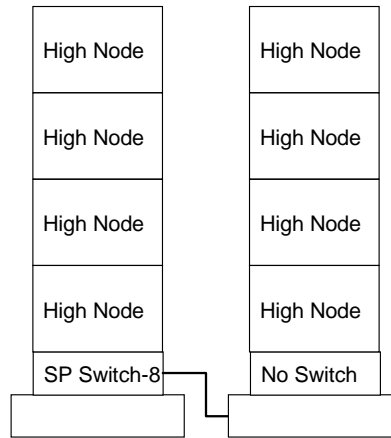


Figure 27. Maximum SP Switch-8 Configuration Using High Nodes

Figure 28 on page 57 shows a case using mixed nodes to expand your SP system.

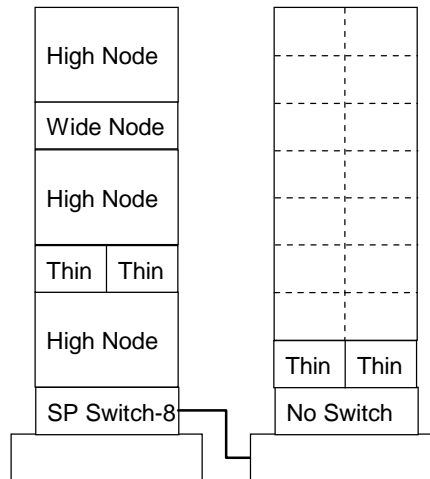


Figure 28. Maximum SP Switch-8 Configuration Using Mixed Nodes

3.6.4.3 Single Stage SP Switch Configuration

This configuration has a switch and mounts 1-80 nodes.

A minimum single-stage SP Switch configuration is formed by the following components (shown in Figure 29 on page 58):

- One tall model frame equipped with an SP Switch
- With your choice of two thin nodes, one wide node, or one high node

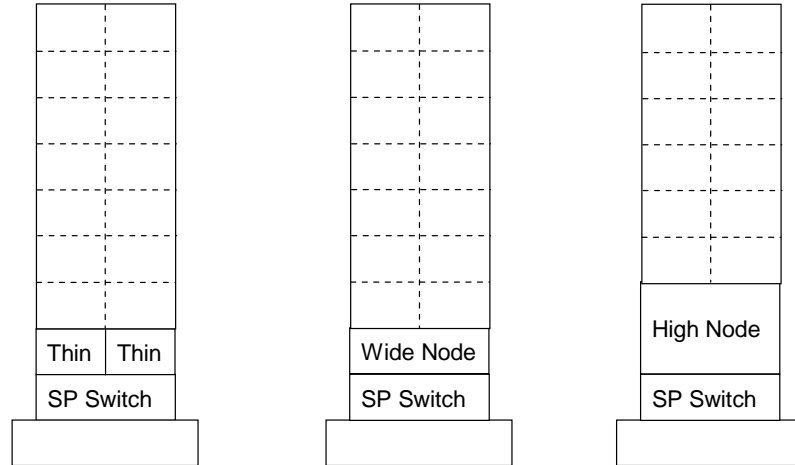


Figure 29. Minimum Single Stage SP Switch Configuration

Depending on the number of nodes in your SP system, switched expansion frames and/or nonswitched expansion frames may be added. There are three methods of expanding your SP system:

1. Adding switched expansion frames
2. Adding nonswitched expansion frames
3. Adding both switched expansion frames and nonswitched expansion frames

Let us look at these methods.

Adding switched expansion frames

The first method for expanding your SP system is to add switched expansion frames. Depending on the number of nodes, up to four switched expansion frames may be added.

A minimum single-stage SP Switch configuration with switched expansion frames (SEF) is formed by the following components (shown in Figure 30 on page 59):

- One tall model frame equipped with an SP Switch
- One tall switched expansion frame

- With your choice of two thin nodes, one wide node, or one high node for both the model frame and switched expansion frame

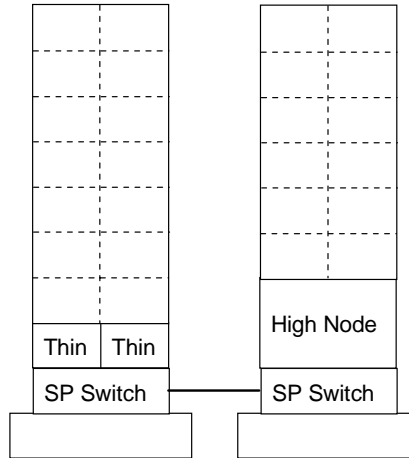


Figure 30. Minimum Single Stage SP Switch Configuration with SEF

There are actually six combinations for a minimum configuration. Figure 30 on page 59 only shows the case of thin and high nodes combination.

Maximum configurations are shown in Figure 31 on page 59 through Figure 33 on page 60. Figure 31 on page 59 shows the case of using only thin nodes to expand your SP system. In this case, the SP system mounts 80 thin nodes.

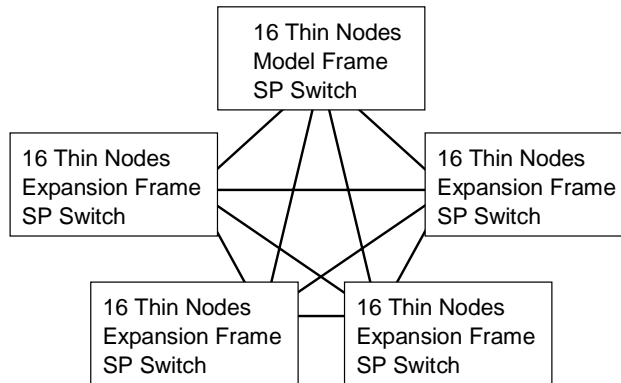


Figure 31. Maximum Single Stage SP Switch Configuration with SEFs (1)

Figure 32 on page 60 shows the case of using only wide nodes to expand your SP system. In this case, the SP system mounts 40 wide nodes.

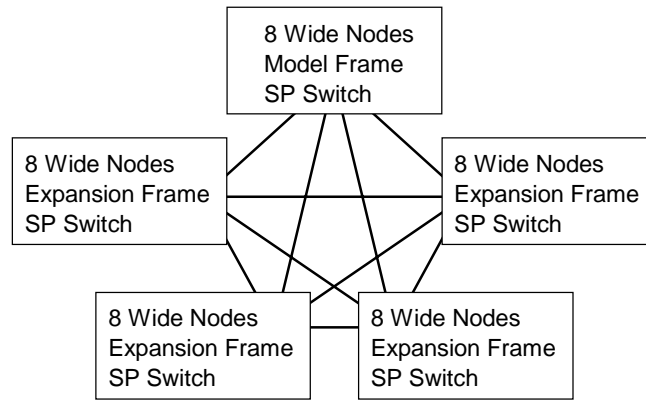


Figure 32. Maximum Single Stage SP Switch Configuration with SEFs (2)

Figure 33 on page 60 shows the case of using only high nodes to expand your SP system. In this case, the SP system mounts 20 high nodes. You can place any nodes in any slots. There is no limitation.

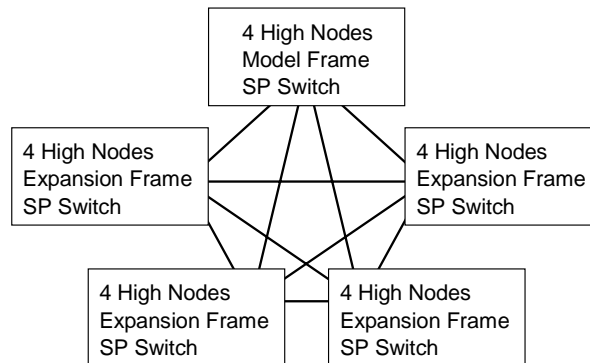


Figure 33. Maximum Single Stage SP Switch Configuration with SEFs (3)

Adding nonswitched expansion frames

The second method of expanding your SP system is to add nonswitched expansion frames. Depending on the number of nodes in your SP system, up to three nonswitched expansion frames may be added to a model frame. Nodes in the nonswitched expansion frames share unused switch ports that may exist in the model frame. Therefore the maximum number of nodes is limited to 16.

A minimum single-stage SP Switch configuration with nonswitched expansion frames (NEF) is formed by the following components (shown in Figure 34 on page 61):

- One tall model frame with an SP Switch
- One tall nonswitched expansion frame
- With your choice of one wide node, or one high node for each frame

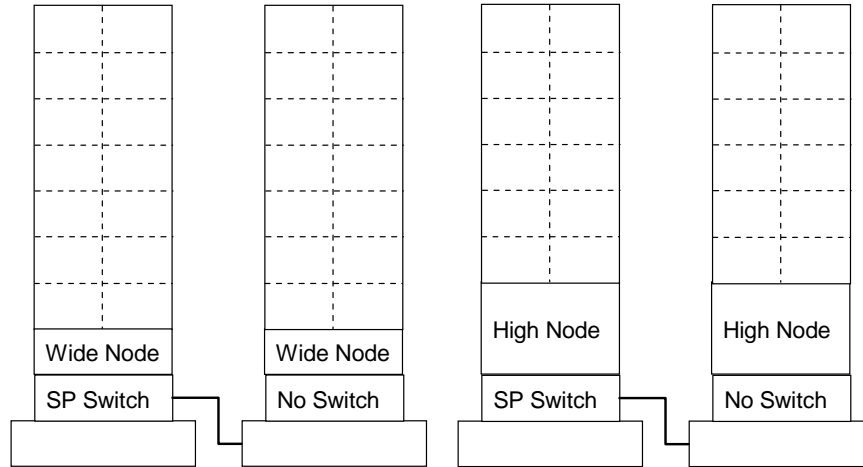


Figure 34. Minimum Single Stage SP Switch Configuration with NEF

You may notice that there is no minimum configuration with thin nodes. The reason comes from the following configuration rule:

Configuration Rule 9

If an expansion frame has thin nodes, it cannot be a nonswitched expansion frame. Similarly, if a model frame has thin nodes, it cannot have a nonswitched expansion frame.

Maximum configuration examples are shown in Figure 35 on page 62 and Figure 36 on page 62.

Figure 35 on page 62 shows the case of using only wide nodes to expand your SP system. In this case you need one nonswitched expansion frame.

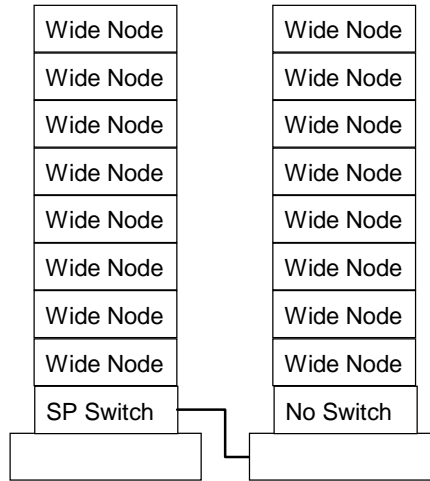


Figure 35. Maximum Single Stage SP Switch Configuration with NEF (1)

Figure 36 on page 62 shows the case of using only high nodes to expand your SP system. In this case you need three nonswitched expansion frames. In both case, the SP system mounts 16 processor nodes.

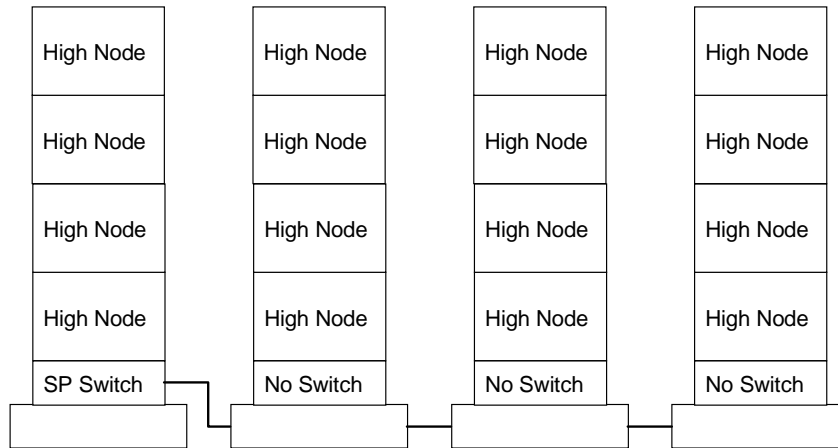


Figure 36. Maximum Single Stage SP Switch Configuration with NEF (2)

Adding both switched and nonswitched expansion frames

The third method of expanding your SP system is to add both switched expansion frames and nonswitched expansion frames. This method combines the previous two methods, adding switched expansion frames and nonswitched expansion frames. The connection between frames with an SP

Switch uses the method of adding switched expansion frames. The connection between frames with an SP Switch and frames without an SP Switch uses the method of adding nonswitched expansion frames.

A minimum single-stage SP switch configuration with both switched expansion frames and nonswitched expansion frames is formed by the following components (shown in Figure 37 on page 63):

- One tall model frame equipped with an SP Switch
- One switched tall expansion frame
- One nonswitched tall expansion frame
- With your choice of two thin nodes, one wide node, or one high node for a frame that is not connected to a nonswitched expansion frame
- With your choice of one wide node, or one high node for both a nonswitched expansion frame and a frame that is connected to a nonswitched expansion frame

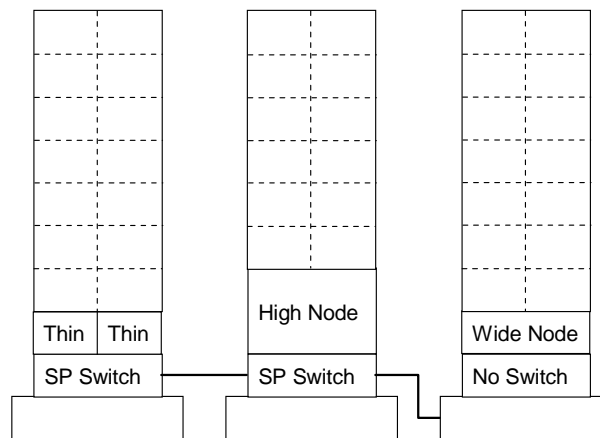


Figure 37. Minimum Single Stage SP Switch Configuration with SEF and NEF

Figure 37 on page 63 is just an example of many possible minimum configurations.

Maximum configuration examples are shown in Figure 38 on page 64 and Figure 39 on page 65.

Figure 38 on page 64 shows the case of using only wide nodes to expand your SP system. In this case, the SP system mounts 80 wide nodes.

In the case you use only thin nodes to expand your SP system, you do not need to use a nonswitched expansion frame, even though you cannot use nonswitched expansion frame by “Configuration Rule 9” on page 61. So a maximum configuration using only thin nodes becomes completely the same as Figure 31 on page 59.

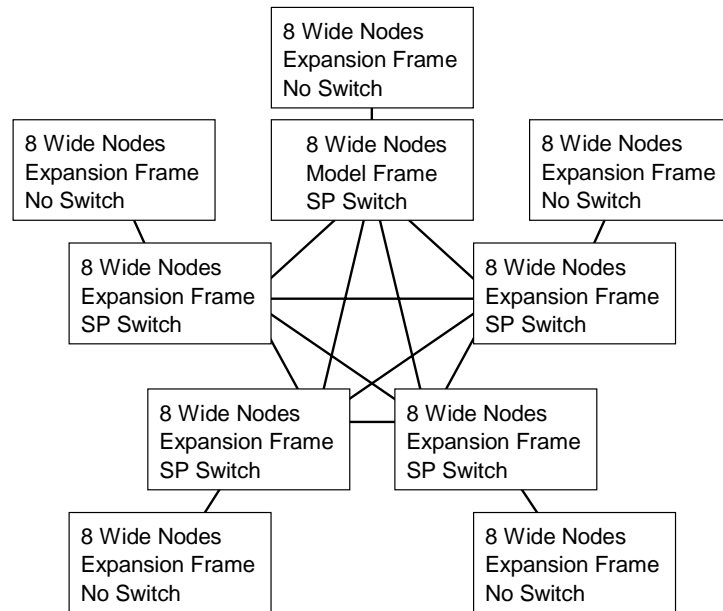


Figure 38. Maximum Single-Stage SP Switch Configuration with SEFs and NEFs (1)

Figure 39 on page 65 shows the case of using high nodes to expand your SP system. In this case, the SP system mounts 64 high nodes.

You may think the maximum number of high nodes could be 80. But actually, the following configuration rule limit the number of high nodes per system:

Configuration Rule 10

You cannot use more than 64 high nodes in your SP system.

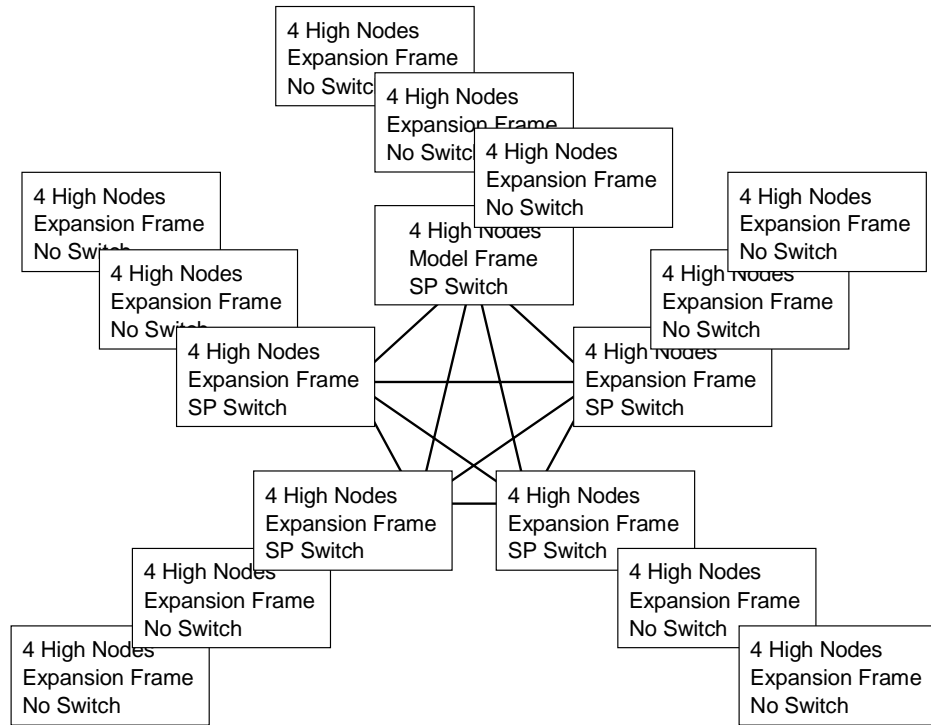


Figure 39. Maximum Single-Stage SP Switch Configuration with SEFs and NEFs (2)

3.6.4.4 Two-Stage SP Switch Configuration

This configuration has a switch and mounts 1-128 processor nodes. This is the maximum standard configuration.

This system configuration requires an SP Switch frame which forms the second switch layer. The second stage switches in the SP Switch frame are used for high performance parallel communication between the SP switches mounted on the model frame and switched tall expansion frames.

A minimum two-stage SP Switch configuration is formed by the following components (shown in Figure 40 on page 66):

- One tall model frame equipped with an SP Switch
- One SP Switch frame
- With your choice of two thin nodes, one wide node, or one high node

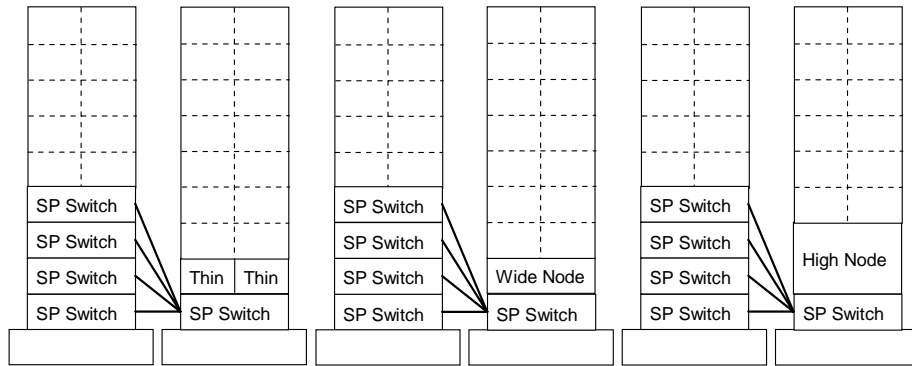


Figure 40. Minimum Two-Stage SP Switch Configuration

In this minimum configuration, there is no need for an SP Switch frame. However, adding an SP Switch frame will provide better system scability. The dedicated switch frame minimizes downtime by reducing the need to recable the switch fabric with each switched expansion frame.

Depending on the number of nodes in your SP system, switched expansion frames and/or nonswitched expansion frames may be added to your SP system. There are three methods for expanding your SP system:

1. Adding switched expansion frames
2. Adding nonswitched expansion frames
3. Adding both switched expansion frames and nonswitched expansion frames

Let us look at these methods.

Adding switched expansion frames

The first method for expanding your SP system is to addi switched expansion frames. Depending on the number of nodes, up to seven switched expansion frames may be added.

A minimum two-stage SP Switch configuration with switched expansion frames is formed by the following components (shown in Figure 41 on page 67):

- One tall model frame equipped with an SP Switch
- One tall switched expansion frame
- One SP Switch frame

- With your choice of two thin nodes, one wide node, or one high node for both model frame and switched expansion frame

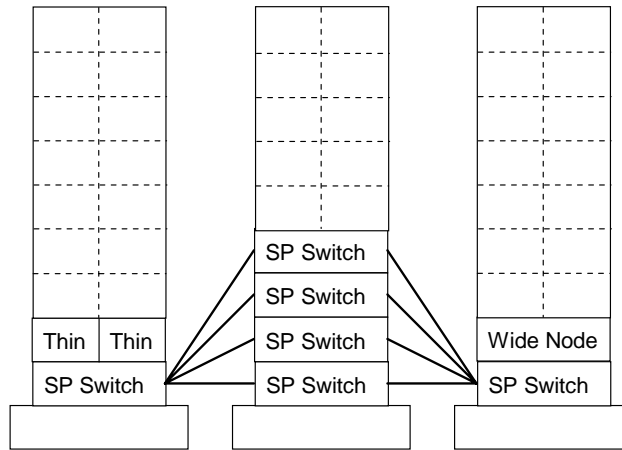


Figure 41. Minimum Two-Stage SP Switch Configuration with SEFs

Maximum configurations are shown in Figure 42 through Figure 44 on page 69.

Figure 42 shows the case of using only thin nodes to expand your SP system. In this case, the SP system mounts 128 thin nodes.

Refer to “The Switch Port Numbering Rule” on page 75 for more rules.

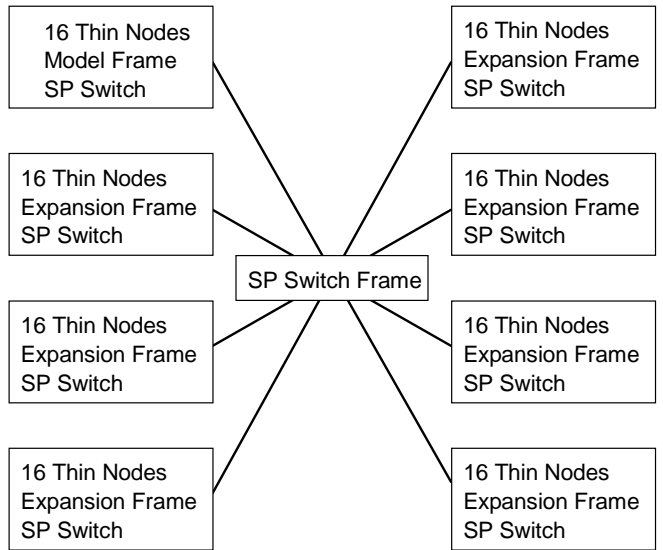


Figure 42. Maximum Two-Stage SP Switch Configuration with SEFs (1)

Figure 43 shows the case of using only wide nodes to expand your SP system. In this case, the SP system mounts 64 wide nodes.

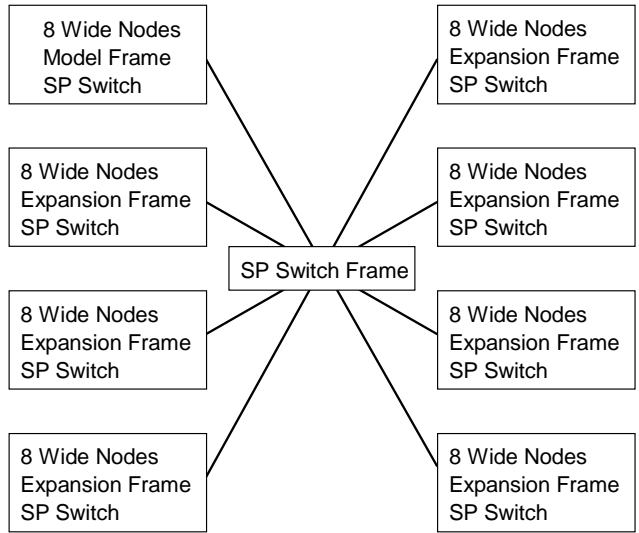


Figure 43. Maximum Two-Stage SP Switch Configuration with SEFs (2)

Figure 44 shows the case of using only high nodes to expand your SP system. In this case, the SP system mounts 32 high nodes.

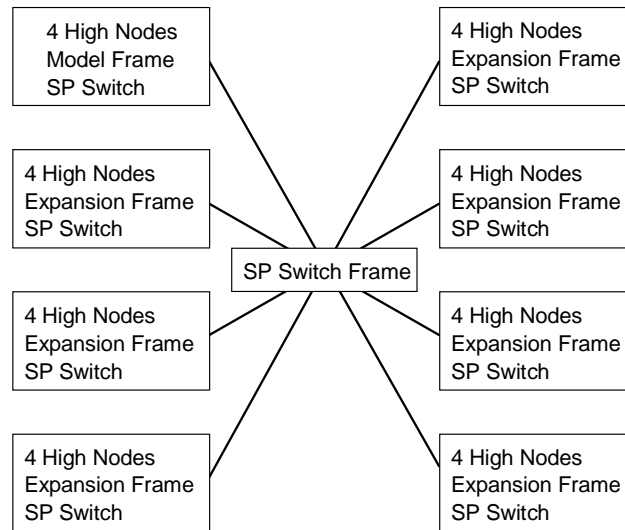


Figure 44. Maximum Two-Stage SP Switch Configuration with SEFs (3)

Adding nonswitched expansion frames

The second method for expanding your SP system is to add nonswitched expansion frames. Depending on the number of nodes, up to three nonswitched expansion frames may be added.

A minimum two-stage SP Switch configuration with nonswitched expansion frames is formed by the following components (shown in Figure 45 on page 70):

- One tall model frame equipped with an SP Switch
- One nonswitched tall expansion frame
- One SP Switch frame
- With your choice of one wide node, or one high node for both model frame and nonswitched tall expansion frame

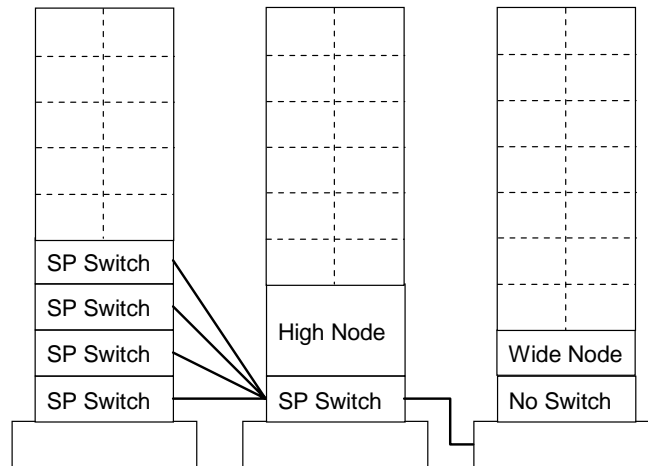


Figure 45. Minimum Two-Stage SP Switch Configuration with NEF

The expansion method and maximum configuration are completely the same as for the case of single-stage SP Switch configuration, except that the two-stage SP Switch configuration has a SP Switch frame while the single-stage SP Switch configuration does not. Refer to “Adding nonswitched expansion frames” on page 60.

Adding both switched and nonswitched expansion frames

The third method for expanding your SP system is to add both switched expansion frames and nonswitched expansion frames. This method combines the previous two methods, adding switched expansion frames and adding nonswitched expansion frames. The connection between frames with an SP Switch uses the method of adding switched expansion frames. The connection between frames with an SP Switch and frames without an SP Switch uses the method of adding nonswitched expansion frames.

A minimum two-stage SP Switch configuration with both switched expansion frames and nonswitched expansion frames is formed by the following components (shown in Figure 46 on page 71):

- One tall model frame equipped with an SP Switch
- On switched tall expansion frame
- One nonswitched tall expansion frame
- One SP Switch frame
- With your choice of two thin nodes, one wide node, or one high node for a frame that is not connected to a nonswitched expansion frame

- With your choice of one wide node, or one high node for both a nonswitched expansion frame and frame that is connected to a nonswitched expansion frame

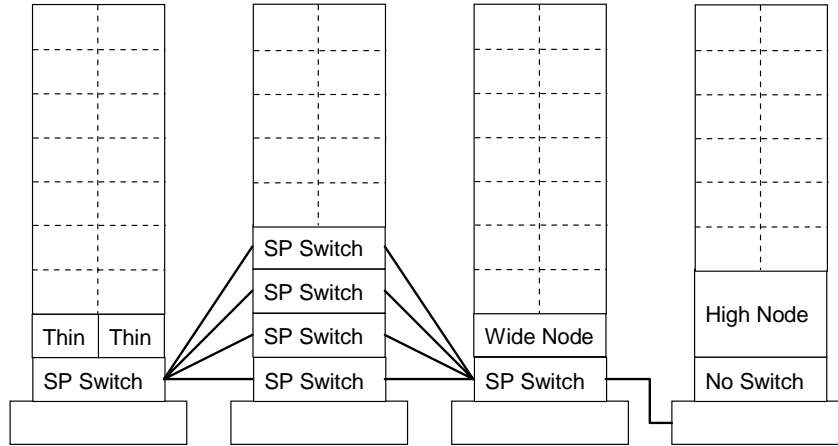


Figure 46. Minimum Two-Stage SP Switch Configuration with SEFs and NEF

Maximum configurations are shown in Figure 47 on page 72 and Figure 48 on page 73.

Figure 47 on page 72 shows the case of using only wide nodes to expand your SP system. In this case, the SP system mounts 128 wide nodes.

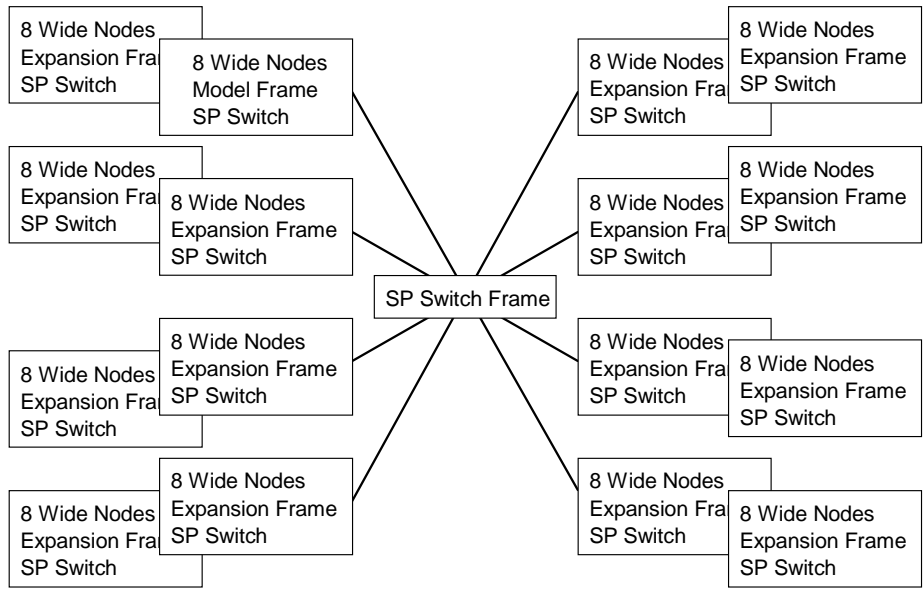


Figure 47. Maximum Two-Stage SP Switch Configuration with SEFs and NEFs (1)

Figure 48 on page 73 shows the case of using only high nodes to expand your SP system. In this case, the SP system mounts 64 high nodes.

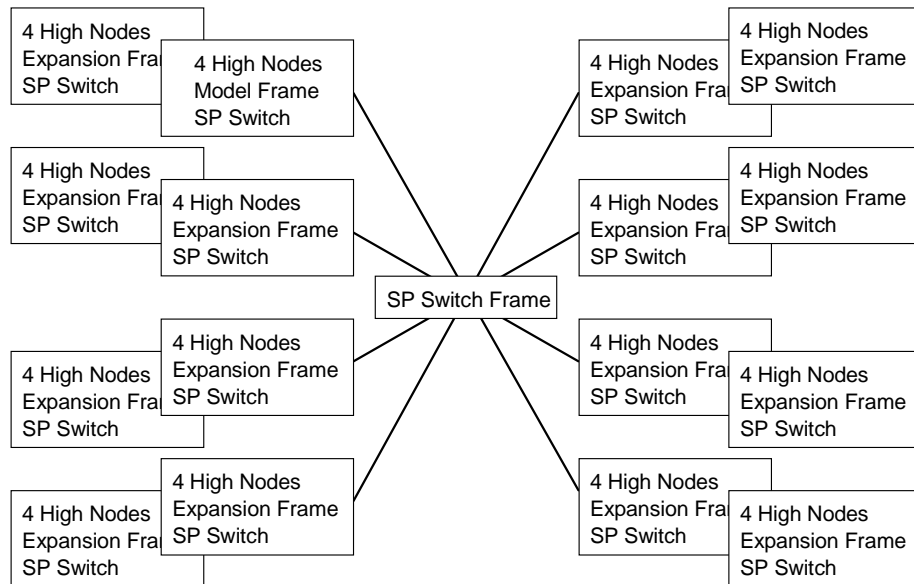


Figure 48. Maximum Two-Stage SP Switch Configuration with SEFs and NEFs (2)

3.6.5 Numbering Rules

We now need to be able to identify individual frames and nodes. To do this, the following numbering rules are used:

- The frame numbering rule
- The slot numbering rule
- The node numbering rule
- The SP Switch port numbering rule

3.6.5.1 The Frame Numbering Rule

The administrator establishes the frame numbers when the system is installed. Each frame is referenced by the TTY port to which the frame supervisor is attached and is assigned a numeric identifier. The order in which the frames are numbered defines the sequence in which they are examined during the configuration process. This order is used to assign global identifiers to the switch ports and nodes. This is also the order used to determine which frames share a switch.

3.6.5.2 The Slot Numbering Rule

A tall frame has a total of 16 slots. A short frame has a total of eight slots. When viewing an SP frame from the front, the slots are numbered sequentially from bottom left to top right. The slot number starts with one, and the maximum number is 16.

The position of a node in an SP frame is sensed by the hardware, and that position is the slot to which it is wired. This slot is the slot number of the node, as follows:

- A thin node's slot number is the corresponding slot.
- A wide node's slot number is the odd-numbered slot.
- A high node's slot number is the first (lowest-number) slot.

3.6.5.3 The Node Numbering Rule

A node number is a global ID assigned to a node. It is the method by which an administrator can reference a specific node in the SP system. Node numbers are assigned by the following formula:

$$\text{node_number} = ((\text{frame_number} - 1) \times 16) + \text{slot_number}$$

Node numbers are assigned independently of whether the frame is fully populated.

Figure 49 on page 74 shows an example of frame, slot, and node numbers.

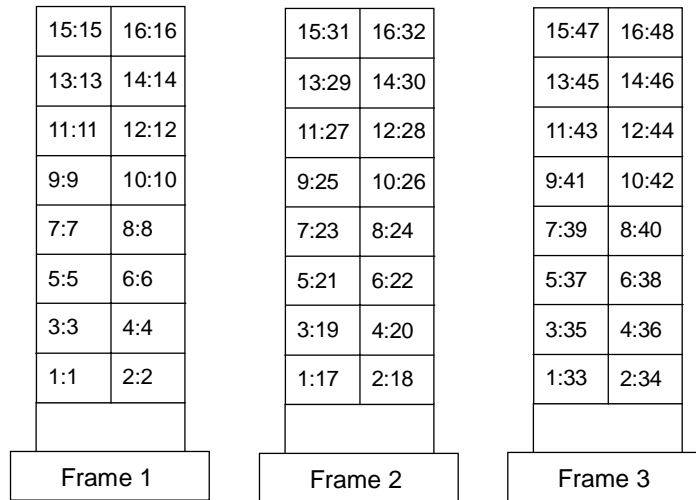


Figure 49. Frame, Slot, and Node Numbers

3.6.5.4 The Switch Port Numbering Rule

This rule is also known as the Node Placement Rule, because you can place nodes only in the slots that have a switch port number.

For the SP Switch-8, and the High Performance Switch LC-8, different algorithms are used for assigning nodes their switch port numbers.

SP Switch and High Performance Switch

For the 16-port SP and High Performance Switches, you can use the following formulas to determine the switch port number to which a node is attached:

1. If the node is connected to a switch within its frame:

$$\text{switch_port_number} = (\text{switch_number} - 1) \times 16 + (\text{slot_number} - 1)$$

2. If the node is connected to a switch outside of its frame:

$$\text{switch_port_number} = (\text{switch_number} - 1) \times 16 + \text{port_number}$$

Here, `switch_number` is the number of the switch board to which the node is connected, and `port_number` is the port position on the switch board to which the node is connected.

Let us take a look at more details, using examples. When you add nonswitched tall expansion frames to tall model frames, the SP system currently supports three frame configurations. Figure 51 on page 76 through Figure 53 on page 78 illustrates the supported configurations for switch port number.

In Figure 50 on page 76, the model frame has an SP Switch that uses all 16 of its switch ports. Since all switch ports are used, the frame does not support nonswitched expansion frames. When you mount at least one thin node pair on a model frame, this configuration is applied.

Also remember “Configuration Rule 9” on page 61.

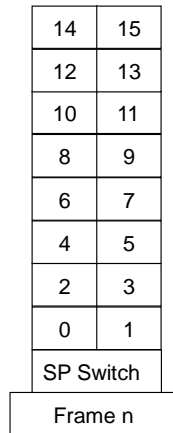


Figure 50. Base Configuration With no Nonswitched Expansion Frame

If the model frame has only wide nodes, it could use at most eight switch ports, and so has eight to share with nonswitched expansion frames. These nonswitched expansion frames are allowed to be configured as in configuration 1 in Figure 51 on page 76 or in configuration 2 in Figure 52 on page 77. Configuration 1 has a single nonswitched expansion frame using wide nodes, while configuration 2 has two nonswitched expansion frames.

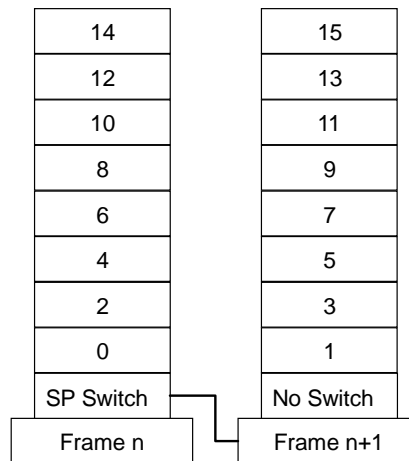


Figure 51. Configuration With One Nonswitched Expansion Frame

In configuration 2, only high nodes are used in the two nonswitched expansion frames. However, wide nodes can be substituted for high nodes if

and only if the wide node is placed in the same address point that the high node would go into. In other words, only four wide nodes could be placed in either nonswitched expansion frames in configuration 2.

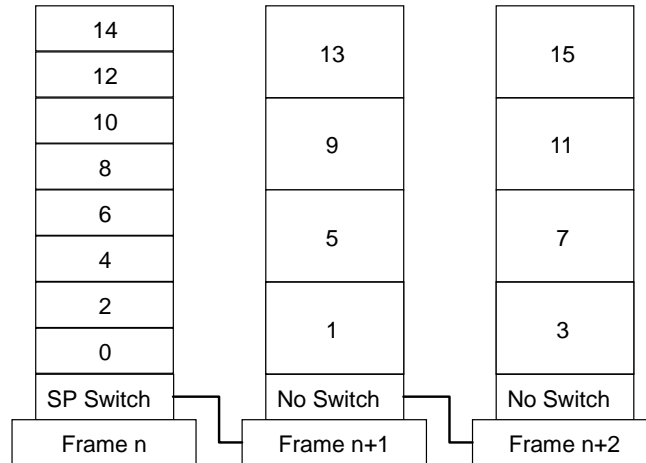


Figure 52. Configuration With Two Nonswitched Expansion Frames

In configuration 3, only high nodes are used in a tall model frame. Such a tall model frame uses only four switch ports. Therefore, its switch can support 12 additional nodes. A restriction here is that each nonswitched expansion frame has to be filled with only high nodes. Therefore, there can be up to three nonswitched expansion frames. Each of these nonswitched expansion frames can house a maximum of four wide or high nodes. Once again, if wide nodes are used, they must be placed in the high node address points.

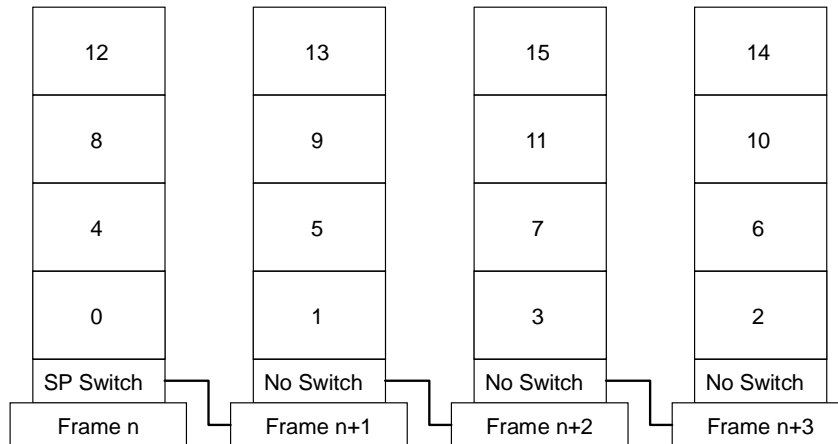


Figure 53. Configuration With Three Nonswitched Expansion Frames

The following configuration rule applies to these configurations:

Configuration Rule 11

Nodes in nonswitched expansion frames will be attached to the switch in the first preceding frame equipped with an SP Switch.

So you have to skip numbers between frames equipped with an SP Switch, if you plan to expand your SP system by the addition of nonswitched expansion frames at a later time. Nonswitched expansion frames must be numbered consecutively from the base frame number that contains an SP Switch.

SP Switch-8 and High Performance Switch LC-8

A system with SP Switch-8 or HiPS LC-8 contains only switch port numbers 0 through 7.

The following algorithm is used to assign nodes their switch port numbers for systems with eight port switches:

1. Assign the node in slot1 to Switch_port_number = 0. Increment switch_port_number by 1.
2. Check the next slot. If there is a node in the slot, assign it the current switch_port_number, then increment the number by 1.

Repeat until you reach the last slot in the frame or switch port number 7, whichever comes first.

Figure 54 on page 79 shows a sample configuration.

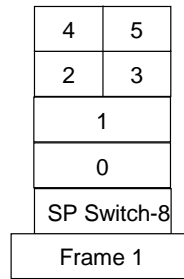


Figure 54. SP Switch-8 and HiPS LC-8 Configuration

Chapter 4. Software Components

This chapter details the major software components of the SP.

Figure 55 on page 81 shows the SP system software architecture.

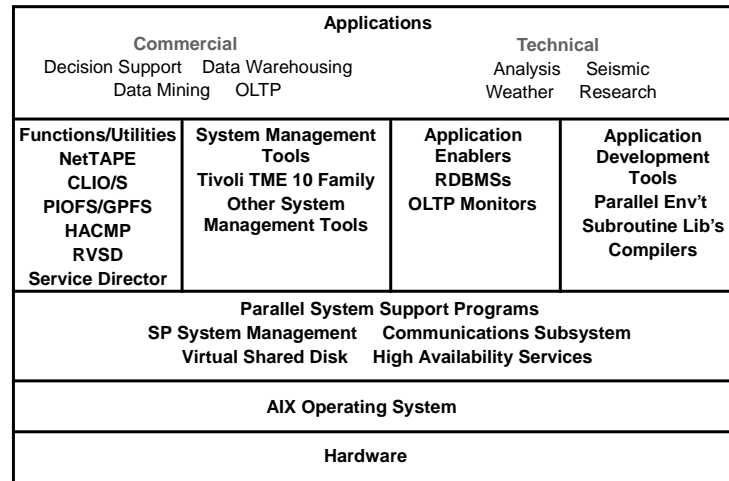


Figure 55. RS/6000 SP System Software Architecture.

The layers that are covered here are the basic ones from a conceptual and management point of view, such as AIX and Parallel System Support Program. Also are discussed some functions/utilities, such as CLIO/S, PIOFS and RVSD.

As shown in Figure 55 on page 81, the basic software layer in every machine is the operating system that makes the machine's resources available for useful work. It also provides services used by applications, such as file and device management, I/O control, and so on. Next in the hierarchy is PSSP. PSSP provides a number of services that help the administration look at the SP system as one entity, and it also provides services that help the system to be robust, reliable and available.

Running on top of the PSSP layer are a number of tools. This chapter discusses functions/utilities. The objective is to give the reader the knowledge to understand the functions, external to PSSP, that are available to make system resources as usefull as possible and to provide the system with new features employing already existing elements.

We next introduce the software components of an SP system, while the remainder of the chapter discusses these in greater detail, focusing on PSSP building blocks.

4.1 System Data Repository (SDR)

The System Data Repository (SDR) contains the SP-specific configuration information. It is a central repository that only resides on the Control Workstation from where the configuration information is distributed to the nodes.

4.1.1 Introduction

The SDR is an SP subsystem that stores SP configuration and some operational information, such as:

- Information about the frames, nodes and switches and how they are configured
- Job Manager operational data
- VSD configuration information
- The current values of `host_responds` and `switch_responds`, two indicators of the health of a node

This information is stored on the Control Workstation, but is made available through the client/server interface to the other nodes.

4.1.2 SDR Data Model

The SDR data model consists of classes, objects and attributes. A *Class* is a type of device or feature, for example: adapter, frame, node, SP, and so forth.

Attributes are the details about a class. For example, the attributes of class adapters are: `adapter_type`, `enet_type`, `netmask`, `netaddr`, and `node_number`.

Objects are a specific set of attributes. Objects have no unique ID, but their combined attributes must be unique.

Refer to *Appendix F, IBM Parallel System Support Program for AIX: Administration Guide*, GC23-3897 for details on classes, objects, and attributes. Figure 56 on page 83 shows an example of the SDR data model.

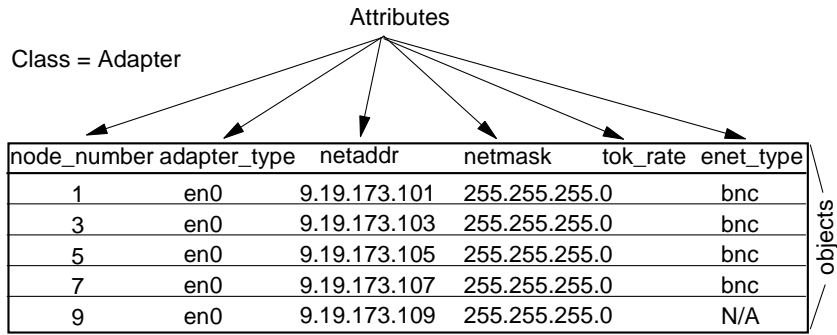
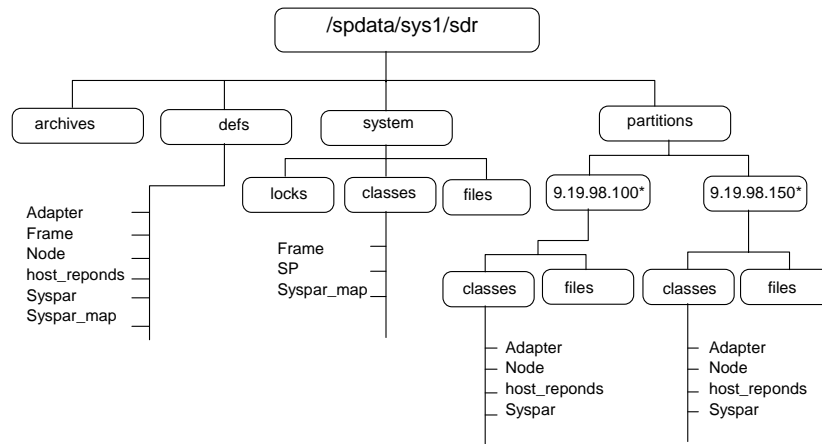


Figure 56. SDR Data Model

The SDR stores class data in the /spdata/sys1/sdr directory on the Control Workstation. The complete SDR directory structure is presented in Figure 57 on page 83.



9.19.98.100 & 9.19.98.150 are used as an example of two partition system. Actual IP address would be different on your system.

Figure 57. SDR Directory Structure - Sample

/spdata/sys1/sdr/system

Contains classes and files that are global to the system.

/spdata/sys1/sdr/partition

A /spdata/sys1/sdr/partition/<partition_ip_address> subdirectory is maintained for each partition. Object classes are replicated for each partition and each class keeps information on the objects pertinent to the partition.

/spdata/sys1/sdr/archives

Contains files of the SDR for backup purposes.

/spdata/sys1/sdr/defs

Contains the header files for all the object classes. Each file describes what fields and variables are used to define the object of the class.

Before any changes are committed to a class, the contents are moved to backup or shadow files located in the same directory as the class. These files are named `class.shadow`. If there is a catastrophic error, such as a power loss or kill of the SDR daemon while updates are being written, the SDR could get corrupted. The shadow file, if one exists, could be used for recovery. Another method to insure against the loss of SDR data is to back up the SDR regularly using the `SDRArchive` command. `SDRArchive` takes a snapshot of the SDR contents and saves it in a tar file. It is a good idea to use this facility before making any system configuration changes.

4.1.3 User Interface to SDR

There are three ways users can get access to the SDR information:

SMIT

SMIT is by far the most commonly used method for interacting with the SDR. As a result of a successful installation of PSSP, some SMIT menus will be added to let you manipulate database information.

Command line interface

Sometimes it can be quicker and very helpful to query SDR information directly. For example: The command `SDRGetObjects` with appropriate arguments can be used to do a query on the SDR database directly.

Through SP commands

The SDR can also be accessed through high-level PSSP commands. One such command is `splstdata`, which can be used to query node, frame, and boot/install server information from the SDR.

4.1.4 SDR Daemons

The daemon that handles the SDR inquiries and updates is `sdrd`. Daemon activity is written to the log `/var/adm/SPlogs/sdr/sdrlog.syspar_ip_addr.pid`, where `syspar_ip_addr` is the IP address of the system partition and `pid` is the PID of the SDR daemon. This means that, based on the number of partitions, you may have multiple `sdrd` daemons running. The logs contain the data and time the process started along with any problems the daemon may have run into.

The SDR daemons are controlled by System Resource Controller (SRC). The `startsrc` command uses the `-g` flag in order to start multiple daemons. It runs the `/usr/lpp/ssp/bin/sdr` script to start the daemons. The script passes the IP addresses associated with the partitions to the respective daemons.

4.1.5 Client/Server Communication

There is one SDR daemon for each partition. Each daemon is uniquely identified by an IP address. In the case of a default partition, this is the IP address associated with the *hostname* of the Control Workstation. In the case of other partitions, this address is the IP alias that was defined for each partition.

A look at `/etc/service` shows that SDR daemons are listening on TCP port 5712. Each node can connect to this port by specifying the IP address associated with its partition (the IP alias or the hostname of the Control Workstation) and making a socket connection. In this way, each SDR daemon is able to communicate only with the nodes in the partition that it associates with.

When a node is initially booted, the `rc.sp` script, started from `inittab`, looks for the `SP_NAME` variable to get the IP address of the partition it must communicate to. The `SP_NAME` variable is not set by default. The user needs to explicitly set and export this variable. If it is not set, the `/etc/SDR_dest_info` file is referenced. This file contains the IP address and name for the *default* and *primary* partitions. The default is set to the IP address associated with the *hostname* of the Control Workstation. The primary is the IP alias given to the partition it belongs to.

The default entry in the `/etc/SDR_dest_info` file serves two purposes. The first is that whenever the node is rebooted, the `rc.sp` script that runs from `inittab` does the following as part of its function:

- Gets the default IP address from the `/etc/SDR_dest_info` file on that node.
- Contacts the SDR and checks `Syspar_map` to find out whether the IP address entry in the `/etc/SDR_dest_info` file for the primary is correct. The `Syspar_map` is a global class that describes the mapping of nodes onto partitions. It contains one object for each node and the node number (switch node number will show, if applicable).
- Updates the `SDR_dest_info` with the correct entry, if it is not correct.

This is a critical step in getting the partitioning to work and is one of the reasons why the affected nodes should be shut down. The reboot ensures that the `/etc/SDR_dest_info` file is updated with the new IP alias address.

The second reason for having a default IP address is that if the partition gets deleted by restoring an archive of the SDR, or by applying a different partition, then the node can still reach the SDR by using this IP address (after failing to connect to the primary).

Figure 58 on page 86 shows an example of how nodes in different partitions interact with sdrd to get access to the SDR.

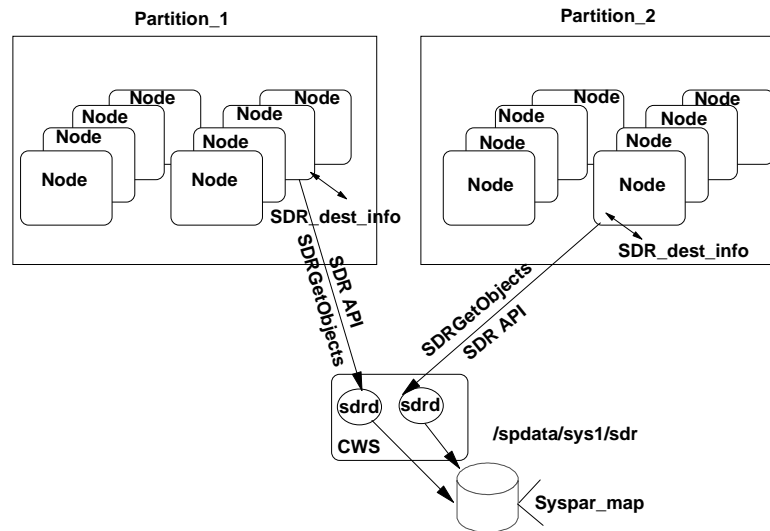


Figure 58. Example: How Nodes Communicate with the SDR

In our example, we are using two partitions, partition_1 and partition_2, so there will be two instances of the sdrd daemon running on the Control Workstation. Once the socket is successfully opened, the client can query the SDR using the SDR interface. In our example, the `SDRGetObjects` command is used to query SP objects.

4.1.6 Locking

Now that there is the potential to have multiple SDR daemons accessing the same system-wide or global object classes, there is the requirement to have a persistent locking mechanism to prevent simultaneous updates of object classes. The lock files are created in the `/spdata/sys1/sdr/system/locks` directory. The file that is created contains details of the node updating the SDR and the partition to which it belongs.

Whenever the SDR daemons are started up, they check for the presence of any lock files in case they were left behind when an update of the SDR ended abnormally. If any are found at this point, they are removed. This is another reason why it is a good idea to recycle the SDR daemons when there are apparent SDR problems.

4.1.6.1 Holding Locks

SDR clients (Job Manager, the SDR command-line routines and so on) can lock an entire class to get exclusive write access. While the client holds the lock, other clients can read the data as it was before the lock, but cannot write to the class. If the client that holds the lock fails to unlock the class before it exits, the server releases the lock without making any changes to the class. The only cases where a class may be locked and the server can release the lock are the following:

- The client that has the class locked is stuck in an infinite loop.
- The node which the client is running crashes or is powered down. The SDR server is not notified of the broken connection until the connection times out. This occurs after the time indicated in the *tcp_keeptime* variable of the `no` command output. The default is 14,400 half-second increments, or two hours.

Only when an administrator is sure that one of the above conditions has happened, should he use the `SDRClearLock` command.

The `SDRWhoHasLock` command can be used to determine if a class is locked and by whom.

4.1.7 Manipulating SDR Data

Before making any changes to the SDR, it should be backed up with the `SDRArchive` command. This enables you to recover from any catastrophic problems such as a power loss. The `sprestore_config` command can be used to restore the SDR to a known working state.

Making changes to a Class

Attention!

You must take extreme care in making changes to the SDR data. If not handled properly, this might result in a corrupted SDR with unpredictable consequences.

To make changes to attributes of existing objects in a class, you can use the `SDRChangeAttrValues` command. With this command, you query the attribute that needs changes and assign it a new value. If a particular object is no longer needed, `SDRDeleteObjects` can be used. Again, be very careful when using these commands.

Another characteristic of the SDR is its ability to store and distribute other files. Commands such as `SDRCreateFile`, `SDRRetrieveFile`, `SDRReplaceFile`, and `SDRDeleteFile` are available to manipulate the files stored in the SDR.

Some of the subsystems that use this SDR facility are:

- The Switch, to distribute the topology files
- Topology services, to distribute configuration files
- GPFS, to distribute configuration files

4.1.8 Useful SDR Commands

We list here some handy SDR commands for quick reference. For details on these commands, refer to *IBM Parallel System Support for AIX: Command and Technical Reference*, GC23-3900.

SDR_test

This command verifies that the installation and configuration of the SDR completed successfully. The test clears out a class name, creates the class, performs some tasks against the class and then removes it.

SDRListClasses

This command simply lists the class names that exist in the SDR.

SDRWhoHasLock

This command returns the transaction ID of a lock on a specified class. The form of the transaction ID is `host_name:pid:session`, where `host_name` is the long name of the machine running the process with the lock, `pid` is the process ID of the process that owns the lock, and `session` is the number of the client's session with the SDR.

SDRClearLock

This command unlocks an SDR class. Use this command carefully and only when a process that obtained a lock ended abnormally without releasing the lock. Using it at any other time could corrupt the SDR database.

4.1.9 The SDR Log File

Serious SDR problems are reported in `/var/adm/SPlogs/sdr/sdrlog.<pid>`, where `<pid>` is the process ID of the SDR daemon. Often, this log contains little more than a message showing a return code of 110, indicating that a client exited without closing its session with the SDR. These messages have no adverse effect on the SDR daemon and can be ignored.

There may be many SDR log files. A new one is created every time a new SDR daemon is invoked. Once the contents are checked, the files can be either archived or discarded, except for the file that corresponds to the daemon that is currently running. To find the current log file, find the process ID of the SDR daemon using `lssrc -g sdr`. The current log file has that process ID as its extension.

4.1.10 Backup and Restore of SDR

The `SDRarchive` command is provided to back up the SDR. We highly recommend that this always be done before making any changes to the SDR. This will ensure that you restore back to your original state.

The `SDRrestore` command is used to restore the backup.

4.2 The Hardware Control Subsystem

The objective of this section is to give the reader an understanding of what aspects of the SP hardware can be monitored and controlled. Monitoring is the activity performed to review the system state. Controlling means to change some aspect of the system state.

4.2.1 What Can Be Monitored and Controlled?

There are three main hardware components of the SP that can be monitored and controlled:

- Frame
- Node
- Switch

Frame: Information that can be obtained from monitoring the frame includes electrical and environmental conditions, such as voltage, current, temperature, power supply status and serial link (RS232) status.

SP software enables you to control the power of the frame.

Note: Data that can be monitored from a node includes three-digit display, electrical and environmental conditions such as power, voltage, temperature and fan failure.

SP hardware-control software enables you to control or change the state of the node's power, key position and reset button.

Switch: Data that can be monitored from a switch includes electrical and environmental conditions such as power, voltage, temperature and fan failure.

Hardware control enables you to control the state of the power of the switch and the multiplexor clock setting.

4.2.2 How It Works

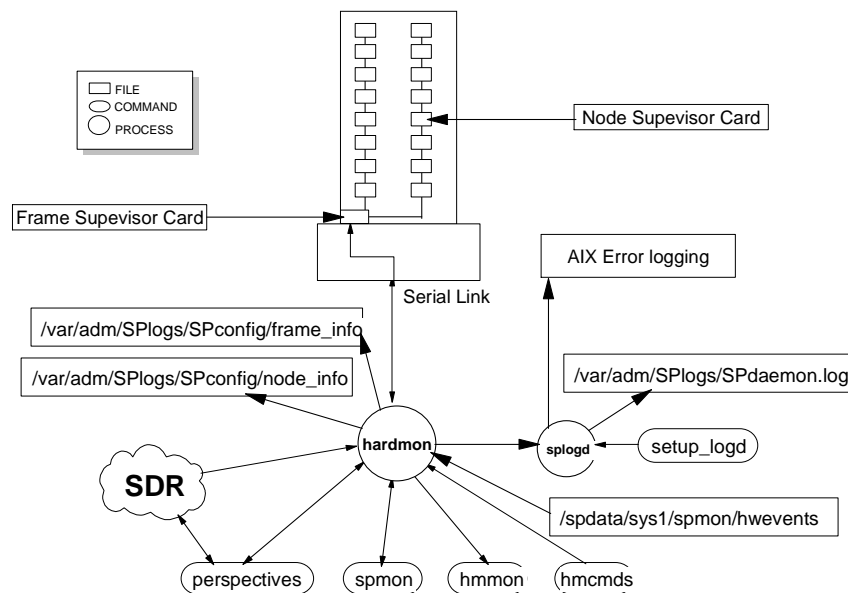


Figure 59. SP System Monitor

Figure 59 on page 90 shows the logical structure of the SP Hardware Monitoring and Control System. At the heart of this SP subsystem is a daemon, the Hardware Monitor daemon, called *hardmon*. Surrounding the core there are client commands. *hardmon* runs on the Control Workstation and communicates with the frames, effectively polling them, using the serial link that every frame has to the CWS. Across this line, *hardmon* sends commands to the frames and receives responses from them. Commands that

are sent across this link include orders to monitor the system and orders to change the system hardware state in some way.

hardmon manages a series of *variables* that represent the state of the frames, nodes and switches in the SP. The value of these variables can be queried using commands such as `spmon` and `hmmon` or the SP System Monitor Graphical User Interface (GUI). The `spmon` and `hmmon` commands are hardmon clients. For a complete list of the available variables that are maintained by hardmon, refer to *IBM Parallel Support Programs for AIX: Administration Guide, GC23-3897, Appendix D: System Monitor Variables*.

`hmcmds` is a command to change the SP system hardware, like powering off and on, to select the mux clock for a switch, and so on. Depending on the device to be controlled, there are a number of options to use.

Also, as can be seen in Figure 59 on page 90, the hardware monitoring daemon does error logging. All the errors captured by hardmon are logged in the standard AIX error log file. This work is done by `splogd`, another daemon whose purpose is to help log errors on behalf of hardmon both to the normal AIX error log file and also to the BSD error logging facility.

As can be seen from Figure 59 on page 90, there a number of files related to hardmon:

`/var/adm/SPIogs/SPconfig/frame_info`: A hardmon generated file that contains the frame types an SP system has.

`/var/adm/SPIogs/SPconfig/node_info`: A hardmon generated file that contains the node types an SP system has.

`/spdata/sys1/spmon/hwevents`: A file that holds the hardware events that hardmon has to watch for and inform `splogd` about.

`/var/adm/SPIogs/SPdaemon.log`: A file where the BSD error logging facility of `splogd` stores its activity.

`setup_logd`: A script that sets up the environment for the SP system error logging to run in, such as the log directory and file creation under `/var/adm`, addition of error log templates for SP messages, addition of the `splogd` daemon to SRC as the `splogd` subsystem, inclusion of `inittab` entries to startup `splogd` and so on. Also this script helps when the SP system administrator wants to run the `splogd` in a different machine than the CWS.

Now that the architectural software components that make up the SP hardware control mechanism have been discussed, it is easier to understand

how this subsystem works. Its active agent is `hardmon`, which polls the frames, nodes and switches every five seconds inquiring the state of the different elements, and based on the results it updates a series of state variables that can be queried by clients. It is not advisable to change the poll rate without a good reason. Also, `hardmon` can receive requests from a client program to modify the state of a component; it then generates a command to perform the required tasks, if possible. If `hardmon` detects errors, they are logged, using the `splogd` services, to the normal AIX error log file and the BSD error logging facility.

As can be seen, `hardmon` enables you to carry out tasks that can disrupt the system services, so its functions are strongly restricted, but this aspect can be customized by the System Administrator. So before any of its services can be accessed, the identity of the user for whom `hardmon` is performing the operation must be authenticated to the authentication server of the SP system.

To do this, the user who wants to have `hardmon` services access must be registered to the SP system authentication server as a principal.

Once this is done, the principal must get a *ticket* from the *ticket-granting service* which is used to request a *service ticket* from the `hardmon` daemon.

Next, the system administrator has to include the principal's name in the `/spdata/sys1/spmon/hmacls` file to give authorization to this principal to use the `hardmon` services. The `hmacls` file controls the authority level the user has to access `hardmon` services. Once the `hmacls` file has been changed, `hardmon` must reread it. To accomplish this, the command `hmadm setacls` must be issued.

The `hardmon` services are SP system wide. This means that this subsystem is not partition-sensitive, because it polls the SP hardware components, which are not affected by partitioning.

Further information about `hardmon` services authorization can be found in *IBM Parallel Support Programs for AIX: Administration Guide*, GC23-3897, in the chapter "Using the SP System Monitor".

4.2.3 User Interfaces

This section focuses on presenting the interfaces the SP software provides to enable the user to work with the hardware monitor services.

There are two user interfaces that let the user interact with the `hardmon` services: the command line and the GUI.

The command line interface consists of the `hmcmds`, `hmmon` and `spmon` commands. `hmmon` and `spmon` allow the query of a specific state variable value or its monitoring. `hmcmds` and `spmon` let the user request a change-of-state operation, like power-on, power-off or reset. If not explicitly stated, these commands do their work on system partition boundaries.

The GUI allows the hardware objects to be monitored and controlled either on the active system partition or across partition boundaries, using the pointer device to perform the desired operation. The GUI can be invoked using the `spmon` command `spmon -g` or the perspective interface command `sphardware`.

Detailed information on how to use and operate the line interface command can be found in *IBM Parallel System Support Programs for AIX: Command and Technical Reference*, GC23-3899. Information about the GUI is in *IBM Parallel System Support Programs for AIX: Administration Guide*, GC23-3897, in the chapters "Using the SP System Monitor" and "Using SP Perspectives".

4.3 Overview of Switch Software

By this point we expect that from earlier chapters of this book you must have familiarized yourself with the hardware architecture of the switch. In this section we concentrate on the switch's working principles and the topology files used by it. We will also discuss concepts such as Primary node, Primary backup node, and Fault Service daemon.

4.3.1 Introduction to the Switch

The switch is, in essence, a high speed communication network. It supports the following protocols to communicate between the nodes:

- TCP/IP protocol suit
- User Space Message Passing

The system facilities based on IP, such as NFS, and licensed programs such as AFS, can be configured to use the switch. With the possible exception of applications that depend on network-specific functions (LAN broadcasts, for example), most applications work over the switch without modification.

The User Space Message Passing mode of the switch is used by Parallel Virtual Machine (PVMe, no longer marketed) and Parallel Operating Environment (POE) for AIX. The parallel programs use the Message Passing Libraries (MPL) or Message Passing Interface (MPI) over the switch to

communicate between the tasks in a running program. With PSSP 2.3, only one user-space process may access this mode in a partition. PSSP 2.4, however, allows support of *Multiple User Space Processes Per Adapter* (MUSPPA). The Resource Manager controls access to the User Space Message Passing mode.

4.3.2 Switch Operating Principles

It is important to understand that although the switch provides data communication across the nodes in much the same way as Ethernet, Token Ring, FDDI, or ATM, it works on an entirely different operating principle.

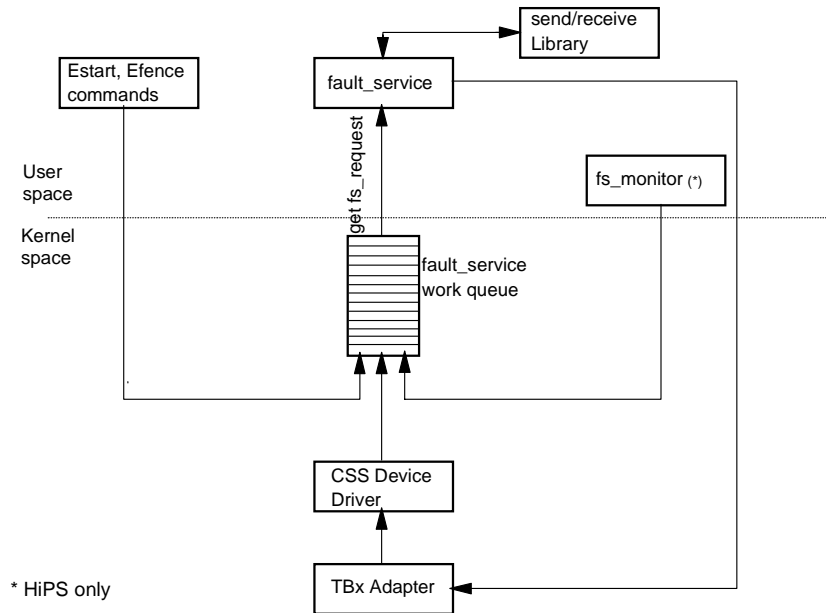


Figure 60. Fault Service Architecture

Figure 60 on page 94 describes the Fault-Service architecture of the switch. The fault-service daemon (`fault_service_Worm_RTG_SP`), which contains the Worm and the *Route Table Generator*, is central to the functioning of the switch. It can be viewed as having three main personalities:

- Primary Worm
- Backup Worm
- Secondary Worm

The Primary Worm plays a key role in the coordination of the switch network. It is a non-concurrent server and therefore can only service one switch event to completion before servicing the next. Examples of switch events (faults) are switch initialization (`Estart` command), switch chip error detection/recovery, and node (switch port) failure. Either the Worm on the nodes or the switch chips themselves can report a fault to the Worm daemon on the primary node.

Switch events are queued by the `fs_monitor` process on the work queue one element per event. The `fs_monitor` uses the system call `get_fs_request` to get single work queue elements to the fault service. It should be noted that HiPS uses the `fs_monitor` daemon, whereas in SP Switch (TB3) this functionality is incorporated in the Worm daemon, so you will see only the `fault_service_Worm_RTG_SP` daemon running in nodes connected to the SP Switch (TBS) switch board.

The Worm daemon must be running in each node that is part of the switch network. If not, that node is considered down from the primary, even if everything else is up and running.

4.3.3 Primary Node Behavior

By default, the first node in the partition is designated as primary node. This node processes switch commands, issued from the Control Workstation. A backup of this node, called primary backup node, is also created. By default, this is the last node in the partition. These defaults can be overridden.

Error recovery on the SP Switch is done locally instead of globally as on the High Performance Switch. On the HiPS Switch, a fault on a link would cause the entire fabric to be interrupted while recovery was performed. On the SP Switch, detection and fault isolation is done at the link level while normal traffic flows across the rest of the fabric. Detected faults are forwarded to the active primary node for analysis and handling. When the primary node completes assessing the fault, the remaining nodes on the fabric are nondisruptively informed of status changes.

The primary backup node passively listens for activity from the primary node. When the primary backup node detects that it has not been contacted by the primary node for a predetermined time, it assumes the role of the primary node. This takeover involves nondisruptively reinitializing the switch fabric, selecting another primary backup, and updating the System Data Repository (SDR) accordingly.

The primary node also watches over the primary backup node. If the primary node detects that the primary backup node can no longer be contacted on the switch fabric, it selects a new primary backup node.

Oncoming primary and oncoming backup primary are used only by the SP Switch. With oncoming nodes, you are now able to change the primary and the primary backup for the next restart of the switch. This is convenient for maintenance purposes, since it allows you to change the oncoming nodes without taking the switch down. Actually, these values in the SDR are only informative, and will be read by the switch subsystem when an `Estart` is issued.

4.3.4 Secondary Nodes Behavior

The nodes that are neither primary nor primary backups are referred to as secondary nodes. The main tasks performed by secondary nodes are:

1. Build the optimal database based on the topology file. This step is performed only when the daemon is started (for example, at boot time).
2. Modify the database based on *deltas* (message indicating un-initialized/fenced nodes and/or uninitialized/fenced links) received from the primary node.
3. Call the route table generator to build the routes from this node to all other nodes.
4. Inform the switch protocol what destinations are available.
5. Load the new routes down to the adapter.

Secondary nodes send/receive service packets to/from the primary node only: The secondary node acknowledges the primary node for each database *delta* message it receives from the primary node.

4.3.5 Switch Topology File

Topology files are located on the Control Workstation at `/spdata/sys1/syspar_configs/topologies` with symbolic links to the `/etc/SP` directory.

The switch topology file is critical to the successful working of the switch. The switch daemon uses this topology file as the basis for creating the routing tables which get propagated to every node on the switch network. The topology files are shipped as standard with the PSSP software, and during the installation process you are required to choose the appropriate

one, based on the number and type of the switch boards that you have installed.

The topology file, `expected.top.NSBnumnsb.ISBnumisb.type`, describes the wiring configuration for the switch. *NSBs* (Node Switch Boards) are switches mounted in frames containing nodes and *NSBnum* is the number of *NSBs* you have installed. *ISBs* (Intermediate Switch Boards) are used in large systems (typically installations with more than four *NSBs* or switches) to connect frame switches so that bandwidth is not compromised on the switch network. For example, a simple installation with two frames and two switches would use `expected.top.2nsb.0isb.0` file. Since there are no *ISBs* used in this configuration, this number is 0.

The Switch-8 (8-port, non-scalable switch) uses unique topology files:

- `expected.top.1nsb_8.0isb.0` used by HiPS
- `expected.top.1nsb_8.0isb.1` used by SP Switch

Figure 61 on page 98 shows a sample topology file. The contents of a given topology file are enhanced by running the `Eannotator` command against it. This adds comments to the records of the file, namely jack information which helps the reader understand the cabling between elements of the system. The `Eannotator` command embodies an understanding of the cabling used in the system of a given switch type. The system uses these only labels when reporting errors in the error log or the `css` logs. If these labels are incorrect, it will be difficult to debug problems because the errors may be attributed to the wrong node.

`Eannotator` physically checks which connections it is going through (that is, the jack socket or the switch connection) to get to a node or switch chip and updates the topology file accordingly. It converts what is initially logical information to actual physical information. For example, L06 for logical frame 6 may be converted to E8 because this may be the 8th physical frame.

```

format 1
16 18
# Node connections in frame L01 to switch 1 in L01
s 15 3 tb0 0 0 L01-S00-BH-J18 to L01-N1
s 15 2 tb0 1 0 L01-S00-BH-J16 to L01-N2
s 16 0 tb0 2 0 L01-S00-BH-J20 to L01-N3
s 16 1 tb0 3 0 L01-S00-BH-J22 to L01-N4
s 15 1 tb0 4 0 L01-S00-BH-J14 to L01-N5
s 15 0 tb0 5 0 L01-S00-BH-J12 to L01-N6
s 16 2 tb0 6 0 L01-S00-BH-J24 to L01-N7
s 16 3 tb0 7 0 L01-S00-BH-J26 to L01-N8
s 14 3 tb0 8 0 L01-S00-BH-J10 to L01-N9
s 14 2 tb0 9 0 L01-S00-BH-J8 to L01-N10
s 17 0 tb0 10 0 L01-S00-BH-J28 to L01-N11
s 17 1 tb0 11 0 L01-S00-BH-J30 to L01-N12
s 14 1 tb0 12 0 L01-S00-BH-J6 to L01-N13
s 14 0 tb0 13 0 L01-S00-BH-J4 to L01-N14
s 17 2 tb0 14 0 L01-S00-BH-J32 to L01-N15
s 17 3 tb0 15 0 L01-S00-BH-J34 to L01-N16
# On board connections between switch chips on switch 1 in Frame L01
s 14 7 s 13 4 L01-S00-SC
s 14 6 s 12 4 L01-S00-SC
s 14 5 s 11 4 L01-S00-SC
s 14 4 s 10 4 L01-S00-SC
s 15 7 s 13 5 L01-S00-SC
s 15 6 s 12 5 L01-S00-SC
s 15 5 s 11 5 L01-S00-SC
s 15 4 s 10 5 L01-S00-SC
s 16 7 s 13 6 L01-S00-SC
s 16 6 s 12 6 L01-S00-SC
s 16 5 s 11 6 L01-S00-SC
s 16 4 s 10 6 L01-S00-SC
s 17 7 s 13 7 L01-S00-SC
s 17 6 s 12 7 L01-S00-SC
s 17 5 s 11 7 L01-S00-SC
s 17 5 s 11 7 L01-S00-SC

```

Figure 61. Sample Topology File

To understand the nomenclature used in the topology file, refer to Figure 62 on page 99. Each line in the topology file describes how the nodes and switches are connected to each other. Fields 2 to 4 describe the switch end of

the connectivity, whereas fields 5 to 7 describe the second end (node or another switch) of the connectivity.

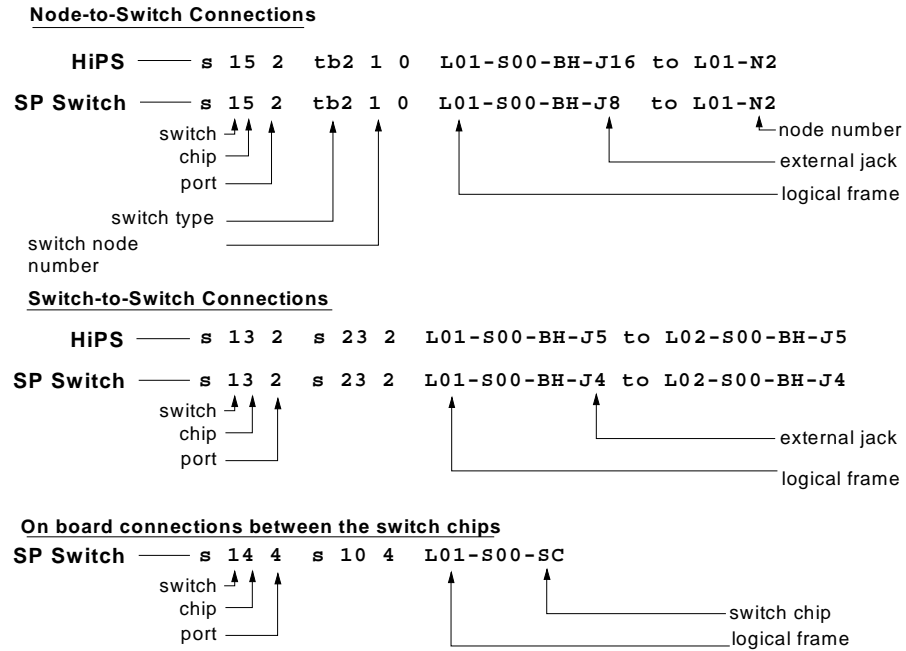


Figure 62. Nomenclature Used in Topology Files

1. **Field 1:** This is always s, signifying that this end of the cabling is connected to a switch board.
2. **Field 2:** This describes the switch number it is connected to. 1 in the figure means that it is switch number 1 and it is inside the node frame. If this is a switch in a switch frame, 1000 is added to differentiate it from the node frame.
3. **Field 3:** This is the switch chip number. This number could range from 0 to 7, as there could be 8 switch chips on a switch board. This number is actually part of field 2 above, but we have separated it to make it simple to understand. A number such as 15 would mean that it is switch chip number 5 on switch number 1. The number 10025 means that it is switch chip 5 on switch number 2, which is housed in a switch frame.
4. **Field 4:** This field denotes the port number on the switch chip.

5. **Field 5:** Notations such as tb0 or tb2 have no relation to the actual type of adapters used in the nodes. They only signify that this end of the connectivity is used to connect a node.
6. **Field 6:** This denotes the switch node number and could range from 0 to 15. In reality this number is 1 less than the actual node number. While interpreting this file, you may add 1 to this number so that it maps correctly with the node number (N numbers, range between 1 to 16).
7. **Field 7:** This field provides the same connectivity information in a more elaborate manner, using the jack number, logical frame number and actual node number. In the figure, bulk head (BH), jack 8, on switch 1 (s00), in frame 1 (L01), is connected to node number 2 (N2) in frame 1.

Figure 62 on page 99 also provides an example of switch-to-switch connections. You may notice that Fields 5, 6, and 7 are now showing the switch connectivity in the same manner as discussed above. The third and last section of this figure provides on-board connections between the switch chips.

Figure 63 on page 100 provides an example of how you can map the notations used in the topology file (Figure 61 on page 98) on the switch, switch chips, ports and nodes, to understand the connectivity.

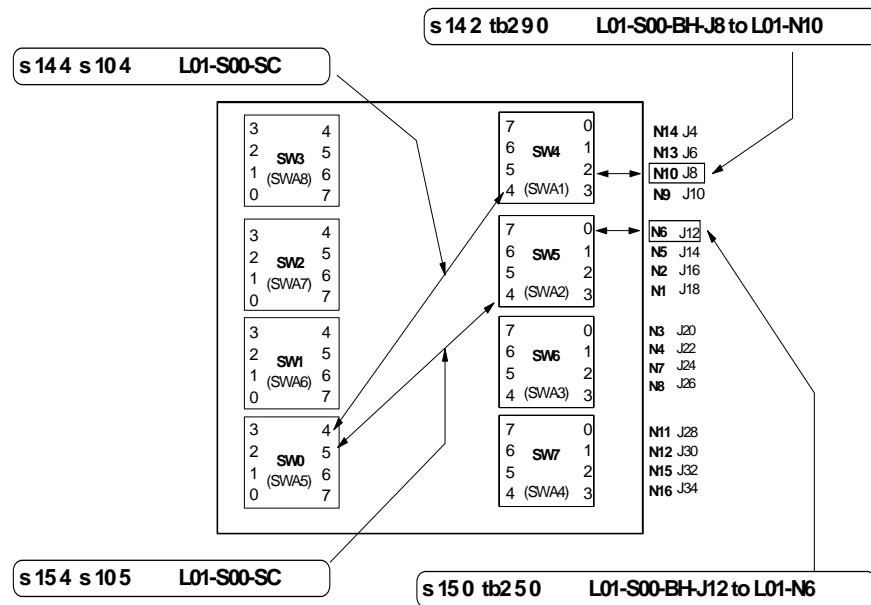


Figure 63. Example: How a Topology File Maps Physically on the Switch

4.3.6 Routing in SP Switch

We learned at the beginning of this book that each switch board in SP switch contains eight logical switch chips. In Figure 66 on page 103, the switch chips numbered SW4, SW5, SW6, and SW7 handle all the communications with the nodes using chip ports 0, 1, 2, and 3. The other four chip ports (4, 5, 6, and 7) are used to communicate with the other switch chip side. It is possible for each of the chips that service the nodes to reach the other three node chips through four different routes.

Lets have a look at the routing mechanism a node uses to communicate to others on the same or a different switch. The switch chips connect together to form multi-stage switching networks.

In general, a packet contains two things: a route and some data. The route specifies where (that is, to which output port) the data is to be delivered. This format supports variable length routing. One nibble (that is, four bits) of routing information is required for each switch chip traversed by the packet. As a packet traverses the network, it discards used routing data. Figure 64 on page 101, shows a typical multstaged packet.



Figure 64. Structure of the Data Packet

The Data Packet format is:

- **BOP** is a Beginning-of-Packet control character. It is a packet delimiter that indicates that the data that follows is route and packet data.
- **Route Bytes** follow the BOP control character; their number depends on the switching system topology.
- **Data Bytes** follow the routing bytes. A packet can contain zero bytes of data.
- **EOP** is the End-of-Packet control character. It is a packet delimiter that indicates the end of the current packet data stream.

The use of the BOP and EOP control characters allow for packets of variable and unlimited length.

Service Packet Format

The servicing of the switch fabric is accomplished by using specially formatted Switch Service Packets. Figure 65 on page 102 shows a sample of a Service Packet.



Figure 65. Service Packet

A switch chip determines that the service packet is destined for it when the packet arrives without route bytes between the BOP character and the Service Control Character (SVC). The switch will route this packet to its service logic. If route bytes exist in a service packet when it reaches a receive port, it is treated as any other normal data traffic.

There are six different types of service packets:

- **Initialization Packets** contain information the switch needs to operate correctly in its current environment, such as links to enable and disable. It also contains information about where to report a detected error. An error/status packet (see below) is returned as an acknowledgment to the initialization packet.
- **Read Status Packet** is used to interrogate a switch about its current status. An error/status packet is returned in response to receiving an area status packet.
- **Reset Packet** is used to reset error conditions detected by the switch chip. Once an error has been reported, it must be reset before the next error can be reported. An error/status packet is returned in response to receiving a reset packet. Other errors will still be detected and stored during this interval, but will not be reported until the current error condition has been reset.
- **Set Local Time-of-Day (TOD) Packet** is used to correctly set the Time-of-Day on the switch. The TOD is not used by the switch, but merely used to help in setting the TOD on the node adapter. An error/status packet is returned in response to receiving a set local time-of-day packet.
- **Set Remote Time-of-Day Packet** is issued to a switch when that switch's TOD is to be propagated to a device (switch or adapter) attached to the current switch. No acknowledge is returned for this packet. The acknowledge will come from the device being sent the Time-of-Day.
- **Error/Status Packet** is returned as a switch-generated packet that contains current information about the switch. The information includes: current error information, current port state information, current receive buffer utilization, current central queue buffer utilization, and current token utilization.

Routing Example:

To see how routing actually works, let us take an example from Figure 66 on page 103. In this example, with node 14 (N14) communicating to node 16 (N16), the path a packet could take is:

The packet passes SW4 through port 0, and can exit the switch chip through the following four routes, which are the chosen routes:

1. Port 7 across to SW3, onto that chip through port 4, exiting on port 7, over to SW7, onto the chip on port 7.
2. Port 6 across to SW2, onto that chip through port 4, exiting on port 7, over to SW7, onto the chip on port 6.
3. Port 5 across to SW1, onto that chip through port 4, exiting on port 7, over to SW7, onto the chip on port 5.
4. Port 4 across to SW0, onto that chip through port 4, exiting on port 7, over to SW7, onto the chip on port 4.

Once onto SW7, it will exit through port 3, go through J15 to the cable, and to the switch adapter on node 16 (N16).

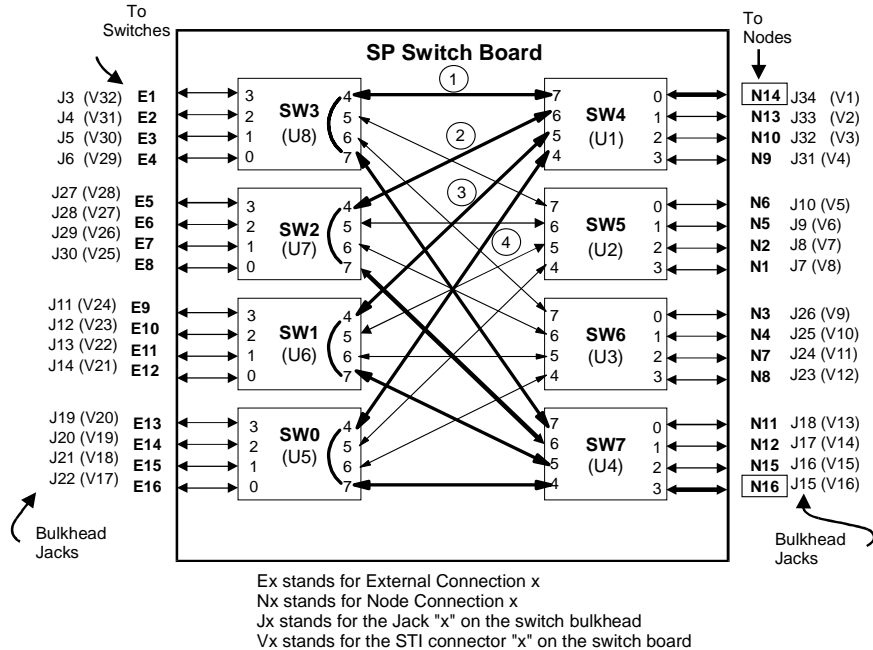


Figure 66. Example: Routing in the SP Switch

4.3.7 Switch Initialization

The following steps are needed during a switch initialization:

1. An `Estart` command is issued for a system partition from the Control Workstation through SMIT or the command line, and this is relayed to the primary node of the partition as an `Estart_sw`.
2. The primary node consults the SDR file for the current topology file and the current fenced nodes.
3. The primary node distributes the topology file to the bootserver nodes of its partitions. Then, it directs each bootserver to distribute the topology file to its client nodes.

Each bootserver distributes the topology file, tests for success, and returns the resulting status to the primary.

The primary node logs and evaluates failure data.

4. The primary runs the Worm code to verify the partition's fabric pool and nodes as follows:
 - Each chip is sent a read status service package to check cabling.
 - Each chip is sent an initialization service package to set the chip ID and specify routes to the primary.
 - Each non-primary unfenced node is sent an initialization service packet to set its personality and specify the topology file for this partition.
 - Each non-primary unfenced node is sent a read status node service package to verify the switch address, personality, and readiness.
 - The primary node updates its device database in accordance with the finding.
5. The primary node sets the Time-of-Day (TOD) across the fabric pool as follows:
 - Traffic is quiesced, if necessary.
 - The TOD is propagated.
 - Traffic is resumed, if necessary.
6. The primary node downloads its route table to its TB3 adapter.
7. The primary node distributes device database *deltas* to the node over the switch through the device database service package.
8. On each non-primary, unfenced node:
 - The local device database is updated with the *deltas*.
 - Routes are computed.

- The routes are loaded to TB3 adapter SRAM.
9. On each node, the communications protocols and Load Leveler code, as appropriate, are notified that the switch is available.
 10. The primary node updates the SDR switch_responds class for its partition. For each node in the partition, the (autojoin, isolated) pair is set to one of the following:

(0,0)	Initial
(0,1)	Fenced
(1,0)	On
(1,1)	Suspended

4.3.8 Switch Clocks

The switch network is a synchronized network with a clock (oscillator) source, usually within the same frame. For systems with multiple frames, there is only one clock source is distributed. One of the frames is designated as Clock Master: the one that distributes its internal clock to all other frames.

The clock subsystem is critical for the proper functioning of the switch fabric. In order to assure that every frame is receiving the same clock signal, usually alternative sources are designated in case the primary alternative fails. However, this switchover is not automatic but requires manual intervention.

The `EcLock` command establishes a master clock source after the system is powered up or when an alternate must be selected. It can set the appropriate clock input for every switch in the system or for a single switch after power-on.

The `/etc/SP/Eclock.top.NSbnumsb.ISBnumisb.0` file contains the clock configuration.

4.3.9 Switch Log Files

There are numerous log files associated with the switch, some of which can provide essential data for resolving switch problems. It is beyond the scope of this book to furnish details about each log file and provide enough information on how they can be read and interpreted. You may refer to *IBM Parallel System Support Programs for AIX: Diagnosis and Messages Guide*, SC23-3899 and *RS/6000 SP: PSSP 2.2 Survival Guide*, SG24-4928 for more details. We will, however, cover the `out.top` file in some detail. A utility, `css.snap`, is provided which creates a compressed tar image of all the switch logs. This image is provided to local IBM Support Centers for further analysis, if required. The switch logs are located as follows:

On the Control Workstation:

- /var/adm/SPlogs/css/Eclock.log
- /var/adm/SPlogs/css/Ecommands.log
- /var/adm/SPlogs/css/SDR_config.log
- /var/adm/SPlogs/css/sdrlog.<cws_ip_address>.<PID>

On the primary Node:

- /var/adm/SPlogs/css/rc.switch.log
- /var/adm/SPlogs/css/out.top
- /var/adm/SPlogs/css/fs_daemon_print.file
- /var/adm/SPlogs/css/flt
- /var/adm/SPlogs/css/dtbx.trace
- /var/adm/SPlogs/css/dtbx.trace.failed
- /var/adm/SPlogs/css/daemon.stderr
- /var/adm/SPlogs/css/daemon.stdout
- /var/adm/SPlogs/css/cable_miswire
- /var/adm/SPlogs/css/router.log

/var/adm/SPlogs/css/css.tab

On a Node:

- /var/adm/SPlogs/css/rc.switch.log
- /var/adm/SPlogs/css/out.top
- /var/adm/SPlogs/css/fs_daemon_print.file
- /var/adm/SPlogs/css/flt
- /var/adm/SPlogs/css/dtbx.trace
- /var/adm/SPlogs/css/daemon.stderr
- /var/adm/SPlogs/css/daemon.stdout
- /var/adm/SPlogs/css/css.tab

4.3.10 The out.top File

The /var/adm/SPlogs/css/out.top file is a dump of the topology file with errors and fault information, created when the switch is initialized. In this file you can find, at a glance, some common wiring and node problems.

Figure 67 on page 107 provides an example of out.top file output.

```
s 14 2 tb0 9 0      E01-S17-BH-J32 to Exx-Nxx
s 14 2 tb0 9 0      E01-S17-BH-J32 to Exx-Nxx -4 R: device has
                    been removed from network - faulty (wrap plug is installed)
```

Figure 67. Example of out.top File Output

The out.top file is broken into a logical and physical notation. There are also comments in the file that are designed to help you understand the basic connections that are occurring. The comments at the top of the file should help you to remember the format. Other comments describe a group of connections (such as nodes connected to board 1). These comments precede the connections they describe.

After the logical and physical nomenclature is given on a line the fault information is given with an error number, an indication of which side of the connection found the error, and a description of the error: -4 R: device has been removed from network - faulty (wrap plug is installed).

4.3.11 Managing the Switch

4.3.11.1 Setting up the Switch Clock Source

The switch clock is synchronous across all nodes active on the switch. Its value is set to zero when the primary node powers on, and is later distributed by the `Estart` command to all nodes on, or joining, the switch.

The `Eclock` command establishes a master clock source after the system is powered up or when an alternate must be selected. It can set appropriate clock input for every switch in the system. After `Eclock` completes, the `Estart` command must be run.

4.3.11.2 Setting up the Primary and Backup Primary Nodes

By default, the primary is the first node in the partition and the primary backup is the last node in the partition. These defaults can be changed through the `Eprimary` command or SMIT interface. The criteria to select a new primary node is as follows:

- Try to select a node on a switch assembly other than the switch assembly to which the primary backup is attached.
- If no other switch assembly exists, select a node attached to a switch chip other than the one to which the primary backup is attached. This is to

avoid failure of both primary and primary backup in the event of a single chip failure.

- If no other switch chip exists, select any available node on the switch chip to which the primary backup node is attached.

The above considerations are also applicable if you decide to select a new primary backup instead of a new primary.

After running the `Eprimary` command and until the next `Estart` command is issued, these nodes are reflected as the oncoming primary and oncoming primary backup nodes. Once `Estart` completes, the primary field is updated based on the oncoming primary field and the backup field is updated based on the oncoming backup field.

4.3.11.3 Starting the Switch

Once all the processor nodes are running, the switch can be started. The steps involved are:

1. Ensure that `fault_service_Worm_RTG_SP` (the fault service daemon that checks, initializes and prepares the switch for operation) is running. This daemon is started by the `rc.switch` script. If it is not running, use `rc.switch` to start the daemon first.
2. On the Control Workstation, run the `Estart` command. If you encounter any problem, refer to diagnosis information in the *IBM Parallel System Support Programs for AIX: Diagnosis and Message Guide*, SC23-3899 and *RS/6000 SP: PSSP 2.2 Survival Guide*, GC24-4928.

You can monitor switch connectivity to the nodes by using SP Perspective or in *All Node Summary* for the switchReponds, in the `spmon` GUI. On switched systems, the node's status is represented by the following three colors in the *ALL Node Summary*:

- **Green:** indicates the node is active on the switch
- **Red:** indicates that a node is not active on the switch for one of the following reasons:
 - It was fenced without the `-autojoin` option.
 - It defaulted and was removed from the switch.
 - Its adapter failed to configure properly.
 - Its adapter was recently reset by the `/usr/lpp/ssp/css/rc.switch` script.

To determine the reason why the error occurred, query the `switch_responds` object in the SDR with the `SDRGetObjects` `switch_responds` commands.

- **Yellow:** indicates that a node is not active on the switch because it was *fenced* with the `-autojoin` option.

4.3.11.4 Fencing and Unfencing the Nodes

The node fencing function provides for isolating nodes that will be rebooted or powered off from the switch in order to prevent such nodes from impacting other nodes. The capability to allow the node to rejoin (the `-autojoin` option) the switch once it is again operational is also provided.

You may run the `Efence node1 node 2` command to fence node 1 and node 2, or you can use the graphical user interface (perspective).

`Eunfence node 1 node 2` can be used to return the node to the switch.

Notes:

1. The primary node and primary backup node for an SP switch cannot be fenced.
2. If you specify the `-autojoin` option while fencing, your node cannot return via `Eunfence` to the switch network. The node will rejoin the switch on a switch fault, node reboot, or `Estart`.

4.3.11.5 Monitoring Nodes on a Switch

You can monitor the availability of nodes on a switch with the `Emonitor` daemon. `Emonitor` is controlled by the System Resource Controller (SRC). One instance of the daemon exists for each partition and is named `Emonitor.partition_name`. A system-wide configuration file, `/etc/SP/Emonitor.cfg`, lists all node numbers (one per line) on the system to be monitored.

`Emonitor` is invoked with `Estart -m`. Once invoked, it is SRC controlled and so will restart if halted abnormally. To end monitoring, run `/usr/lpp/ssp/bin/emonctrl -s`, which stops the daemon in the system partition.

4.3.11.6 Global Shutdowns and Reboots of Nodes with a Switch

The `Equiesce` command disables switch error recovery and primary node takeover. It is used to shut off the normal error actions when global activities are performed. For example, when the nodes are shut down or rebooted, they are not fenced from the switch.

The `Equiesce` command causes the primary and primary backup nodes to shut down their recovery actions. Data still flows over the switch, but no faults are serviced and primary node takeover is disabled. Only the `Eannotator`, `Eclock`, `Eprimary`, `Estart`, and `Etopology` commands are functional after the `Equiesce` command is issued.

`Estart` must be issued when the global activity is complete to reestablish switch recovery and primary node takeover.

4.4 Time Service

To present a single system image to distributed applications and system management layers, consistent time across nodes is critical. The PSSP partition-sensitive daemons use timestamps in their communications and will become quickly confused if time is not synchronized. Service tickets and keys of kerberos, the SP authentication mechanism, tolerate five minutes of difference in the time setting of communicating hosts. Original SP nodes did not have a system battery to preserve the system clock across a shutdown; therefore, a time management protocol was mandatory.

4.4.1 Why Network Time Protocol?

IBM chose Network Time Protocol (NTP) for the SP because it is public domain code that scales well. Customers already using NTP can quickly integrate the SP into their time management regime. NTP is optional: a customer can implement another time synchronization protocol such as timed and Digital Time Service.

Since the SP can support different applications across different sets of nodes, a flexible time service is needed; for example, a customer may need to implement three different time zones, corresponding to 3 applications running on three independent node groups. Different NTP domains can easily be configured within the SP. With the entire SP synchronized, any internal timestamp for an application or management function is the same at any given moment on any node.

4.4.2 NTP Architecture

NTP uses a hierarchical, master-slave configuration. Refer to Figure 68 on page 111. Atop the hierarchy are the primary servers, which maintain their own, presumably highly accurate, clocks. Below the primaries are secondary servers. There may be many levels (hops) in the hierarchy of secondary servers. Each level is referred to as a stratum, with the primaries at stratum 0, the first level of secondaries at stratum 1, and so on. The stratum of a server

specifies exactly how many *hops* are required to the primary clock source. The higher a server's stratum number, the more hops are required up the hierarchy, and the less accurate that server's time value.

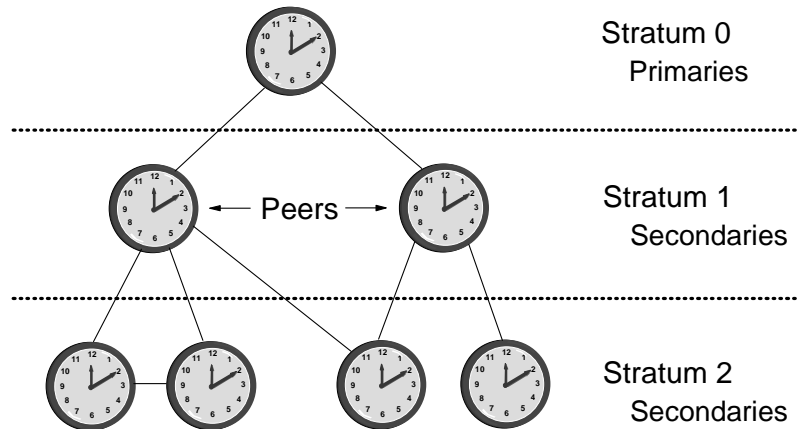


Figure 68. NTP Hierarchy

Secondary servers are both clients of lower-stratum servers and servers to higher-stratum servers. Servers in the same stratum are called peers, and may give or request time from each other. A host configured to have multiple time servers employs a selection algorithm to determine the most accurate and reliable server with which to synchronize. The accuracy of synchronization depends on a host's position in the stratum and the duration of synchronization. The duration of synchronization refers to how frequently a host measures to resolve the skew, or clock drift, among clocks. The more frequently the host compares its clock to its time server(s), the better the accuracy.

NTP servers communicate via UDP/IP messages, thus any IP-capable network can form an NTP hierarchy. NTP servers are IP hosts with corresponding IP addresses. NTP is implemented by the `xntpd` daemon.

4.4.3 NTP Modes

Four NTP implementation options are offered on the SP. The selection of an option governs the NTP configuration on the Control Workstation, boot/install server nodes, and regular nodes.

consensus Run NTP on the SP nodes and Control Workstation. This configuration uses the Control Workstation as the primary time server, or time master. Any boot/install servers in the

SP become peer secondary servers. Regular nodes can request time from either boot/install server nodes or the Control Workstation, as all are connected on the SP private Ethernet.

timemaster	Use the site's existing time server to synchronize the SP nodes and control workstation. The Control Workstation and boot/install server nodes are peers and request time from the time server. Regular nodes can request time from the Control Workstation and boot/install server nodes, and depending on their connectivity, from the time server directly.
internet	Use an NTP server from the Internet to synchronize the SP nodes and control workstation. The Control Workstation is assumed connected to the Internet time server(s) and is configured alone in a stratum. Boot/install servers become peers at a higher number stratum (one level down the hierarchy from the Control Workstation). Regular nodes request time from the control workstation and boot/install server nodes.
none	Do not use NTP.

When configuring NTP for timemaster or Internet, we suggest specifying multiple time servers on separate networks to ensure the SP always has access to a time source.

The default configuration is consensus. On the CWS, IP address 127.127.1.10 is configured as the time server. This address signifies local time service to NTP and makes the control workstation a stratum 10 time server, allowing you to hook into a higher stratum time server at a later date.

When you choose any of the options that involve using NTP, the installation scripts determine the IP address of the host(s) specified as the primary server(s) and create the `/etc/ntp.conf` file on the Control Workstation, boot/install servers, and processor nodes. This file indicates which hosts can provide time service to which other hosts.

4.5 High Availability Infrastructure (HAI)

This section aims to concisely explain the key concepts of HAI, illustrated by real-life examples of HAI in action. There are entire books devoted to the topic, such as

- *RS/6000 SP High Availability Infrastructure*, SG24-4838

- *SP Monitoring: Keeping it Alive*, SG24-4873
- *IBM Parallel System Support Programs for AIX: Event Management Programming Guide and Reference*, SC23-3996
- *IBM Parallel System Support Programs for AIX: Group Services Programming Guide and Reference*, SC28-1675

See these references for the details of HAI implementation and exploitation.

4.5.1 "In the Beginning ..."

In 1994-95, IBM focussed on high availability on the SP. The inherent design of many nodes simplified the hardware aspect - nodes could back each other up. The challenge was, and still is, in the upper layers of the system, such as the database and application.

IBM's High Availability Cluster Multiprocessing (HACMP) product was available at the time (HACMP is still marketed today). It robustly protected hardware resources of small clusters of RS/6000 machines or SP nodes. HACMP had two major limitations:

1. **Scalability.** In HACMP, every node in the cluster communicated with every other node. This was fine for a cluster of up to 8 nodes, but on an SP with 512 nodes, you would have 512x511 simultaneous conversations. The overhead of the basic node communication would have overwhelmed network and processor resources. This is known as the n^2 problem, because as the number of nodes, n , increases, the number of conversations approaches n^2 .
2. **Scope.** HACMP concerned itself fundamentally with network adapters, networks, disks and nodes. Hooks were provided to manipulate the upper layers of software, but HACMP did not have a strong application view of availability.

IBM took HACMP's cluster manager code - the heart of the product - and reworked it to scale. Around this IBM built High Availability Infrastructure (HAI). Internally, HAI was known as Phoenix. HAI provides a set of services to monitor and manage the availability of applications.

Although HAI is currently implemented only on the SP, IBM's vision for HAI is far greater than just for the SP. HAI is designed to be a cross-platform set of services, potentially implemented on IBM's and even other manufacturer's server platforms. The HAI has been renamed to RSCT (RS/6000 Cluster Technology) and it will be packaged different in next PSSP releases. This way, non-SP RS/6000 customers could implement HAI. If RSCT is accepted

on new platforms inside and outside of IBM, perhaps it will be renamed to simply "Reliable Scalable Cluster Technology".

4.5.2 HAI Packaging

HAI made its first appearance on the SP with PSSP Version 2.2 in 1996. It is a standard part of PSSP, and is packaged in the ssp.ha file set. The Resource Monitor component, closely linked with HAI, partially relies on the Performance Agent of IBM's Performance Toolbox/6000 product. This is why Performance Agent must be ordered with any new SP.

4.5.3 HAI Architecture Overview - "The Picture"

No discussion of HAI is complete without "The Picture", as shown in Figure 69 on page 114.

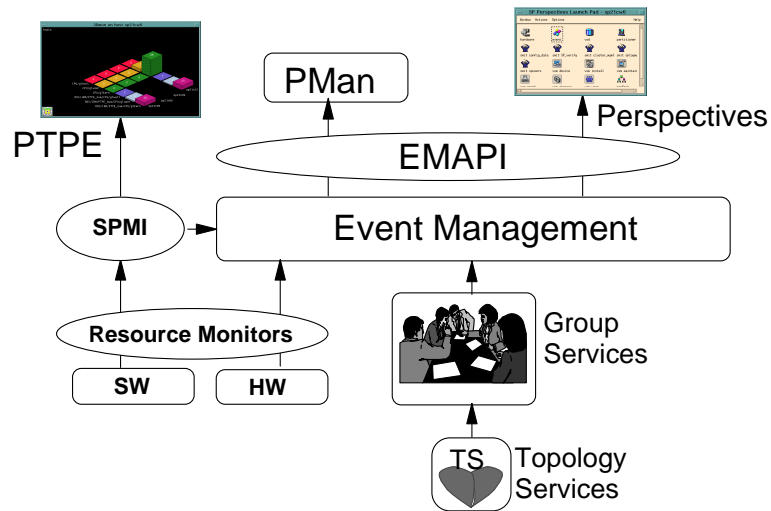


Figure 69. HAI Infrastructure - "The Picture"

Although many components are presented in the figure (all of which will be explained throughout the book), HAI comprises three principle elements:

1. Topology Services (TS)
2. Group Services (GS)
3. Event Manager (EM)

Sitting above EM is the Event Manager Application Programming Interface (EMAPI). Through this API, higher level components gain knowledge of the state and availability of system resources. Examples of these components

include Problem Management (PMAN), a system monitor such as the Perspectives graphical user interface, and others (such as HACMP ES, and potentially databases, enablers and applications).

Feeding EM with SP resource information are Resource Monitors (RM) and the System Performance Measurement Interface (SPMI). SPMI is a construct of Performance Agent. SPMI and EM supply data to higher level performance monitoring tools such as Performance Toolbox Parallel Extensions (PTPE). This is further discussed in Section 5.6, "Performance Management" on page 247.

RM, SPMI, and PTPE are an SP-specific implementation of resource and performance monitoring. They are not strictly part of base HAI. They changed as HAI moved from platform to platform, whereas TS, GS, and EM are designed to be platform-independent services.

We now discuss the core elements of HAI: TS, GS, and EM.

4.5.4 Topology Services

TS is the foundation of HAI. A distributed subsystem, TS is implemented by the hatsd daemon on each node and the CWS. TS lives to maintain availability information about nodes and their network adapters. TS considers a node to be "up" if it can be reached through at least one communication path. Determining valid communication paths is why TS concerns itself with network adapters.

Currently, TS supports only the SP administrative Ethernet and switch adapters. Eventually it will support all types of network adapters. TS monitors all SP node types and the CWS.

TS works tirelessly to update node and network adapter states, but does not care about the implications of, for example, a node going down. GS cares. GS subscribes to TS for node and network adapter availability. To take action on this information, GS needs a reliable network roadmap to broadcast its messages to the nodes. This leads us to TS's other primary responsibility: maintaining the network roadmap, or Network Connectivity Table (NCT) for use by GS's Reliable Messaging component. GS accesses the NCT via a shared memory segment. This is illustrated in Figure 70 on page 116

The hatsd daemons communicate via UDP. As UDP is an unreliable communication protocol, the daemons must validate and acknowledge the UDP packets. TS and GS communicate via UNIX Domain Stream (UDS) sockets.

Before describing how TS performs its duties, a few terms are explained:

Adapter Membership is the process of monitoring the availability of network adapters in the environment and establishing routes between nodes.

Adapter Membership Group is an association of adapters, governed by monitoring protocols. A Singleton group is an Adapter Membership Group with exactly one member.

Node Membership is the process of maintaining node availability based on Adapter Membership. If adapters on different nodes are in the same Adapter Membership Group, then the nodes can communicate. Nodes can also communicate indirectly across different Adapter Membership Groups. This is all captured in TS's NCT.

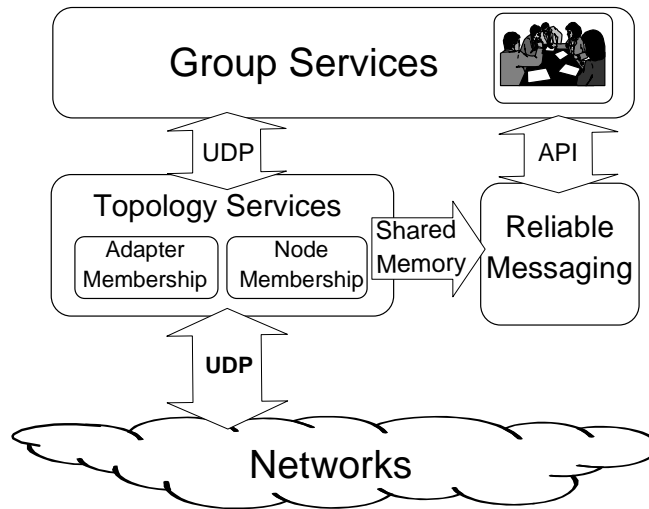


Figure 70. TS and GS Interfaces

4.5.4.1 Group Leaders, Crown Princes, and Death in the Family

You power up an SP. As each node comes up, it starts its hatsd daemon. TS comes alive. How does TS perform Adapter Membership? Since it uses UDP, TS must eventually become aware of the IP addresses of the network adapters on the other nodes. TS goes to the source of SP configuration: the SDR.

Each node builds a *machine list* after receiving information from the SDR. The machine list defines all nodes in the system, and the switch and administrative Ethernet IP addresses for each node. Note that the

administrative Ethernet interface is typically en0, but could be en1. You specify which interface when defining the reliable hostname in the SDR at installation time, and the corresponding IP address is sent in the machine list. When new nodes are added to a running SP system, the SDR does not automatically send the configuration changes to the nodes. The HAI stack must be refreshed for all affected partitions to update the machine lists and membership groups. TS could easily be ported to different platforms by essentially changing the front-end process, served by the SDR on the SP, that supplies the initial node and adapter information.

Machine lists are partition-specific and always include the CWS. TS has the dependency that each node has to be able to communicate with the CWS, thus the CWS is included in every machine list. The CWS runs one instance of hatsd for each partition.

Before continuing the description of the formation of Adapter Membership Groups, a little more terminology is offered. Every group has two special adapters with specific personalities:

1. **Group Leader.** This adapter has the highest IP address in the Adapter Membership Group. The Group Leader maintains the topology and connectivity information for the group and distributes it to the members.
2. **Crown Prince.** This adapter takes over as Group Leader when the Group Leader disappears. Possible causes of this succession are failure of the Group Leader's node, adapter, or IP communication subsystem. The Crown Prince has the second highest IP address in the group.

Now each node knows, via its machine lists, all the potential members of Adapter Membership Groups for the Ethernet and switch. Each node first initializes itself into a Singleton group - making it its own Group Leader and Crown Prince. The Group Leaders periodically send proclamations to all lower IP address adapters. The lower IP address Group Leaders eventually join the rings of the higher IP address Group Leaders, which incorporate the joiners' topology. Group Leaders, Crown Princes and member lists are constantly updated in the new, bigger rings. Finally, one ring for each adapter type exists and the final personality of the adapters is determined by IP address order. With the rings established, Adapter Membership Groups are created.

Every five seconds, using a Proclaim Packet, the Group Leader in an Adapter Membership Group invites other adapters that are in the machine list but not currently part of the group, to join the group. This is how adapters in new or rebooted nodes, for example, become part of the group.

In the steady state, each adapter in a group is responsible for passively monitoring its *Neighbor*, which is the adapter with the next highest IP address. By passively monitoring we mean that the Neighbor sends heartbeat packets to the adapter, which simply takes note of receiving them. For example, given Adapter_A with an IP address of x.x.x.10 and Adapter_B with an IP address of x.x.x.9, Adapter_A is the Neighbor of Adapter_B and sends Adapter_B heartbeat packets. Adapter_B does not acknowledge the packets from its Neighbor.

The Group Leader "wraps around": its Neighbor is the adapter with the lowest IP address in a group.

The monitoring process continues until a change to the group topology occurs, such as an adapter failure. The n^2 problem is avoided because in steady state, a given adapter is in communication with only one other adapter, independent of the number of nodes installed.

By default, heartbeats are sent every second (the *frequency*), and if four successive heartbeats are not received (the *sensitivity*), the Neighbor adapter is declared unavailable. Heartbeat frequency and sensitivity are tunable. If an adapter declares its neighbor unavailable, the hatsd daemon notifies the Group Leader with a DEATH_IN_FAMILY message. The Group Leader updates the membership status of the group and distributes it to the remaining group members, starting with a Prepare To Commit (to a new topology) message.

4.5.4.2 Network Connectivity Table - The Network Roadmap

TS creates several layers of information about nodes and adapters, culminating in the Network Connectivity Table.

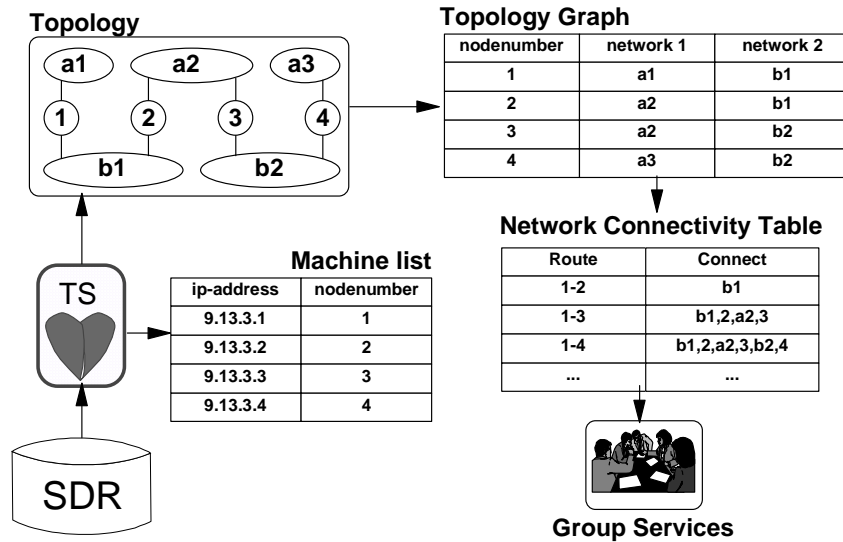


Figure 71. TS Process Flow

With reference to Figure 71 on page 119, TS goes through the following steps:

- Receive node and adapter configuration information from the SDR at the CWS
- Build machine lists
- Form the adapter rings
- Create Adapter Membership Groups
- Build a topology table and graph to indicate individual node connectivity, therefore availability
- Build the Network Connectivity Table in shared memory specifying all the valid routes between all node combinations
- Accept connections from local clients, primarily GS (up to this point TS would refuse connections because it has not completely stabilized its information)
- Monitor adapters via heartbeats and update the tables as necessary

Now that we know what nodes are available and how to talk to them, GS steps into the picture.

4.5.5 Group Services

Applications that best exploit the SP architecture typically comprise several cooperating processes running on multiple nodes. How does a distributed application first notice something has gone wrong in the system, then coordinate its own recovery? GS is a general purpose facility for coordinating and monitoring changes to the state of applications running on a set of nodes. The application view of high availability in HAI begins with GS.

With reference to Figure 72 on page 120, a GS daemon, hagsd, runs on all nodes and the CWS. Like TS, GS is partition-sensitive and multiple hagsd instances can be running at the CWS. Also like TS, GS daemons exchange their own reliable-protocol messages over UDP/IP. The communication routes selected are determined from the NCT created by TS.

As the name implies, groups are an important concept in GS. To take advantage of the coordination and monitoring functions of GS, an application must first form a group. The active members of the group correspond to the processes on each node that support a distributed application. Such group members are called *providers*. Another process may only wish to be informed of the group's activities. Such a process can ask to subscribe to a group, and if accepted, becomes a *subscriber* to the group. Subscribers are not group members. Subscribers just watch. Providers and subscribers are collectively called *GS clients*.

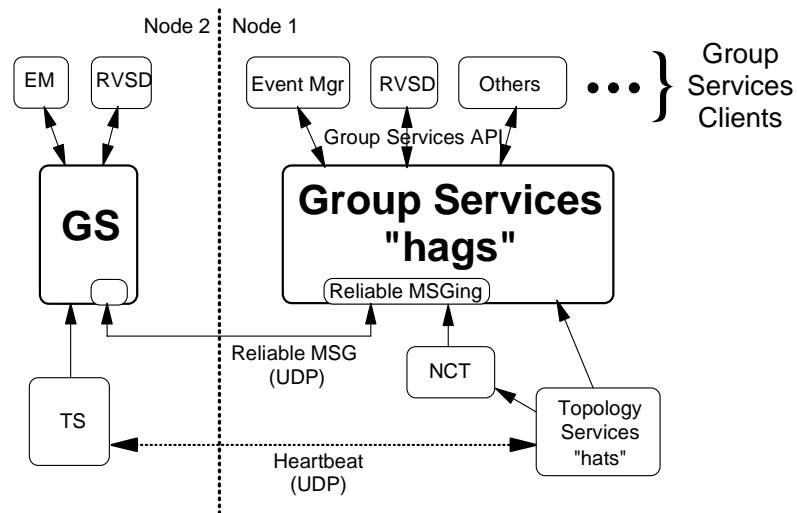


Figure 72. Group Services Structure

Groups are defined by three attributes:

1. Name. This must uniquely identify the group.
2. Membership List. This is a list of one or more providers. The list is maintained in order of age, with the oldest - the first provider to join the group - at the top and the youngest at the bottom. The provider is identified by an identifier, which comprises the instance ID and node number on which the provider is running. Instance IDs are defined by the provider itself when it joins the group. Identifiers must be unique in the group; for example, on a given node two identical processes may be running in support of a distributed application. They must be uniquely identified to participate properly in GS functions.
3. Group State Value. This is a byte field up to 256 bytes long. It is not interpreted by GS. It can be used as a group's billboard for advertising its state. EM uses the state value of one of its groups for version control of the Resource Manager database, to ensure each node in the group is monitoring from the same vintage of resource variables.

By default, the SP maintains three *internal* groups: host membership, Ethernet adapter membership, and switch membership. GS clients can subscribe to these groups to keep track of hardware status. In addition, GS clients can create *external*, or user, groups.

Who can be a GS client? Who can create groups? The answer is: any process of an application or subsystem that uses the Group Services Application Programming Interface (GSAPI). Examples from IBM include EM, RVSD, GPFS, and HACMP ES. If you are writing your own applications or updating an existing one, you could embed GSAPI function calls to:

- Coordinate your peer processes running on other nodes.
- Create a "bulletin board" to advertise your application state to other applications.
- Subscribe to changes in the state of other applications that yours may depend on.

Use of GS is optional. GS itself does not perform recovery actions, but provides the synchronization and commit protocols that application programmers typically find complex, expensive, and error-prone. Some applications have already incorporated their own methods for fault tolerance; GS does not demand that you re-engineer these facilities. For these applications GS could be used only for inter-application coordination.

4.5.5.1 GS Components - Functional Overview

This section describes the internal components of GS and how GS clients can get services from GS - by forming a group.

With reference to Figure 73 on page 123, GS consists of several components:

- **TS Client Module.** GS receives services from TS here.
- **Reliable Messaging Module.** This module, using the NCT created by TS in shared memory, provides reliable, sequenced delivery of messages between GS daemons.
- **Name Server Module.** This module controls the GS namespace. GS provides a single group namespace per domain, managing all internal and external groups. A domain is the set of nodes within a partition, and only one GS daemon is elected GS nameserver in a domain. After the TS Client Module successfully connects to TS, GS determines the lowest-numbered operational node by sorting node membership data. The GS daemon corresponding to this node becomes the GS nameserver for the domain. Should the GS nameserver daemon fail, the GS daemon on the next lowest-numbered node is elected nameserver. If the original GS daemon comes back on-line, it does not resume its responsibilities, but goes to the end of the succession line.
- **Client Control Module.** This module accepts and manages connections to GS clients. GS clients employ UNIX domain socket connections with the GS daemon.
- **Meta-Group Control Module.** This module handles communications between the GS daemons. A *meta-group* is the collection of GS daemons that support a GS client's group. An example of communications handled by this module is the following: a request comes in from the Client Control Module (interfacing a GS client, such as Application_A) to create a new group called *elliott*. After reviewing the names of the groups it already supports, the Meta-Group Control Module communicates with the Name Server GS daemon to ensure that no other group by that name exists in the domain. The group *elliott* may already exist, created by Application_B on some other group of nodes in the domain.

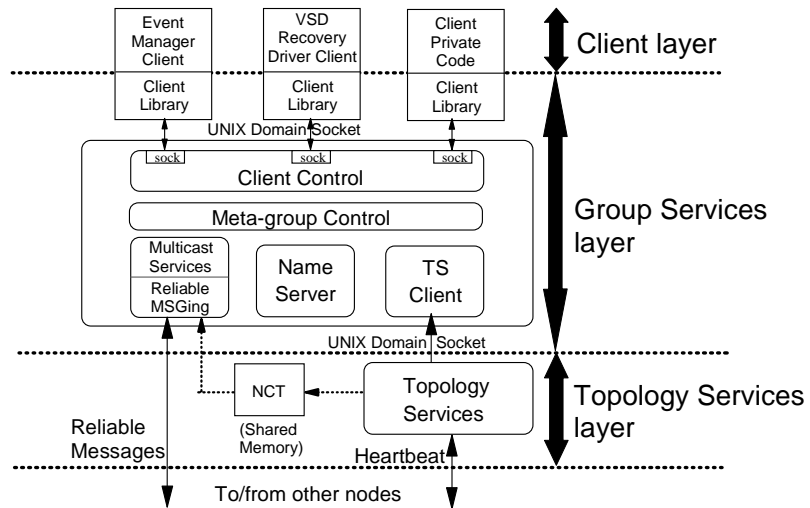


Figure 73. GS Internals

The GS subsystem comprises two daemons:

1. hagsd - provides most of the services of GS.
2. hagsglsm - a client of hagsd that provides global synchronization services for the SP Switch adapter membership group.

4.5.5.2 Creating GS Groups

A GS client creates an external group by executing a join request function call in the application program. The first join request creates a group, provided that group name does not already exist in the domain. Subsequent join requests to the group add new providers to the group. Internal groups, those for host and adapter membership, are created automatically via the same process by which GS daemons come alive. In effect, TS is a pseudo-GS client in that membership of the internal groups derives from TS data.

It is important to understand that although GS is globally aware of all groups existing in the system (the GS nameserver must have them all registered, complete with membership lists), not every GS daemon knows about every group. For example, refer to Figure 74 on page 124. In a 9-node SP, a parallel application is distributed across nodes 1, 2, 3, 4, 5 and 6. The processes on those nodes form a group called *glynn*. Similarly, the application *clarabut* is spread across nodes 2, 6, 7, and 8. Both groups *glynn* and *clarabut* are registered to the GS nameserver on node 9. If you query nodes 7 and 8 for the defined external groups, they will report only on *clarabut*. To maintain

knowledge of *glynn* and its gyrations in GS daemons independent of *glynn* wastes processor and communications bandwidth on nodes 7 and 8. Similarly, nodes 1, 3, and 5 know nothing of *clarabut*.

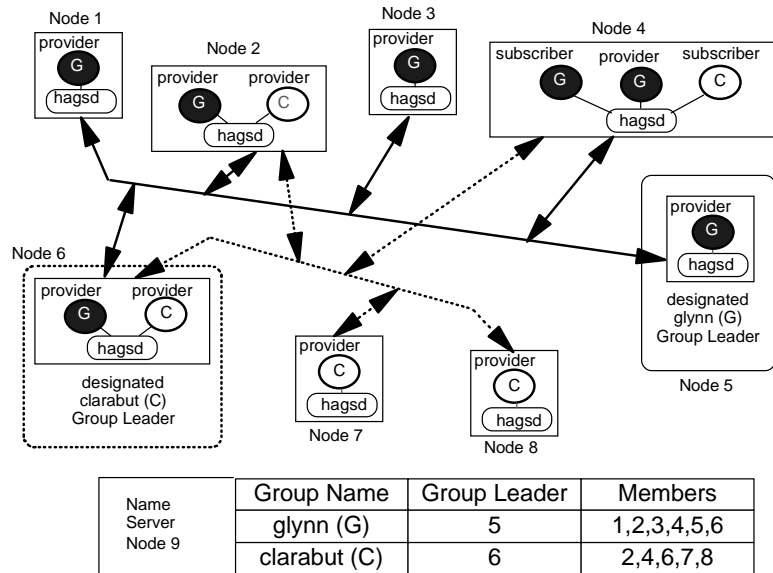


Figure 74. "Group Awareness" of GS Daemons

For each new group, one GS daemon will become the Group Leader. Yes, this is the same term as in TS, but GS Group Leaders are chosen by the age of the provider in the membership list, not by the IP address. The GS daemon on the node with the oldest provider becomes Group Leader. A Group Leader acts as the coordinator for all operations in the group, including:

- Managing group information (membership lists, state value)
- Managing group meta-data (items such as protocols, coordination of voting, default votes, recovery actions)

Group Leaders replicate data to all GS daemons on nodes where the group's providers reside for availability in case the Group Leader node fails.

4.5.5.3 Democracy Comes to the SP - Voting Protocols

So now we have created groups and are all set to enjoy the rich function of GS. How does a member act in a group in a controlled, synchronized manner? via *protocols*.

Protocols do the work in GS. A protocol is a mechanism that coordinates state changes and membership in a group. Consider typical actions that might occur in a group:

- **Membership Changes.** Providers can willfully join or leave the group. They can also leave by failing or being cast out by the other providers.
- **State Value Changes.** A provider may want to update the state value or the group's "billboard" for interpretation by another application; for example, a database wants to let a dependent application know that it is unavailable.
- **Message Broadcasts.** A provider may wish to send a message to all other members.

Protocols are somewhat analogous to a well-mannered democracy. Providers must respect group integrity and follow due process. Protocol communications are handled by the Group Leader. Protocols execute in the following sequence:

1. Proposal
Any provider, or GS itself, can start or propose an action. Suppose a process wants to join a group. It issues a join request, and GS will submit the request to the existing members of the group.
2. Asking Vote
Depending on the proposal, the group may have to vote. In our example, there are circumstances when the group does not want new members to join. If a distributed application is trying to recover from a failure, it may not necessarily want new processing nodes, ready to do application work, to come online until recovery is complete.

Voting is defined by the number of phases, or rounds of voting, required. The proposer of the protocol asks for n phases, where voting is repeated $n-1$ times. If a proposal requires 1-phase voting, the proposal is automatically approved. A 2-phase protocol requires one round of voting. The flexible n -phase voting mechanism allows application programmers to tailor their synchronization and coordination requirements.
3. Voting
If the protocol is 2 phases or higher, the members vote. Vote responses are Approve, Reject, or Continue. Unlike in most democracies, voting must be unanimous for proposal acceptance - one Reject vote causes the proposal to be refused. A Reject ends the protocol, regardless of additional rounds of voting requested. Continue is a conditional Approve, but with the desire to go to another voting round. A single Continue vote causes another voting round.

If a provider does not vote in the time allowed (for example, the provider's node fails), the Group Leader uses the group's default vote for that provider.

4. Notification of Result

GS notifies the provider of the result, and updates the membership list and state value accordingly. In our example, the potential provider has been voted into the group, is made a member, and put on the membership list of the other providers.

An illustration of GS protocol execution is part of Section 4.5.7, "Putting It All Together - A Simple Example of Exploiting HAI" on page 136.

GS guarantees orderly and coherent execution of protocols. For a given group, GS may receive many simultaneous protocol requests. GS will execute exactly one at a time (unless it is advantageous to batch up requests, such as multiple joins or failure leaves and cast outs). GS uses *barrier synchronization* in protocols. All providers must proceed to a certain point, or barrier, before the protocol continues. Completion of voting is a barrier: GS will wait until every vote is received, or the voting time limit expires and the default vote is cast for a provider, before taking any other action in the protocol.

4.5.5.4 Other GS Facilities

This section generally describes two other services of GS: Source-Target Groups, and sundered network recovery. Details on these services may be found in *RS/6000 SP High Availability Infrastructure*, SG24-4838.

Source-Target Groups

GS does not normally support services between groups beyond subscription. The Source-Target facility provides some level of synchronization between groups. Consider two applications: a disk availability subsystem, and a distributed database. Each forms a group in GS. If a node crashes, the database application may wish to begin its recovery protocols only after the disk recovery application uses GS to help make the database disks available again.

Sundered Network Recovery

Given an unfortunate set of network failures in a partition, GS daemons may become split and unable to communicate across the split. The GS namespace becomes *sundered*, although each GS daemon in each sundered portion of the partition has enough information to reconstruct the original group.

Bad things can happen in sundered namespaces. Two nodes, each owning a tail of a twin-tailed disk subsystem, could be on other sides of the split. The disk recovery application, relying on GS for node availability, may mistakenly inform each node in the split group that its availability partner is gone. Each node may try to acquire the disk subsystem, leading potentially to data corruption. A quorum mechanism in the application (GS does not provide such a mechanism) would be useful in this case.

When the network is repaired, GS has protocols to automatically reintegrate the sundered namespace and groups.

4.5.6 Event Management

GS provides applications with a rich infrastructure of coordination and synchronization services. In terms of giving the application detailed knowledge of the resources in the environment, about all GS can report is what it knows from TS: whether nodes and network adapters are up or down. How do distributed applications know about CPU utilization on a critical database node, or whether a file system is about to fill up, or if network errors are mounting on a key communications link? From the point of view of availability, any distributed application would be very interested in this type of information.

EM provides an application for comprehensive monitoring of hardware and software *resources* in the system. A resource is simply an entity in the system that provides a set of services. CPUs execute instructions, disks store data, database subsystems enable applications. You define what system events are of interest to your application, register them with EM, and let EM efficiently monitor the system. Should the event occur, EM will notify your application. Figure 75 on page 128 illustrates EM's functional design.

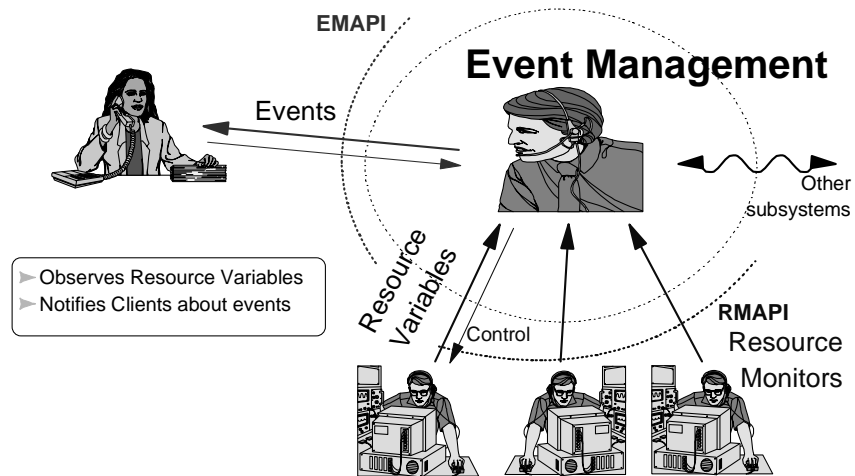


Figure 75. EM Design

EM gathers information on system resources using *Resource Monitors (RM)*. RMs provide the actual data on system resources to the event-determining algorithms of EM. RMs are integral to EM, therefore HAI, but how do RM get their data? Data-gathering mechanisms would vary according to platform. The SP-specific implementation of resource data-gathering mechanisms is described later.

EM is a distributed application, implemented by the EM daemon (haemd) running on each node and the CWS. Like TS and GS, EM is partition-sensitive, thus the CWS may run multiple instances of haemd. To manage its distributed daemons, EM exploits - that's right - GS. GS lives to serve applications like EM. As EM must communicate reliably among its daemons, it uses - right again - the Reliable Messaging information built from TS. This is shown in Figure 76 on page 129.

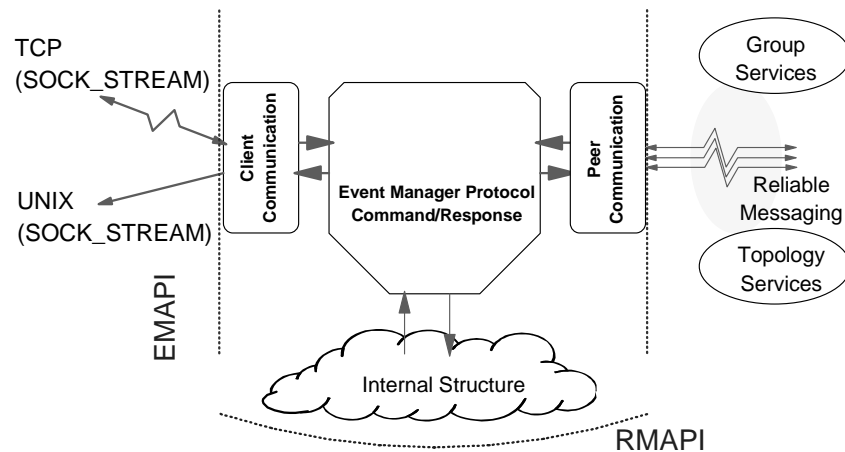


Figure 76. EM Client and Peer Communication

EM receives resource data across the Resource Monitor Application Programming Interface (RMAPI). Clients communicate with EM through the Event Manager Application Programming Interface (EMAPI). An EM client can comprise many processes spread across nodes in a partition. A local process, that is, one executing on the same node as a given EM daemon, uses reliable UNIX domain sockets to talk to EM. On the CWS, a local process connects to the EM daemon that is running in the same system partition as the overall client. In this manner, the client can get events from anywhere in its partition.

Remote clients, that is, clients executing in a separate partition or outside the SP entirely, use TCP/IP sockets, which is a less reliable method because of the protocol that cannot always properly deal with crashed communications sessions between programs. Remote clients usually connect only to the EM daemon on the CWS. When connecting, a remote client specifies the name of the target partition on the call to the EMAPI. The remote client will then connect to the EM daemon on the CWS that is running in the target partition. A client could connect directly to any EM daemon in the target partition and get the same events, but you would need an algorithm to determine the target node. It is easier to just connect to the appropriate daemon on the CWS.

4.5.6.1 Scalable Design

An EM client can efficiently monitor an event SP-wide by exploiting the distributed EM architecture. This is critical to the scalability of the monitoring and notification infrastructure. A client, executing on node *steve*, registers for

events occurring on node *scott* by communicating only with the EM daemon on node *steve*. Peer communication paths, based on Reliable Messaging, allow the EM daemon on node *scott* to transfer event notifications to node *steve*'s EM daemon, and therefore to the client. If node *steve*'s EM daemon tried to remotely connect to the RM on node *scott*, a great deal of resource data would go across the network for haemd on node *steve* to analyze for events. Why waste system resources when you have a node *scott* haemd there to look at the data and transmit registered events only as required? *RM are always local to the EM daemon.*

Clients must use the *event registration* process to inform EM of their interest in specific events. Applications perform event registration through function calls to the EM API. Perspectives provides a menu-driven interface to register events, and further specifies actions to take when events occur. Event registration defines:

- The *resource variable*, or attribute of a specific resource. A resource variable can be a counter (such as the number of packets transmitted on en0), quantity (such as average CPU busy), or state (such as the network interface up or down). Resource variables must be associated with a RM so that EM knows who to ask for data. This association is made via the *resource class*, discussed later.
- The *instance vector* of the resource variable. You want to monitor average CPU busy, but on which node? Which occurrence of that resource in the system? Instance vectors pinpoint specific resources in the system.
- The *predicate* or event-triggering rule to apply against the resource variable (such as "average CPU busy is greater than 90%"). Each variable has a default predicate.

You can also specify a *rearm predicate* for a resource variable. If set, the rearm predicate is monitored after the initial predicate is true. When the rearm predicate is true, then the initial predicate is once again monitored. For example, you are interested in CPU busy on node *sam*. Your predicate will trigger an event whenever *sam*'s CPU busy rises above 90%, but if the CPU workload is fluctuating, the resource variable value may repeatedly cross the 90% threshold, generating many identical events. If you specify a rearm predicate of CPU busy going below 60%, then after *sam* goes above 90% CPU busy, EM will not generate another event on this resource variable for *sam* until *sam*'s CPU busy falls below 60%. You assume that the situation has stabilized on *sam* if the CPU busy falls back below 60%.

Whenever a resource is added or changed in the SDR, the EM Configuration Database (or EMCDB) must be compiled, time-stamped for version control, and redistributed to the EM daemons. Restarting all EM daemons accomplishes the redistribution. This is discussed in more detail later.

4.5.6.2 Your Application and EM

To enable your application to use EM, you follow these steps:

1. Check the availability of a RM that gives you the data you need. If none exists, you would have to write a program using the RMAPI.
2. Check the SDR for your event definition. If it is there, great, else you will have to go through event registration.
3. Using the EMAPI, register your application as a client and receive events from EM. In addition to event notification, the EMAPI provides two query services for your application:
 1. The current value of a resource variable, whether or not it is triggering an event
 2. A list of defined resource variables and their default predicates

4.5.6.3 EM and GS

EM initialization illustrates how EM exploits GS. When haemd starts on a node, it fetches the EMCDB version from the SDR. Operating with the correct version of the EMCDB is crucial to service EM clients. After more initialization steps, haemd connects to GS.

All EM daemons are providers of a group called ha_em_peers. This group allows EM to synchronize the joining and leaving of EM daemons, which collectively form EM's distributed execution. When the *first* EM daemon requests to join the ha_em_peers group, the group does not actually exist. The daemon will create the ha_em_peers group and initialize the group state variable with the EMCDB version stored in the SDR.

Consider the case when an ha_em_peers group already exists. With reference to Figure 77 on page 132, EM is running and the ha_em_peers group's state variable contains the current operational version of EMCDB. Once the new EM daemon has successfully joined the ha_em_peers group, it must load the correct version of the EMCDB.

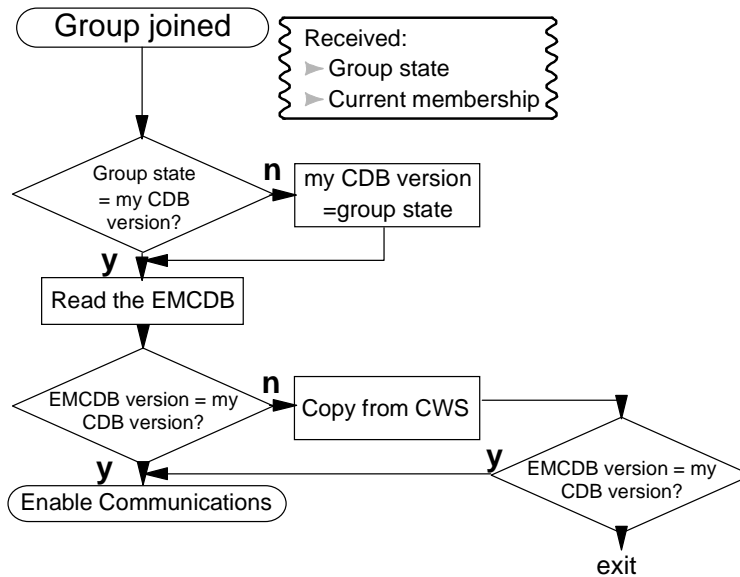


Figure 77. haemd Joining Group Services

From the group state value, it knows what version of EMCDB is being used by the already-running EM daemons. It compares the state value with the one it obtained from the SDR earlier in the initialization. If they do not match, the EM daemon uses the state variable value to remain consistent with the group. Now the EM daemon attempts to load the database from a local copy. It examines the EMCDB in the appropriate file to verify consistency with what the group is currently using. If the versions match, EM daemon loads the database and is ready to accept clients. If not, the EM daemon checks one more place: an EMCDB staging area on the CWS. If still no match, then the EM daemon is not configured and refuses any requests for event monitoring.

4.5.6.4 Resource Monitors - The SP Implementation

Figure 78 on page 134 summarizes the key elements of resource monitoring for EM. As you can see, EM is one busy little daemon. EM uses a number of RM, both internal (inside the EM daemon) and external (below the RMAPI line) to supply the data of the resource variables. The RMs will be discussed in more detail later; first we describe how the data of the different types of resource variables is made available to the EM daemon.

Resource variable data of type counter and quantity are placed in System Performance Measurement Interface (SPMI) shared memory. SPMI technology comes from the Performance Agent component (perfagent.server)

of Performance Toolbox/6000 (PTX/6000). The developers of HAI chose SPMI for two main reasons:

1. SPMI technology is kept in step with new releases of AIX. By simply making the correct version of `perfagent.server` (for the appropriate AIX level) a prerequisite for PSSP, HAI development avoids the laborious task of writing and maintaining their own mechanism for collecting AIX-level resource data.
2. SPMI allows multiple simultaneous clients: EM can use it for event monitoring, and PTX/6000 and Performance Toolbox Parallel Extensions PTPE/6000 (discussed in Section 5.6.2, “Performance Management Tools” on page 250) can use it at the same time for performance monitoring.

You monitor resource variables of type counter and quantity with respect to time. You control how you monitor a resource by associating it with a *resource class*. The resource class defines the following for a resource variable:

- The associated RM
- The observation interval, in seconds
- The reporting interval, in seconds, to the associated RM

The observation and reporting values may be different if the resource variable values are also being passed to PTPE/6000 and/or PTX/6000. For performance measurements, you may want to report the values more frequently, but for EM, there is no need to observe the values as often.

All resource variables that are located in shared memory with the same observation interval are observed on the same time boundary. This minimizes the observation overhead, no matter when the request for a resource variable is made.

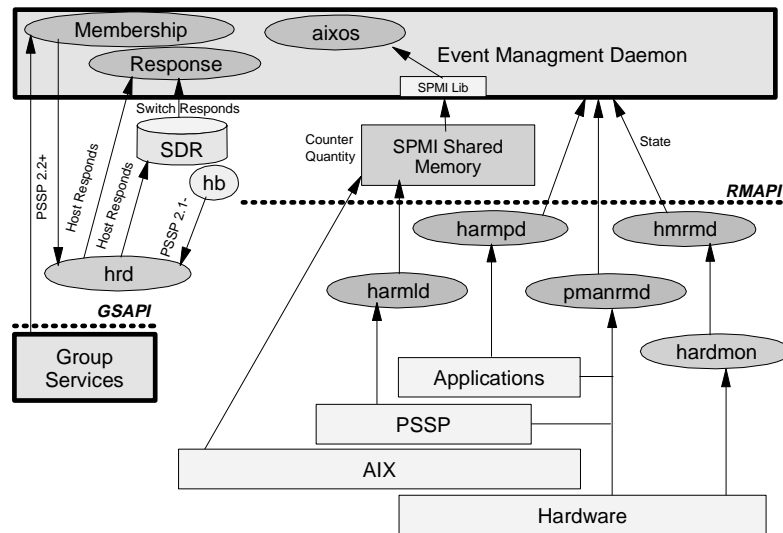


Figure 78. SP Resource Monitors

Resource variables of type state are handled differently. Every time they change, the resource monitor responsible for supplying the resource variable sends a message containing the data directly to the EM event processor. The rationale is that a state change may need to be acted upon immediately.

The EM daemon also subscribes to GS to receive data about node and adapter resource variables. The EM daemon is a data client and supplier of the Host_Responds daemon (hrd). Notice also that the SDR supplies switch responds data. On the SP, the EM daemon is truly versatile in obtaining data on its resource variables!

The SP implementation provides 370 defined resource variables. In the spirit of cross-platform HAI support, the names of resource variables and RMs are minimally prefixed with the vendor and product, so that uniqueness can be maintained across platforms.

We now examine the RMs.

External Resource Monitors

The SP ships with four external resource monitors that supply data to the EM daemon. Again with reference to Figure 78 on page 134, the four external RMs are:

- **IBM.PSSP.hmrmd**

This monitor provides the state of the SP hardware. The information is

obtained from the PSSP hardware monitoring subsystem (hardmon). The resource variables are of type state and sent directly to the EM daemon as a message.

- **IBM.PSSP.harmpd**

This monitor examines processes that are executing a particular application. The resource variables can be used to determine whether or not a particular system daemon is running. The resource variables are of type state and sent directly to the EM daemon as a message.

- **IBM.PSSP.harmlld**

This monitor provides switch, VSD, Loadleveler, processor online information, and internal variables. It uses SPMI, as it only reports on resource variables of type counter and quantity.

- **IBM.PSSP.pmanrmd**

This monitor supplies the resource variables of the Problem Management subsystem (PMAN). On your behalf, pmanrmd can execute a program, script, or command to report on various aspects of the system. The resource variables are of type state and sent directly to the EM daemon as a message.

Internal Resource Monitors

The three internal RMs are:

- **Membership**

This monitor supplies the resource variables that correspond to node and network adapter state. The information is obtained directly from GS by subscribing to the system groups hostMembership, enMembership, and cssMembership.

- **Response**

The Host_Responds daemon (hrd) obtains node states by using EMAPI. It receives events from the Membership monitor. This is fine for nodes at PSSP Version 2.2 or higher, but for earlier releases, HAI did not exist. For nodes on PSSP Version 2.1 or lower, hrd uses the PSSP heartbeat daemon (hb). From this composite of sources, hrd provides the Response monitor with host states and simultaneously updates the SDR host_responds class.

The Response monitor takes switch response states directly from the SDR.

- **aixos**

This monitor deals with resource variables of the AIX operating system (such as CPU usages - idle, kern, user, wait). It uses the SPMI to obtain data.

The EM daemon incorporates the SPMI Library to interface with SPMI shared memory. The SPMI library provides the means to directly query AIX structures (such as kernel and Logical Volume Manager) to supply data for a range of operating system resource variables. The aixos resource monitor calls the SPMI library, which by design places the AIX info data into SPMI shared memory. In one step, aixos supplies data for its resource variables and updates SPMI shared memory for PTX/6000 or PTPE/6000 to exploit.

You may wonder why these resource monitors are internal. Primarily, the reason is performance for membership and response monitors. For aixos, since the EM daemon takes data out of shared memory managed by the SPMI Library, it would be very inefficient to have an external resource monitor use the SPMI Library to get the AIX data, only to put it back into SPMI so EM could access it.

4.5.7 Putting It All Together - A Simple Example of Exploiting HAI

To illustrate many of the concepts in this section, we present a common scenario on the SP: after booting your nodes, how does their host_reponds state change to that satisfying green color on the Perspectives hardware monitor?

The example assumes a default SP installation, with PSSP Version 2.4 installed on each node.

4.5.7.1 Scenario

Your SP was turned off for the weekend due to scheduled power testing in the building. Early Monday morning (or Saturday morning, depending on your part of the world), you come into the office to restart the SP. You have already booted the CWS, and using Perspectives, are ready to boot the nodes. Your SP has one frame, four thin nodes, a switch, and one (default) partition. The hostnames and IP addresses for the system are as follows:

Slot	Reliable Host Name	en0	css0
1	node1	5.5.5.10	6.6.6.10
2	node2	5.5.5.20	6.6.6.20
3	node3	5.5.5.30	6.6.6.30

Slot	Reliable Host Name	en0	css0
4	node4	5.5.5.40	6.6.6.40
CWS	spcw	5.5.5.1	N/A

You start the nodes booting from their internal disks. You are staring at the Perspectives monitor, waiting for the icons representing your nodes to change from red to green.

4.5.7.2 TS

Since the CWS is already up, TS has already initialized node and adapter membership groups and created an NCT. The CWS is the only member of the en0 group and node group, and there is no internode connectivity in the NCT. The CWS is both the Group Leader and Crown Prince of the en0 ring, and from its machine list is aware there should be four other members. It is periodically issuing proclaims to the en0 IP addresses of the nodes, asking them to join its Singleton group.

The nodes begin processing the /etc/inittab entries. The TS daemons begin on each node and obtain configuration data from the SDR. After a flurry of proclaims, joins, distributions of topology, and ring mergers, we end up with the following adapter membership groups:

- en0: Group Leader is node4; Crown Prince is node3; members are node2, node1 and spcw.
- css0: Group Leader is node4; Crown Prince is node3; members are node2 and node1.

Figure 79 on page 138 shows the topology graph for en0 (Ethernet) and css0 (switch), both before and after an Estart command is issued to bring up the switch. The Group Leaders are designated GL. Notice that each switch adapter is in a Singleton group before the Estart. Heartbeating begins. Each TS daemon builds (or updates in the case of the CWS) its topology tables and NCT, puts the NCT in shared memory, then begins accepting requests from clients, namely GS.

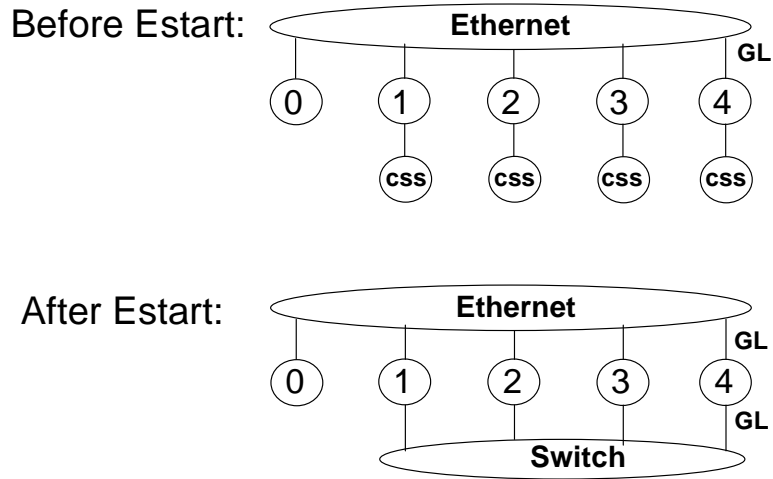


Figure 79. Topology Table for Scenario

4.5.7.3 GS

Before node booting, the CWS was already up and GS on the CWS had already created the internal node and adapter membership groups. The CWS is the only member of the en0 group and node group, and the switch group does not exist. The CWS is both the Group Leader of the en0 membership group and GS nameserver.

From processing /etc/inittab, GS daemons initialize on each node. The GS daemon determines the node number and partition it is running on, then connects to TS (if TS is not accepting connections yet, GS retries every second). Based on the information from TS, GS becomes aware of the other GS daemons that should be out there, and via NCT and Reliable Messaging, starts communicating with them. The CWS, with the lowest node number (0), is confirmed as the GS nameserver.

Since the CWS has the oldest GS daemon, it becomes Group Leader for the en0 and node membership internal groups. The first GS daemon on the node to issue a join request to the css0 internal group becomes the Group Leader of that group (in our case, say it was node3). Membership lists are maintained in the order of GS daemons joining the group. The providers of the groups are in effect the TS daemons running on each node. We have the following groups:

- en0 adapter group: Group Leader is CWS; other providers are node3, node4, node1, and node2. The membership list is maintained in the order of GS daemons joining the group.
- css0: Group Leader is node3; other providers are node4, node1 and node2.
- ha_em_peers: created by the CWS EM daemon, Group Leader is CWS, known only to the CWS GS daemon (which is the nameserver and GSAPI server for the EM daemon on the CWS).

The GS daemons now accept client requests. The hagsglsmd daemon on each node connects to its GS daemon.

4.5.7.4 EM

Before node booting, the CWS was already up and EM on the CWS had already created the ha_em_peers group. The CWS is the only member of the group, and has set the state variable to be the version of the EMCDB it retrieved from the SDR at initialization.

From processing /etc/inittab, EM daemons initialize on each node. Each determines its node number and fetches the version of the EMCDB from the SDR. The EM daemon will not accept any clients until it successfully joins the ha_em_peers group. EM tries to connect to GS every 5 seconds.

An EM daemon request to join the ha_em_peers group involves a 2-phase voting protocol. The first phase deals with group establishment and the second with actual membership request voting. Let us watch node3, the first node to connect to GS, join:

- node3 issues a join request to the ha_em_peers group. GS puts a vote request to the other group members including node3, in our case the CWS EM daemon is the only other member. The vote request contains the group state value, which could be empty if this is the first node, or could contain the EMCDB version number.
- In our case, the group state value contains the EMCDB version number. The CWS and node3 EM daemons vote *approve*. If the state variable was null, then this would be the first daemon to join. It would vote *approve*, and the group would be newly formed (this happened to the CWS).
- GS submits a second vote request to the group (second phase), this time as to whether node3 should be allowed to formally join the group. The node3 EM daemon, eager to join, votes APPROVE. The CWS EM daemon votes APPROVE, and the node3 is allowed to join. A join is refused only if a group member is still recovering from a prior termination of the joining

daemon. EM daemon will try to join ha_em_peers every 15 seconds until successful.

- The GS nameserver, the CWS GS daemon, updates its membership list for ha_em_peers.

After each EM daemon successfully joins ha_em_peers, it gets the proper version of the EMCDB as described in Section 4.5.6.3, “EM and GS” on page 131. Once this is done, EM accepts client requests.

At this point, we have TS, GS, and EM successfully initialized throughout the SP. The EM daemons’ internal Membership RMs update the IBM.PSSP.Membership.LANadapter.state resource variable from its connection to GS, which includes data on the state of the en0 and css0 interfaces by node. Happily, the news is good: TS is successfully heartbeating all en0 and css0 adapters, which GS learns of by subscribing to TS, which EM learns of by its Response monitor connecting to GS.

4.5.7.5 ES Client - Host_Responds Daemon

On the CWS, the Host_Responds subsystem (hrd) began just after TS was started. hrd runs only on the CWS. It has been patiently waiting for EM to fully initialize, and now connects to the EM daemon as a client.

Across the EM API, hrd periodically receives events from the EM daemon. The events describe the state of the en0 and css0 adapters for all nodes. Using this information, hrd supplies the IBM.PSSP.Response.Host.state resource variable in EM; in effect, hrd is both a client and resource monitor for EM.

4.5.7.6 Perspectives Shows Green!

Perspectives, an EM client, subscribes to events concerning Response.Host.state. It receives the events reporting the new state of the nodes, and turns their icons green.

4.6 SP Security

Most people want to believe that security is not an issue on the RS/6000 SP system because it has its own internal network and often is physically separated from the outside. But security is always an issue.

There is the story of the RS/6000 SP on the submarine: the captain of the ship thought that his system was safe, and that he need not worry about security at all. But when asked if he would let all the seamen know root's password, he was horrified because of the classified data that was on the system which only root had access to. Not caring about security is equivalent to letting everybody know root's password. Now he cares about the security on his system.

In this section, the basic mechanisms of RS/6000 SP security are introduced. The security related work for the system administrator is covered and Kerberos management and the sysctl application are presented. When the RS/6000 SP authentication system is AFS based, there are some additional issues that must be discussed.

4.6.1 General Security Concepts

Security is necessary to protect the system on the inside from being violated by intruders from the outside, and to prevent users on the inside doing bad things on the outside. There are often company rules about sensitive documents, and the security policy should ensure that this information is kept where it belongs. There must be some sort of barrier or fence to keep users and data where they each belong.

One of the earlier concepts of computer security is the security perimeter, developed in the mainframe age. The security perimeter is an imaginary line that separates the mainframe's CPU, memory and disks from the rest of the world. These three components are inside the security perimeter; everything else is outside. This perimeter-based security environment is easy to define: those on the outside cannot see the inside, those on the inside cannot see the outside, and they cannot see each other. This basic security concept is shown in Figure 80 on page 142.

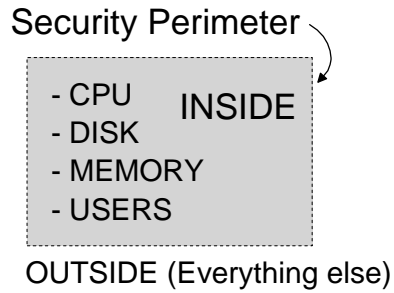


Figure 80. Security Perimeter

This concept is defined as the imaginary line that separates the mainframe's CPU, memory and disks from the "rest of the world". So all these components are "inside" the security perimeter. Everything else is outside. From a security point of view, in such an environment it is fairly easy to define what must be protected: the perimeter content from the outside and from the inside. The *perimeter security* prevents an entity from outside the perimeter from getting inside it, and the *inside security* prohibits the users inside to interfere with the outside or with one another.

Before going on, there is the need to introduce two security-related concepts: *identification* and *authentication*. Identification is the process by which an entity tells another who it is; authentication is the process by which the other entity verifies this identity. So the identification process is a two-phase process: telling the other party who you are and proving it by showing some sort of credential.

There is another related concept, namely authorization. Authorization is the process performed by an entity to check if an agent, whose identity has previously been authenticated, has or does not have the necessary privileges to carry out some action. This will be discussed later.

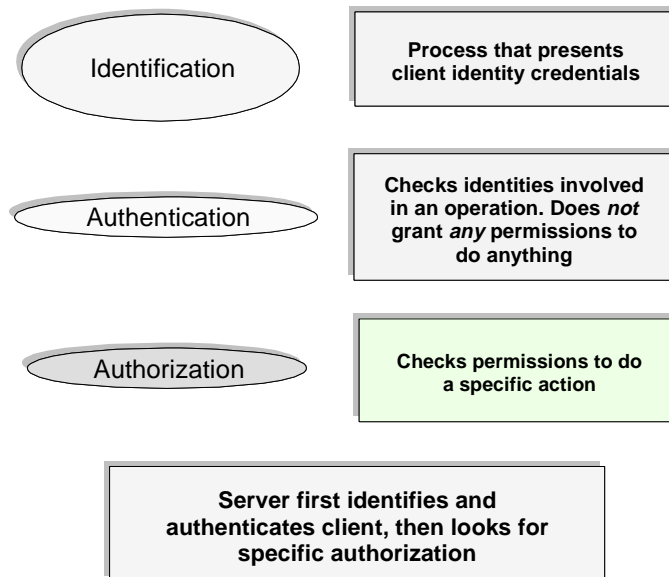


Figure 81. Authentication and Authorization.

So, going back to the security perimeter - authentication was implicit from the users' point of view: they sat down in front of a terminal *knowing* that the terminal was connected to the intended machine. If the line used to communicate with the mainframe was dial-up, then the users trusted the telephone company to connect them to the right machine. From the machine point of view, authentication was carried out using a *userid* and *password*, the former being the user identification and the latter the user credentials.

As can be seen, under this scheme everything is under control. Now what is going on in the client/server security arena? By client/server system is understood a set of networked machines. In this scenario, it is not easy to apply perimeter security because the perimeter concept is not clear. The biggest concern in this new scheme is the network. Why? Because every networked computer can be programmed to listen, receive and process *all the information* that travels across the link it is connected to. Furthermore, a machine can be programmed to send information across the network *impersonating* another machine. *Impersonation* means to assume another machine's identity or another user's identity. One way to protect the machines and users from being impersonated is to authenticate the packets when crossing the net - and it is in this area that Kerberos comes into play.

Another important flaw that the networked environment introduces is the fact that the information can be captured and registered, if it crosses the net in an uncyphered or decrypted format. The solution for this is to exchange packets of cyphered or encrypted data. The standard method to do this is the *Data Encryption Standard* (DES).

In an SP system there are, at least, two levels of security: AIX and PSSP. The next sections describe these and some additional SP security topics.

4.6.2 AIX Security

AIX is the operating system that supplies the basic infrastructure needed to carry out work. As such, it provides the basic security elements such as access control for user accounts, files and directories, devices, networks and so on. Even though AIX provides a reasonably secure environment (see the related redbook: *Elements of Security: AIX 4.1*, GG24-4433, for a full discussion of AIX security), it does not address the concerns exposed in the introduction, for example, it does not prevent impersonation and does not have encryption support. Also, it must be stated that this is not only true for AIX, but for every UNIX system. So it is necessary to add some elements to get the needed security.

4.6.3 Kerberos

Kerberos

Also spelled Cerberus - The watchdog of Hades, whose duty was to guard the entrance (against whom or what does not clearly appear) ... it is known to have had three heads.

- Ambrose Bierce, *The Enlarged Devil's Dictionary*

The problem that Kerberos solves has already been stated: authentication and exchange of information over an insecure network. When dealing with authentication and Kerberos, three entities are involved: the client, who is requesting service from a server; the second entity; and the *Key Distribution Center* (KDC) or Kerberos server. KDC is a machine that manages the database, where all the authentication data is kept and maintained. Basically, the records that make up the database are composed of the user name, user private key, user expiration date and other administrative data. Associated with the Kerberos server there is an administration server (who is in charge of the database administration), the authentication server (who does all the authentication work), data propagation programs that deal with the database

replication problem, and some utilities components. On the client side, there is an administrative client who provides database read-only services. As a server may well be a client of itself, this administrative client may also run on the server.

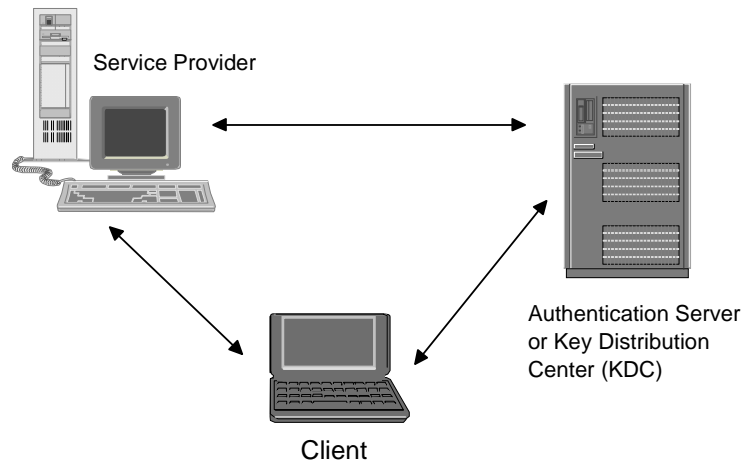


Figure 82. Entities Involved in a Third-Party Authentication System

This distributed authentication system was designed and developed at the Massachusetts Institute of Technology in the 1970s, as part of the Athena project.

As seen, Kerberos is a third-party system used to authenticate users or services, known to Kerberos as *principals*. The very first action to take regarding Kerberos and principals is to register the latter to the former. When this is done, Kerberos asks for a principal password, which is converted to a principal (user or service) 56-bit key using the DES algorithm. This key is stored in the Kerberos server database.

When a client needs the services of a server, the client must prove its identity to the server so that the server knows to whom it is talking. This is accomplished through the authentication mechanism provided by Kerberos, that is described next.

Tickets are the means the Kerberos server or the Key Distribution Center gives to clients to authenticate themselves to the service providers and get work done on their behalf on the service servers. Tickets have a finite life, known as the ticket life span.

In Kerberos terms, to make a Kerberos authenticated service provider work on behalf of a client is a three-step process:

- Get a ticket-granting ticket
- Get a service ticket
- Get the work done on the service provider

The main role of the *Ticket-Granting Ticket* service is to avoid unnecessary password traffic over the network, so the user should issue his password only once per session. What this ticket-granting ticket service does is to give the client systems a ticket that has a certain time span, whose purpose is to allow the clients to get service tickets to be used with other servers without the need to give them password every time they request services. It works as follows: the user or service "logs in" the authentication client, who is supposed to be known by the Kerberos server, and issues a request for the ticket-granting ticket service. One reason to do this is because the life span of the previous ticket has expired. The authentication client (AC) sends a message in plain text to the authentication server (AS) containing the user's name and the machine's AC name. See Figure 83 on page 146.

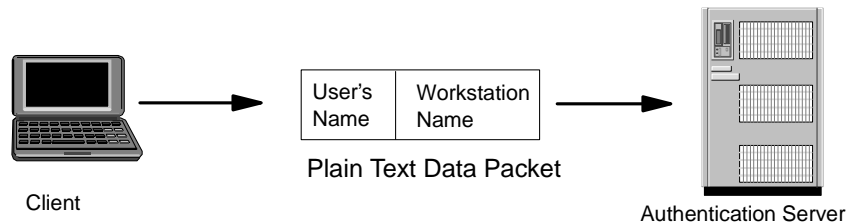


Figure 83. Client Logs into AS.

AS generates a data packet, known as the ticket-granting ticket, that is composed of:

- User's name
- Current time
- Ticket life-span
- Machine AC name
- Randomly generated DES key, known as the *session key*, AC - TGS key.

The purpose of the session key (AC - TGS key) is to serve as the encryption key for messages from the ticket-granting ticket service and the AC user

process. As will be seen shortly, every ticket has a session key, which is only stored in the AC and the service server (SS), not in the AS. In the special situation where the SS is the ticket-granting ticket server (TGS), AS and SS machines are the same and the session key is the common key between the AC user process and the TGS. For the sake of clarity, this session key is named AC - TGS session key. See Figure 84 on page 147.

User's Name	Current Time	Ticket Life-span	Machine AC name	Session Key
-------------	--------------	------------------	-----------------	-------------

Ticket-Granting Ticket encrypted with TGS key

Figure 84. Authentication Server Ticket-Granting Ticket

Next, the AS looks for the TGS secret key and encrypts the ticket with it. This way, only the TGS can read the ticket. Then, the AS appends the recently generated session key, the AC - TGS key, and makes up a message that is wholly encrypted with the AC user's secret key and sends it back to AC. This way, both the AC and the TGS know the AC - TGS session key. The TGS knows because it generated it and the AC user process also knows it because it received it from the TGS encrypted with its own secret key. See Figure 85 on page 147.

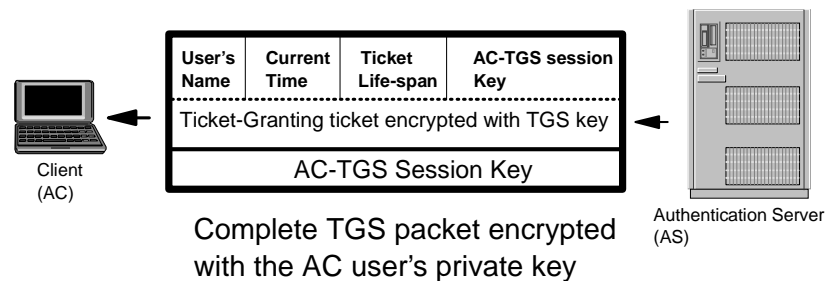


Figure 85. Complete Ticket-Granting Ticket Message from the AS to AC.

For the following explanation, refer to Figure 86 on page 148.

1. The AC receives the ticket-granting ticket packet from the AS.
2. The AC machine prompts the user for his password.
3. The password is transformed into a 56-bit DES key.
4. The AC machine tries to decypher the message with this key.
5. If it succeeds, then the process was really the AC user process. If not, it was another task and the process stops. So now this way, the

AC user process has a ticket-granting ticket that can be used to get tickets for other services.

It must be remembered that at this point, the AC user process can only know the AC - TGS session key and not the ticket-granting ticket because it does not know the ticket-granting ticket secret key to decrypt it.

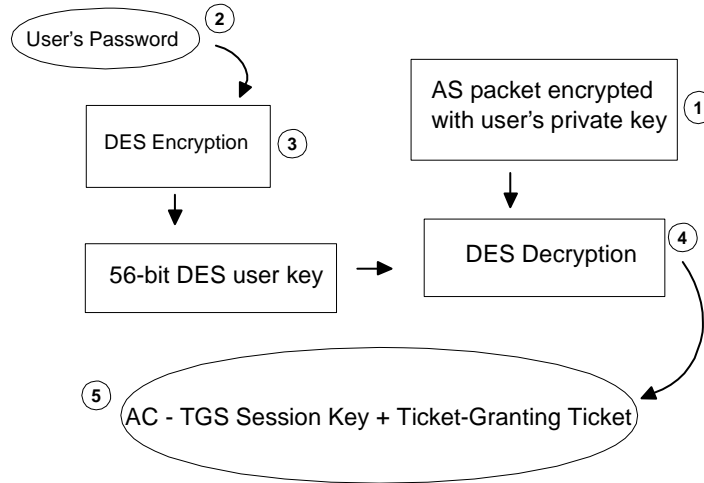


Figure 86. AC Receives TG Ticket and Session Key from KAS.

Of course, it is not necessary to get a new ticket-granting ticket every time there is a need for a Kerberos authenticated service. Once there is a ticket-granting ticket, it can be used until the end of its life span.

When the AC user process needs the services of the SSX server, it generates a packet, known as the *authenticator*, that contains the following information:

- Current time
- User's name or service name
- AC machine address

The AC encrypts this message with the session key it got from the ticket-granting ticket message exchange, the AC - TGS session key, and forwards it to the TGS along with the ticket-granting ticket. It must be remembered that the second part of this message, the ticket-granting ticket, is encrypted using the TGS secret key.

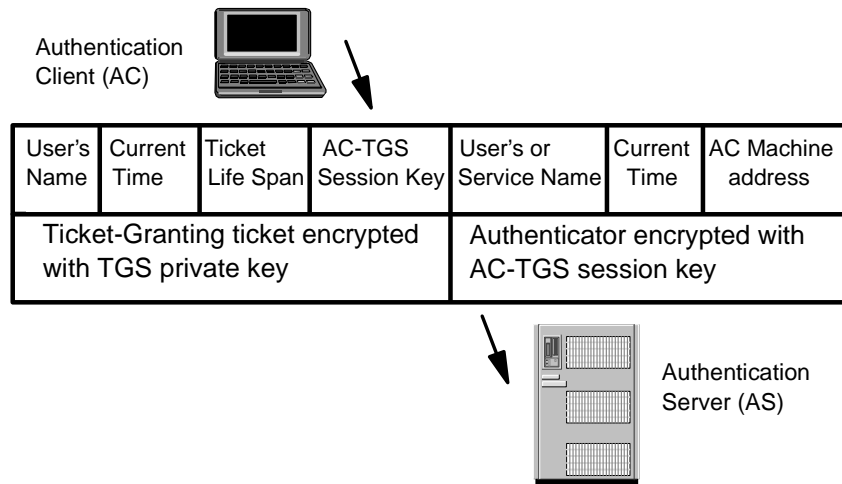


Figure 87. Authenticator and Ticket-Granting Ticket

The TGS decrypts the ticket with the TGS private key and now is able to retrieve the session key established at the beginning of the procedure, AC - TGS session key, and can validate the time stamps, the user name and the AC machine address that are retrieved from the decyphered authenticator and ticket.

Once everything is OK, TGS generates a different ticket, called the *service ticket*, to request server SSX services. This ticket, again, has the following content:

- User's name
- Current Time
- Ticket life span
- AC machine address
- SSX - AC session key, (56-bit DES key)

This new ticket, the service ticket, is encrypted using the SSX private key, so only SSX can read it. TGS appends to this encrypted ticket the SSX - AC session key and cyphers the whole packet with the AC - TGS session key, the session key gotten when the AC contacted TGS to get the ticket-granting ticket. So, when the AC user process receives this message, it is able to decypher it and has a valid service ticket and session key, the SSX - AC session key, to be used with server SSX.

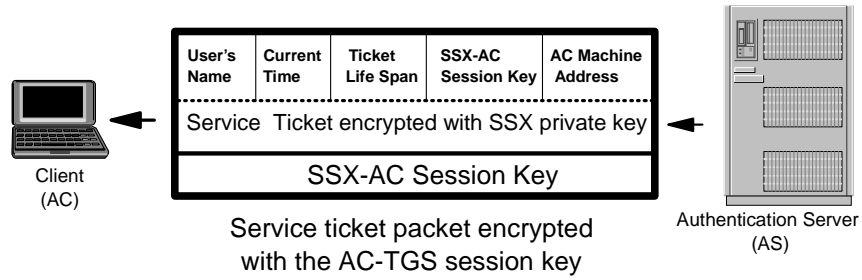


Figure 88. SSX Server Packet Ticket, Sent from AS to AC

The next step is to contact the SSX server and request the desired service. The AC user process creates another authenticator containing:

- Current Time
- User's name
- AC machine address

The AC machine sends a message to the SSX server containing the ticket it got from the TGS, which is encrypted with the SSX private 56-bit DES key, and the authenticator. The authenticator is cyphered with the SSX - AC session key.

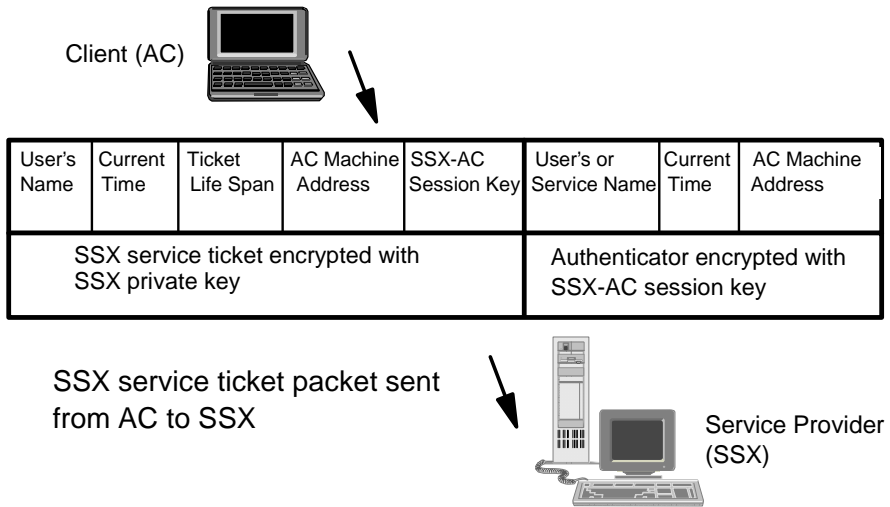


Figure 89. Service Packet Ticket Sent from AC to SSX.

Now the SSX gets this packet and decyphers the ticket it receives using its own private 56-bit DES key. From this key, it gets the SSX -AC session key, which it uses in turn to decrypt the authenticator, and to check the validity of the user's name, time stamp and AC machine address. If the check is successful, server SSX will perform the requested service on behalf of the AC machine. See Figure 90 on page 151.

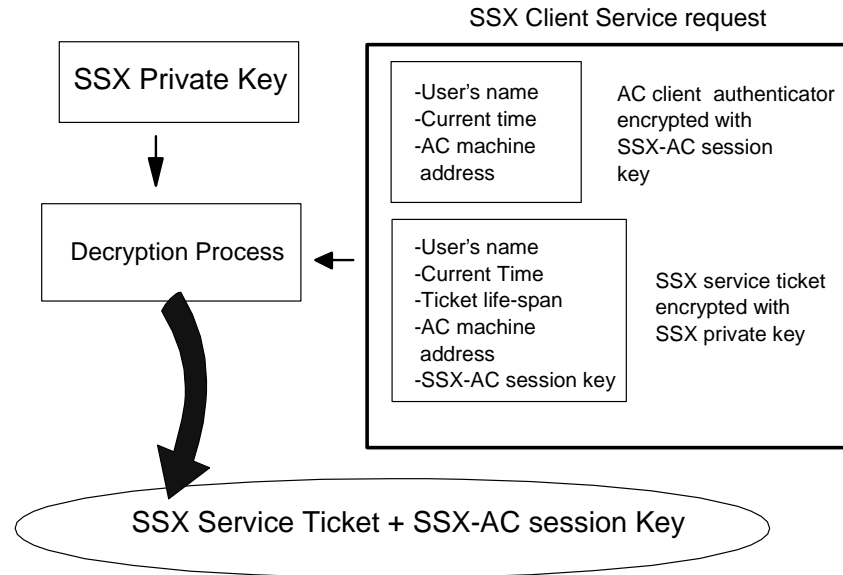


Figure 90. Service Provider Decypher Process

Because Kerberos uses DES to encrypt its messages, it makes it almost impossible for a third party to capture and make data readable over the network. And because of its authentication mechanism, it highly reduces the risk of impersonation.

In Kerberos terms, a *realm* is the domain an authentication server serves. At a minimum it is composed of a main server and a set of clients. Kerberos secondary servers can be used in case of a main server failure or for load balancing of the network. When there are secondary servers, the Kerberos database propagation software keeps the databases synchronized. The secondary databases are read-only and only the main server can update them.

As previously stated, a principal is a user or service known to Kerberos that can request a service that is authenticated by a Kerberos server. There is a method to provide subcategories to a Kerberos principal, using *instances*.

When dealing with users, a user instance represents the Kerberos authority granted to the user to, for example, carry out administrative tasks. When talking about services, the instance is the occurrence of the server. For example, there may be one global service that runs on every machine of a certain network. An instance of the server will be associated to each host.

Strictly speaking, a principal's name is a three-part name that represents a user or a service. The first part, known as the *primary name*, corresponds to the user or service name, the second is the instance and is a primary name modifier, and the third part is the realm name or Kerberos domain.

There are two Kerberos implementations in use: Version 4 and Version 5 are, unfortunately, incompatibles.

4.6.4 SP Security Management

4.6.4.1 Managing Kerberos Authentication

Kerberos authentication is the process by which an entity A proves its identity to an entity B using Kerberos authentication services. It will be assumed that A is an AIX user and B is a service provider of some sort. A number of steps must be followed for this process to be feasible:

- Both entities must be known to the Kerberos authentication server (KAS).
- A has to get credentials from the KAS to prove its identity to B.
- A has to deliver its credentials to B to be acknowledged.

The first step assumes that the KAS setup has already been done. This means that the basic structure (database, authentication administrator and services, database master password, KAS additional files and so on), is up and running. In the SP system this is done by the `setup_authent` command, used initially when the SP system is installed. `setup_authent` also creates some basic SP principals related to some SP services, like `hardmon` and `rcmd`, that are needed by the installation process itself.

The second step is accomplished by *adding principals* to the KAS database. It must be pointed out that all the KAS services are authenticated, so there is no plain text traffic over the network when these services are used. When adding principals, KAS creates an entry in the Kerberos database, specifically in `/var/kerberos/database/principal*`. After this is done, the new principal is known to KAS and has his private key, derived from his Kerberos password. This private key is stored in the `/etc/krb-srvtab` file in B's machine and in the KAS database. For the A user, the private key only gets stored in the KAS database.

For entity A, along with the registration process, it gets a ticket-granting ticket that gets stored in the file pointed to by the environment variable `KRBTKFILE`. If this variable does not exist, it is stored in `/tmp/tkt<uid>`, where `<uid>` is the AIX user A identification number. This file is known by the name *ticket cache file*. The lifetime of the ticket-granting ticket is the longest time available for a ticket: almost 30 days. In fact, it depends upon the command used to add the principal.

In the second step, user A gets a service ticket from KAS using the ticket-granting ticket from phase 1. This is automatically done when the user runs the client side of B, assuming it is a kerberized client. The service ticket, again, is stored in the file pointed to by `$KRBTKFILE` by an append operation. If this variable does not exist, it is appended to `/tmp/tkt<uid>`. All this activity is part of the KAS protocol and must be carried out by the kerberized client side of B, which runs on behalf of user A.

The third step is the shortest and easiest, because the B client side, user A, sends its request to sever B along with the service ticket and the encrypted authenticator recently built. The *authenticator* is decyphered by B using its own private DES key, which is stored in `/etc/krb-srvtab` and is able to retrieve the session key generated by KAS to be used in the A-B dialog. With the session key, it can decode the authenticator sent by the client and carry out the corresponding checks. Finally it executes the requested process and sends back to the user A client process the results encrypted with the A-B session key.

Kerberos Directories and Files: When managing SP Kerberos, a number of directories and files play an important role that must be understood to keep the KAS and Kerberos clients healthy; these are:

On KAS:

- `/.k`
- `$HOME/.klogin`
- `/etc/krb-srvtab`
- `/etc/krb.conf`
- `/etc/krb.realms`
- `/var/kerberos/database/*`
- `$KRBTKFILE` or `/tmp/tkt<uid>`

On the nodes:

- `$HOME/.klogin`

- /etc/krb-srvtab
- /etc/krb.conf
- /etc/krb.realms
- \$KRBTKFILE or /tmp/tkt<uid>

/.k: This is the master key cache file that holds the 56-DES key derived from the Kerberos master password, which is set initially when the KAS is set up. At any time, if the file gets lost or corrupted, it can be restored using the `kstash` command which, in turn, asks for the master password and stores it in the `/.k` file. Kerberos administration utilities and the `kadmind` daemon read the master key from this file, instead of prompting for it. If the file is lost or corrupts, Kerberos servers still work, but they won't be able to restart unattended.

\$/HOME/.klogin: This is a plain text file that holds a list of principals in the format `principal_name.instance@realm`. In this way, these principals get authorization to execute commands from another machine using the local account. It can be thought of as the equivalent of `$/HOME/.rhosts` when dealing with Berkeley `r`-commands.

\$/KRBTKFILE or /tmp/tkt<uid>: This is the user ticket cache file (<uid> refers to the user identification number) and contains the tickets owned by the user. In Kerberos terms, the user is an authentication client, created when the user executes the `kinit` command. The first ticket contained there is the ticket-granting ticket. Its contents can be shown using `klist` and it can be destroyed executing `kdestroy`. When the variable `KRBTKFILE` does not exist, the ticket cache file gets created in `/tmp` under the name `tkt<uid>`. If `KRBTKFILE` exists, then the ticket cache file is the one that is pointed to by it.

\$/etc/krb-srvtab: This is the binary server key file. It holds the names and private keys of the local instances of Kerberos authenticated servers. This file exists on CWS and on each node. For example, the CWS `/etc/krb-srvtab` file has an entry for `hardmon` and `rcmd` principals. On the nodes, at least the `rcmd` principal has to have an entry there. To list the contents of this file, the commands `klist -srvtab` or `ksrvutil list` can be used. Both commands show the key versions that are being used. The key versions must correspond to the versions that are held in the Kerberos database. The command `ext_srvtab` can be used to generate new server keys.

\$/etc/krb.conf: This is a plain text file that holds the local authentication realm (Kerberos server domain) and the Kerberos authentication servers for the local Kerberos domain. The first line defines the local realm name. The lines that follow specify the hostnames of the realm authentication servers. On the

primary authentication server, KAS, this file is set up by the `setup_authent` script.

/etc/krb.realms: This is a plain text file that holds a pairing between hostnames or network names and realms. It is used to record by which realm a host is being managed.

/var/kerberos/database: This directory contains binary files that make up the KAS database. The files `principal*` (the Kerberos database), and some ACL files that control the use of the `kadmin` command are kept in here. These files are set up by the `setup_authent` script.

To perform Kerberos management tasks, it is recommended to include in the `PATH` environment variable the directories that hold Kerberos commands. So it is desirable to set `PATH` to:

```
PATH=/usr/lpp/ssp/rcmd:$PATH:/usr/lpp/ssp/bin:/usr/lpp/ssp/kerberos/  
bin:/usr/lpp/ssp/kerberos/etc; export PATH
```

Also, if access is needed to the Kerberos commands man pages, it is useful to set the variable `MANPATH` to:

```
MANPATH=$MANPATH:/usr/lpp/ssp/man; export MANPATH
```

When the SP authentication system is based on Kerberos V4, which is supplied with the PSSP software, there are a number of tasks to be performed by the System Administrator (SA) to keep the system up and running. Among these tasks are:

- Managing Kerberos principals
- Managing Kerberos tickets
- Managing the Kerberos database
- Managing Kerberos daemons
- Managing secondary Kerberos authentication servers

4.6.4.2 Managing Kerberos Principals

As stated before, a Kerberos principal (KP) is a user known to KAS that can be a true AIX user and needs access to Kerberos authenticated services or the name of an authenticated service provider. The principal name can have an instance modifier, which, when the KP is a user, grants some different authority degrees to the user; or when it is a service, indicates in which host the service is running; and a realm modifier that identifies under which Kerberos domain the principal is being managed.

Note: Kerberos Principals

A Kerberos principal's name need not be an AIX user name. It can be *any* wanted name. It is not related to AIX users.

Regarding service providers running in a multiple network interface host, there must be an instance of the server per network interface. For example, if host *kaseg* has three network interfaces, say *en0*, *tr0* and *fi0*, then the *rcmd* principal may have three instances named *rcmd.kaseg_en0*, *rcmd.kaseg_tr0* and *rcmd.kaseg_fi0*, where *kaseg_en0*, *kaseg_tr0* and *kaseg_fi0* are the names associated with each of the IP addresses of each network interface.

When the SP system is installed, *setup_authent* creates the Kerberos database, generates the configuration files */etc/krb.conf* and */etc/krb.realms* unless they have been supplied, creates the *.k* file and the principals *rcmd*, with instances related to each network interface the CWS may have, and *hardmon* with random passwords in KAS. Then, the *setup_server* script completes Kerberos setup including *rcmd* principals, with the corresponding instances representing the machine network interfaces, in the KAS database for the nodes. It then updates the root's *\$HOME/.klogin* with node information so that some node's principals, such as *rcmd*, can access Kerberos-protected services on the KAS as the root user.

The *setup_authent* script also generates a KAS administrative user, *root.admin*, so the SP system has a Kerberos "account" to perform administrative tasks.

The main management tasks related to principals are to add, list, and delete principals, and change their attributes.

To add principals, there are number of commands, such as *kadmin* and its subcommand *add_new_key*, (*ank* for short), *kdb_edit*, *add_principal* and *mkkp*. The last one is non-interactive and does not provide the capability to set the principal's initial password, so this must be done with *kadmin* and the related subcommand *cpw*. But in general, what they do is to update the KAS database with the principal's name and private key derived from its Kerberos password. At the time the private key is set, the appropriate utilities get a ticket-granting ticket from the ticket-granting service and store it in the ticket cache file, *\$KRBTKFILE* or */tmp/tkt<uid>*.

A special situation arises when adding an authenticator administrator, the Kerberos administrator. The user who will be the Kerberos administrator must be added in the normal way, but with the *admin* instance name. Next the principal name must be added to one or more of the following files:

- /var/kerberos/database/admin_acl.add
- /var/kerberos/database/admin_acl.get
- /var/kerberos/database/admin_acl.mod

These files control Kerberos principals who have permission to add, list and modify, respectively, entries in the authentication database.

This way the new administrator will have database read/write permission and it granted Kerberos administrative command execution.

To list Kerberos principals, there is the command `lskp`. The following is an example of its output:

```
[sp2en0:/tmp]# lskp
KM          tkt-life: 30d      key-vers: 1  expires: 2037-12-31 23:59
changepw.kerberos  tkt-life: 30d      key-vers: 1  expires: 2037-12-31 23:59
default     tkt-life: 30d      key-vers: 1  expires: 2037-12-31 23:59
hardmon.sp2cw0  tkt-life: Unlimited key-vers: 1  expires: 2037-12-31 23:59
hardmon.sp2en0  tkt-life: Unlimited key-vers: 1  expires: 2037-12-31 23:59
hardmon.sp2en1  tkt-life: Unlimited key-vers: 1  expires: 2037-12-31 23:59
hardmon.sp2en2  tkt-life: Unlimited key-vers: 1  expires: 2037-12-31 23:59
krbtgt.MSC.ITSO.IBM.COM
             tkt-life: 30d      key-vers: 1  expires: 2037-12-31 23:59
rcmd.sp2cw0    tkt-life: Unlimited key-vers: 1  expires: 2037-12-31 23:59
rcmd.sp2en0    tkt-life: Unlimited key-vers: 1  expires: 2037-12-31 23:59
rcmd.sp2en1    tkt-life: Unlimited key-vers: 1  expires: 2037-12-31 23:59
rcmd.sp2en2    tkt-life: Unlimited key-vers: 1  expires: 2037-12-31 23:59
rcmd.sp2n01    tkt-life: Unlimited key-vers: 1  expires: 2037-12-31 23:59
rcmd.sp2n02    tkt-life: Unlimited key-vers: 1  expires: 2037-12-31 23:59
```

As can be seen, this command shows the principal's name, including its instance, the ticket lifespan, the key version number and the principal expiration date.

To delete principals, the command `rmkp` should be used. There is an alternate way to perform the same task. It consists of dumping the KAS database to a flat file, with `kdb_util dump Database_Dump_File`, editing the `Database_Dump_File` file, removing the principal line, and reloading the KAS database with `kdb_util load Database_Dump_file`. The last form overwrites the entire Kerberos database with the `Database_Dump_File` contents.

The principal attributes that can be changed are the password, (with `kpasswd`), the expiration date (with `kdb_edit` or `chkp -e`), and the ticket life time (with `chkp -l`). The value supplied to `chkp -l` is an integer in the 0 -

255 range, interpreted in a special way. For example, numbers ranging from 0 to 128 represent multiples of 5 minutes, so 126 means 10 hours and 30 minutes. The range from 129 to 255 represent lifetimes from 11 hours and 24 minutes to 30 days.

More about managing SP Kerberos principals can be found in *IBM Parallel Systems Support Programs for AIX: Administration Guide, GC23-3897*, chapter entitled "Security Features of the SP System".

4.6.4.3 Managing Kerberos Tickets

Tickets are the way a client authenticates itself to a Kerberos protected service provider. They represent the credentials delivered by a central authority, KAS, to identify the client to a server.

A ticket normally contains the client name, the name of the server that provides the needed service, the time when the ticket was issued, the ticket lifetime, the client machine IP address, and a session key to be used when the client and the server make a link. The ticket is encrypted with the server's private key.

When a principal is created and the password is set, a ticket-granting ticket, with a limited lifespan, is used to obtain service tickets to use with Kerberos-protected service providers.

During their life, tickets get stored on each user's cache file, `$KRBTKFILE` or `/tmp/tkt<uid>`.

Managing tickets deals with getting tickets, listing current tickets, changing tickets' attributes and deleting tickets.

Normally, the only ticket that has to be obtained manually is the ticket-granting ticket, because the service tickets are obtained automatically using the ticket-granting ticket. To get a ticket-granting ticket, the Kerberos principal must issue the `kinit` command, which gets and stores a ticket-granting ticket in the user's cache file with a default lifespan of 30 days, unless the `-l` flag is used.

To list the tickets, the `klist` command displays the user's cache file, showing the date the ticket was issued, its expiration date and the complete principal name to whom the ticket belongs. An example follows:

```
[sp2en0:/]# klist
Ticket file:      /tmp/tkt0
Principal:       root.admin@MSC.ITSO.IBM.COM
    Issued          Expires          Principal
```

```
Feb 13 14:31:26 Mar 15 14:31:26 krbtgt.MSC.ITSO.IBM.COM@MSC.ITSO.IBM.COM
Feb 13 14:31:27 Mar 15 14:31:27 hardmon.sp2en0@MSC.ITSO.IBM.COM
Feb 17 21:21:10 Mar 16 03:14:06 rcmd.sp2n16@MSC.ITSO.IBM.COM
[sp2en0:/]#
```

In the example, the root user has valid tickets to use ticket-granting tickets on the KAS, hardmon on host sp2en0 (the CWS), and rcmd on node sp2n16 within the realm MSC.ITSO.IBM.COM.

The ticket attributes that can be changed are its cache file and its lifetime. The cache file is changed by the `KRBTKFILE` environment variable. The lifetime gets modified by using `kinit -l`. `kinit` asks for the new lifespan in minutes.

`kdestroy` is the command to destroy a user's ticket. It writes zeros to the current ticket cache file and then destroys it, so the user's tickets are effectively destroyed. If for some reason, the ticket cache file is not found, `kdestroy` issues the message `No tickets destroyed`.

It is advisable to generate new tickets for every working session and to destroy them at the end of the working session. This way the SP Kerberos system will be kept healthy.

More about managing Kerberos tickets can be found in *IBM Parallel Systems Support Programs for AIX: Administration Guide*, GC23-3897, in the chapter entitled "Security Features of the SP System".

4.6.4.4 Managing the Kerberos Database

There are two main points to deal with when managing the KAS database: backing it up and changing its master password.

Backing up the KAS database: This task is easily accomplished by using the `kdb_util` command and its subcommand `dump`. When dumping the database, the `kdb_util` command creates a flat file that contains the KAS database. This file can be manipulated as text file, and can be edited to perform some specific tasks like deleting principals or checking the server key version. The reverse operation, restoring the database, is done with the same command, `kdb_util`, and the subcommand `load` (the argument is the file containing the database to be restored). When restoring the database, it must be remembered that the whole database will be overwritten by the `load` operation, so it is advisable to have a backup of the database, just in case.

Changing the KAS Database Master Password: Doing this often improves SP system security. When this operation is done, the system generates a new master key and then reencrypts the KAS database with this new key, which does not affect the principal's password. This operation is performed by

logging in to Kerberos as the admin principal used to set up authentication, (`kinit root.admin`), and using the `kdb_util` command and the subcommand `new_master_key`, specifying the name of a file where the command will dump the KAS database. After that, the database must be reloaded using the `kdb_util load` command, and the `/.k` file must be updated using `kstash`. Finally, Kerberos daemons must be stopped and restarted. If there are secondary authentication servers, the new database and the master key cache file must be transferred to them and the secondary Kerberos server daemons should be stopped and restarted to reread the information propagated to them.

More about managing the Kerberos database can be found in *IBM Parallel Systems Support Programs for AIX: Administration Guide*, GC23-3897, in the chapter entitled "Security Features of the SP System".

4.6.4.5 Managing Kerberos Daemons

There are three daemons that deal with Kerberos services:

kerberos: This daemon runs on the primary and secondary authentication servers. Its work is to process all the authentication requests coming from the clients. This means that it is in charge of getting ticket-granting and services tickets for the authentication clients. For load-balancing purposes, there can be more than one kerberos daemon running on the realm.

kadmind: This daemon is the authentication database server for the password-changing and administration tools, like `kpasswd`, `kadmin` and so on. It uses the master key for authorization and runs only on the primary authentication server. There can be only one kadmind daemon running on the primary authentication server.

kpropd: This daemon runs only on secondary authentication database server hosts, listening for a TCP connection on the `krb_prop` service. When it receives a request, it updates the Kerberos secondary server database.

All these daemons run under SRC control. They are started and stopped using the normal `startsrc` and `stopsrc` commands, but they can only be queried using signals.

More about managing Kerberos daemons can be found in *IBM Parallel Systems Support Programs for AIX: Administration Guide*, GC23-3897, chapter entitled "Security Features of the SP System".

4.6.4.6 Managing Secondary Authentication Servers

Secondary Kerberos authentication servers can provide two services to the realm they serve: primary Kerberos server backup and network load balancing, as any kind of secondary servers would. The daemons `kerberos` and `kpropd` run on these servers.

The tasks related to the Kerberos secondary servers are:

- Setting up a secondary Kerberos server
- Updating the Kerberos secondary server database
- Managing Kerberos secondary server daemons

Setting up a secondary Kerberos server: One of the first steps in this procedure is to update the `/etc/krb.conf` file on the primary authentication server, to reflect the existence of the secondary Kerberos server, and transfer it to the secondary server. Also, there must be a register in the secondary server that states to what realm a host belongs, so the `/etc/krb.realms` file must be copied to the secondary server. Also, the Kerberos `/usr/kerberos/etc/push-kprop` script must be activated on the primary KAS, if this is the first secondary server being set up. This is better accomplished through cron, making an entry for this script to run.

The complete procedure for setting up a Kerberos secondary server is the same as for setting up CWS as a secondary Kerberos authentication server, and can be found on *IBM Parallel Systems Support Programs for AIX: Installation and Migration Guide*, GC23-3898, in the chapter "Installing and Configuring the High Availability Control Workstation".

Updating Kerberos secondary server database: This is an automatic task performed by `kpropd` Kerberos daemon, that is always running on the secondary server. This daemon comes up when the secondary server boots and is under the SRC control, so it being monitored by it. This means that, if for any reason, it dies it will be automatically restarted by AIX System Resource Controller and there will be a log entry in the `/var/adm/SPlogs/kerberos/kpropd.log` file stating this fact.

Managing Kerberos secondary server daemons: The daemons that run on secondary servers are `kerberos` and `kpropd`, whose functions have already been explained. They run under SRC control, so they can be started and stopped with the `startsrc` and `stopsrc` commands, though they only support signal communication.

4.6.4.7 Keeping Kerberos Secure

The fact that an SP system uses Kerberos authentication services does not ensure that the system is secure. Some other precautions must be taken to have KAS secure. If this point is disregarded, one user may have unauthorized access to KAS and, for example, alter the database, steal private keys, add an unknown KAS administrator and so on, giving himself unauthorized and complete KAS access, which threatens the security of the SP system.

The basic security practice with every system, to start with, is to place the machine in a physically secure area.

Also, do not create ordinary user accounts on KAS and disable remote server access through Telnet, rlogin, ftp and tftp. This reduces security risks by only allowing authorized people to log in to KAS.

Configure and activate AIX security auditing to track down all security-relevant events. This will provide a system security log to be checked in case of troubles.

All Kerberos machine users' passwords, particularly that of root, must be kept in an extremely safe place. The use of authority delegation is strongly recommended to avoid abuse of the root account. For example, this can be accomplished by using the sysctl PSSP application, creating AIX users that can run commands and scripts on behalf of root but without explicitly giving them root authority.

Enforce rules to securely build good password so they cannot be guessed. Also, change Kerberos-related passwords often.

Some good references to Kerberos are:

PSSP Scalable POWERparallel Systems: PSSP Version 2.2 Technical Presentation, ITSO Poughkeepsie Center, SG24-4542.

IBM Parallel Systems Support Programs for AIX: Administration Guide, GC23-3897.

Kerberos: An Authentication Service for Open Networks Systems, Jennifer G. Steiner and Jeffrey I. Schiller, Project Athena, MIT, and Clifford Neuman, Department of Computer Science, Fr-35, University of Washington, March 30, 1988.

4.6.5 Andrew File System (AFS)

From a security point of view, AFS services are all authenticated using a Kerberos V4 compatible mechanism, so it is very reliable in security terms. An SP system can use an AFS Kerberos authentication server, but there are a number of differences to be recalled. AFS uses a different set of protocols, utilities, daemons and interfaces to manage the principal database. AFS uses its own tools, which are not compatible with the SP Kerberos V4 commands. Furthermore, AFS uses a different algorithm to transform the user's passwords into an 8-byte encryption key, so as a consequence the encryption keys generated by AFS and SP Kerberos are not the same, starting with the same user password. This has some implications for the cypher and decypher processes, regarding the programs involved, for example, they have different sets of commands to manage this process.

Even though AFS uses Kerberos V4 to authenticate its services, it is a two-way authentication mechanism because the client and the service provider must prove their identity to each other. And to control access to directories and files, AFS relies on ACLs.

AFS files are stored in volume structures that are located on the AFS server's disks. To improve performance, frequently accessed data may be replicated to secondary servers offering read-only access to this copy. This also improves network load because the cache managers that run on the AFS clients can use this replica to get the requested information in shorter time. And this leads to another AFS strength: robustness when servers crash, because the replicated data is available on secondary read-only servers and/or a client's local disk cache.

Additional material concerning AFS can be found at the following URL's:
<http://www.transarc.com/Product/AFS/FAQ/faq.html>,
<http://www.css.itd.umich.edu/docs/tutorial/AFS/> and in the white paper *Security on the RS/6000 SP System*, that can be found at
http://www.rs6000.ibm.com/resource/technology/sp_security.ps.

4.6.5.1 AFS Authentication Management

The SP authentication system supports AFS authentication services in two flavors: CWS may be an AFS server, or CWS may be an AFS client. Anyway, it is assumed that the AFS configuration work has already been done, both in the server and in the client, including the nodes themselves.

One important consideration to remember when working with AFS authentication services is that the SP system authentication package only

provides an interface to the AFS authentication server; it does not supply any AFS executable program.

Another important issue when dealing with AFS authentication servers, is the authentication services default administrator. At SP system authentication configuration time, running the `setup_authent` script, the authentication services administrator is the AFS authentication services administrator.

Because the AFS server used for authentication services must be Kerberos V4 based, there is great compatibility between the AFS authentication server and the SP authentication package. But there are also some differences in the implementation, mainly in the command interface used.

As said before, AFS provides its own set of commands and daemons. Among these are:

- *afsd*: A daemon used to connect AFS clients and server
- *kas*: The authentication administrator interface
- *klog.krb*: The user interface to get Kerberos tickets and AFS tokens
- *pts*: The AFS protection services administration interface:

`pts` subcommands are:

<code>adduser</code>	add a user to a group
<code>chown</code>	change ownership of a group
<code>creategroup</code>	create a new group
<code>delete</code>	delete a user or group from database
<code>examine</code>	examine an entry
<code>listowned</code>	list groups owned by an entry
<code>membership</code>	list membership of a user or group
<code>removeuser</code>	remove a user from a group
<code>setfields</code>	set fields for an entry

- *token.krb*: The command user interface to list current AFS tokens and Kerberos tickets

The main tasks an SP SA must deal with when AFS authentication services are used, are the following:

- Managing AFS principals
- Managing AFS tickets
- Managing the AFS database
- Managing AFS daemons
- Managing secondary AFS authentication servers

Most of these tasks should be performed using AFS commands, but PSSP software allows the use of some of its commands to perform the first two management tasks. The following PSSP software commands work in an AFS authenticated environment:

- `kinit`
- `klist`
- `kdestroy`
- `ksrvutil -afs change`
- `ksrvtgt`
- `rcmdtgt`
- `setup_authent`

To get more information about these AFS tasks, refer to the appropriate AFS documentation.

Managing AFS principals: Principals play the same role under AFS and Kerberos. The tasks involved here are adding, listing, deleting and changing the principal's attributes.

The main command involved in all these tasks is `kas`, which has corresponding subcommands to perform addition (`create`), list (`examine`), deletion (`delete`) and change attributes (`setfields`) of an AFS principal.

Managing AFS tickets: Tickets or AFS tokens are used in the same way as Kerberos tickets, and in most situations the `kinit` command can be used to get a ticket-granting ticket. AFS allows the use of the `klog.krb` command to get a ticket-granting ticket. The same is true for displaying the tickets held by a user: it can be done using `klist` or the AFS command `tokens.krb`. To delete user's tickets, the PSSP `kdestroy` command can be issued, but this will not delete the AFS Cache Manager service ticket. To do this, use the `unlog` AFS command.

4.6.6 Sysctl

Sysctl is an SP Kerberos-based client/server system designed to run commands remotely and in a parallel fashion with a high degree of *authorization*. The orders submitted to the sysctl system are run in parallel, if the request specified more than one machine where the command is to be executed, under root authority.

Root authority is dynamically assigned to the submitted processes. The submitter does not need to be root and does not need to know root's password. So sysctl allows the delegating of root privileges to non-root users,

based on their identity (which is fully authenticated by Kerberos), the work to be done, access control lists (ACL) and any other criteria. In this way, many of the classic root-performed tasks can be delegated to Kerberos principals without risking system security.

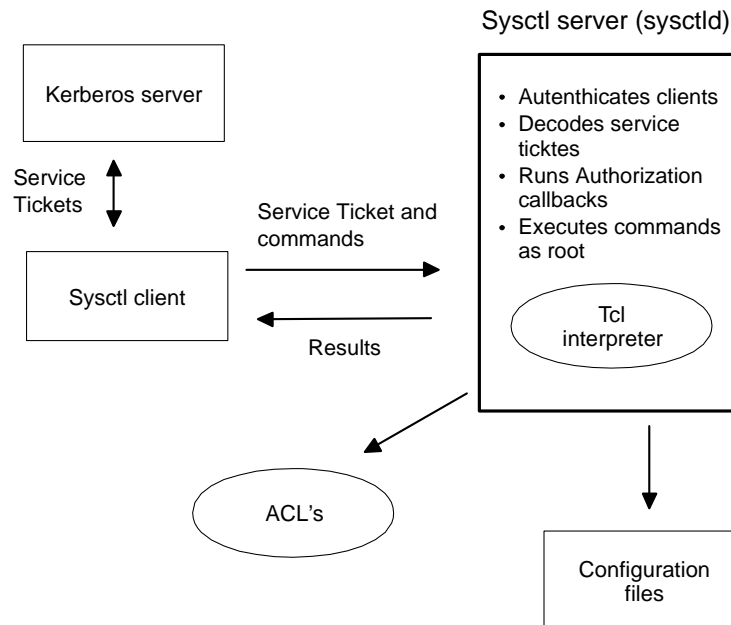


Figure 91. Sysctl Architecture.

From an architectural point of view, there is a sysctl server on each machine of the pool running as root, which provides the sysctl services to a client named sysctl. The client generates the requests that are to be performed and gets the authentication information from the SP authentication server (SPAS), then sends them in parallel to each of the machines where the requests will be executed. At each target machine, each sysctl server daemon, sysctld, performs the authentication checks according to the SP authentication protocol, checks the *authorization* that can be ACL and/or program based, executes the requests as root and sends back any stdout and stderr produced to the sysctl client.

Sysctl has its own command language, which is based on the *Tcl* and *Extended Tcl* interpreters. For details, see *Tcl and the Tk Toolkit*, John Ousterhout. Addison-Wesley, Reading, MA, 1994, ISBN 0-21-63337-X. The

sysctld daemon provides access to a Tcl-based set of commands that can be separated in to three classes:

- Base Tcl commands
- Built-in sysctl commands
- User-written scripts

Base Tcl commands: These are the basic Tcl interpreter commands. As every language, it has control commands, compute orders, I/O control, and commands that allow the execution of other commands, including externals such as AIX commands. Tcl commands that can be used by sysctl are defined in the `/etc/sysctl.conf` file.

Built-in sysctl commands: These are Tcl-based IBM-written applications ready to be used by the sysctl programmer. Examples of these are ACL processing commands (`acladd`, `aclcheck`, `aclcreate`, `acldelete` and so on), service commands (`svclogevent`, `svcrestart` and so on) and miscellaneous commands (`checkauth`, `listfs`, `quit`, `safeexec`, `whoami` and so on). For a detailed list of built-in sysctl commands, refer to *IBM Parallel System Support Programs for AIX: Administration Guide, GC23-3897*, in the chapter entitled "Sysctl".

User-written scripts: These are applications written using the base Tcl or built-in sysctl commands.

One of the strongest features of sysctl is its security capability, which can be divided in to the following components:

- Client authentication
- Client authorization

Client authentication is SP security based, that is the user who wants to run sysctl must be a known principal to the SP authentication system and must follow the normal procedures to have access to a protected service.

Client authorization is the phase where the sysctld server on the machine where the commands are about to run, checks to see if the user on behalf of whom it is trying to execute some procedure has the necessary authorization to do that work. There are two authorization mechanisms:

Authorization callbacks: This is a Tcl routine that uses a sysctl command, such as `aclcheck`, to check if the client has the necessary privileges to run the procedure it wants to run. The callback is executed just before running the request. If the callback succeeds, the procedure is run.

Access Control Lists: These are the normal AIX ACLs. The system administrator specifies a file where he grants authorization to specific users to execute some commands. There are two types of ACL files: global to a machine, `/etc/sysctl.acl`, and user-supplied. The latter are used to customize the machine's environment to some particular `sysctld` need, such as a more restrictive command execution policy than the global policy.

The `sysctl` subsystem has the following security components:

- Client authentication: Kerberos ticket
- Authorization callback: Tcl routine
- Access Control List: flat file

Every user wanting to use the `sysctld` services must be a Kerberos principal and get a `sysctld` service ticket. In this way the client authenticates with the `sysctld` server. So to grant basic access to `sysctld` services to a user, he or she must be a Kerberos principal. This is accomplished following the normal procedures to add a Kerberos principal.

Next, the client needs to be authorized to execute the intended command. The first authorization level is the callback mechanism, which is checked just before the command is run. The callback is a procedure, written in Tcl, that returns a Tcl error if the client is not authorized to run the procedure. The following callbacks are built into the `sysctl` server:

NONE: This callback always returns a successful status when called. This means that no special authorization is needed to run a command that has this callback associated. Even users that are not Kerberos principals can execute the procedure.

AUTH: Any client that has been successfully authenticated by KAS can run the command associated with this callback.

ACL: To be able to run the associated procedure, the client must be listed in the ACL file that can optionally be specified with this callback. If there is no ACL file as the ACL callback argument, then the file `/etc/sysctl.conf`, the `sysctl` server global ACL file, is used.

SYSTEM: This type of callback has been designed to be used when the programmer does not want to grant execution permission to certain procedures from the command line or from an interactive `sysctl` session. If a user tries to execute a procedure that has this callback associated from the command line or from a `sysctl` interactive session, the callback itself will always return a Tcl error, so the procedure will not run. This, effectively,

bypasses the callback and the associated procedure can only be executed from within the sysctl server, sysctld. There are three situations when the callbacks are bypassed:

- sysctld reads configuration files (sysctld startup)
- sysctld executes an authorization callback
- sysctld executes the body of a procedure that a user was granted authority to run

Summarizing, SYSTEM callback only allows the procedure to execute from within the sysctl server, sysctld.

sysctl configuration files: There are two main configuration files related to sysctl: /etc/sysctl.conf and /etc/sysctl.acl.

/etc/sysctl.conf: The sysctl.conf file configures the local sysctl server daemon by optionally creating variables, procedures and classes, setting variables, loading shared libraries, and executing sysctl commands. These items are used by the server in the processing of requests from local and remote sysctl clients. The default location of this file is /etc/sysctl.conf. An alternate configuration file location can be specified by using the -f flag when starting the server. The /etc/sysctl.conf file contains sysctl commands that are read and executed by the sysctl server during its initialization. More about this file in the corresponding man page.

As an example, part of /etc/sysctl.conf is shown as follows:

```
create var buildTop /usr/lpp/ssp AUTH
```

This line creates the variable buildTop, assigns it a value of /usr/lpp/ssp, and links it to an authorization callback of AUTH. The variable buildTop can be referenced within sysctl commands and procedures.

There must be a /etc/sysctl.conf file on every machine that runs the sysctld daemon.

/etc/sysctl.acl: The sysctl.acl file contains the list of users authorized to access objects that are assigned the ACL authorization callback. Each line of the file must conform to one of the following formats:

Comment: A line that begins with a number sign (#) is considered a comment.

Principal name: A line of the form "_PRINCIPAL principal_name" defines a principal on the ACL. If principal_name is not fully qualified (for example, it

does not contain an instance or realm separator), the realm defaults to the local realm and the instance defaults to NULL.

ACL file: A line of the form "_ACL_FILE file_name" references another ACL file, the contents of which are included in this file.

The first line of this file (or any sysctl ACL file) must begin with the line:

```
#acl#
```

An example of an /etc/sysctl.acl file is:

```
#acl#
_PRINCIPAL root.admin@MSC.ITSO.IBM.COM
_ACL_FILE /etc/mycmd.sysctl.acl
```

The first line authorizes the Kerberos principal root.admin, when it has a valid sysctl service ticket issued in MSC.ITSO.IBM.COM realm to access objects (such as commands) that have the associated callback ACL. The next line tells sysctld to include the /etc/mycmd.sysctl.acl file in this file, so any principal listed there will also have access to those objects that have the ACL callback associated.

It must be remembered that the ACL callbacks can also have particular ACL files as arguments. These files have the same syntax as the /etc/sysctl.acl file. When a command has an associated ACL callback that has an ACL file as an argument, all the users that should execute that command must be included in the ACL file. This is also true for root.admin.

Using sysctl: Before using sysctl, the user must have a Kerberos principal, and the sysctld must be running and properly configured on the target machine. There are two ways to run sysctl: interactive and non-interactive.

Interactive mode: This mode is invoked by `sysctl` typing at the shell prompt. The `sysctl` command answers with:

```
Sysctl (Version 1.1) on sp2en0.msc.itso.ibm.com
sysctl>
```

At this point, the system is ready to execute commands:

```
sysctl> svcversion
1.1
sysctl> df
sp2en0.msc.itso.ibm.com:
Filesystem                Size(KB)  Used(KB)  Free(KB)
/                          32768     12784    19984
```

```

/usr          593920    589860    4060
/var          81920     25652     56268
/tmp          81920     43580     38340
/home        155648    58592     97056
/tftpboot    28672     19392     9280
/spdata      2506752   818036    1688716
/notesbench  53248     9540      43708
/hari        40960     13968     26992
/usr/lpp/info/lib/en_US 102400    90280     12120
/pgc        2048000   269352    1778648
sysctl>

```

In the example, the commands `svcversion` and `df` were executed.

Non-interactive mode: This mode is invoked by typing `sysctl` at the shell prompt, followed by the `-h` flag and the name of a remote host and/or the name of the command to be executed, as in the following example:

```

[sp2n0:/etc]# sysctl -h sp2n09 sys:info
sp2n09.msc.itso.ibm.com AIX 4.2

```

This example shows how to execute the command `sys:info` on node `sp2n09` to retrieve the nodename and its AIX version and release.

A command can be issued in parallel to a series of machines following the `sysctl` command with multiple `-h` flags, each having the machine name as argument.

sysctl scripts: Included in the `/etc/sysctl.conf` `sysctl` configuration file there are a number of ready-to-use `sysctl` scripts that are helpful in SP system administration. Some of them are shown in Table 8.

Table 8. List of `sysctl` Scripts included in the `/etc/sysctl.conf` File

Function	File containing the scripts
Parallel file system commands (pdf and pfck)	<code>/usr/lpp/ssp/sysctl/bin/pdfpfck.cmds</code>
Process management commands (pfps cmds)	<code>/usr/lpp/ssp/sysctl/bin/pfps.cmds</code>
Problem management commands	<code>/usr/lpp/ssp/sysctl/bin/pman.cmds</code>
CWS root commands	<code>/usr/lpp/ssp/sysctl/bin/cwsroot.cmds</code>
Log management commands	<code>/usr/lpp/ssp/sysctl/bin/logmgt.cmds</code>
Kerberos database management commands	<code>/usr/lpp/ssp/sysctl/bin/kdbadmin.cmds</code>
VSD <code>sysctl_vsd.cmd</code> commands	<code>/usr/lpp/csd/sysctl_vsd.cmds</code>

More on sysctl can be found in *IBM Parallel System Support Programs for AIX: Administration Guide*, GC23-3897, in the chapter entitled "Sysctl".

4.7 Parallel I/O

A serial application can exceed the I/O capacity of a single node, even after the application's data has been spread across many disks and I/O adapters of the node. Parallel applications need access to data that is spread across many nodes to improve I/O performance. Ideally, any node should be able to access data residing on any other node or node group, to offer the customer flexibility in where applications are installed. In any case, high availability of critical data is crucial.

How does I/O capacity scale to ultra-large size on the SP? How does the SP offer flexible and highly-available I/O configurations?

4.7.1 Network File System

Network File System (NFS) offers a potential solution. As a base component of AIX, NFS is widely used on the SP. In fact, NFS is a base technology exploited by the node installation and customizing processes. It is very flexible, easy to configure, and with products like IBM's High-Availability NFS (HANFS) can be made more available.

NFS has severe restrictions with respect to scalability and performance. An NFS server can reside only on a single node, so we still have the problem of exceeding the I/O capacity of a single node. NFS is not a high-performance data sharing solution, although NFS Version 3 offers good improvements, particularly with respect to write performance.

4.7.2 Andrew File System and Distributed File System

Andrew File System (AFS), from Transarc, and Distributed File System (DFS), from the Open Software Foundation (OSF) offer more secure, manageable file system implementations, compared to NFS. They offer little more in the way of I/O scalability and performance, as they fundamentally remain mechanisms for distributed data access.

4.7.3 Virtual Shared Disk

Virtual Shared Disk (VSD) allows data in logical volumes on disks physically connected to one node to be transparently accessed by other nodes. Importantly, VSD supports only raw logical volumes, not file systems. VSD is in the ssp.vsd fileset of PSSP.

IBM developed VSD to enable Oracle's parallel database on the SP. Oracle's database architecture is strongly centralized. Any processing element, or node, must be able to "see" the entire database. In the case of the parallel implementation of Oracle, all nodes must have access to all disks of the database, regardless of where those disk are physically attached.

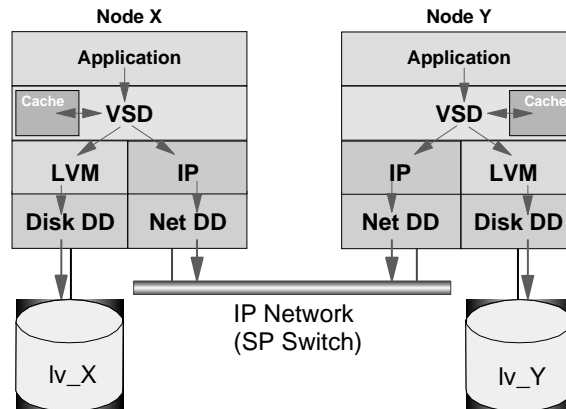


Figure 92. VSD Architecture

With reference to Figure 92 on page 173, imagine two nodes, Node X and Node Y, running the same application. The nodes are connected by switch and have locally-attached disks. On Node X's disk resides a volume group containing the raw logical volume lv_X. Similarly, Node Y has lv_Y. For the sake of illustration, let us suppose that together lv_X and lv_Y constitute an Oracle Parallel Server database to which the application on each node makes I/O requests.

The application on Node X requests a piece of data in the database. After the node's Virtual Memory Manager (VMM) determines that the data is not in memory, it talks not to the regular Logical Volume Manager (LVM) but rather to the VSD device driver. The VSD device driver is loaded as a kernel extension. Thus VSDs configured in the SP are known to the appropriate nodes at the kernel level.

The VSD device driver can fetch the data from one of three places:

1. VSD cache, if the data is still there from previous requests. VSD cache is shared by all VSDs configured on a node. Data is stored in 4KB blocks, a size optimized for Oracle Parallel Server. If your I/O patterns involve I/O operations larger than 4KB, we recommend disabling VSD cache because its management becomes counterproductive.

2. lv_X, in which case the VSD device driver exploits Node X's normal LVM and Disk Device Driver (Disk DD) pathway to fetch the data.
3. lv_Y, in which case the VSD device driver issues the request through the IP and Network Device Driver (Net DD) pathway to access Node Y. For performance, VSD uses its own stripped-down IP protocol. Once the request is passed up through Node Y's Net DD and IP layers, Node Y's VSD device driver accesses the data either from VSD cache or from lv_Y.

The VSD server node uses the *buddy buffer* to temporarily store data for I/O operations originating at a client node and to handle requests that are greater than the IP message size. In contrast to the data in the cache buffer, the data in a buddy buffer is purged immediately after the I/O operation completes. Buddy buffers are used only when a shortage in the switch buffer pool occurs or on certain networks with small IP message sizes (for example, Ethernet). The maximum and minimum size for the buddy buffer must be defined when the VSD is created. For best performance, you must ensure that your buddy buffer limits accommodate your I/O transaction sizes to minimize the packetizing workload of the VSD protocol. Buddy buffers are discussed in detail in *IBM Parallel System Support Programs for AIX Managing Shared Disks*, SA22-7279.

The VSDs in this scenario are mapped to the raw logical volumes lv_X and lv_Y. Node X is a client of Node Y's VSD, and vice versa. Node X is also a direct client of its own VSD (lv_X), and Node Y is a direct client of VSD lv_Y. VSD configuration is flexible. An interesting property of the architecture is that a node can be a client of any other node's VSD(s), with no dependency on that client node owning a VSD itself. You could set up three nodes with powerful I/O capacity to be VSD servers, and ten application nodes, with no disk other than for AIX, PSSP, and the application executables, as clients of the VSDs on these server nodes.

VSDs are defined in the SDR and managed by either SP SMIT panels or the VSD Perspective. Part 5.1, "SP Monitors". VSDs can be in one of five states as shown in Figure 93 on page 175.

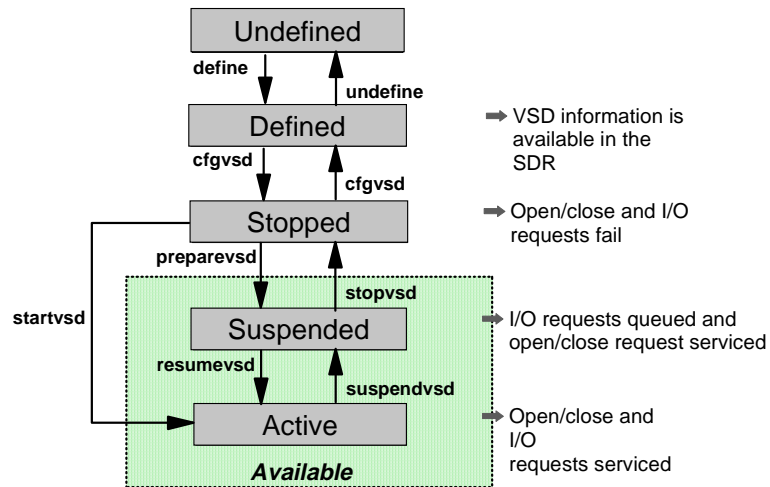


Figure 93. VSD State Transitions

This figure shows the possible states of a VSD and the commands used to move between states. VSD configuration changes, or manual recovery of a failed VSD, require you to move the VSD between various states.

The distributed data access aspect of VSD scales well. The SP Switch itself provides a very high-bandwidth, scalable interconnect between VSD clients and servers, while the VSD layers of code are efficient. The performance impact of servicing a local I/O request through VSD relative to the normal VMM/LVM pathway is very small. IBM supports any IP network for VSD, but we recommend the switch for performance.

VSD provides distributed data access, but not a locking mechanism to preserve data integrity. A separate product such as Oracle Parallel Server must provide the global locking mechanism.

4.7.4 Hashed Shared Disk

While VSD enables efficient distributed access to raw logical volumes, the performance of any VSD is restricted by the I/O capacity of the VSD server node. To improve I/O bandwidth, IBM provides Hashed Shared Disk (HSD) in base PSSP. HSD is a striped version of VSD. If the I/O load on a specific VSD is too heavy, you can use HSD to distribute the load across other VSDs and nodes.

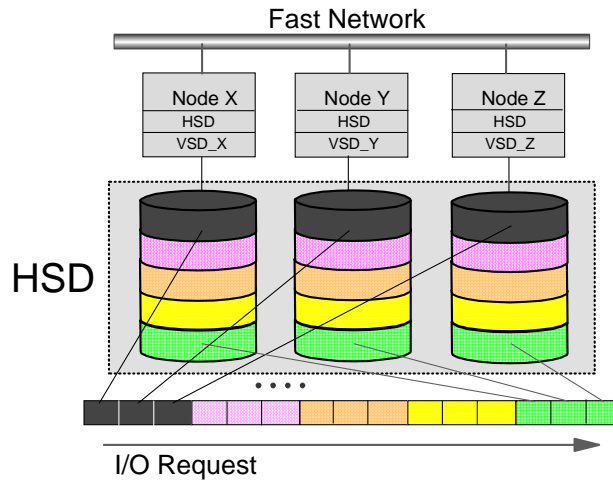


Figure 94. HSD Architecture and Function

HSD automatically distributes data across VSDs on different nodes. The HSD driver sits above the VSD layer. With reference to Figure 94 on page 176, Nodes X, Y, and Z each have a VSD (VSD_X, VSD_Y, and VSD_Z, respectively). When an application issues a write request, HSD breaks the I/O transaction into a smaller number of I/O operations, depending on the number of VSDs defined to the HSD and the stripe size. The stripe size defines the maximum size of a block of data to be stored on a single VSD in an I/O operation. HSD uses a hash function to distribute the I/O operations across the VSDs. In our case, a write transaction on Node X would result in several I/O operations spread across VSD_X, VSD_Y, and VSD_Z.

HSD offers improved I/O bandwidth for applications, but is tricky to set up and difficult to change afterwards. For example, if you wish to define a new VSD on a new node to your HSD set, you must back up all the existing HSD data, redefine the HSD to include the new VSD, and reload the data. Oracle Parallel Server has its own data striping mechanism, so HSD is not widely used.

4.7.5 Recoverable Virtual Shared Disk

Recoverable Virtual Shared Disk (RVSD) adds availability to VSD. RVSD allows you to twin-tail disks, that is, physically connect the same group of disks to two or more nodes, and provide transparent failover of VSDs among the nodes. RVSD is a separately-priced IBM LPP.

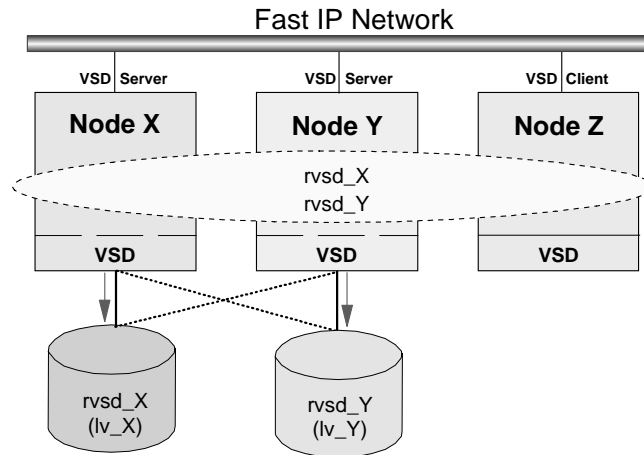


Figure 95. RVSD Function

With reference to Figure 95 on page 177, Nodes X, Y, and Z form a group of nodes using VSD. RVSD is installed on Nodes X and Y to protect VSDs `rvsd_X` and `rvsd_Y`. Nodes X and Y physically connect to each other's disk subsystem where the VSDs reside. Node X is the primary server for `rvsd_X` and the secondary server for `rvsd_Y`, and vice versa for Node Y. Should Node X fail, RVSD will automatically failover `rvsd_X` to Node Y. Node Y will take ownership of the disks, vary-on the volume group containing `rvsd_X` and make the VSD available. Node Y serves both `rvsd_X` and `rvsd_Y`. Any I/O operation that was in progress, and new I/O operations against `rvsd_X`, are suspended until failover is complete. When Node X is repaired and rebooted, RVSD switches the `rvsd_X` back to its primary, Node X.

RVSD subsystems are shown in Figure 96 on page 178. The `rvsd` daemon controls recovery. It invokes the recovery scripts whenever there is a change in the group membership. When a failure occurs, the `rvsd` daemon notifies all surviving providers in the RVSD node group, so they can begin recovery. Communication adapter failures are treated the same as node failures.

The `hc` daemon is also called the Connection Manager. It supports the development of recoverable applications. The `hc` daemon maintains a membership list of the nodes that are currently running `hc` daemons and an incarnation number that is changed every time the membership list changes. The `hc` daemon shadows the `rvsd` daemon; recording the same changes in state and management of VSD that `rvsd` records. The difference is that `hc` only records these changes after `rvsd` processes them, to assure that RVSD

recovery activities begin and complete before the recovery of hc client applications takes place. This serialization helps ensure data integrity.

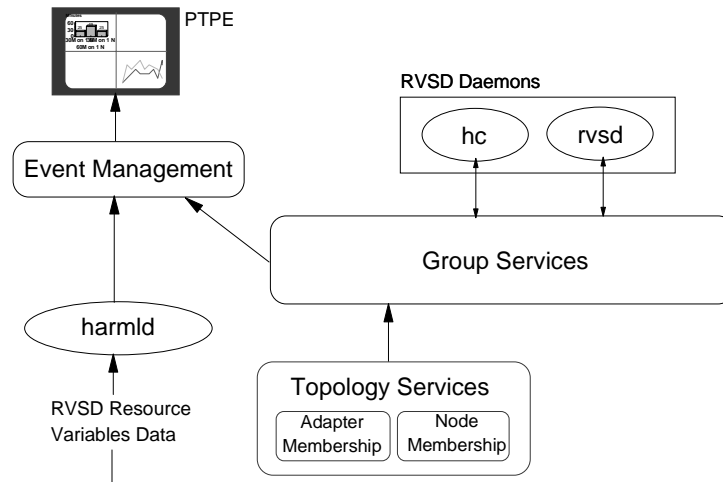


Figure 96. RVSD Subsystems and HAI

4.7.6 Parallel Input/Output File System

VSD, HSD and RVSD improve the flexibility, performance, and availability of access to large amounts of data on the SP, but only for raw logical volumes. What about applications that need to talk to a file system?

IBM's first parallel file system for the SP was Parallel Input/Output File System (PIOFS). We need to be careful here with the two base concepts of PIOFS: parallel, and file system. With reference to Figure 97 on page 179, we can see how parallelism is achieved in PIOFS. When a client node requests data in a PIOFS file system, it is asking for a service from a PIOFS server. The PIOFS server, typically comprised several server nodes, presents the file system as one entity to the clients.

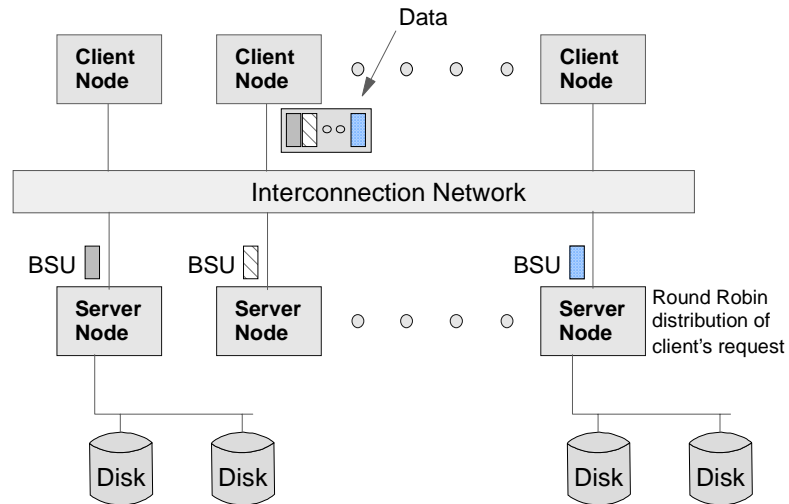


Figure 97. PIOFS Design

When a client creates a data file on a PIOFS server, the file is striped across the server nodes in chunks with the size defined by the Basic Striping Unit (BSU). The BSU default size is 64KB, but can be increased to provide higher bandwidth transfers. The data is striped across the server nodes using a round-robin algorithm. PIOFS further stripes data across the disks supporting the PIOFS file system on each server node. PIOFS supports 64-bit addressing; thus, files greater than 2GB and very large file systems are supported. Each server node can support up to 256GB of file space. Since a PIOFS server can span up to 512 nodes, PIOFS offers ultra-large parallel I/O scalability. The primary candidates for PIOFS are applications that use large sequential files.

Files less than 2GB can be loaded into a PIOFS file system with standard UNIX commands such as `cp`, `dd`, and `mv`. For files greater than 2GB, you can use the same commands if you have AIX V4.2 or later. Otherwise, specific procedures are necessary to read and write between AIX and PIOFS file systems.

PIOFS uses UDP/IP and can be implemented over any IP-capable network adapter. Most commonly, the SP Switch is used as the interconnection network. Servers must use the same adapter types; you cannot have some nodes switch-attached and some Ethernet-attached.

So PIOFS is parallel, but what about the file system aspect? Implemented as an AIX Virtual File System, PIOFS can coexist with other AIX file systems on AIX-based client nodes and server nodes. PIOFS supports both sequential and random file access, and users have access to the parallel files via *most* POSIX functions, but PIOFS is not a standard file system. Its unique aspects offer advantages and disadvantages:

- **Checkpointed file versions.** After a program checkpoints a file, all file changes are made to a new block of storage. The old storage blocks are kept unchanged. Two file versions, active and optionally checkpoint, may exist. When a file is deleted, the checkpoint version is also deleted.
- **Logically partitionable files and views.** An application can achieve even greater parallelism by logically dividing the PIOFS file into multiple subfiles, or views, to be processed simultaneously by different parts of the application. It is possible to dynamically specify subfilings that corresponds to specific rows, columns, or blocks of a file. Parallel access to data is achieved without the overhead of managing multiple data files, or changing or moving the PIOFS file.
- **Not fully POSIX-standard.** PIOFS does not conform to all of the ISO/IEC 9945-1 POSIX standard. For example, explicit locking of portions of files is not supported, and the 64-bit unsigned offsets used for large files are not supported by POSIX.
- **Poor availability.** If any disk or any server node in a Parallel I/O File System fails, then the entire Parallel I/O File System fails. There is, in general, no recovery from a disk failure except to recreate all the existing files and restart whatever jobs were running at the time of failure. No file system repair utility such as `fsck` can be used with PIOFS. PIOFS should not be viewed as a permanent structure.

A typical application of PIOFS is a high performance scratch file system for number-crunching. An organization would read in a large data set from tape, load it on to PIOFS, execute a parallel or serial batch application on the data, and move the output data set(s) to permanent storage.

PIOFS is a separately-ordered IBM LPP requiring only AIX. PIOFS Version 1.02 became available in August 1996. It is scheduled to be withdrawn from IBM Service support on 31 December 1998. The product to replace it is the General Purpose File System (GPFS).

4.7.7 General Parallel File System

The criteria set out at the beginning of Chapter 4.7, "Parallel I/O" on page 172 called for I/O capacity on the SP that is scalable, flexible, and highly

available. RVSD covers all these bases, but only for raw logical volume data. General Parallel File System (GPFS), IBM's second generation of parallel file systems on the SP, addresses all requirements for file systems.

4.7.7.1 General Description

GPFS originated in the IBM Almaden Research Center. The "Tiger Shark" research project aimed to create a high-performance file system for multimedia applications, which are I/O bandwidth hungry by nature. Tiger Shark has spawned three products:

1. Multimedia LAN Server - an audio and video server on a LAN with real-time processing capabilities
2. Video Charger - video across a LAN embedded in a Web Server product
3. GPFS

The RS/6000 Division modified the GPFS product of Tiger Shark for implementation on the SP. Because of the multimedia heritage, most GPFS commands are prefixed by "mm".

GPFS provides a standard, robust file system for serial and parallel applications on the SP. From a user's view, it resembles NFS, but unlike NFS, the GPFS file system can span multiple disks on *multiple nodes*. GPFS exploits VSD technology and the Kerberos-based security features of the SP, and thus is only supported on SP systems.

A user sees a GPFS file system as a normal file system. Although it has its own support commands, usual file system commands such as `mount` and `df` work as expected on GPFS. GPFS file systems can be flagged to mount automatically at boot time. GPFS supports relevant X/OPEN standards with a few minor exceptions, which we discuss later. Large NFS servers, constrained by I/O performance, are likely candidates for GPFS implementations.

GPFS is a separately-priced and ordered IBM LPP. It is only supported on PSSP V2.4 or later, and requires the RVSD Version exploiting PSSP V2.4. RVSD and VSD in PSSP V2.4 have enhanced functions, such as VSD node fencing, used by GPFS. Normal RVSD implementation requirements, such as twin-tailing the disk subsystems, are not actually mandatory with GPFS.

4.7.7.2 Implementation

NFS and VSD have a clear client-server concept imbedded in their implementation. One of the keys to understanding GPFS is that it does not

have a client-server concept *at the GPFS level*. Consider Figure 98 on page 182.

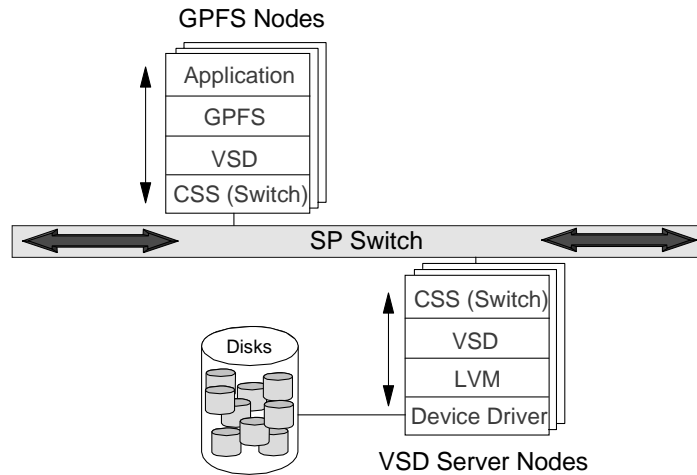


Figure 98. GPFS Structure

The nodes with the application and GPFS layer are simply called GPFS nodes. They have GPFS code, and are capable of "seeing" any GPFS file system in the SP. When the application makes an I/O request to a GPFS file system, a GPFS node uses VSD technology to service the request. Notice that the nodes with the actual disks have only a VSD layer; no GPFS code is installed on these nodes. The nodes with the disks are called VSD servers, not GPFS servers. In essence, GPFS provides the file system interface, while VSD technology furnishes the distributed data access. A GPFS node can also be configured as a VSD server, depending on the I/O access pattern of the application. A set of GPFS nodes and its VSD servers is referred to as a pool.

We now review the software architecture of GPFS in more detail. Please refer to Figure 99 on page 183.

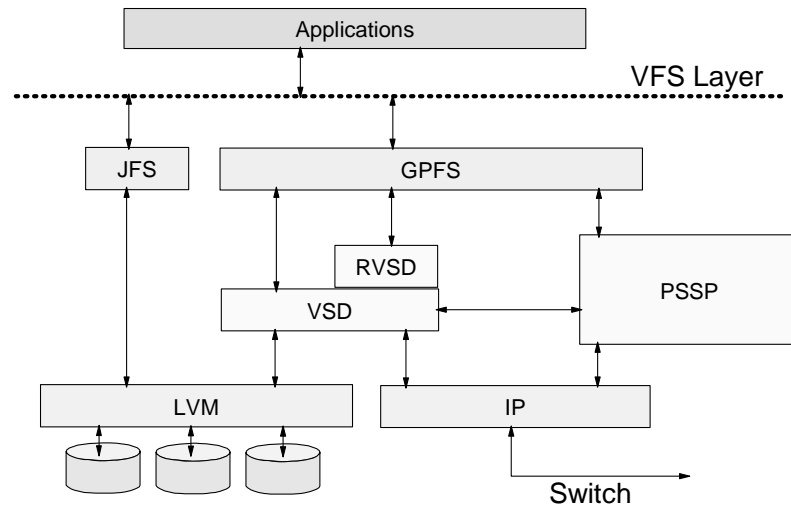


Figure 99. GPFS Software Architecture

GPFS is a kernel extension sitting atop the RVSD/VSD layers. To AIX, it is just another virtual file system (VFS) such as Journaled File System (JFS) or NFS. From an application viewpoint, GPFS resembles JFS, except the I/O operations are handled through RVSD/VSD instead of the local LVM. Like any JFS, a GPFS file system can be NFS-exported. Non-GPFS nodes or any NFS client machine in the enterprise can access an NFS-exported GPFS file system. This is why large NFS servers are targets for GPFS replacement; from the NFS client point of view, nothing changes, except the increased performance!

Although GPFS exploits VSD technology (which can be implemented over any IP network), it is only supported on the SP Switch or HiPS Switch. GPFS is designed for high performance. As such, access to GPFS file systems obeys SP partition boundaries.

GPFS uses the familiar, traditional UNIX file system structure of i-nodes, indirect blocks and data blocks. You configure the parameters of some of these components at creation time of a GPFS file system, which affect things such as maximum file size in a GPFS file system. A detailed discussion can be found in *GPFS: A Parallel File System*, SG24-5165. Each GPFS file system has a log file containing all transactions executed against the file system. GPFS logs are similar to jfslogs for a JFS file system. GPFS logs are replicated; up to two copies are maintained for availability.

GPFS configuration is stored in the SDR on the CWS. Whenever a GPFS node boots, it checks configuration files in the SDR to see if anything has changed. If changes are detected, they are applied to the node while it boots. This configuration management process allows GPFS changes while not all nodes are available.

4.7.7.3 mmfsd - The Multiple-Personality GPFS Daemon

GPFS relies on a daemon, mmfsd, to provide data and meta data management (such as disk space allocation, data access, and disk I/O operations), security, and quota management. To maintain consistency and a single view of the GPFS file system in a parallel environment, the daemon performs extra tasks by assuming different personalities.

A single GPFS daemon can assume one or more of the following personalities: Configuration Manager, Stripe Group Manager, Token Manager Server, and Metadata Manager.

Configuration Manager

This daemon is responsible for global configuration tasks in a GPFS pool, such as selecting a Stripe Group Manager for each file system and determining whether a GPFS pool has quorum (similar to the quorum concept of LVM for volume groups). One daemon per pool, usually the first to start, will assume this personality.

Group Services of HAI will select the next "oldest" node for this personality should the initial Configuration Manager become unavailable.

Stripe Group Manager

Each file system in the pool has one of these daemons. It is responsible for:

- Adding disks to grow the file system
- Changing disk availability
- Repairing the file system (GPFS actually has an `mmfsck` command!)
- Mounting and unmounting file systems
- Controlling disk space allocations
- Passing token requests for file access to the Token Manager

Token Manager Server

Each file system in the pool also has one of these daemons. The Token Manager Server and associated Stripe Group Manager always reside on the same node. The Token Manager Server cooperates with the Token Manager, which is a kernel extension in each GPFS node. A request for a file requires a

token. For availability, two copies of tokens are maintained in separate Failure Groups (discussed later). There are two scenarios, depending on the file location:

- The file resides on the requesting node

The application requests the token from the Token Manager on the node. If the token is available, it is granted to the application and the Token Manager informs the Token Manager Server for the file system. If the token is not available, as in the case of another node writing to the file, the Token Manager negotiates with the other node(s) to get the token.

- The file resides on a remote node

The Token Manager requests a token from the Token Manager Server for the file system. If the token is unavailable, the Token Manager Server tells the requesting node which other node(s) have the token. Again, the Token Manager must negotiate directly with the other node(s) for the token.

Parallel applications can efficiently access non-overlapping regions, or byte-ranges, of a GPFS file. Tokens can be requested at either the byte-range or file level, and these GPFS locks are enforced. Also, an application can use POSIX locks (fcntl system call), which are advisory - in other words, for the application's purposes only. POSIX locks are not enforced at the system level.

Metadata Manager

Each file in a GPFS file system has one of these managers. A GPFS daemon can be a Metadata Manager for numerous files. The node having the longest continuous access to a file usually has its mmfsd elected as Metadata Manager for that file.

4.7.7.4 I/O Striping

GPFS automatically spreads data across VSD disks for performance and disk capacity balance. GPFS is particularly suited for large sequential files with large block sizes.

Each file system comprises a number of VSD disks, which form a stripe group. The Stripe Group Manager controls data and metadata allocations for the stripe group. At the time of file system creation, you can request the Stripe Group Manager to use one of three allocation algorithms:

1. RoundRobin

I/O operations are performed on all VSD disks in strict order. The data is placed one allocation at a time per disk through the same loop sequence of disks. For example, if a file system were supported by three VSD disks,

the allocation sequence would be 1-2-3-1-2-3-1-2-3 and so on. This is the default method, and ensures good performance and I/O balance.

2. **Random**

Allocations are done purely randomly across the VSD disks. This method does not assure I/O balance.

3. **BalanceRandom**

This is a hybrid of RoundRobin and Random. Allocations are done randomly, but a VSD disk cannot be chosen again for an allocation until all other disks have had an allocation. For example, an allocation sequence could be 1-3-2-2-1-3-1-2-3 and so on.

4.7.7.5 Replication and Failure Groups

Properly configured and with sufficient system resources (and budget), GPFS can be made incredibly available. Some might say insanely available.

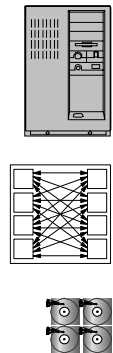
File system data can be protected at three levels:

1. **Disk Hardware.** AIX mirroring and RAID hardware can be employed, independent of any products or mechanisms discussed so far.
2. **Node.** With twin-tailed disk subsystems, RVSD can protect against the failure of a VSD server node by moving volume groups to secondary VSD servers. Without twin-tailing, GPFS is made aware of data access errors. After four successive errors, GPFS suspends the VSD.
3. **GPFS.** Using failure groups and replication, GPFS further expands availability.

GPFS allows up to two copies of data and metadata via replication. Replication is configured at the file or file system level; by default, no replication occurs except for the file system transaction logs. Replicas are allocated in accordance with failure group configuration. A failure group is a set of nodes in a pool that share a single point of failure. Each disk in a GPFS file system is assigned to a failure group, thus file system replica allocations are spread across distinct failure groups whenever possible to maintain the highest possible level of availability. Failure groups by default are identified as the node number plus 1000. For example, a disk on node 7 belongs to failure group 1007. If a file configured to have a single replica is initially allocated to a disk on node 7, GPFS will search for another disk not in failure group 1007 to allocate the replica.

AIX mirroring allows three copies of a logical volume. In combination with GPFS replication, it is possible to have six copies of your data spread across VSD servers in the system!

4.7.7.6 Performance



	write	read	Transport	Aggregate
NFS	800KB/s 5MB (V3)	10MB/s	UDP TCP (V3)	N/A
PIOFS	8MB/s	12MB/s	IP	Scalable
GPFS	20-30MB/s	20-30MB/s	IP (VSD)	Scalable

Figure 100. GPFS Performance vs. NFS and PIOFS

Figure 100 on page 187 gives a performance comparison of the mechanisms that provide shared data access at a file system level. Note that GPFS has considerably more I/O bandwidth than NFS and PIOFS, and has balanced read and write performance. The values presented are application dependent; your results may vary.

4.8 File Collection

File Collection is used to simplify the task of maintaining duplicate files on the multiple nodes of the SP system. In a standard SP system, the files that are required on the Control Workstation, boot/install servers, and processor nodes belong to file collections. By grouping these files into file collections and using the provided tools, you can easily maintain their consistency and accuracy across the SP nodes.

This section describes the characteristics and organization of file collections. You will also see how to report, update and maintain the file collections on your SP system. We also describe how to build file collections for your own purposes. Significance of scan and refuse files and their usage is covered in some detail.

4.8.1 Introduction

A file collection is a set of files and directories that are duplicated on multiple nodes in the SP network and managed by tools that simplify their control and maintenance. The Parallel System Support Program (PSSP) is shipped with a

program called `supper`. `/var/sysman/supper` is a perl program that uses the Software Update Protocol (SUP) to manage the SP file collections and maintain them across the system.

4.8.2 Characteristics of File Collections

A file collection directory does not contain the actual files in the collection. Instead, it contains a set of master files to define the collection. Some master files contain rules to define which files can belong in the collection; others contain control mechanisms such as time stamps and update locks.

The `supper` has a set of subcommands. The files in a collection are handled with special procedures using these `supper` commands. The `supper` commands interpret the master files and use the information to install or update the actual files in a collection.

A file collection uses some special files to define and control the collection. The `/var/sysman/` directory contains these files.

File collections require a unique, unused ID for `supman`, through which the file collection daemon, `supfilesrv`, can communicate. The default installation configures the user ID, `supman_uid`, 102 and the port, `supfilesrv_port`, to 8341. These values could be changed through SMIT or the `spsitenv` command.

A file collection can be either Resident or Available. A Resident file collection is one that is installed in its true location and able to be served by other systems. A file collection that is not installed in its true location but is able to be served by other systems is called an Available collection.

File collections can be either primary or secondary. Primary file collections can contain secondary file collections. When a primary collection that contains secondary collections is installed, the secondary collections are available but not resident.

The file collection daemon, `supman`, requires read access to any files that you want managed by file collections. For example, if you want a security file such as `/etc/security/limits` managed, you must add the `supman` ID to the security group. This provides `supman` with read access to files that have security group permission and allows these files to be managed across the SP by file collections.

4.8.3 Predefined File Collections

PSSP is shipped with four predefined file collections. They are:

sup.admin

The sup.admin collection is a primary collection that is available from the Control Workstation. It is resident (that is, installed) and available on the boot/install servers, and resident on each processor node.

This file collection is important because it contains the files that define the other file collections. It also contains the file collection programs used to load and manage the collections.

user.admin

The user.admin collection is a primary collection that is available from the Control Workstation, resident and available on the boot/install servers and resident on each processor node. This collection is a good place to add other files related to user management, such as /etc/environment, /etc/security/limits, and /etc/passwd.

power_system

The power_system collection is used for files that are system dependent. It is a primary collection that contains one secondary collection called node.root collections. The power_system collection contains no files other than those in the node.root collection.

The power_system collection is available from the Control Workstation and available from the boot/install servers.

node.root

This is a secondary file collection under the power_system primary collection. The node.root collection is available from the Control Workstation, resident and available on the boot/install servers and resident on the processor nodes. It contains key files that are node-specific.

4.8.4 File Collection Organization

By default, the file collections are organized in a hierarchy with the Control Workstation as the master server for all the collections. Figure 101 on page 190 shows this organization.

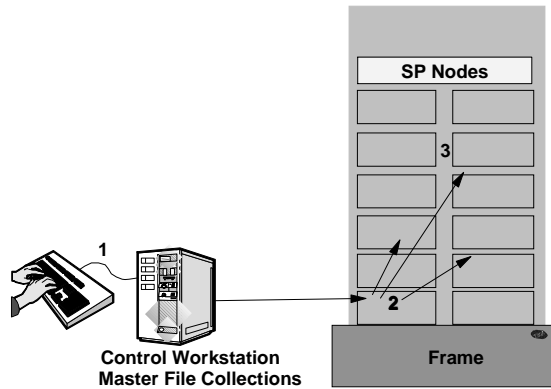


Figure 101. SP Default File Collection Hierarchy

The commands to update the file collections are set up in a crontab file to run hourly in a staggered sequence, as follows:

1. Files are updated on the Control Workstation in the master file collections.
2. The boot/install server uses `supper update` to request the changes from the Control Workstation.
3. Nodes use `supper update` to request the changes from the boot/install server.

4.8.5 Directory and Master Files

The `/var/sysman/sup` directory contains master files for all the file collections. Figure 102 on page 190 shows this directory structure.

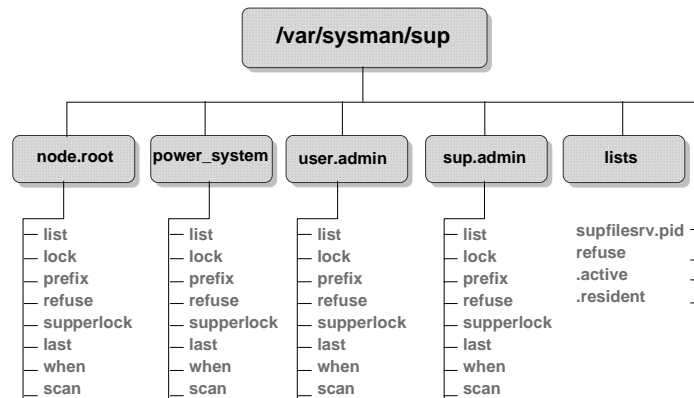


Figure 102. `/var/sysman/sup` Files and Directories

Following is a brief description of these files and directories.

supfileserv.pid: The process ID of the SUP supfilesrv process.

.active: A file identifying the active volume group

.resident: Lists each file collection in the SP system

refuse: Resides on a client system and contains the list of the files which need to be excluded. If it is under any master file collection directory, then the files included in that file collection will be excluded. Global refuse occurs when it is in the `/var/sysman/sup` directory.

last: This file is a system-maintained list of the files and directories that have been updated.

lock: This is an empty file used by supper block to lock a file collection at update time, preventing more than one update of a collection at the same time.

supperlock: This file is similar to the lock file and is used by supper to lock a file collection during update.

prefix: This file describes the starting point that supper uses when searching or scanning for files. If the root (`/`) directory is specified, all files on the system will be scanned.

when: This contains the time of the last file collection update. It protects against accidental update with older versions.

scan: This file contains a list of all the files in this file collection. If it exists, it is used by the supper process as the definitive list of all files. If it does not exist, supper does a scan to determine which file should be included.

It is normally recommended that you perform a `scan` to create this file at the time the file collection is created or modified. If a scan has been performed and the scan file exists, you must ensure that it is up-to-date. If it is there, it will be used even if the file collection has been modified and additional files have been added to the file collection in the list file.

list: This file contains details about the files that are part of this file collection. Rather than merely list them, however, it may be easier to define rules for identifying the files. For example, you could define a collection of all files within a directory or all files on the system that have `.myfile` at the end of the

name. The supper program scans the file systems as specified to find these files.

lists: A directory that contains links to the *list* files in each file collection.

node.root: Directory of the master files in the node.root collection.

power_system: Directory of the master files in the power_system collection.

sup.admin: Directory of the master files in the sup.admin collection.

user.admin: Directory of the master files in the user.admin collection.

4.8.6 Managing the File Collections

PSSP software is shipped with four predefined file collections, namely sup.admin, user.admin, power_system, and node.root. It gets installed during sysman fileset installation, but its usage is optional. In order to maintain the above file collections, you need to perform basic reporting of the file collection information, updating files that belong to an existing file collection, and possibly adding/deleting files from an existing file collection.

4.8.7 Basic File Collection Maintenance

Following are the four major tasks that you might want to perform in order to maintain the predefined file collections.

4.8.7.1 Reporting File Collection Information

To become familiar with the delivered collections you might list their name, access points, and whether or not they are resident. You might also want to verify the complete list of the files in a collection. The `/var/sysman/supper` is a perl program that provides subcommands that report information about file collections. These commands are useful for verifying information before performing file collection manipulation or for checking results at some point during a procedure. Refer to *IBM Parallel System Support for AIX: Command and Technical Reference*, GC23-3900, to learn more about these subcommands.

4.8.7.2 Verifying File Collections Using scan

The scan file provides an inventory of all the files and directories in the file collection. By looking at the scan file you can quickly see which files are in a collection. You can find the scan file in the file collection's directory in `/var/sysman/sup`.

Although scan files require maintenance whenever files are added, deleted or changed in the master collection, they save processing time on larger file systems and are recommended.

You must update the scan file each time you modify the file collection or you risk accidentally updating the collection from an outdated master file.

4.8.7.3 Updating Files in a File Collection

The SP file collections contain system control files that you may want to change. When you update files that are part of a file collection, you do not have to make the same update to each replication of the files. Instead, you make the update to the copy on the master server (generally the Control Workstation) and run the `supper update` command on all systems you want updated. The `supper` program does the updates for you.

At the time of installation, the `supper update` command is included in the crontabs file with the default set to run hourly. If you opt *not* to run the `supper update` command manually, your file changes will get updated anytime between 1 to 59 minutes later, depending on the system clock.

Like any other cron entry, you can modify the crontabs file to run the `supper update` more or less frequently.

In short, to update files in a file collection you need to perform the following three steps:

Step 1: Make the changes to the files on the master file collection.

Step 2: Run the `supper scan` command on the server.

Step 3: Run the `supper update` command, first on any secondary server, then on clients.

4.8.7.4 Adding and Deleting Files in a File Collection

Adding or deleting files in a file collection requires special considerations beyond those for adding or deleting files in a standard directory. The prefix, list and refuse master files contain criteria that define which files are in a particular collection. The master files are read and processed by the `scan`, `install`, and `update` commands. When you add or delete files in a file collection, you need to review the contents of these master files.

When adding files, you need to consider whether the files are in a primary or secondary collection, what the prefix and list files in that collection contain and the position of the new files in the file tree. Remember, the `supper`

command begins its search starting at the point defined in the prefix file. All files encountered that pass the criteria defined in the list file are considered part of the file collection.

4.8.8 Building a File Collection

Once you have become familiar with file collections and have worked with the delivered system collections, you may want to create your own. You can build a file collection for any group of files that you want to have identically replicated on nodes and servers in your system. These file collections must all reside on the Control Workstation or a boot/install server. Some good candidates for file collections are application data files or local tools.

The following steps describe how you can build a file collection of your own.

Step 1: Identify the files you want to collect

The files you want to collect as a group must all reside on the same system from which you want to serve this collection, and their permissions must allow them to be readable by everyone.

Step 2: Create a file collection directory

Create a directory in `/var/sysman/sup` with the name of your file collection.

Example: create `/var/sysman/sup/tools` if your file collection name is tools. Change owner and group to bin for tools directory.

Step 3: Create a list file

Create a list to describe the rules for including and excluding files in that directory.

Hint: The easiest way to create these files is to copy them from an existing file collection directory and modify as required.

Step 4: Add a link to the list file

The `lists` file in the `/var/sysman/sup` directory contains a symbolic link to the `list` file in each file collection. When you create a new file collection you must add a link to this file.

Step 5: Update the file.collections file

Edit the `/var/sysman/file.collections` file using your favorite text editor. Add the name of your new file collection as either a primary or secondary collection.

Step 6: Update the .resident file

The `.resident` file contains a list identifying all the resident SP file collections. Edit the `.resident` file on your Control Workstation or your master server and

add your new collection, if you want to make it resident, or use the `supper install` command.

Step 7: Build the scan file

The scan file only works with resident collections. It provides you with a list of files in the collection that you can use for verification and eliminates the need for `supper` to do a directory search on each update. If your directory tree is extensive, this can save processing time on larger file systems.

You must keep the scan file current. When a scan file is present, the update commands read it as an inventory of the files in the collection and do not do the directory search. If you fail to create a new scan file when you add, modify, or delete files in the master collection, the file will not be current and `supper` will not upgrade the collection correctly.

4.8.9 Installing a File Collection

During the utilization and customization process, the required SP file collections are first built on the Control Workstation and then installed on your boot/install servers and processor nodes. If you create your own file collections, you have to install them yourself on each server node.

To install a collection on a boot/install server you must perform the following steps:

Step 1: Update the sup.admin file collection

The `sup.admin` file collection contains the files that identify and control all the file collections. It contains `.collections` and `.resident` files. Whenever you make changes to these files, you need to update the `sup.admin` collection to distribute these updates.

You can wait for the scheduled update in the `crontabs` file or issue the command yourself on every boot/install server and node that needs the new collection.

Step 2: Run the supper install command

Run `supper install` on each boot/install server or node that needs this collection.

Step 3: Add the supper update to crontabs

You need to add the `supper update` command to the `crontabs` file on each server or node that has this collection. This insures that the new collection will be updated on schedule with the other file collections.

4.8.10 Refusing Files in a File Collection

The refuse file allows you to customize the file collection at different locations. You can create a file collection with one group of files and have different subsets of that group installed on the boot/install servers and the processor nodes.

This could be done by creating a refuse file in the `/var/sysman/sup` directory on the boot/install server or processor node. This master file lists all the files in every file collection that you do not want installed or updated on that system.

4.8.11 Removing a File Collection

Attention

Do not remove any of the predefined file collections that shipped with the SP system.

The file collections that come with the SP are required. If you remove them, the PSSP may not run properly.

You can use the `supper scan` command to build a scan file for the file collection you are going to remove. The scan file shows you a list of all the files in a collection. You can check it to verify that you do not need any of these files before you remove the collection.

Please note that removing a file collection does not delete it. It removes it from the place where it was installed. To completely delete a file collection, you must remove it from every place where it was installed. Be sure that you do not need any files in a file collection before you remove it.

4.9 Managing AIX Automounter

All the users or applications on a single stand-alone workstation have access to all the files on the system (that their AIX permissions allow).

On a network with two machines, the users on one machine would not necessarily have access to the files on the other machine. This is shown in Figure 103 on page 197.

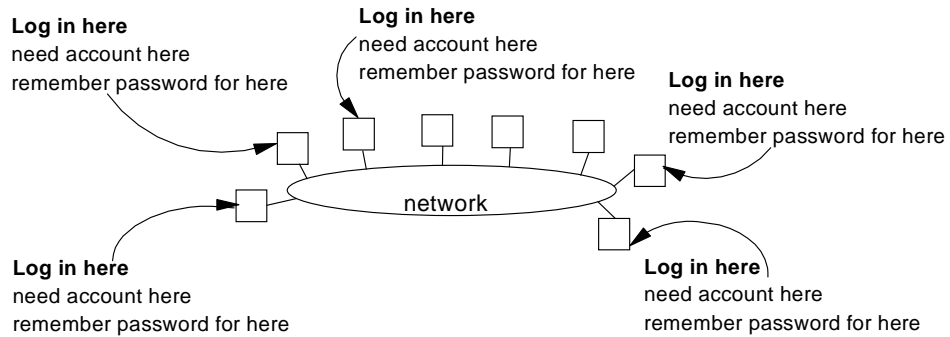


Figure 103. The World before NIS and NFS Automounter

To give users access to other file systems on other machines, there is the regular AIX concept of network file system mount, NFS. To use this, each user would have to NFS mount the file system that he wants to use on a directory that he has access to. Then the remote file system would appear local to the user.

It is also possible, with standard AIX commands, to set it up so that when a user starts to access a particular directory on his local machine, the local machine communicates with a remote machine to automatically mount a predefined remote file system on that local directory. This is most commonly done for the user's home directory. The remote file system then seems local to the user.

Sometimes, when customers have a network of like machines used by a group of people, they assign one password file for the whole network. This is NIS, or Network Information Services. They can then take this automatic mounting a step further so that there is one home directory file system per user or group of users.

When NIS and NFS Automounter are set up, the user is presented with a local login prompt at every machine on the network. When the user types in his userid and password, the local machine verifies this information with the central machine, which keeps one userid and password for each user who wants to log into the collection of machines. Once the user has typed in the correct authentication details, the local machine puts the user into an empty directory with no files, and then NFS Automounter can be set up to mount the user's home directory (from wherever it is on the system) over the empty local directory. So the user can log in anywhere on the network, using the

same userid and password, and have the same home directory no matter which machine he uses to log in.

Life after the advent of NIS and NFS Automounter is shown in Figure 104 on page 198.

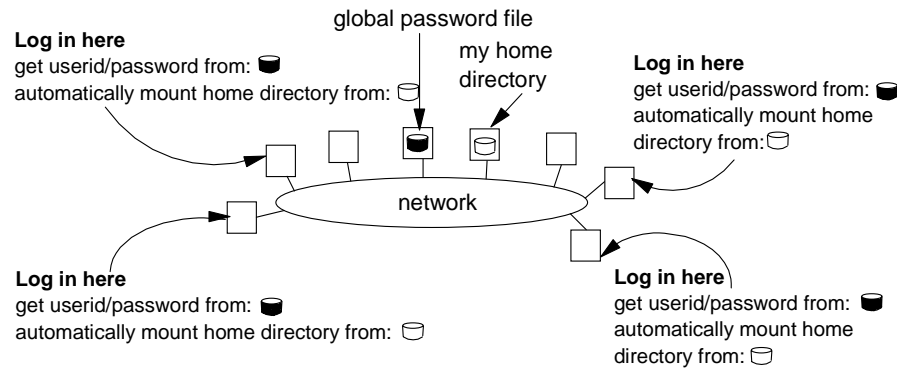


Figure 104. The World after NIS and NFS Automounter

All this is done quickly and the user is unaware that his password was stored on a different machine. He will not necessarily even know which machine holds his home directory. His home directory appears locally to him, no matter where in the NIS domain he logs in.

As the system gets larger and larger, the system administrator sleeps less and less. For each new user he has to do all the NIS setup and NFS Automounter mounting manually.

On the RS/6000 Scalar Parallel machine, this problem of manual setup for users has been addressed. At installation time, one has to make the following decisions:

- Whether to let PSSP start up the AIX Automounter daemon or not.
- Whether to use SP user management or not.

These settings can be changed after installation.

Having chosen to let PSSP start up the Automounter, and having chosen to use SP user management, new options are added to SMIT. Now, instead of using the user fastpath to go directly to the SMIT option for adding a user, one would go to the spuser fastpath and then SMIT will take care of the automounting configuration, implementation and setup under the covers for

the user's home directories at the time of the user's creation. This is the AIX Automounter daemon.

If the SP is configured to manage file collections, then the AIX Automounter map files will be automatically distributed to all the nodes by the supper daemon. The AIX Automounter map files are part of the user.admin file collection. If one does customize any of the SP Automounter functions, one would then need to edit the user.admin file to distribute the modified files or remove the default distributed ones.

If you want to use the NFS automounting facility for other than home directories, you have to do it manually, just as for a network of normal UNIX workstations. But the Automounter map files would be distributed automatically.

4.9.1 The Other Automounter

In PSSP 2.3, the BSD Automounter was replaced by the AIX Automounter, which is the AIX version of the SunOS 4.x Automounter. The AIX Automounter daemon software is part of NFS in the Network Support Facilities of the AIX Base Operating System (BOS) Runtime, and is fully supported by AIX.

The BSD Automounter, also known as AMD, is still available under licence on an as is basis. This Automounter is very flexible and has many added functions not available in the AIX Automounter. Unfortunately, the BSD Automounter is hard to service, has presented many problems in the field, so it not recommended.

If you are using BSD Automounter (PSSP 2.2 or below), and want to migrate to the AIX Automounter (included in PSSP 2.3 or later), you do not have to worry unless the old BSD Automounter was using a function not supported by the new AIX Automounter. When the control workstation first boots up at PSSP 2.3 or later after being migrated from a prior level of PSSP, `services_config` is called and this migrates the BSD Automounter map files to the AIX Automounter map files.

If the BSD Automounter map files were changed, or if the BSD to AIX conversion was attempted manually, errors might occur. These errors are logged, and a file of the successful map file migrations is created.

It is possible to have a system running both the AIX Automounter and the BSD Automounter. This is shown in Figure 105 on page 200. The Control Workstation distributes both the BSD and the AIX Automounter map files. When a node boots up, it runs `services_config`, which does the following:

- On the nodes running PSSP 2.3, services_config starts the AIX Automounter daemon.
- On the nodes running PSSP 2.2 and older, the BSD Automounter daemon is started.

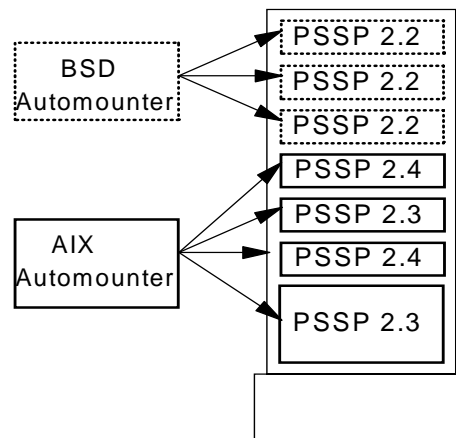


Figure 105. BSD Automounter and AIX Automounter Coexistence

The map files for each Automounter reside in a different place, and so nodes with any level of PSSP are able to mount all the file systems. The map files are distributed to all nodes, and each Automounter uses its own configuration files. A node effectively chooses which set of map files to use depending on the Automounter it runs. It is not advisable to start more than one Automounter on a given node to manage the same file system; it may hang that file system.

4.9.2 The AIX Automounter

The AIX Automounter allows system administrators to customize its functions to suit their site. During the execution of every AIX Automounter function, the daemon checks to see if there are any user scripts to replace or enhance a native function. If such a script exists, it is executed instead of, or in addition to, native AIX Automounter functions. For example, the startup script could be changed so that the Automounter daemon lists all the map files that it will be using before it starts up.

All internal and external errors and error conditions related to or detected by the AIX Automounter daemon are logged in a file. The output from the AIX Automounter configuration process and from the scripts that start and refresh the Automounter daemon are also stored in the error file.

If the automounted file system is not being used, it is disconnected. The disconnection time can be set if something other than the default 300 seconds is required.

4.9.3 Differences between AIX and BSD Automounter

With the BSD Automounter, it is possible to prioritize the interfaces over which to mount the directories. The BSD Automounter configuration file can be configured so that if the switch is down, the mount will happen using the Ethernet adapter. In the AIX Automounter configuration file, two interfaces can be specified, but they are not prioritized. The first host that answers the AIX NFS-mount operation is the host from which the AIX-mount gets done. This lack of prioritization works the same way in both the AIX and the BSD Automounters when one of the adapters is down, but when they are both up, the AIX Automounter will pick the fastest responding adapter, which may or may not be the best choice.

With the BSD Automounter, it is possible to override the default mount point for a specific directory by updating the BSD Automounter configuration file. This is not supported on the AIX Automounter; the default mount point is always used.

With the AIX Automounter, there is no support for selectors to control the use of a location entry. On the BSD Automounter, however, the use of location entries can be specified using equivalence operators in the configuration file.

If two users had the same home directory, BSD Automounter would NFS-mount the home directory twice onto the stub directories used by the two users. So, if these two users were logged on from the same machine, there would be two mounts. This dual mounting increases network traffic and is not an efficient use of resources, so AIX Automounter will only NFS mount the home directory for the first of the two users to log in, and the second user would receive a symbolic link to the already mounted home directory.

In the spirit of trying to efficiently use the network, the AIX Automounter will automatically unmount the remote file system and erase the symbolic links if the particular directory is not used for a specified amount of time. If two users were both using the same home directory, and if they were both inactive for a period, the home directory would be automatically unmounted and the symbolic links removed. If one user then reaccessed his home directory, the link would be reestablished, but possibly in a different order than before. So instead of the first user having the NFS-mounted directory and the second user having the symbolic link, it could be reversed after a period of inactivity. For users of the C shell, this may be confusing, since the C shell follows links,

and if the `pwd` command were run, it would show the actual path of the directory and not the path that was taken to get to the directory.

4.10 Job Management

In a world where one has many nodes in a machine, there has to be some way of sharing the batch work on the various nodes. If not, some nodes would be used optimally, and some nodes would not be used at all. It would require a very diligent system administrator to notice the inconsistencies and an even more dedicated system administrator to work out a way of optimally using the various nodes.

4.10.1 AIX Job Management

In AIX itself, all the traditional AIX commands that are used to handle job management are still available when AIX is running on a node. Here is a summary of the various AIX job management commands. This list should make it obvious that these commands alone are not sufficient to effectively manage any number of nodes greater than one.

- `at` allows the system administrator to specify a shell script to be run once at a specific time. In this shell script, one specifies which program or programs to run on which node or nodes. But this script has to be well written to:
 - Handle all possible error conditions.
 - Deal with useful redirection of all possible outputs to known places.
 - Cope with high processor load on the destination machine at the requested run time.
- `batch` could be used in conjunction with `at` to specify a list of jobs that will be run only when the processor load level allows.
- `cron` is used on each node to specify a shell script to be run at a regular time. This command, like the others, uses shell scripts that have to be well written to handle all possible error conditions.

The good thing about these AIX commands is that they are relatively simple to use and most system administrators have lots of experience with them. However, they are not sufficient to effectively manage the many different jobs to be run on the various nodes. Managing parallel jobs on multiple nodes would also not be easy with these traditional commands. Enter LoadLeveler.

4.10.2 The LoadLeveler Approach

LoadLeveler was designed with the original high-level system characteristics in mind, specifically "Manageability and Ease of Use". To accomplish this goal, LoadLeveler allows one to conceptually submit all the jobs that need to be run into a pool, and then farm them out to the various nodes in an efficient manner.

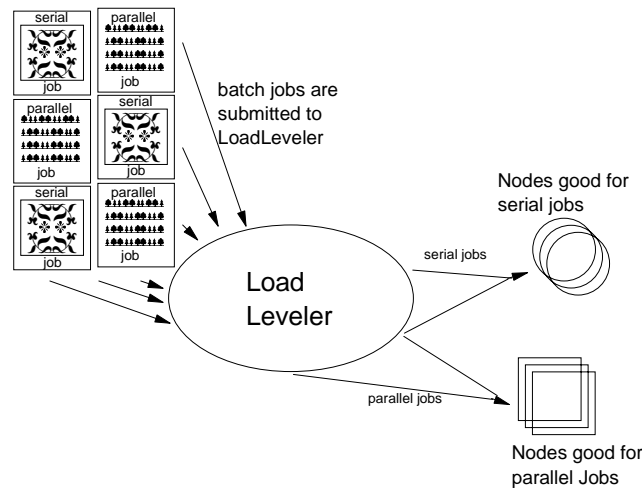


Figure 106. LoadLeveler Divides Work Among Nodes

First, specify which node is the primary Central Manager, and then nominate the alternate Central Manager to ensure availability of the system. Usually, the Control Workstation is the primary Central Manager. The Central Manager collects the various jobs and delegates them out to specific nodes.

Then categorize the various nodes by the type of work that they are best suited to perform, and categorize the jobs by the types of nodes that they run on. One can also specify nodes to be submit-only nodes, and these nodes then do not receive any work from the Central Manager.

Then categorize the various users and groups according to:

- Their job priority.
- The limit of the number of jobs that they can submit at a time.
- The type of job to be run as a default if none is given.

Once the nodes that do the work and the users that submit the work have been classified, it only remains to classify the kinds of work that can be submitted. Then the benefits of using LoadLeveler can be reaped. A whole

host of commands have been developed to manage and use LoadLeveler, but there is also an easy-to-use Graphical User Interface (GUI). This GUI gives different colors for the activity levels of all the nodes.

4.10.3 Resource Manager

The problem with LoadLeveler is that it only directly handles serial jobs, or jobs that run on one node at a time. When the job request indicates that the job must be run in parallel, LoadLeveler sends the request to the Resource Manager who takes care of delegating the job to the required nodes. The graphical representation of this relationship between the LoadLeveler and the Resource Manager is shown in Figure 107.

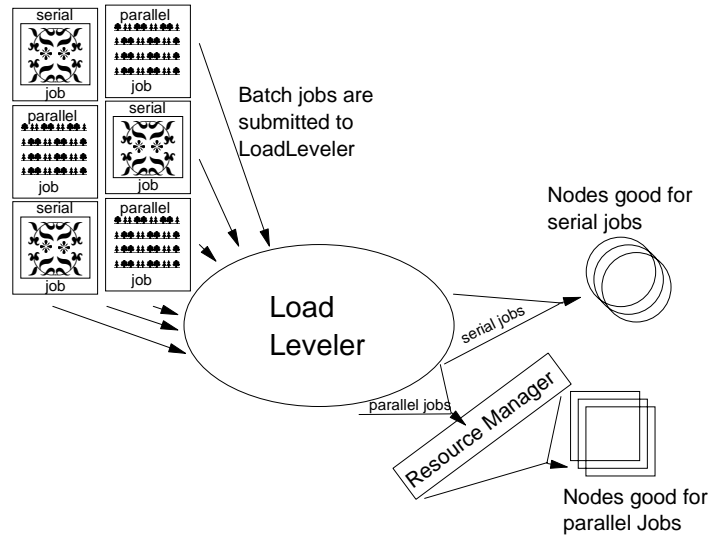


Figure 107. Interaction of LoadLeveler and Resource Manager

The Resource Manager is a program that runs on one of the nodes. It prevents parallel jobs from interfering with each other and reports job-related node information back to LoadLeveler. The system administrator has to add a couple of options to the configuration of LoadLeveler to enable it to interface with Resource Manager.

4.11 Parallel Environment (PE)

IBM Parallel Environment presents a UNIX-like serial program interface to the end user, while providing powerful message passing libraries optimized for the RS/6000 SP. It also provides tools for compiling, debugging, and

analyzing the performance of parallel applications. Parallel Environment supports both interactive and batch execution via the LoadLeveler product on the SP.

Parallel Environment includes components and tools for developing, executing, debugging and profiling parallel programs. These are:

- Parallel Operating Environment (POE)
- MPI/MPL Message Passing Libraries
- LAPI (Low_level API)
- VT Visualization Tool
- Parallel Debuggers
- Xprofiler Profiler Analyzer

4.11.1 POE Operating Environment

This environment includes the `poe` command, a run-time daemon (`pmd`) for controlling a parallel computation task, compiler scripts to include the appropriate libraries, and sample programs.

What POE does: The `poe` command, which appears to an end user as a serial Unix (AIX) command, identifies a set of N compute nodes on which to run an executable specified by the user, causes that executable to be started on each of the identified nodes, and waits for all executables to finish before returning to the user. The compute nodes can consist of a specific list of IP names or addresses, or can be selected implicitly by LoadLeveler when the compute nodes are all on an RS/6000 SP system. Standard output from the executables running on the compute nodes is routed back to the `poe` command and displayed as standard output from POE.

The executable can be a serial program (for example, the AIX command `hostname`), a shell script, or a parallel program compiled with one of the scripts provided by POE (for example, the `mpcc` script). These scripts link the user's application code with the message passing library and with a POE initialization stub that provides coordination between all of the executables started by POE.

One of the sample programs is a fast copy program that uses the MPI message passing library to copy a file from one node to another node on the SP.

Figure 108 on page 206 describes how applications make use of the parallel environment architecture. Applications compiled with the parallel compilers have been optimized for parallel execution.

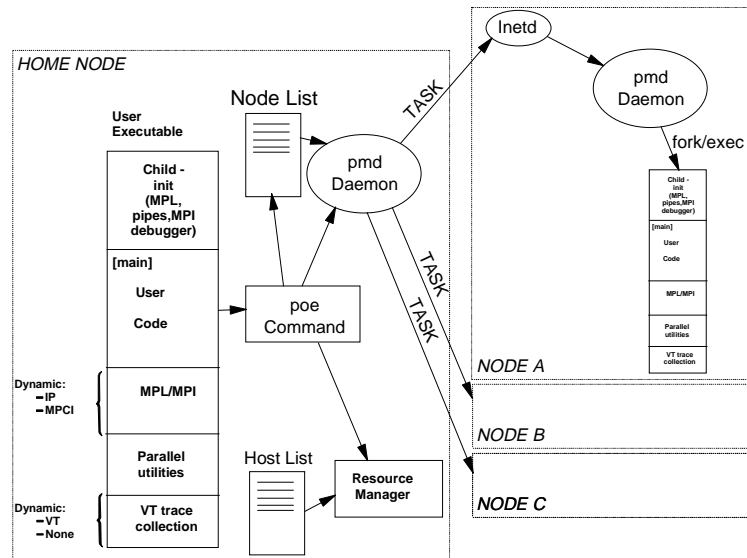


Figure 108. POE Architecture

The generated code also includes the auto selection of TCP/IP or User Space as the transport layer for the message passing libraries (MPI/MPL). This selection can be fixed to one or the other from inside the application.

The partition manager in the home node will contact the remote pmD daemon to dispatch the different tasks. Or, it will use LoadLeveler to allocate nodes. The use of one or the other will depend on your hardware, configuration, and any specific need the application may have. POE does not realize any allocation.

4.11.2 MPI/MPL Message Passing Libraries

These libraries provide APIs for message passing between tasks of a parallel program. MPI is an industry standard, and IBM's version conforms to the Version 1.1 level of the standard. There are two versions of the MPI library; one optimized for execution on the RS/6000 SP (using a "user space" packet protocol), the other supporting execution on any cluster of RS/6000 workstations. The choice of libraries is made dynamically at run time, thus permitting binaries created on a workstation to be executed optimally on the SP. The MPI libraries also include support for POSIX threads.

What MPI does: The MPI message passing library provides functions, callable from FORTRAN, C, and C++, that allow a program to reliably send and receive messages from one parallel task to another. MPI provides both blocking and non-blocking forms of communication, as well as a variety of waiting and testing functions. MPI promotes portability and extendability of applications and libraries by defining "communicators", which correspond to logically independent communication domains, and by defining a wide variety of data types that free the programmer from having to worry about the actual size of a data element.

If the user's executable uses AIX POSIX threads, a thread-safe version of the MPI library is automatically linked in when the executable is created by the appropriate POE compiler script. In this version of the library, AIX threads are used to provide overlap of computation and communication within a single executable running on a multiprocessor (SMP) node.

If the compute nodes selected by the user are on an SP with the SP Switch, the user may request the version of MPI optimized for direct communication over the switch. This is called "user space" access, and provides the lowest latency and highest bandwidth available to a user application. On other nodes, the MPI library uses UDP/IP as the packet transport protocol, working over whatever LAN network is available. The user's executable is the same, no matter which library is used.

MPL is an IBM-defined message passing library that was provided on the SP prior to the definition of the MPI standard. It still provided for compatibility with existing user applications. New applications should be coded using MPI. A program can have a mix of MPI and MPL calls; however, an MPI send can not match an MPL receive, or vice-versa.

4.11.3 LAPI (Low-Level API)

This API provides a reliable, "active-message" style message passing programming interface. It is distributed as part of the SP system software, and runs only on the SP. LAPI is intended to be incorporated into higher-level messaging subsystems. These subsystems typically are event-driven and support a distributed shared memory interface to application programmers.

What LAPI does: LAPI provides low level support for an "active message" style of messaging, in which the sender (origin) of a message provides the address of a target node handler that is to process the message. Typically, the target node handler is expected to run asynchronously with any other activities on the target node. LAPI provides this capability by creating a POSIX thread dedicated to running handlers. This thread is totally invisible to

the user application; the user may create additional threads to increase the degree of asynchronous activity, if desired. LAPI is supported only on the RS/6000 SP using direct access (user-space) to the SP switch.

Figure 109 on page 208 depicts how LAPI interacts with the hardware layer.

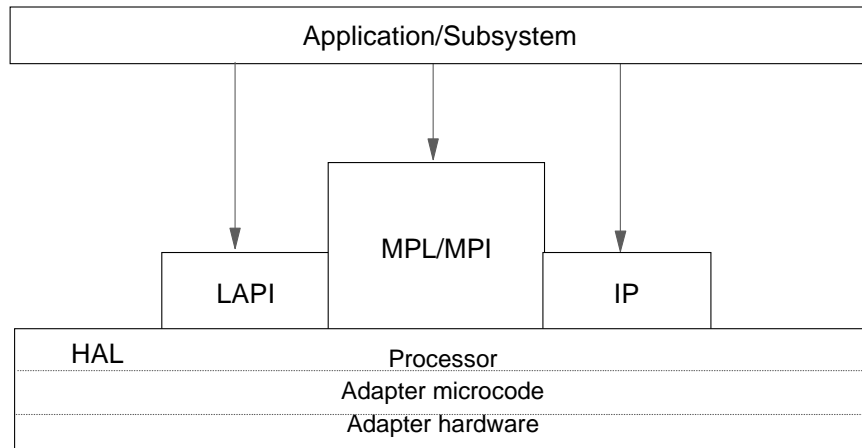


Figure 109. Communication Protocol Layers

LAPI interacts directly with the hardware layers using the Hardware Abstraction Layer (HAL). Applications using LAPI can get much better latency for inter-tasks communication than those using direct TCP/IP sockets or the standard message passing libraries.

4.11.4 Visualization Tool (VT)

Provides a graphical tool for displaying and analyzing the message passing patterns in a parallel program. A running program produces VT trace records that can be saved and studied to determine if alternative message passing techniques might improve overall performance. VT also provides an interface to display how a running parallel job uses system resources such as CPU, paging space, and I/O.

What VT does: If the user sets a particular environment variable before executing POE, each MPI/MPL message passing call will generate a trace record that is written to a file during execution. At the end of the parallel job, all the trace files from each task are merged into a single time sequence and copied to the user's current directory. At any subsequent time, the `vt` command can be run, and can be used to interpret the trace file, displaying

the data on an X windows display using a variety of display paradigms, including a time line, task pie charts, and so on.

The `vt` command can also be run without accessing a particular trace file. In this mode, `vt` can display performance information from a selected set of nodes (possibly being the set of nodes on which a parallel job is being run).

4.11.5 Parallel Debuggers

The `pdbx` debugger is a parallel interface to the AIX `dbx` debugger. It allows grouping of tasks so that a single `dbx` command can be applied to multiple tasks. The `pedb` debugger is a multiple X window debugger for persons unfamiliar with `dbx`. It provides the capability to look at individual tasks or to group tasks.

What `pdbx` does: The `pdbx` command is a variant of the `poe` command, and supports the same command line and environment options that `poe` supports. When the compute nodes have been selected, `pdbx` starts the `dbx` command on each of the compute nodes, and then starts the user's executable under the control of `dbx`. It then provides a "multilog" (dialog with more than one other party) between the originating (home) node and all of the parallel tasks. The user may define subsets to partition the tasks into those under direct control of `pdbx` and those that are allowed to run free.

What `pedb` does: `pedb` is an X windows debugger that is also a variant of the `poe` command. After the compute nodes have been selected, `pedb` starts a remote debugger agent on each node and then has that remote agent start the user's executable under debug control. From the X windows display, the user may select individual tasks or group tasks, display the source line, global and stack variables, etc. The `pedb` debugger also supports a window that allows the user to see the details of the message passing queues. This would allow a user to examine which messages have not been received, or which messages have been sent incorrectly, for example.

4.11.6 Xprofiler Profile Analyzer

AIX provides a profiling capability for user applications, and two alphanumeric summary tools: `prof` and `gprof`. Xprofiler provides a graphical user interface as an alternative to `gprof`.

What Xprofiler does: Xprofiler is a serial command which reads and interprets "gmon.out" files produced when the compiler flag "-pg" is used to create an executable. `gmon.out` profile files capture library call counts and instruction addresses at 10 millisecond sample intervals. In AIX, the `gprof` command can be used to display a tabular printout of this data. Xprofiler

provides an interactive display of the data for one or more profile files, which may make interpretation of the data easier. It is not required that the profile files be generated by a parallel job.

4.11.7 User Space and IP Protocols

The SP Switch (and its predecessor, the High Performance Switch) provide a low latency, high bandwidth interconnection between SP nodes (See Part 3.4, "High-Performance Communication Network" and Part 4.3, "Overview of Switch Software" for details). The PSSP software provides an IP device driver, and many of the PSSP subsystems use the switch as a high performance LAN, accessing it via TCP/IP. An additional path, called "user space access", is available over the switch. This path is provided by the appropriate SP switch adapter microcode, and is completely independent of IP or any service activity over the switch.

Since PSSP 2.4, many processes per SP node can make use of this preferred access path. However, in early versions of PSSP, only one task per SP node was allowed to use the "user space" path. The name for this new facility is called MUSPPA (Multiple User Space Processes Per Adapter). The decision which set of tasks can use which set of nodes in this way is made by the PSSP Resource Manager, and supported by POE and LoadLeveler.

Depending upon the application's communication characteristics, the low latency and high-bandwidth available over this path might be required to provide optimal performance. Typical values for SP uniprocessor nodes and the SP Switch are: latency - 35 microseconds; bandwidth - 100 MB/sec.

Chapter 5. System Management

This chapter covers SP installation, partitioning, performance management, workload management, and user management.

In installation we describe the various steps of the installation process, such as:

- Install the CWS
- Input node configuration information
- Install and customize nodes

This section provides an overview of the Network Installation Management (NIM) mechanism. It tells you about the tasks accomplished during the execution of the `setup_server` script. The boot process and node customizing process are covered in some detail. Migration and PSSP upgrades are addressed, keeping in mind the current software coexistence limitation and constraints.

The SP User Management (SPUM) section provides information about the user database, files that hold user information such as `/etc/passwd`, `/etc/group`, `/etc/security/passwd`, and `/etc/security/group`, and how they remain consistent across nodes.

Tasks related to user management, such as *add user*, *delete user*, *list user*, *change attribute of user*, and *change password of the user*, are also covered. SP Access Control (SPACS) implementation and its interface with job submission system, such as Resource Manager (RM) are explained in the SP user management section.

The Problem Management section, addresses the issues involved in logging and reporting SP errors.

The SP reports and acts on system problems in many ways. In general, SP problem management leverages AIX and the rich services of the High Availability Infrastructure (HAI). It also provides three types of error logging and reporting facility:

- The AIX error logging facility is implemented by the AIX error daemon `errdemon`. PSSP, through `sysctl` consolidates the nodes' logs at the CWS.
- BSD error logging facility is well known and widely implemented. It is implemented through the `syslogd` daemon.

- PSSP-specific error logging is built around the splogd daemon which reports on SP hardware errors and state changes.

The Problem Management subsystem (PMAN) provides an infrastructure for recognizing and acting on problem events within the RS/6000 SP system. PMAN is an Event Manager (EM) client and its working principle is also covered in the problem management section.

Under System partitioning section, you will discover why and how partitioning is accomplished in the RS/6000 SP.

System Partitioning is a method for organizing the SP system into non-overlapping groups of nodes for various purposes. System partitioning lets you divide system resources, yet allows you to manage and administer these resources from a single point of control (CWS).

The `syspar_ctrl` command helps you manage *partition-sensitive* subsystems. You may *add*, *start*, *refresh*, and *delete* partition-sensitive subsystems. We have provided you a step-by-step method of partitioning your SP system.

SP tuning help to optimize networks. SP system management, interprocessor communications, shared disk access, interconnection to other systems and programming, depend heavily on thoughtful tuning and exploitation of networks. Performance Management section briefly describes general performance considerations for SP, and introduces related tools to help you tune your SP.

The SP accounting builds upon the standard System V accounting facilities of AIX. Enabling SP accounting support is optional. The SP accounting extends the function of base AIX accounting. The section on accounting mainly presents you the basic differences between the SP and stand-alone AIX accounting procedures. The accounting section also lists the special considerations required in SP accounting.

The section on managing resource workload explains how LoadLeveler and Resource Manager can be used to utilize the SP system effectively.

SP administrators often need to distribute the workload of their batch jobs in a manner to optimally exploit the SP resources. Through using LoadLeveler approach, they can submit their jobs to a pool of nodes and let the Loadleveler run these jobs on the most suitable nodes.

Finally, the backup and recovery section provides the basic techniques to backup your vital SP system files.

5.1 SP Monitors

PSSP offers two different applications that help administrators monitor and manipulate SP resources: SP Perspectives and SP System Monitor (spmon).

5.1.1 SP Perspectives

SP Perspectives integrate system management tasks with a Graphical User Interface (GUI). They are called Perspectives because each one of them offers a different perspective of the RS/6000 SP. SP Perspectives can:

- Monitor and control hardware.
- Create and monitor system events.
- Define and manage VSDs.
- Generate and save system partition configurations.
- Set up performance monitoring hierarchies and archiving.

All Perspectives applications can be launched from a single common GUI, automatically display a graphical image of the RS/6000 SP hardware configuration, have a scalable design, accommodate additions to the SP, and perform system management functions via direct manipulation of icons.

To use the Perspectives GUIs, no special authorization is required. However, the applications that the GUIs start may require special authorization. Figure 110 on page 214 presents a typical opening window when SP Perspectives starts.

SP Hardware Perspective

The Hardware Perspective is responsible for presenting information about system hardware. Depending on the resource that is monitored, the Hardware Perspective gets its data from the SDR, Event Manager, and Problem Manager.

In order to control SP system hardware or open a serial link from the Hardware perspective, an SP Perspectives client needs the appropriate hardmon (see 4.2, “The Hardware Control Subsystem” on page 89) ACLs set up on the Control Workstation (See 4.6, “SP Security” on page 141). Then the client needs to have a Kerberos token that has the desired hardmon authority.

There are five commands that root can issue on the CWS that a non-root user can be given access to:

- cshutdown
- cstartup

- Efence
- Eunfence
- Estart

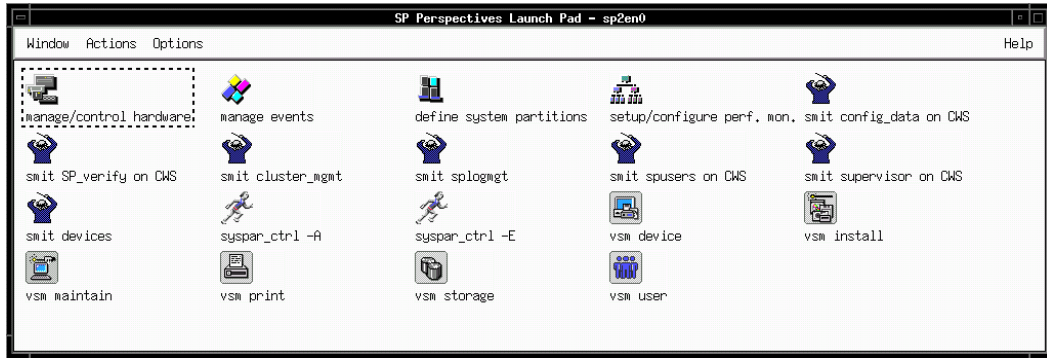


Figure 110. SP Perspectives

These commands can be used by any user for whom sysctl ACLs have been set up on the Control Workstation for the principal that the SP Perspectives client is using.

To create, modify, or delete node groups from the Hardware perspective, you must be root on the Control Workstation. Figure 111 on page 215 shows a sample screenshot of the Hardware Perspective.

IBM Parallel System Support Programs for AIX, Administration Guide, GC23-3897 has an extensive example of using the Hardware Perspective.

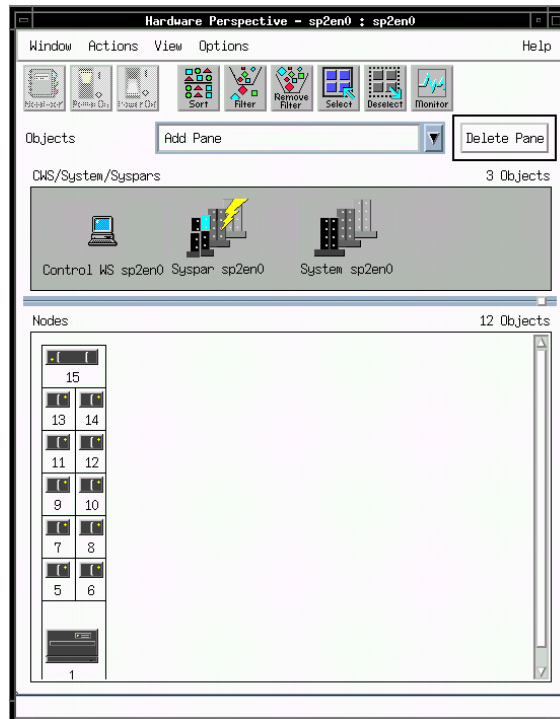


Figure 111. Hardware Perspective

Event Perspective

The Event Perspective allows you to define and manage event definitions within a system partition. In this sense it is not a GUI for displaying information; rather, it allows you to define monitors and triggers for other perspectives to display.

An *event definition* allows you to specify under what condition the event occurs and what actions to take in response. Using the Event Perspective, you can:

- Create an event definition
- Register or unregister an event definition
- View or modify an existing event definition
- Create a new condition

An event is triggered when a boolean expression involving a resource evaluates to true. A *resource* can be any entity in the system that can be

observed, for examples processors, disk drives, memory, adapters, database applications, processes, and filesystems. A more detailed description of the Event Perspective, along with examples including defining and monitoring events, is given in *RS/6000 SP Monitoring: Keeping it Alive*, SG24-4873. Figure 112 shows a screen shot of the Event Perspective.

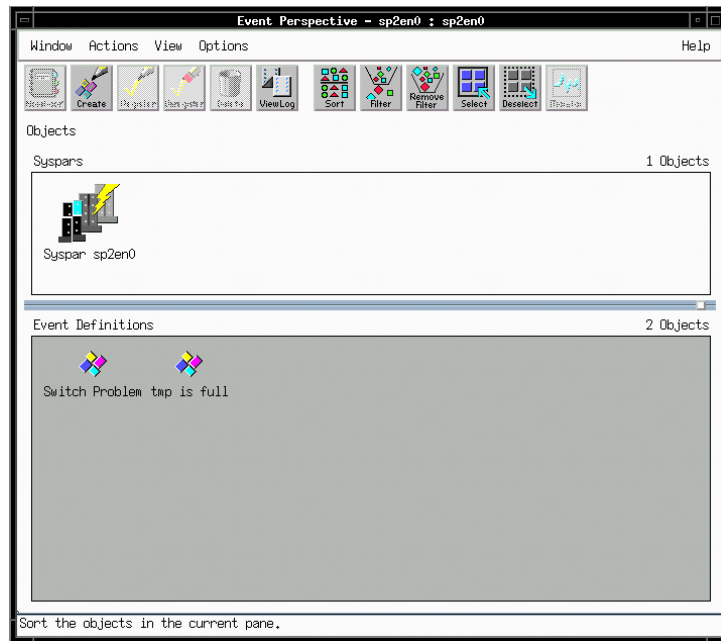


Figure 112. Event Perspective

No special authorization is needed to access the Event Perspective. However, to use the Problem Management subsystem, a appropriate Kerberos principal must be added on the Control Workstation and nodes.

VSD Perspective

The VSD Perspective is designed to allow monitoring of events specifically connected with the VSD resources of the system. Although one can use the Hardware Perspective to accomplish the same task, VSD Perspective offers a better graphical representation of VSD resources.

As with all other Perspectives, you must start with defining monitors for VSD resources using the Event Perspective. The VSD Perspective can then group such events and present them in an organized and concise form.

SMIT Panels

Besides the Hardware and Event Perspectives, the main SP Perspectives window has several icons that trigger SMIT GUI front-ends.

5.1.1.1 SP System Monitor

The SP System Monitor provides an alternative way for authorized operators and administrators to monitor the system hardware. Because the SP Perspectives GUI provides access to a wider range of system management tasks, IBM recommends the use of Perspectives over the SP System Monitor. However there are times when you have no choice but to use a command-line interface to monitor system resources. The System Monitor GUI is presented in Figure 113 .



Figure 113. System Monitor GUI (spmon)

There are four main parts to the SP System Monitor:

Hardware Monitor. A set of commands and a daemon used to monitor and control the SP hardware platform.

SP System Monitor GUI. Shown in Figure 113. Although this graphical interface is quite popular, it is being replaced for SP Perspectives.

Command Interface. The `spmon`, `hmon`, `hmcnds`, `sitem`, and `nodecond` commands, with their flags and parameters, provide a way to monitor and control the system at an individual system partition level or at a global (all system partitions) level.

Logging daemon. The `splogd` logging daemon logs SP hardware state changes, reports SP hardware errors, and provides a state change alert.

With the `spmon` and `hmon` commands you can query the value of a state variable or monitor for a change of state. With `spmon` and `hmcnds` you can control the nodes, frames and switches. The `hmon` and `hmcnds` commands provide equivalent function to the `spmon` command but provide a more compact interface than `spmon`. The `spmon` command is provided for compatibility with earlier SP software releases. Each of the commands has a `-G` flag that removes system partition boundaries, allowing the command to work on any hardware on the SP. The `-G` option is always needed to monitor or control frame or switch hardware. Frames and switches are considered to be outside all system partitions.

5.2 Installation

The purpose of this section is to outline the installation process for an SP system from a software point of view.

5.2.1 Installation Process

Before going to the actual process, there are several planning tasks that should be performed in advance. Some are shown here. For a complete reference, review *RS/6000 SP: Planning, Volume 1, Hardware and Physical Environment, GA22-7280*, and *RS/6000 SP: Planning Volume 2, Control Workstation and Software Environment Guide, GA22-7281*.

From a software perspective there must be planning for at least:

- Hostnames, IP addresses, netmasks and routes
- Boot/Install servers
- Authentication servers
- SP system partitioning

Boot/Install servers: These machines are used to boot and install the rest of the nodes, across the network. In simple installations, the Control Workstation (CWS) is the only boot/install server, but in a complex one there will be more boot/install servers.

Authentication servers: Because of the importance of some tasks in an SP system, there must be an entity that authenticates users and services, beyond normal AIX security. For this purpose, the SP uses Kerberos V4 as the authentication mechanism. PSSP offers the possibility that the SP can be a Kerberos V4 client and/or server.

SP system partitioning: There is a way to configure an SP system such that there can be node groups that logically do not relate to each other, for different purposes. For instance, there may be a production and a development partition, so the environment has different contexts.

The SP system installation involves three major steps:

- Install the CWS
- Input node configuration information
- Install and customize nodes

Install the CWS: In this phase, the CWS can be regarded as any RS/6000 machine, so its installation follows the normal installation process, which is outside the scope of this work.

After loading AIX and all the necessary PTFs onto the CWS, check out the necessary RS-232 connections to the frames, configure the CWS ethernet interfaces and review name resolution. This means that all names and addresses of all the IP nodes' interfaces must be resolvable on the CWS before the installation process takes place. Major steps when installing the CWS are the PSSP software installation and the file set download of all products that are about to be installed on the nodes. Also make sure that there is plenty of disk space to hold the required software to install the nodes.

Another important task to be performed when installing the CWS is the definition of the SP authentication services, a process that is performed with the program `/usr/lpp/ssp/bin/setup_authent`. The type of authentication service the SP will use can be one of the following:

- The CWS is the primary authentication server.
- The CWS is a secondary authentication server.
- The SP system makes use of an AFS authentication server, which has already been set up.
- The SP system is an authentication client system.

Detailed information about the CWS installation can be found in *IBM Parallel System Support Programs for AIX: Installation and Migration Guide*, GC23-3898.

Input node configuration information: This step can be accomplished using PSSP commands or SMIT. Basically, it consists of entering data in the System Data Repository (SDR), which is the SP configuration database. All the required data to perform this phase is extracted from the previously

filled-in worksheets that can be found in *RS/6000 SP: Planning, Volume 2, Control Workstation and Software Environment Guide, GA22-7281*.

The final step in this phase is to set up the network installation environment to install the nodes. The node installation is performed using AIX Network Installation Management (NIM). It provides an environment to install machines through the network using a previously made mksysb image.

NIM setup is done by the script `setup_server`. Among its functions is the configuration of the CWS as a NIM server. This machine provides resources so the other machines can be installed and because of this, it is called a *boot/install server*. But not only the CWS can be a boot/install server. In an SP system some of the nodes are also boot/install servers, serving other nodes during the installation process. But the first time `setup_server` is run, only the CWS is configured as a NIM server, so the boot/install servers can be installed and configured.

Install and customize nodes: The node installation process is a three-step process:

- Node network installation
- Node customization
- Boot processing

Node network installation: As stated before, the basic building block of this phase is AIX NIM. It provides an environment to install machines through the network using remote resources stored in a NIM server machine. In particular, this server stores a previously made mksysb image. So, in this phase, what basically gets done is the node installation from an mksysb image.

An overview of network installation looks like this:

- The node is powered up and performs its internal tests, as any RS/6000 machine.
- The node issues a boot protocol (bootp) request broadcast to locate its installation server, which is the machine that has the necessary resources to boot and install the node. In this case what is needed is a boot image and an AIX mksysb.
- The appropriate boot/install server receives the boot protocol request and answers the node, letting the node retrieve the correct boot image using `tftp`. The boot image selected depends on the type of node (uniprocessor or multiprocessor), and network boot

interface (currently only Ethernet is supported by the RS/6000 SP). See Figure Figure 114 on page 221.

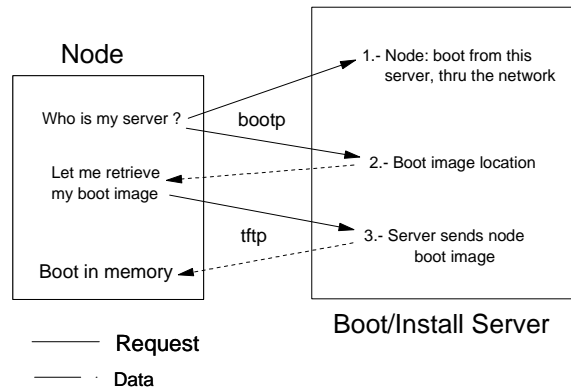


Figure 114. Node Network Boot

- The boot image contain as small AIX kernel and the root file system (/). After the node transfer this boot file (using tftp), it load it into memory, and then it executes the /sbin/rc.boot script. This is boot phase 1. It configures the node's network interfaces and routes, sets up the NIM environment, and NFS-mounts the Shared Product Objects Tree (SPOT), better know as /usr from the boot/install server. The SPOT selected depends on the *lppsource* being used to install the node, and it is located at /spdata/sys1/install/<lppsource name>/spot/spot_<lppsource name>/usr directory on the boot/install server.
- The next step is to poll the machine buses to detect devices, such as adapters, disks and so on. Those devices that have entries in the Predefined Devices Database (PdDv), will get configured.
- Now the boot device, after the installation, is set up. This is boot phase 2.
- During this phase, some of the configuration data is restored, the root volume group gets created and mksysb is restored. The mksysb file is NFS-mounted from the boot/install server.
- Along with the mksysb restoration, the node customization script file, pssp_script, is transferred to the node and is now executed.

Node Customization: During this step, all the pssp_script activity is logged in the node file /var/adm/SPlogs/sysman/hostname.config.log.\$\$ (hostname

is the node hostname and \$\$ is pssp_script PID), and the following tasks are performed:

- Gets the *hostname.install_info* and *hostname.config_info* using tftp. These files contain some environmental variables. *hostname.install_info* holds, for example, the CWS IP address, the CWS hostname, node initial hostname, node installation device, PSSP code version, and node processor type. *hostname.config_info* holds the IP address and netmask for the node network interfaces.
- Gets Kerberos files, *script.cust*, *scriptfb.cust* and *tuning.cust*, using tftp.
- Detects the processor type (uniprocessor or multiprocessor) and installs the correct Base Operating System runtime, if it is not already installed. This allows the use of any mksysb to install the nodes. Also updates the node boot logical volume.
- Also, if they are not already installed, pssp_script installs the prerequisite file sets *perfagent.server*, *bos.rte.tty* and *devices.sio.sa.diag*, which are required by the SP monitoring processes.
- PSSP software is installed now. The PSSP version that gets installed is determined from the SDR class node.
- Node dump logical volume is created. This has been designed this way because the nodes do not have a console, so when they boot and there is a pending dump to be transferred from the AIX V4 default dump device, paging space *hd6*, the system warns of this fact through the console. But the nodes have no console, so to avoid losing the dump or the node getting stuck waiting for the administrator to decide what to do with it, the node customization process creates a special dump logical volume, where the dump can be stored until retrieved.
- The *script.cust* and *tuning.cust* scripts are run. *script.cust*, among other tasks, imports existing volume groups, defines name resolution and paging space, gets a remote commands service ticket from the server, and copies user management files from the server. *tuning.cust* basically sets network options using the AIX `no` command.
- Finally, pssp_script gets the partition name and changes the boot/install server response to disk, so when the node boots again it will boot off the internal disk on which AIX was installed.

Boot Processing: After the installation and customization phases, the node boots off the disk where AIX was installed and performs additional customization steps depending on the options that were set in the PSSP Site Environment SDR database. Basically, this is related to how to manage Network Time Synchronization (NTP), SP System Accounting, SP User Home Directories and File Collection file set installation.

At the end of this process, the SP system is fully installed and customized, ready to be used.

5.2.2 Software Upgrade

Upgrading an SP system to the latest levels of AIX and PSSP is not a trivial task and requires careful planning. The current configuration must be fully documented and understood before going on to the upgrade steps. Detailed upgrading information can be found in *RS/6000 SP: Planning, Volume 2, Control Workstation and Software Environment, GA22-7281*.

Before going to the discussion about upgrading, it is necessary to understand a couple of concepts related to this kind of work, such as AIX levels, *migration* and *upgrade*. AIX levels are identified using the following convention: VV.RR.MMMM.FFFF, where:

- VV is a numeric field of 1 or 2 digits that indicates the version. The current version is 4.
- RR is a numeric field of 1 or 2 digits that identifies the release level within a version. There are three releases: 1, 2 and 3.
- MMMM is a numeric field from 1 to 4 digits that identifies the modification level within the release. For AIX Version 4 release 1, there are five possible modification levels: 1, 2, 3, 4 and 5. In AIX 4.2, there is only one modification level, being it 1. AIX Version 4 release 3 also has only one modification level whose number is 1.
- Within any VV.RR.MMMM level there is a fourth identifier, FFFF, that deals with the fix level and is not used to designate an AIX level, but is used when dealing with the file sets that make up AIX.

Migration activity takes place when there is a change in the AIX operating system version or release, for example when going from AIX V3.2.5 to AIX V4.2.1 (starting VV is 3, final one is 4; starting RR is 2, latest is 2; starting MMMM is 5, latest is 1), or when the starting point is AIX V4.1.5 and the goal is AIX V4.3.1. Upgrading work has to be done when there is only a change in AIX modification level or PSSP level.

Note: Migration vs. Upgrade

Migration: Work to be done when changing AIX version or release.

Upgrade: Activity to be carried out when changing the AIX modification level or PSSP release.

When migrating or upgrading the nodes of an SP system to some PSSP level and its AIX prerequisite level, the path does not include any configuration changes to the system. Also, the CWS must be at the highest software levels of AIX and PSSP.

Attention

When migrating to or upgrading an SP system, the Control Workstation must be at the highest AIX and PSSP level in the system.

Beginning with PSSP 2.4, there will only be migration or upgrade support from the latest modification level of a specific AIX release. For example, PSSP 2.1 works under AIX 4.1.4. But this mix is unsupported for a PSSP upgrade to PSSP 2.4. The machine must first be upgraded to AIX V4.1.5. The same is true for AIX V4.2: it must be at AIX V4.2.1 at the time of this writing.

The supported change (migration or upgrade) paths are listed in the following table:

Going from	To
PSSP 2.1 and AIX 4.1.5	PSSP 2.4 and AIX 4.2.1, 4.3.1
PSSP 2.2 and AIX 4.1.5, 4.2.1	PSSP 2.4 and AIX 4.2.1, 4.3.1
PSSP 2.3 and AIX 4.2.1	PSSP 2.4 and AIX 4.2.1, 4.3.1
PSSP 2.3 and AIX 4.3.1	PSSP 2.4 and AIX 4.3.1
PSSP 2.4 and AIX 4.2.1	PSSP 2.4 and AIX 4.3.1

Table 9. Supported Change (Migration or Upgrade) Paths under PSSP 2.4

Note: There is no change path from PSSP 1.2 and AIX V3.2.5.

The very first step when upgrading is to apply specific PTFs to any node at PSSP level 2.1, 2.2 and 2.3. The document Memo to Users (Read This First), that is shipped with the PSSP has all the necessary information to perform this work. This document is also available at the Web at <http://www.rs6000.ibm.com>. Before going on, be sure to have all the necessary backups, such as mksysb's, System Data Repository Archive (SDRArchive), data backups and so on.

The next step is to upgrade the CWS. The upgrade that has to be done on the CWS can be split into two main tasks:

- CWS AIX upgrade
- CWS PSSP upgrade

The CWS AIX upgrade is a normal OS upgrade and is done using the install media. The CWS PSSP upgrade is fully described in *IBM Parallel System Support Programs for AIX: Installation and Migration Guide*, GC23-3898, in the chapter "Migrating to the Latest Level of PSSP".

Following the CWS upgrade, it must be decided if partitioning is needed, because there may be a need to have independent environments for testing purposes, coexistence limitations or some other reason. The procedures to carry out this partitioning task are detailed in *IBM Parallel System Support Programs for AIX: Administration Guide*, GC23-3897, in the chapter entitled "Managing System Partitions". If there is a need to partition the system, it is recommended to include a test phase to make sure everything is working as expected.

After the partition issue, it is strongly recommended to upgrade a single node to the right AIX and PSSP levels for testing purposes. This will show up any problem that may arise due to the upgrade work, and offer the opportunity to fix it without disrupting the whole SP environment. The boot/install server of this node must be the CWS, because it has already been upgraded to the correct software levels. The detailed steps to upgrade a node to the latest software levels of AIX and PSSP can be found in *IBM Parallel System Support Programs for AIX: Installation and Migration Guide*, GC23-3898.

When the test environment has been fully tested and everything is working as desired, it is time to migrate the rest of the nodes to the correct software levels of AIX and PSSP. Before carrying out this activity, the boot/install servers must be upgraded to the proper AIX and PSSP levels, because they are needed to upgrade the rest of the nodes. The procedure to accomplish this is described in *IBM Parallel System Support Programs for AIX: Installation and Migration Guide*, GC23-3898.

The next phase is to upgrade the rest of the nodes to the desired PSSP and AIX levels, for example PSSP 2.4 and AIX 4.2.1 or 4.3.1. The necessary steps are detailed in *IBM Parallel System Support Programs for AIX: Installation and Migration Guide*, GC23-3898.

The final step is to perform some post upgrade tasks. These are related to cleaning up the CWS's disk space that was used by previous versions of the PSSP software and AIX NIM of the previous operating system software, because of having to support, at some transition stage, a mix of PSSP and AIX levels. Specific procedures to follow can be obtained from *IBM Parallel System Support Programs for AIX: Installation and Migration*, GC23-3898, in the chapter "Migrating to the Latest Level of PSSP", section "Post-Migration Activity". Also, if there is a specific set of tasks to be carried out because of the customer's environment, this is the right time to do it.

5.2.3 Coexistence

In the SP system context, coexistence addresses the problem of having different nodes, *within a system partition*, running different PSSP levels and different AIX-supported levels and/or supported LPPs, with clearly stated constraints. This issue began to be addressed by PSSP V2.2 and continues to be addressed by the later PSSP releases.

With PSSP 2.4, IBM supports any mix of the following PSSP versions: 2.1, 2.2, 2.3 and 2.4, for nodes in the same system partitions.

Also, there is a minimum AIX level that corresponds to a PSSP version:

PSSP Level	AIX Level
2.1	4.1.5
2.2	4.1.5, 4.2.1
2.3	4.2.1, 4.3.1
2.4	4.2.1, 4.3.1

Table 10. PSSP and AIX Levels

Another important consideration regarding the coexistence issue is the CWS AIX and PSSP levels. In an SP system, the CWS must be at the highest AIX and PSSP levels that it will serve. This means that if the latest AIX and PSSP levels are 4.3.1 and 2.3, respectively, then the CWS must be installed with

AIX 4.3.1 and PSSP 2.3. The current PSSP level is 2.4 and the corresponding AIX level is 4.2.1 or 4.3.1.

Coexistence of LPP and PSSP components: There are restrictions for some LPPs and PSSP components when coexisting in the same system partition. If an installation is running any of the LPPs or PSSP components that are listed below, consult *RS/6000 SP: Planning, Volume 2, Control Workstation and Software Environment, GA22-7281*, to get more specific information about these constraints.

- IBM Virtual Shared Disks or IBM Recoverable Virtual Shared Disks
- LoadLeveler
- PIOFS, CLIO/S, NetTape
- General Parallel File System (GPFS)
- Parallel Application Products, such as PESSL, PVME and PE
- High Availability Group Services API
- Switch Support

5.3 SP User Management

This section discusses the activities the SP system administrator (SSA) has to perform to manage users, such as how they are created and maintained.

The objective of any user management system is to make sure that the user database, that is, the files that hold the user management information (*/etc/passwd*, */etc/group*, */etc/security/passwd* and */etc/security/group*), is consistent in the environment where the users work. This can be accomplished in a number of ways offered by the SP system software: SP User Management (SPUM), Network Information System (NIS), or to manage each user individually over each machine that comprises the network environment. The last method has a huge associated effort, because the system administrator has to maintain many files consistently across the network, so this technique is not the most effective.

So there are two realistic choices: NIS and SPUM. When using NIS, the NIS master machine holds the NIS maps that hold the user management information such as logname, home directory, hostname (where the home directory is located), password, group information, and default shell. With NIS as the chosen method for user management, there is the need to use automounter support to have the user's home directory automatically mounted, at the right point, when needed. As a consequence, automounter

maps must be created and administered. This automounter issue can be solved using the AIX automounter support, which creates and maintains the automounter maps. But the SP will not take care of the NIS administrative work. Furthermore, the SP does not interface with NIS.

If the selected choice is SPUM, then the SP will manage the user database using file collections and SP automounter support. This means that most of the user management work is totally automatic.

5.3.1 Configuring SP User Management

This task is done at SP system installation time, using the SMIT fastpath `site_env_dialog`, as shown:

```

Site Environment Information
Type or select values in entry fields.
Press Enter AFTER making all desired changes.
[MORE...7]
Automounter Configuration           true           +
Print Management Configuration      false          +
Print system secure mode login name [""]
User Administration Interface       true           +
Password File Server Hostname      [sp2en0]
Password File                       [/etc/passwd]
Home Directory Server Hostname     [sp2en0]
Home Directory Path                 [/home/sp2en0]
File Collection Management           true           +
File Collection daemon uid          [102]
[MORE...8]
F1=Help           F2=Refresh       F3=Cancel        F4=List
Esc+5=Reset      Esc+6=Command    Esc+7=Edit       Esc+8=Image
Esc+9=Shell      Esc+0=Exit       Enter=Do

```

Fields that have to be set to activate SPUM are:

- Automounter Configuration True
- User Administration Interface True
- File Collection Management True

Once this is done, the SP system will manage users from a single point of control, the CWS. Users will have a unique home directory across the whole SP system, mounted automatically when needed via NFS by the AIX automounter.

Automounter Configuration set to true means that this SP system will have configured and running this kind of service.

User Administration Interface set to true tells the installation process to integrate SP user management SMIT screens into the Security and Users SMIT menu.

Note: SP User Commands

When User Administration Interface is set to true and NIS is not used, the following AIX user management commands are linked to SPUM commands:

```
/usr/bin/chfn -> /usr/lpp/ssp/config/sp_chfn
```

```
/usr/bin/chsh -> /usr/lpp/ssp/config/sp_chsh
```

```
/usr/bin/passwd -> /usr/lpp/ssp/config/sp_passwd
```

The original commands are kept in the /usr/bin directory, renamed as cmd_name.aix.

Password File Server Hostname points to the host that will be the password file server for the SP system, and Password File tells the password file full pathname within the specified machine.

Home Directory Server Hostname is the name of the machine that will serve the user home directories, and Home Directory Path tells the directory name under which the user home directories will be created. This can also be set for users.

File Collection Management set to true establishes that the File Collection software must be configured.

5.3.2 AIX Users vs. SP Users

By "AIX user" is meant an account that has been created in a node using the standard AIX `mkuser` command or some of its derivatives. This user is only known to the AIX machine on which he was created. An SP user is one that was set up using the PSSP (`spmuser`) command on the CWS. His "spatial span" is the whole SP system.

Even when SPUM is in use, it is still possible to add, list, and delete users, and change some user attributes on a node basis. But this is not consistent with the SPUM user management policy. If an SP system administrator (SPSA) adds a user in a node that is under SPUM and the file collection is configured, then the user database file collection update mechanism will delete that user from the node in the next hour.

When SPUM is the user management technique used in the SP system, the SPSA can choose not to use file collection updates and to have a different user database synchronization scheme.

When an SP user wants to change his password, he must log on to the CWS. SP will not allow him to change the password when connected to a node.

5.3.3 SP User Management (SPUM)

Tasks related to SPUM are:

- Add an SP user
- List SP users
- Change SP user attributes
- Delete an SP user
- Change a user's password

Add an SP user: The process of adding an SP user is almost the same as adding an AIX user. Almost the same files are modified. The only parameter that `spmuser` needs to receive as argument is the user's name. Default values for primary group, group set and initial program are contained in `/usr/lpp/ssp/bin/spmuser.default`. The user's home directory default location is retrieved from the SDR SP class, `homedir_server` and `homedir_path` attributes. `spmuser` only supports the following user attributes:

- `id`
- `pgrp`
- `home`, in the format `<hostname:home_directory_path>`
- `groups`
- `gecos`
- `shell`
- `login`

`spmuser` generates a random password for the user, which is stored in the `/usr/lpp/ssp/config/admin/newpass.log` file, whose owner is root, and with 600 permissions. So root must read the file and tell the user his initial, system-generated password and delete the file.

When the appropriate period has elapsed, nodes pull the SPUM file collection from the CWS and update its configuration.

List SP users: This task is performed with the `splsuser` command, which shows only the attributes that are managed by the `spmuser` command. For example:

```
[sp2n15.msc.itso.ibm.com:~]# splsuser root
root id=0 pgrp=0 groups=system,bin,sys,security,cron,audit,perfmn
home=/ on sp2n15.msc.itso.ibm.com shell=/bin/ksh gecost= login=true
```

Change SP user attributes: This is done by the `spchuser` command. The attributes that can be modified are the same that are set with `spmuser`, except for the user name.

When the appropriate period has elapsed, nodes pull the SPUM file collection from the CWS and update its configuration.

Delete an SP user: This is accomplished with the `spruser` command. When the appropriate period has elapsed, nodes pull the SPUM file collection from the CWS and update its configuration.

Change a user's password: To change a user's password when pure SPUM is running, the user must log on to the machine that holds the master password file and change the password with `passwd`. If the server that contains the master password file is not the CWS, then the `/etc/passwd` and `/etc/security/passwd` files must be transferred to the CWS after changing the user password. The new files will be updated automatically to the nodes by the file collection update.

If NIS is being used, user passwords are changed with `passwd` or `yppasswd`.

5.3.4 Restricting User Access to the CWS

For security reasons, it is advisable to restrict CWS access. But if the SP system is pure SPUM, then users need CWS access, at least to change their passwords. IBM provides a script that has a configuration file to allow certain users full CWS access and restrict full login to others. The script is located in `/usr/lpp/spp/config/admin/cw_restrict_login`. The users that are allowed full CWS login access must have their names listed in `/usr/lpp/spp/config/admin/cw_allowed`, except for root. The script is designed to allow the root user full CWS access. The `cw_allowed` file format is one user per line, starting at the left-most column. No comments can be on that file. Users whose names are not included in `cw_allowed` and are not root, are

allowed CWS login only to change their password. After that action, they are logged out. To make this script work, it must be included in some login process control file such as `/etc/profile` in the CWS. To delete the `cw_restrict_login` function, it is only necessary to take it out from the CWS login process control file.

5.3.5 SP Access Control (SPAC)

SP systems can be divided into two major groups:

- LAN consolidation machines
- Parallel computing servers

In the first case, SP nodes are divided into several groups each one assuming different types of functions. So the SP system is probably partitioned. One node group could be serving a production database environment, another could be a development area, and maybe a third is being used for testing purposes. It is not desirable to allow all the users to have no login restrictions to all node groups or partitions because of performance and security issues.

In the second class of SP systems, the SP appears to the user community as one machine and it does not matter where a user logs in or from where a job was submitted to be executed. In this later scenario it is very desirable to deny users login capability, so if there is a central entity that manages the node pool that performs parallel job processing, the pool performance will not be degraded at all.

In any case, certain login control must be enforced. And this is where SPAC or Login Control enters in. Its function is to perform login control on the nodes. SPAC was initially designed to work with the SP Resource Manager (RM). The RM is a central entity that receives job execution requests, selects the appropriate nodes to carry out the job, prevents parallel jobs from interfering with each other, and reports job related node data. The idea behind SPAC is to restrict user access on a node. When the parallel job requests are received by the RM, it uses login control to grant node access to these users and to their jobs. Afterwards, when the jobs have completed, the RM again uses SPAC to restrict user access to the machines. SPAC use is not restricted to nodes that are part of the RM pool. Login control can be activated/deactivated with the `spacs_cntrl` command, but this must be carried out with extreme care on nodes that are managed by the RM, because it can interfere with the RM functions.

SPAC bases its work upon the `/etc/security/user` file and the user attributes `login` and `rlogin`. By changing `login` and `rlogin` dynamically, user node login control is achieved. SPAC controls user access through the `login` command (local connection) and `rlogin` restricts user login via `rlogin` and `telnet` commands (network connection). Both attributes may have true or false values.

Implementing SPAC: Login control can be used by the SPSA or by a job submission system, such as RM. The heart of this control is the `spacs_cntrl` command, which uses four keywords to manage login control. The arguments `block` and `unblock` are used by the SPSA, in the node where SPAC needs to be controlled, to grant or prohibit node access to a user or set of users. `allow` and `deny` keywords are used by a job submission system to update the user node access state.

When a job submission system requests a user access change two or more times, using the keywords `allow` or `deny`, the file `/var/spacs/spacs_data` is created.

The first `allow` or `deny` request updates the `/etc/security/user` file accordingly. Starting the second request (`allow` or `deny`) submitted by the job submission system (JSS), the file `spacs_data` file is created and the requests are "stored" there under certain rules.

Also, starting with the second request, a counter is set up. For the sake of clarity, it will be assumed that the second JSS request was a `deny` request. If the third request by the JSS is also a `deny` request, only one pending `deny` request will be held. If the third request by the JSS is to `allow` access, this `allow` request compensates the pending `deny` and the `spacs_data` file will hold no JSS request for a specific user on the node, but the user state may be uncertain. It will depend on what was going on when the file cleared.

When the `block` and `unblock` options are used by an SPSA, there is a SPACS deterministic behavior. The `block` option will clear the `spacs_data` file and deny user access to the node. The `unblock` option will clear the `spacs_data` file and allow user access to the node.

So, as can be seen, it is not advisable to use the keywords `block` and `unblock` on a node where the JSS is submitting jobs. If this is done, there will be a loss of state information. To correctly update the `spacs_data` file, SPAC generates a lock file under `/var/spacs/spacs_cntrl.lock`.

The services that are denied when using `spacs_cntrl` command are:

- `login`

- rlogin
- AIX rsh
- AIX rcp
- AIX rexec
- SP rsh
- SP rcp

To disable FTP, the `/etc/ftpusers` file or the AIX `ruser` command should be used.

Command line example: to disable user access on node f9n13 for user edu, the following `spacs_cntrl` command should be issued on machine f9n13:

```
/etc@f9n13#> spacs_cntrl -v -l block edu
```

The `v` flag specifies verbose output and the `l` flag tells the command to log its output to `/var/adm/SPlogs/spacs/spacs.log`.

The complete `spacs_cntrl` command description can be found in *IBM Parallel System Support Programs for AIX: Command and Technical Reference*, GC23-3900.

There is more about SPAC in *IBM Parallel System Support Programs for AIX: Administration Guide*, GC23-3897, in the chapter "Managing User Accounts".

5.4 Problem Management

The SP reports and acts on system problems in many ways. In general, SP problem management exploits AIX and the rich services of HAI. This section presents the facilities that come standard with the SP:

- AIX, BSD, and PSSP-specific error logging
- HAI clients (Problem Management)
- IBM Support Tools

5.4.1 AIX, BSD and PSSP-specific Error Logging

The SP uses AIX and BSD error logging to report errors. IBM-specific software components generally use the AIX facility. BSD error logging is needed to support the public domain codes, such as AMD, NTP, and File Collections. AIX includes the BSD error logging capability as standard function.

To report on SP-specific hardware like power supplies, fans, and most errors detected by hardmon (the hardware monitoring daemon), PSSP provides the splogd daemon. splogd also interacts with AIX and BSD error logging.

Since every node and the CWS have AIX and PSSP installed, the facilities of these software components are available SP-wide. For this section, reference to a node means SP nodes as well as the CWS.

As well as using these error logging facilities, most SP subsystems create their own error log files. These log files may be on the CWS, node, or both, depending on the SP subsystem. Most of the log files are located in the /var/adm/SPlogs directory. A comprehensive list of SP error log files is given in *IBM Parallel System Support Programs for AIX: Diagnosis and Messages Guide*, GC23-3899.

5.4.1.1 AIX Error Logging

The AIX error log is often one of the first places to check for problems on a node. AIX error logging is implemented by the errdemon daemon and the error log file /var/adm/ras/errlog. The kernel, an application, or a command can all generate errors which errdemon traps, formats, and writes to the log.

Each node runs AIX error logging by default. PSSP provides functions built on sysctl to consolidate the nodes' logs at the Control Workstation (CWS), providing a single point of control.

AIX Error Notification Facility

The AIX Error Logging facility can be configured to take action based on the occurrence of a specific error via the AIX Error Notification Facility (AENF). After logging an error, the errdemon checks to see if that error entry is defined to AENF. If so, the system executes the action specified in the AENF configuration for that error. For example, if an error with label CORE_DUMP is logged, one could define an error notification object for that label to page a system operator.

For details on this handy AIX function, see *AIX V4.1 Problem Solving Guide and Reference*, SC23-2606.

5.4.1.2 BSD Error Logging

BSD Syslog is a widely implemented and well-known logging facility. It is implemented by the syslogd daemon and its configuration file, /etc/syslogd.conf. syslogd routes error information from applications and daemons to user-defined logs, and can organize errors by type and priority. A syslogd client can also log general events, not just errors, to a single log file, which is often a great convenience when debugging problems.

PSSP exploits syslogd for its many daemons, mostly for information reporting as opposed to error reporting. See *IBM Parallel System Support Programs for AIX Command and Technical Reference*, GC23-3900, for more information.

5.4.1.3 PSSP-Specific Error Logging

PSSP provides the splogd daemon to report on SP hardware errors and state changes. splogd forwards errors to both AIX and BSD error logging, and writes hardware state changes to a file. Optionally, you can specify a script to run when a specific state change occurs. By default, splogd executes on the CWS, but it can be installed and configured to run on any node if you wish to off-load work from the CWS.

5.4.2 Problem Management (PMAN) Subsystem

PMAN provides an infrastructure for recognizing and acting on problem events. PMAN is an EM client application, and is packaged with PSSP in the ssp.pman fileset. Use of PMAN requires Kerberos authentication, discussed later. Its function is presented in Figure 115 on page 236.

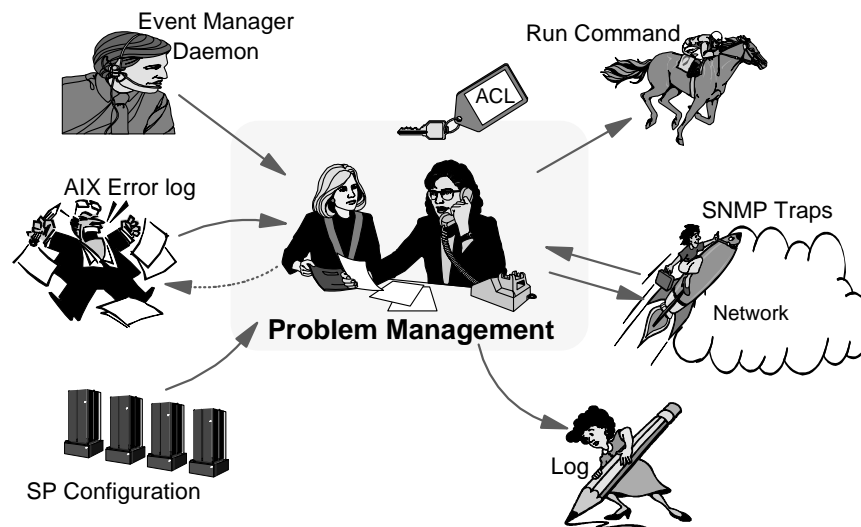


Figure 115. PMAN Design

PMAN receives events from EM, and can react in one or more of the following ways:

- Send mail to an operator or administrator

- Notify all logged-on users via the `wall` command or popping up a window on displays with a graphical user interface
- Generate a Simple Network Management Protocol (SNMP) trap for an enterprise network manager, such as Tivoli
- Log the event to AIX and BSD error logging
- Execute a command or script

Three daemons constitute the PMAN subsystem. They are shown in Figure 116 on page 238.

- **pmand**
This daemon directly interfaces to the EM daemon. `pmand` registers for events, receives them, and takes actions. PMAN events are stored in the SDR, and `pmand` retrieves them at initialization time.
- **pmanrmd**
If EM does not monitor the resource variable of interest, PMAN supplies its own RM, `pmanrmd`, to access the resource variable's data. You configure `pmanrmd` to periodically execute programs, scripts, or commands and place the results in one of EM's 16 user-defined resource variables. `pmanrmd` supplies resultant data to the RMAPI. It also cooperates with `pmand`.
- **sp_configd**
This daemon creates SNMP traps from the event data in `pmand`, and is the interface for an enterprise network manager to access SP event, configuration, and resource variable data. This is further described in 6.3, "Tivoli" on page 288.

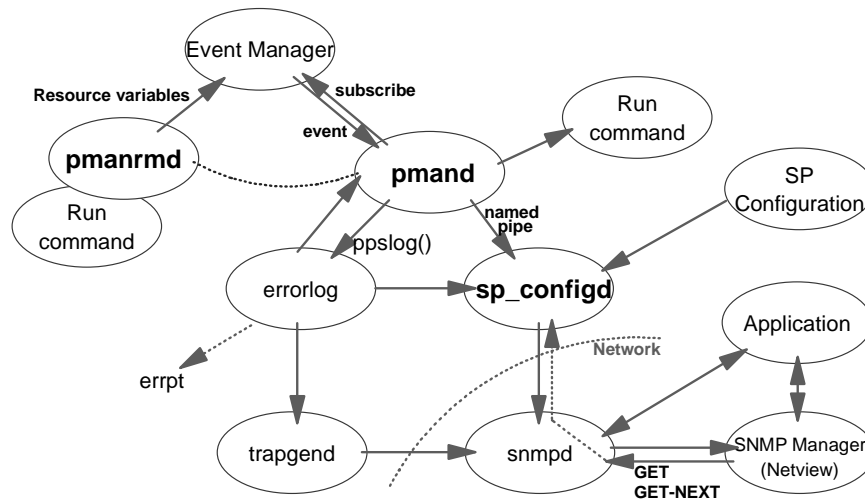


Figure 116. PMAN Daemons

The SP exploits AENF to link the AIX error log and PMAN. When a node is installed, an AENF object is created which sends all AIX Error Log entries to PMAN. PMAN filters the entries based on subscriptions you define. The sp_configd daemon picks up SNMP-alertable AIX errors and passes them to snmpd for forwarding to the network manager.

By default, all PMAN daemons are started on all nodes and the CWS. It is important to understand that PMAN is *not* a distributed application, but an EM client. The PMAN daemons on one node do not know about their counterparts on the other nodes, and do not care. At initialization, each instance of pmand obtains PMAN event subscriptions from the SDR, so each node is aware of all PMAN events to be monitored. Although it is not mandatory to run the PMAN daemons everywhere, we do not recommend disabling them. PMAN must execute:

- Where you want an action to originate, given a specific event from EM.

For example, you could register with EM to monitor for the /tmp file system exceeding 95% on node 5, but have node 10 execute myscript when that event occurs. EM takes care of notifying PMAN on node 10 that node 5's /tmp is over 95% full; PMAN on node 5 does not care about this event. Technically, PMAN does not even need to run on node 5.

- Where you need custom resource monitoring, facilitated by the pmanrmd resource monitor.

As an EM client, PMAN is an efficient infrastructure to respond to events across the system. For example, suppose your system-wide policy is to monitor for /tmp exceeding 95%. Instead of creating and maintaining the required configuration on each node, use PMAN to subscribe to that event for all nodes. The EM distributed infrastructure monitors for you, and notifies your desired PMAN client (usually the node where the event occurs) to take appropriate action; furthermore, newly installed or re-installed nodes will automatically fall under the /tmp monitoring scheme. This is because the PMAN event is globally defined in the SDR, and PMAN is configured to run by default on all nodes.

The pmandefaults script sets the default events to be monitored, and is an excellent starting point for configuration of the PMAN subsystem. The script is documented in *RS/6000 SP: PSSP2.2 Survival Guide*, SG24-4928, and also in *IBM Parallel System Support Programs for AIX Administration Guide*, GC23-3897.

PMAN Security Considerations

PMAN's event registration is based on sysctl, which uses Kerberos for user authentication. If you register an event in PMAN, you must have valid Kerberos credentials. In addition, your Kerberos principal must be listed in the /etc/sysctl.pman.acl file on the local node, in order to store the subscription in the SDR, as well as on all nodes that are affected by the new subscription, in order for the affected Problem Management daemons to be notified of the new subscription.

The Kerberos principal that was in effect when the subscription was created is stored as part of the subscription definition. Only a user with the same Kerberos principal can modify the subscription. PMAN also uses the stored Kerberos principal to decide whether the action caused by an event should be allowed. Before a PMAN daemon on a node performs an action, it checks to see whether the Kerberos principal that is stored in the subscription definition is authorized to perform the requested action on the node. If the requested action is to write an entry in the AIX Error Log and BSD syslog or to generate an SNMP trap, then the Kerberos principal that owns the subscription must have been listed in the root user's \$HOME/.klogin file. If the requested action is to execute a command, then the Kerberos principal must be listed in the \$HOME/.klogin file of the user to run the command, by default the creator of the subscription.

5.4.3 Perspectives

Perspectives is also a client of HAI. It provides a graphical user interface for SP monitoring. Perspectives also provides the configuration function of

PMAN (PMAN can also be configured using the command line). Perspectives is described in 5.1.1, "SP Perspectives" on page 213.

5.4.4 IBM Support Tools

5.4.4.1 Service Director

Service Director is an offering from IBM which helps customers automate the maintenance of their SP system. Service Director does the following:

- Monitors and analyzes system errors, and can place a service call automatically to IBM without customer intervention. A customer-supplied modem and analog line is required. Footnote: The automatic call function may not be offered in every country.
- Monitors and reports on supervisor, HiPS, SP Switch hardware-related faults and checkstops as recorded in the AIX error log.
- Allows manual or custom Problem Management Record (PMR) creation from an X-Windows display. IBM support personnel use PMRs to document customer problems, from initial call to resolution.

Customers receive Service Director free if their SP is covered by an IBM Warranty or IBM Maintenance Services Agreement.

5.4.4.2 Service Collections

Often, IBM Support personnel need to gather a great deal of detailed information to help isolate problems. To ease this process, the SP allows you to create service collections. A service collection can comprise:

- Configuration files
- Error and other log files
- Command line outputs

Each major SP component has a service collection table which defines the information to be gathered to help solve a problem with that component. You can collect data by node or groups of nodes. For example, if your AMD automounts are failing and IBM Support personnel are involved, they may ask you to execute the service collection process for AMD. The output will be a compressed, tar-format file that you can copy to diskette or tape, or send electronically to IBM Support for analysis.

5.5 System Partitioning

System partitioning is a method for organizing the SP into non-overlapping groups of nodes for various purposes such as testing new software and

creating multiple production environments. This section provides an overview of system partitioning and describes how you can partition and then manage the partitions on your SP.

5.5.1 Overview

A system partition is a fairly static entity that contains a specified subset of SP nodes with a consistent software environment. System partitioning lets you divide system resources to make your SP system more efficient and more tailored to your needs while keeping system management simple and efficient. At the most elementary level a system partition is a group of nodes working together.

The Control Workstation does not belong to any system partition. Instead, it provides a central control mechanism for all partitions that exist on the system.

5.5.2 Considerations for Partitioning

There are some basic rules that should always be followed when you set up the partitions. These impose some limitations on the configurations that can be selected for the partitions.

- Nodes in a partition must not cross the switch chip boundaries. Putting it another way, all nodes on a switch chip must be in the same partition. The nodes have predetermined fixed connectivity with each switch board. They are:

Node slots 1, 2, 5, and 6 always connected on same chip

Node slots 3, 4, 7, and 8 always connected on same chip

Node slots 9, 10, 13, and 14 always connected on same chip

Node slots 11, 12, 15, and 16 always connected on same chip

That is, in a single-frame, single-switch system, with two high nodes in slots 1 and 5 (so they are connected to the same switch chip), it would not be possible to have these two nodes in different partitions.

- HACMP nodes must be in the same partition.
- Shared hard disks must be connected to nodes in the same partition.
- Virtual Shared Disks (VSDs) cannot span partitions.

Figure 117 on page 242 shows how a single-switch system with 16 thin nodes could be partitioned using one of the layouts for the 8-8 configurations.

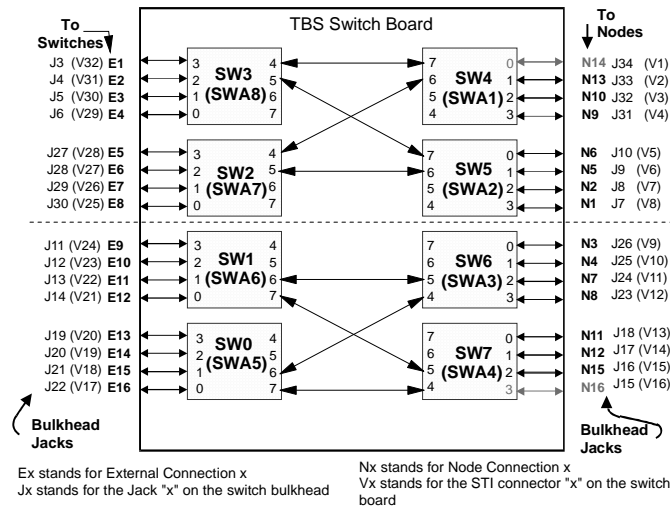


Figure 117. Example of an 8_8 Configuration

Where:

J3 to J34 Bulkhead jacks for data cables

V1 to V32 Data connectors on the switch board (silk screen)

SW0 to SW7 Logical identification of switch chips

- In output files, add 10,000 to the logical number.
- The frame number is used to build the logical number. Multiply it by 10.

U1 to U8 Master switch chip silkscreen on the board

N1 to N16 Node ports (physical numbering)

E1 to E16 Switch-to-switch ports

5.5.3 Partitioning Justification

Partitions are separate logical SPs that need to have separate applications and separate management. There could be many advantages in having partitions on the SP system, some of which are:

- Separate test and production environments: For example, you may wish to configure a system partition to use for non-disruptive testing when you

migrate to a new release of software. Or you may wish to configure system partitions for multiple isolated production environments so that the work load in one environment will not interfere with the work load in another environment.

- There are coexistence constraints, for different levels of PSSP, that is, they cannot reside in the same system partition. For example:
 - PSSP 1.2 and PSSP 2.1 cannot coexist in the same system partition.
 - PSSP 2.2 provides support for coexistence of one additional level of PSSP in the same system partition.
 - PSSP 2.3 provides support for coexistence of up to two additional levels of PSSP in the same system partition.

Note

The Control Workstation must always be at the highest level of PSSP and AIX that is used by the nodes.

- Distributed system management responsibilities: If there is more than one SP administrator (for example, one for the test environment and another for the production environment), then having each environment in a separate partition keeps the system administrators' responsibilities isolated to their respective partition. Separate views in system management tools (perspective) and command line interface also prevent interference between environments. The `SP_NAME` environment variable, if present, identifies the system partition to subsystems. Otherwise, the application uses `SDR_dest_info` file.

5.5.4 Partitioning the SP System

System partitioning can be achieved by applying one of a fixed set of supplied configurations. The `ssp.top` option in the PSSP install package contains a directory structure with all relevant files for partitioning the system with supplied system partition configurations. In addition, the `ssp.top` option contains the System Partitioning Aid, a tool for creating customized system partitions to suit your needs.

In the beginning, after you have installed the PSSP software, but before you have configured system partitions, there is only one system partition corresponding to the name and IP address of the Control Workstation returned by the `hostname` command. This is the *default or persistent* system partition. It always exists. When you configure more than one system

partition, one of them must be defined as the default system partition with this name and IP address.

The steps needed to partition your SP system are as follows:

Step 1: Archive the System Data Repository (SDR).

Step 2: Define alias IP addresses and names for all but the default system partition.

Step 3: Check if the desired configuration is supplied. If not use the System Partitioning Aid to generate and save a system partition configuration.

Step 4: Select a system partition configuration and layout to apply.

Step 5: Customize (name and specify attributes) the system partitions.

Step 6: Shutdown all nodes in the partition that is being configured.

Step 7: Run the `setup_server` command on the Control Workstation.

Step 8: Apply the configuration with the `spapply_config` command. The tasks carried out by this command are as follows:

- The SDR is restructured for the partitions and multiple `sdrd` daemons are created and started.
- The heartbeat (if PSSP 2.1 or earlier), HA subsystems, and `host_responds` are restructured and multiple daemons are created and started.
- The switch gets configured according to the new partition.
- It validates that the VSD configuration is consistent with the partition being set up.

Step 9: It reboots all nodes in the newly created system partitions.

5.5.5 Repartitioning or Modifying an Existing System Partition

If you wish to change or add system partitions, the above steps remain valid, except that you need to run the `Eunpartition` command before making any changes to the partitions (step 5). This command is required only on an SP system with a switch.

5.5.6 Syspar Directory Structure

Figure 118 on page 245 provides an example of a typical directory structure for a single-frame, 16-node system that has two partitions.

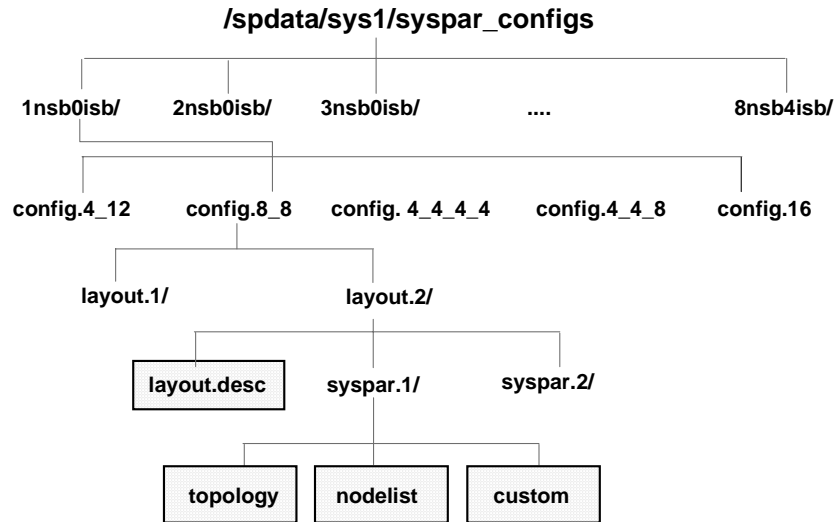


Figure 118. Directory Structure for Topology Files

The `/spdata/sys1/syspar_configs` directory is at the top-most level and holds all the configuration files and directories that contain information about RS/6000 SP partitions. These are:

- The `/spdata/sys1/syspar_configs/NSBnumnsb/ISBnumisb` directory is named for the number of Node Switch Boards (NSB) and Intermediate Switch Boards (ISB) used in your configuration. In this example it is `/spdata/sys1/syspar_configs/1nsb0isb`. Under this directory you will find system-supplied configuration directories.
- The `/spdata/sys1/syspar_configs/config.8_8` directory contains the configuration that we have selected in our example. Here `config_8.8` means that we have selected a configuration that will make partitions of 8 nodes each.
- The `/spdata/sys1/syspar_configs/config.8_8/layout.3` directory determines the selected layout. This layout determines which nodes will go into which partition. The file `layout.desc` (also in this directory), describes the actual layout in terms of which node goes into which partition.

- /spdata/sys1/syspar_configs/config.8_8/layout.3/syspar.1 and /spdata/sys1/syspar_configs/config.8_8/layout.3/syspar.2 are the syspar directories, one for each partition.
- In each of the syspar.n directories, there are three files:
 - **topology**
This file contains the necessary topology information for the switch. Sometimes the topology files provided with PSSP do not meet your needs. In such cases, the system partitioning aid can be used to create custom topology files.
 - **nodelist**
The nodelist file is similar to the layout.desc file, except that it describes the nodes that belong to its partition only.
 - **custom**
Only the custom file gets created at the time of partition creation. The other files are selected as supplied to match your partitioning requirement. This file contains information on partition name, PSSP level, default mksysb name for the partition, primary node for the partition, backup primary node for the partition, and topology file name.

5.5.7 Managing System Partitions

Some of the subsystems that run in the SP environment operate within the domain of a system partition, rather than the domain of the system as a whole. These subsystems are called partition-sensitive.

To help you manage these subsystems, PSSP includes the Syspar Controller, which uses the `syspar_ctrl` command. With this command you can add, start, refresh, and delete partition-sensitive subsystems, as well as control aspects of their operation, such as tracing.

You can list the partition-sensitive subsystems with `syspar_ctrl -E`. The order in which the subsystems will be listed in the output is the order in which they are added or started by this command.

The partition-sensitive subsystems are as follows:

hats - Topology Services subsystem, which provides information to other PSSP subsystems about the state of the nodes and adapters on the SP system.

hb - Heartbeat subsystem, which communicates with several PSSP subsystems as part of providing information about the state of the nodes and

adapters on the SP system. This is provided only for compatibility with PSSP 1.2 and PSSP 2.1.

hags - Group Services subsystem, which provides facilities for coordinating and monitoring changes to the state of a client program that is running on a set of nodes in an SP system.

haem - Event Management subsystem, which compares information about the state of system resources provided by resource monitors with requests for information by client programs, and creates events that notify the clients when a match occurs.

hr - Host_Responds subsystem, which provides information about the state of the nodes on the SP system.

pman - Problem Management subsystem, which provides the ability to execute the actions when events are generated.

emon - Emonitor subsystem, which monitors the status of nodes in a switch network, bringing a node back into the switch network when necessary. This subsystem is not started by the `syspar_ctrl` command. It is started using the `Estart -m` command.

sp_configd - SNMP support portion of the Problem Management subsystem.

emcond - Event Management Conditions subsystem, which loads the SDR with predefined events or conditions of interest. These conditions are used by the Event Perspective GUI.

5.6 Performance Management

Performance management is a complex discipline, requiring clear definition of your performance goals, intimate understanding of your workloads, configuration of a myriad of parameters in a controlled manner, collection of relevant performance data, and expertise in interpreting the data. This section is not a comprehensive how-to on SP tuning, but describes general SP performance considerations and tools.

The RS/6000 SP Network Performance & Tuning Workshop, Q1097 (offered through IBM Education and its partners), is an excellent source for practical SP performance management information and tips. PSSP Version 2.3 and earlier revisions of *IBM Parallel System Support Programs for AIX Administration Guide*, GC23-3897, contain a chapter on SP tuning. This chapter has been removed in the PSSP Version 2.4 revision of the manual.

The performance material is now maintained on the web at
<http://www.rs6000.ibm.com/support/sp>.

5.6.1 General SP Performance Considerations

SP tuning is generally an exercise in optimizing networks. SP system management, interprocessor communications, shared disk access, interconnection to the enterprise, and programming depend heavily on thoughtful tuning and exploitation of networks. Once the SP networks are optimized, you tune an individual SP node the way you would any stand-alone RS/6000 machine.

In this section, we present general SP performance considerations.

5.6.1.1 Large Configuration

Since an SP may grow to 512 nodes, the ground rules and assumptions used in tuning small networks of RS/6000 machines change in an SP environment. SP tuning requires strong change management discipline, which stems from the complexity of large system configurations.

In large SP configurations, a bottlenecked node (such as a communications gateway) can affect all other nodes in the system. Bottlenecks in large systems have correspondingly amplified negative impacts. For example, if the SP administrative Ethernet is overloaded by heavy NFS traffic caused by multiple concurrent node installs, the normally light load of the HAI daemons can exacerbate the network problem. The daemons will step up their communications to cope with the apparent unavailability of resources, just making things worse.

5.6.1.2 SP Subsystems and Daemons

PSSP introduces many subsystems and daemons to manage the SP. In the default installation, these facilities present a light load to the system, but as you start to exploit their function, you must be wary of potential performance impacts. You could monitor for hundreds of resource variables with HAI, or mistakenly monitor one resource variable that triggers a steady stream of events (such as CPU usage greater than 0 on a node, and log the message). SP subsystems are generally efficiently designed so they can scale, but they can also be abused.

5.6.1.3 SP Administrative Ethernet

Care must be taken in the design and loading of the SP administrative Ethernet. Many system management functions and enablers use the SP administrative Ethernet, such as node installation and customizing, LoadLeveler, Parallel Operating Environment (POE), and HAI. The SP

administrative Ethernet may also be connected to your external networks, and open to traffic in your enterprise.

Generally, SPs larger than 16 nodes have multiple boot/install servers on physically separate Ethernet LAN segments. This maintains adequate network performance, particularly for intensive system management tasks such as node installation.

5.6.1.4 High-Speed Interconnect Network: SP Switch

The SP Switch is the interconnect fabric of the nodes. Many network tuning parameters affect all network interfaces on a node. Nodes of a switched SP system always have at least one other network - the administrative Ethernet - and usually other networks connecting into the enterprise. Optimizing these global parameters for such disparate networks as a slow Ethernet and a high-speed SP Switch is not trivial, and usually involves trade-offs.

The SP Switch introduces specific tuning parameters. You can tune the switch's device drive buffer pools, *rpool* and *spool*, by changing the *rpoolsize* and *spoolsize* parameters on the node's switch adapter. These pools are used to stage the data portions of IP packets. Their sizes interact with the node's network options settings. A detailed discussion of *rpool* and *spool* is presented in *IBM Parallel System Support Programs for AIX Administration Guide*, GC23-3897.

The small extra latency incurred by traversing intermediate switch connections in SP systems greater than 80 nodes may also be a consideration where consistent, precision results are required. For example, if a parallel job is dispatched to a set of nodes that all belong to one side of the intermediate switch connection, the execution time will be slightly faster than if the job is dispatched to the same number of identically configured nodes residing on both sides of the interconnect (the next day, for example).

Successful tuning of the SP Switch is fundamental. In a shared-nothing architecture, the interconnect fabric supports interprocessor message passing. Problems with the interconnect fabric potentially degrade performance of distributed and parallel applications.

5.6.1.5 Large Volumes of Data

SP systems typically have large volumes of data distributed across many nodes. Access to and movement of data within the system is critical to overall system performance. NFS, VSDs, PIOFS, and GPFS are all mechanisms for sharing data in the SP. Each has particular tuning considerations and techniques, but fundamentally, an I/O request eventually comes down to a

disk subsystem attached to a node. You must ensure that the I/O subsystem itself is not a bottleneck; otherwise, higher-level tuning will be fruitless.

Detailed tuning information for VSD and RVSD (the basis of GPFS) can be found in *IBM Parallel System Support Programs for AIX Managing Shared Disks*, SA22-7279.

5.6.1.6 External Networks

When you connect external networks to the SP, care must be taken to optimize the exchange of small data packets (typical of Ethernet, Token Ring, and similar slower-speed networks) and the large data packets of the SP Switch.

One of the final SP installation steps is setting all network adapters in SP nodes and the CWS to their maximum transmit queue size. The AIX defaults are typically too small to handle SP system traffic. For example, by default an Ethernet adapter has a transmit queue size of 30 to 64 Maximum Transmission Units (MTU). An Ethernet MTU is usually 1500 Bytes. 30 MTUs represent 45,000 Bytes, which is less than a single packet on the SP Switch, which is 65,520 Bytes. In this case, a node will experience network performance problems if it forwards SP Switch traffic to the Ethernet interface.

5.6.1.7 Application Considerations

Parallel application programmers must be conscious of the architecture of a system with an interconnect fabric for the processors. For example, an application using socket communications across the SP Switch must close the socket after use. Open, unused sockets are a common cause of SP performance problems as they waste system resources, such as communications buffers, which other working sockets cannot access.

5.6.2 Performance Management Tools

As part of the RS/6000 family of AIX machines, the SP inherits a set of mature performance management tools. In addition, IBM has enhanced the tools to scale with the SP.

5.6.2.1 AIX Facilities

AIX provides all the classic UNIX resource monitoring tools. Thus, each node's resources can be interrogated at the operating system level. Examples of these tools include the commands `sar`, `vmstat`, `iostat`, and `netstat`.

AIX Version 4 includes Performance Diagnostic Tool (PDT). PDT optionally runs on individual nodes, and assesses the system state, and tracks changes in performance and workload. It tries to identify impending problems and suggest solutions before they become critical. PerfPMR is another tool available with AIX Version 4. It is used by IBM Software Service to gather performance data on a node for analysis by IBM personnel, but the PerfPMR can also be used by customers.

The SP currently offers no specific reporting consolidation or meta-level commands in this area. AIX resource monitoring capabilities are well explained in *AIX Performance Monitoring and Tuning Guide*, SC23-2365.

5.6.2.2 Reporting Resource Data Information

Performance Agent, a component of Performance Toolbox/6000, is a component of the resource monitoring facility of HAI. By itself, Performance Agent only supplies data from performance related resource variables to the SMPI; it offers little in the way of viewing or analyzing the data. IBM supplies program products specifically to exploit the performance data from Performance Agent.

Performance Toolbox (PTX/6000)

PTX/6000 comprises the Performance Manager (PM) and Performance Agent (PA), or perfagent.server. The latter is the exact same code that ships automatically with the SP. The PM component includes X-Windows and Motif-based applications that provide:

- Realtime, color graphic performance monitors for local and remote systems, such as 3dmon.
- Performance analysis tools.
- Performance tuning controls.

PTX/6000 has a long, successful heritage of monitoring the performance of RS/6000 stand-alone machines. PTX/6000 is a separately orderable and priced IBM Licensed Program Product (LPP).

With reference to Figure 119 on page 252, the xmservd daemon of PA feeds the performance statistics into shared memory. While the aixos resource monitor within the EM daemon can supply AIX-level resource variable data selectively, as per EM client event registrations. xmservd can be further configured for specific resource monitoring needs. PM can simply act as another client of the SPMI interface (or Local Data Consumer in PTX/6000 vernacular), just as the Resource Monitors of Event Manager. The diagram introduces another interface into shared memory: the Remote Statistics

Interface (RSI). This interface allows a PM instance on another node (shown as a Remote Data Consumer, in this case the 3dmon monitoring program of PM) to access the local node's shared memory. This way, one PM installation can monitor a cluster of machines at the same time.

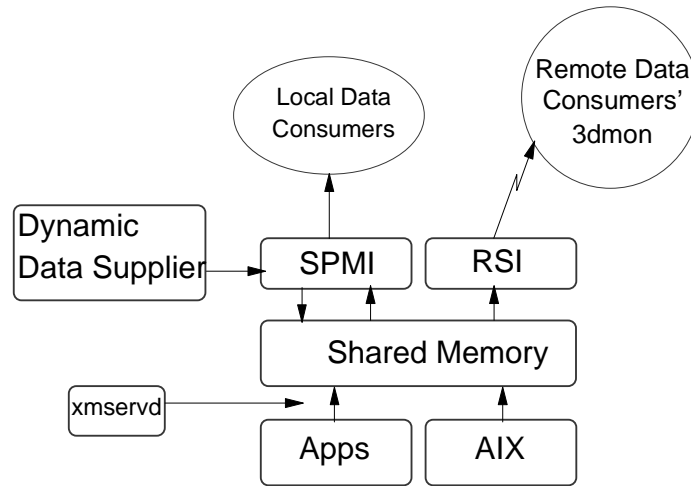


Figure 119. PTX/6000 Functional Overview

Although PA is an integral part of the SP's software implementation, PM offers no SP-specific function. Its usability does not scale: on larger SP installations (greater than 24 nodes), its graphical monitoring tools cannot accommodate on the screen all the performance statistics of all nodes. PM is not aware of specific SP hardware and software enablers.

Performance Toolbox Parallel Extensions (PTPE)

PTPE extends the function of PTX/6000 to better monitor the SP. The implementation of PTPE is shown in Figure 120 on page 253 and described in the discussion of PTPE capabilities.

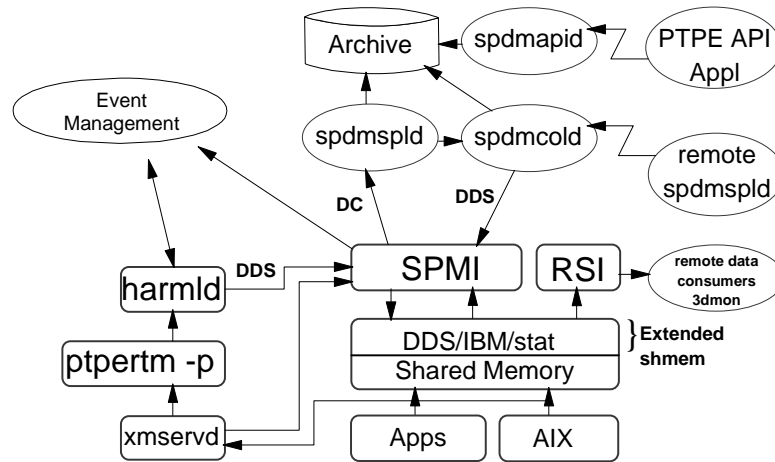


Figure 120. PTPE Architecture

In addition to the capabilities of PTX/6000, PTPE provides:

- **Collection of SP-specific data.** PTPE provides `ptpertm`, an additional data supplier that complements the data `xmservd` collects. The SP-specific performance data is currently implemented for:
 - The SP Switch
 - LoadLeveler
 - VSD
- **SP runtime monitoring.** The system administrator should have a global view of SP performance behavior. With reference to Figure 121 on page 254, similar nodes of the first tier, or Collectors, can be grouped and their performance data summarized by their respective Data Manager node in the second tier. This way, large SP systems can be easily monitored from a single presentation application by viewing node groups instead of individual nodes. The Data Managers are administered by the Central Coordinator in the third tier. The Central Coordinator aggregates the Data Managers' summary data to provide a total performance overview of the SP. Of course, the base PTX/6000 monitoring functions can be used to focus on any particular performance aspect of an individual node.

The PTPE display monitors on the Data Manager receive data from `spdmspld` (SP Data Manager SamPLer daemon) on the local node. To acquire data from remote Collectors, `spdmspld` communicates with `spdmcold` (SP Data Manager COLlector daemon) on the remote nodes, which in turn get the data from their associated `spdmspld` daemon. The

communications between these daemons is via a private protocol. Note that PTPE does not use the same mechanism as PTX/6000 to collect data from remote nodes.

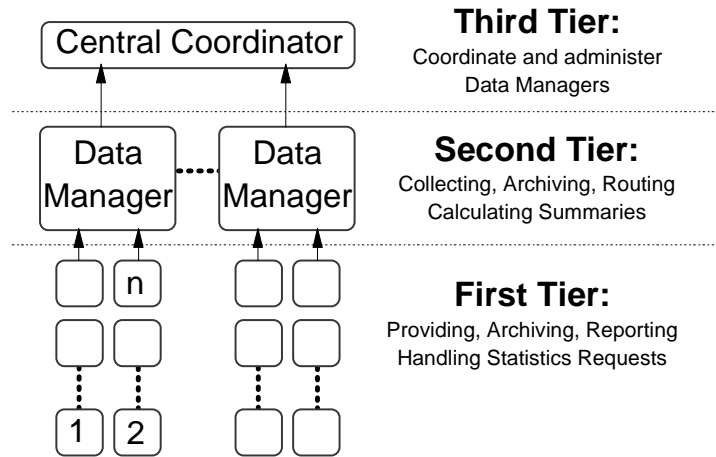


Figure 121. PTPE Monitoring Hierarchy

- Data Analysis and Data Relationship Analysis.** PTPE provides an API to allow analysis applications to sift through all requested data. The data archive created by PTPE exists on every node, and is completely accessible through the API using the `spdmapid` daemon (see Figure 120 on page 253). In base PTX/6000, performance data is analyzed with the `azizo` utility, which is restricted to simple derivatives like maximum, minimum, and average. With the PTPE API, programs of any statistical complexity can be written to find important trends or relationships. Also, data captured for `azizo` use is far more limited with base PTX/6000.

PTPE is a separately orderable and priced feature of the PSSP LPP.

5.6.2.3 Performance Event Monitoring with HAI Clients

Performance-related resource variables from PA are part of Event Manager's standard suite of monitored resources variables. You can always use your own programs or `PMAN` to interface to HAI to monitor and take action on performance-related events. See 4.5.6, "Event Management" on page 127 and 5.4, "Problem Management" on page 234.

5.6.3 Other Performance Management Aides

This section discusses additional program products and aides that help manage SP performance.

5.6.3.1 IBM-supplied Network Tuning Sample Files

The default SP install process configures the nodes with SP-specific network settings instead of standard AIX ones. You will further alter these settings based on your interfaces and workloads, but most customers new to the SP find this a daunting task. To help new customers approach the network tuning task, IBM offers (beginning with PSSP V2.3) sample files containing further-customized network settings for different environments. These files are located in the `/spdata/sys1/install/config` directory on the Control Workstation, and cover three typical SP environments:

1. Commercial (tuning.commercial file)
2. Interactive or development (tuning.development file)
3. Engineering or scientific (tuning.scientific file)

Copying one of these files to the `/tftpboot/tuning.cust` file on the Control Workstation or `boot/install` server before node installation or customizing will propagate the network settings out to the nodes.

These files contain settings for a node's tunable network options. Network options, viewed with the `no -a` and `netstat` commands, affect the operations of all networks on a node. If a node hosts different workloads generating different network traffic profiles, or has several types of network media (such as Token Ring and FDDI), you may be unable to optimize performance for all workloads and network media. The IBM-supplied configuration files provide only a better starting point than the default installation process.

5.6.3.2 Client Input Output/Sockets (CLIO/S)

SP systems often exchange large volumes of data with IBM mainframe systems, presenting a performance challenge to the network administrator. In a network of RS/6000 (including SP) and MVS/ESA systems, CLIO/S provides high-speed file and data transfer, and access to tape devices. CLIO/S is a data-transfer protocol consisting of a set of programs and application programming interfaces (API). CLIO/S runs on both MVS/ESA and AIX platforms and is bi-directional, making it easier to distribute work and data across a network.

CLIO/S is a separately ordered and priced IBM LPP.

5.6.3.3 Capacity Planning

There are no specific capacity planning tools available for the SP, and only a few in the UNIX marketplace. BEST/1 from BGS is a performance and capacity planning tool that uses queueing models to predict performance of a given configuration that processes a given workload. BEST/1 has not been

tested in an SP environment, although it could be used to analyze an individual SP node.

5.7 Accounting

An SP is likely to have different groups of users, and potentially different legal organizations, sharing diverse hardware resources. For example, in an outsourcing environment, a single SP could support independent workloads from separate companies running on a variety of node configurations. Groups of nodes and users may incur different usage fees, and you need to track resource usage for chargeback at the group level. You may charge a different usage fee for compute-intensive jobs that demand exclusive use of hardware resources. Also, you may wish to produce system-wide, consolidated accounting reports.

The SP builds upon the standard System V accounting facilities of AIX. AIX accounting is described in *IBM General Concepts and Procedures for RS/6000*, GC23-2202.

5.7.1 How Standard AIX and SP Accounting Differ

Enabling SP accounting support is optional. It is a separately-installed portion of the PSSP ssp.sysman fileset. SP accounting extends the function of base AIX accounting in three ways:

- **Accounting record consolidation.** You can set up accounting classes, which are groups of nodes for which accounting data is merged together to provide a single report. Accounting classes can be used to impose different charges for nodes of varying capabilities.

Standard AIX runs a script called runacct each night via cron . runacct produces daily usage and cumulative (over some fiscal period, like a month) reports. SP accounting splits the runacct function into two parts. First, the nrunacct script performs on each node the same function as runacct. Second, the crunacct script, executing on an accounting master node, gathers and consolidates the data from each node. The SP provides modified versions of AIX reporting routines to handle accounting classes and exclusive resource use.

- **Exclusive use of resources accounting.** Jobs that demand exclusive use of hardware resources can be handled separately for accounting purposes. Standard accounting bases charges on resources used (processor, disk, and so on). If a user is given exclusive use of a pool of nodes for a parallel job, the actual resource use measures may not be

appropriate for chargeback; instead, billing could be based on wall clock time use of the nodes as no other users can run jobs on those nodes during that time period.

When Resource Manager is asked to run an exclusive-use job, it executes a start job program under the real user ID of the job. When the job completes, Resource Manager executes the corresponding end job command. The start and end job commands bracket all the normal resource accounting records of the user's exclusive-use job. The reporting routines essentially discard the resource-based records between the start and end job commands, which are used to calculate elapsed time that the user had exclusive use of the nodes. Time becomes the resource, and a standard charge fee mechanism is used to bill the user ID of the job.

- **LoadLeveler parallel job accounting.** Independent of the AIX accounting base, LoadLeveler, which is bundled with the SP, provides job-based accounting data on all resources used by processes associated with a specific job.

5.7.2 Tools and Considerations

Like AIX accounting, the SP merges accounting data by user ID and login name. This forces all SP nodes for which accounting is enabled to have identical user ID and login names; in other words, belong to the same user name space. Name space can be managed by NIS, SP User Management, or other mechanisms.

AIX accounting requires the creation of directories, files, and crontab entries, as well as accounting configuration activity. This is appropriate for a single system, but a daunting task for an SP with hundreds of nodes. The SP simplifies this task greatly. After installation of the accounting portion of the SP, default values in the SDR configure accounting across the entire system. Accounting can be enabled or disabled globally by changing the site environment data. Node-by-node accounting activation, accounting classes creation, and exclusive-use support on a node are efficiently configured using SMIT (System Management Interface Tool) menus or SDR commands to update node attributes en masse. The accounting installation process creates all necessary directories and files on the nodes for which accounting is enabled.

See *IBM Parallel System Support Programs for AIX Administration Guide*, GC23-3897, for more information on SP accounting implementation.

5.8 Managing Print Service

The SP Print Management System is removed from PSSP 2.3 and onward. This means that the nodes running PSSP 2.3 can no longer configure Print Management System. We recommend the use of Printing Systems Manager (PSM) for AIX as a more general solution to managing printing on the SP system.

However, if you are running earlier versions of PSSP on some of your nodes, the SP Print Management System is still supported on those nodes. The `print_config` routine running on the Control Workstation will configure the SP Print Management System on nodes running versions of PSSP earlier than PSSP 2.3.

5.9 Managing Resource Workload

In Chapter 4.10, “Job Management” on page 202, LoadLeveler was presented. LoadLever is the human interface to the intricate task of deciding which job runs on which node to get the greatest amount of work out of the system per given length of time. The use of LoadLeveler is easy: there is a graphical user interface provided for this very purpose. Underneath LoadLeveler however is the Resource Manager subsystem without which parallel use of the various nodes would not be possible. It is the management of this Resource Manager which is discussed first.

5.9.1 Managing Resource Manager

Currently there are two main applications which use Resource Manager: Parallel Operating Environment, which uses Resource Manager for interactive parallel jobs; and LoadLeveler, which requests resources for batch parallel jobs. This is shown in Figure 122 on page 259.

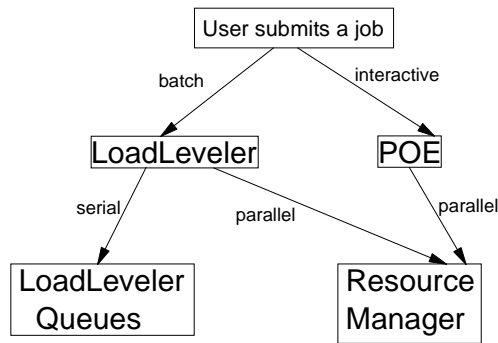


Figure 122. Delegation of Different Workload to Various Nodes

Resource Manager is a server that runs on one of the nodes. It prevents parallel jobs from interfering with each other and reports job related node information. Resource Manager operates per partition, so if there is more than one partition, there is more than one Resource Manager.

5.9.1.1 High Availability of Resource Manager

When the Resource Manager server is started, it automatically tries to start a backup Resource Manager server on another node. If the primary Resource Manager server fails, this backup becomes the primary server and starts a backup on another node. Jobs will continue to run unaffected through this takeover period.

If the primary and the backup Resource Manager servers die at the same time, there is no recovery of the Resource Manager, although the running jobs are not affected. To recover the Resource Manager after this failure, one must manually:

1. Wait for all running parallel jobs to terminate.
2. Reset the user node access if the access control is configured.
3. Restart the Resource Manager.

5.9.1.2 Installation of Resource Manager

Installation of Resource Manager is only required if you submit parallel jobs to LoadLeveler or if you use the POE for interactive parallel jobs.

To install Resource Manager, regular AIX installation tools are used to make sure that the ssp.jm fileset is installed. This fileset must be installed on all the nodes in the particular partition in which the Resource Manager is going to be

used. The original version of PSSP was SSP, and the *jm* extension stands for Job Management.

Resource manager uses TCP/IP to talk to the various nodes that will be managed, and so some TCP/IP files are updated during the installation process to include the Resource Manager details. They are as follows:

- /etc/services changed and original backed up in /etc/services.bak.jm
- /etc/inetd.conf changed and original backed up in /etc/inetd.conf.bak.jm

Resource Manager needs to be started and running each time the node is rebooted, so /etc/inittab is also updated during the installation process with the Resource Manager startup code. The original version of the file is saved in /etc/inittab.bak.jm.

5.9.1.3 Setting Up Resource Manager Server the First Time

Usually, the Resource Manager primary server runs on the Control Workstation, and the SMIT menus described below that do the Resource Manager management only work on the Control Workstation.

1. **Access Control.** Particular nodes can be dedicated to parallel use, and individual users can be prevented from using these nodes unless they access them through the Resource Manager. So this access control must be configured first.
2. **SDR Operation.** The Resource Manager needs to get system configuration information from the System Data Repository. The correct operation of the SDR must be verified. This can be done using SMIT or the regular command line interface.
3. **Configuration.** SMIT is used from the Control Workstation to define the Resource Manager configuration. The SMIT panels allow one to:
 - Change/Show the Resource Manager server list.
 - Change/Show Access Control.
 - List/Add/Delete/Change/Show a Pool. The Pools are lists of nodes within the partition managed by the Resource Manager that are either Batch Nodes, Interactive Nodes or General Nodes, which can handle either Batch jobs or Interactive jobs depending on the requirements.
 - Change/Show *En* Adapter. This is used to specify over which Ethernet adapter communications with/by/from the Resource Manager will happen. This is static data and cannot be changed without restarting the Resource Manager.

4. **Manual Startup.** The first time you use Resource Manager, start it up manually using the RM_options fastpath with SMIT.
5. **Verify Successful Startup.** One can verify that the Resource Manager Server is ready to accept requests from clients by using the SP_verify fastpath with SMIT.

This setup and configuration is shown in great detail in *Administration Guide for IBM Parallel System Support Programs for AIX*, GC23-3897.

5.9.1.4 Stopping Resource Manager

One can stop the Resource Manager subsystem using the RM_stop fastpath with SMIT. Here one is given the option to shut down the system gracefully or immediately. It is better to wait for existing clients to stop and then stop Resource Manager.

5.9.1.5 Reconfiguring Resource Manager

At some point, one may decide to redefine the pools of batch, interactive or general use nodes. Or one might need to redefine the access control. To do these, one need merely to effect the changes using the configure_pool fastpath in SMIT, and then the RM_options fastpath in SMIT to reconfigure Resource Manager.

However, if you decide to modify the SDR attributes that affect node or adapter objects, such as adding or deleting a frame when Resource Manager is already running, you must then stop Resource Manager and restart it in order to reread the SDR data.

5.9.1.6 Querying Resource Manager

There are various commands and options at the command line that enable any user on any node to query the status of the Control Workstation, particular pools, nodes and jobs.

Figure 123 on page 262 shows a graphical representation of what has to be running on the Control Workstation and the various nodes in order to have a usable Resource Manager in a partition.

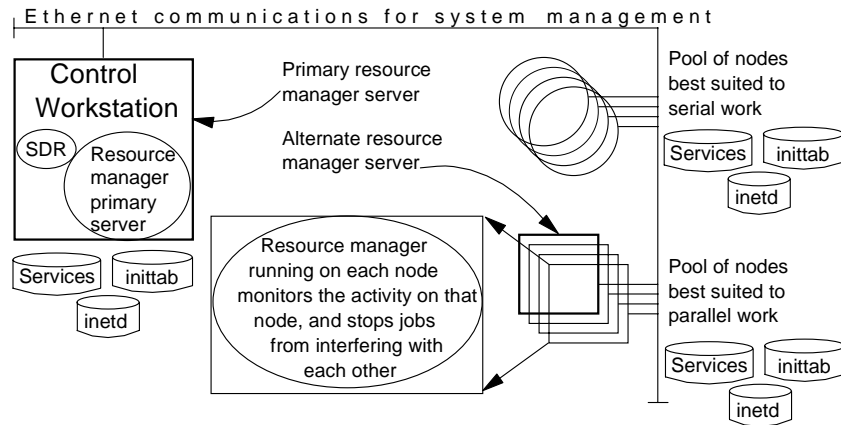


Figure 123. Resource Manager Running on the Nodes in a Partition

5.9.2 Managing LoadLeveler

LoadLeveler is the graphical user interface that the users use to submit their batch jobs to a collection of nodes. LoadLeveler was not designed specifically with the RS/6000 SP in mind. It is able to work on heterogeneous UNIX systems, but only the RS/6000 SP benefits and advantages are presented here.

For more detailed information about how to use and performance optimize LoadLeveler, refer to *IBM LoadLeveler Technical Presentation Support*, ZZ81-0348-00. Although this book was published in 1994, the content and principles are still valid.

LoadLeveler was originally designed for use with collections of serial machines, but its modular design made it easy to change to accommodate parallelism. The parallel aspects are not for heterogeneous UNIX systems yet - this extra feature is confined to the RS/6000 SP.

5.9.2.1 Installation of LoadLeveler

Installation of LoadLeveler is effected through the use of SMIT with the install fast path. The LoadLeveler fileset needs to be installed on each node that is in the partition and is going to participate in the pool.

5.9.2.2 Setting up LoadLeveler the First Time

After installing LoadLeveler, you need to customize it by modifying both the administration file and the configuration file.

The administration file

The administration file optionally lists and defines the machines in the LoadLeveler cluster and the characteristics of classes, users, and groups. Some nodes are *submit-only* nodes, and these cannot run jobs. They are defined here.

The configuration file

The configuration file contains many parameters that you can set or modify to control how LoadLeveler will operate. The LoadLeveler Central Manager node is defined here. This file also contains values which define the list of users that are allowed to administer LoadLeveler, the polling frequency, the number of polls per update, the interval between the polls, the negotiator interval and the machine update interval. The time values in this file are specified in seconds.

LoadLeveler must be started up on all the nodes that are to be in the group. The start up can be either a global startup, or granular. If it is granular, the Central Manager must be started first.

5.9.2.3 The LoadLeveler Daemons

LoadLeveler is an intricate arrangement of daemons which run in harmony on the nodes to evenly distribute the work. Six daemons are started on the nodes. Which daemons are started on which nodes is defined in the administration and configuration file on the Central Manager. The specific organization of the daemons on the different types of nodes is shown in Table 11 on page 263, and the diagrammatic interaction of these daemons is shown in Figure 124 on page 264.

Table 11. Different LoadLeveler Daemons Run on Different Nodes

On every node	master daemon
On central manager	collector daemon negotiator daemon
On schedule nodes	schedd daemon
On execute nodes	startd daemon
On user nodes	kbdd

The master daemon

The master daemon runs on all nodes and is responsible for the availability of all the other daemons on its particular node. If it detects failure of any of the other daemons on its node, it will attempt a specified number of restarts per

unit of time. If the number of restarts per hour is exceeded, then the master daemon notifies the operator and aborts and all daemons on that node are killed. All activity is recorded in mail messages that are sent to any specified destination.

On the Central Manager, the master daemon is responsible for:

- Nomination of the alternate Central Manager node which will become the primary Central Manager in event of failure.
- Startup of the negotiator daemon and the collector daemon.

The master daemon is initialized by a set of parameters which are found in the config and admin files. The config files are either local per node, or global per partition.

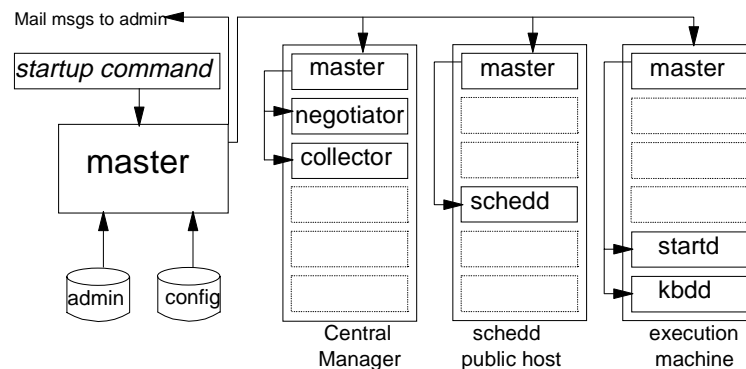


Figure 124. Interactions of LoadLeveler Daemons for Serial Batch Jobs

The collector daemon

The collector daemon only runs on the Central Master node. The collector daemon periodically receives information from schedd and startd daemons running on other LoadLeveler nodes. The LoadLeveler in-memory job tables are maintained by the collector daemon.

The negotiator daemon

The negotiator daemon maintains the status of each job and node in the cluster and responds to status queries from users and the administrator. The negotiator daemon runs on the Central Manager node. This daemon is started, restarted, signalled, and stopped by the master daemon. The negotiator:

- Receives job and user information from the collector daemon.
- Receives and records job status changes from the schedd daemon.

The schedd daemon receives jobs sent by users submitting jobs and schedules those jobs to target nodes. The schedd daemon:

- Is started, restarted, signalled, and stopped by the master daemon.
- Responds to job requests from the negotiator daemon. These requests are sent by an administrator and are passed on by the negotiator daemon. The actions include: run a job, change the priority of a job, remove a job, hold or release a job, send information about all jobs.
- Contacts the startd daemon on the target node when requested to start a job on a node, then the startd on the target node starts and controls the job execution.
- Maintains a log of events that it has processed and is processing so that the Central Manager can reconstruct the job information in event of a failure.
- Maintains a log of accounting information. Some of this information is furnished by startd.
- Sends job events back to the negotiator daemon when it (schedd) is restarting, or a new series of procedures is arriving, a job is started, a job was rejected, completed, removed, or vacated. Determines the status by examining the exit status returned by the startd.
- Requests that a remote startd daemon kill a job.

The startd daemon

When the Central Manager negotiator dispatches a job, it sends the target node name to the schedd daemon. The schedd daemon then contacts the startd daemon on the target machine, which then forks a starter process which runs the startd daemon.

The startd daemon periodically monitors jobs and machine resources on the local node and forwards a snapshot of this information to the collector daemon in the Central Manager.

The administrator can specify that a certain node is only to be used for batch jobs if it is not being interactively used. The startd receives notification of keyboard and mouse activity from the kbdd daemon to determine whether a particular node is available for batch use by LoadLeveler.

The startd daemon periodically examines the process table for LoadLeveler jobs and accumulates resources consumed by those jobs. This resource data is used to determine if a job has exceeded its job limit and for recording in a history file. The startd daemon also sends accounting information to schedd.

The starter process

The startd daemon spawns a starter process after the schedd daemon tells startd to start a job. The starter process is responsible for running the job and reporting status back to startd. The starter process:

- Executes any pre- or post-job programs as defined in the configuration file in LoadLeveler's home directory. If a pre-job program fails, the job itself will not start.
- Handles the authentication to make sure that valid users are submitting valid requests to valid files that they have access to.
- Runs the job by forking a child process that runs with the user ID and all groups of the submitting user. The starter child creates a new process group of which it is the process group leader, and executes the user's program or a shell. The starter parent is responsible for detecting the termination of the starter child. LoadLeveler does not monitor the children of the parent.

The kbdd daemon

In the non RS/6000 SP environment, machines could have been set up to be execution machines but only when that particular machine was not being used for any interactive jobs. The kbdd daemon would monitor keyboard and mouse activity. It would be spawned by the master daemon if the X_RUNS_HERE keyword in the configuration file was set to true. The kbdd daemon would notify the startd daemon when it detected keyboard or mouse activity, and then the startd daemon would suspend LoadLeveler initiated jobs on that node for the duration of the interactive activity.

The use of the keybdd daemon is not necessary on the nodes of an RS/6000 SP as most nodes are dedicated to batch operation, and so the keybdd daemon will not run.

5.9.2.4 High Availability of LoadLeveler

If the node that was nominated as the Central Manager fails, then the negotiator daemon running on the alternate Central Manager takes over control and notifies the master daemon on the alternate Central Manager that it is now the primary Central Manager. This does not affect the status of currently running jobs.

5.9.2.5 LoadLeveler Accounting

The account information is periodically stored on disk on each LoadLeveler node. The administrator can merge all the accounting files on the Central Manager node.

5.9.2.6 Checkpoint/Restart

Loadleveler supports basic checkpoint/restart capability for C and FORTRAN jobs.

Checkpointing is a method of periodically saving the state of a job such that, if for some reason (such as node or network failure) the job does not complete, it can be restarted from the saved state. Checkpointing with LoadLeveler also allows the migration of a long running job to another node when the first node is required for other work. Migration can be initiated automatically (by the LoadLeveler scheduler) or upon command from the system administrator. Migration might be initiated automatically because an inactive workstation becomes active again. The system administrator might migrate a job because he wants to perform maintenance on a node that is running a long job.

5.9.2.7 Diagnostic Tools

LoadLeveler provides a log function that can be used to keep information about LoadLeveler daemons. It is a test and debug tool reserved for system programmers and administrators; to initialize the log function, administrators have to modify daemon parameters in the configuration files.

The log function will return any piece of information about the way LoadLeveler daemons are running and should be used for debug purposes only. Normally the log function is disabled and the administrator or end users find the information they need through the GUI interface.

5.10 SP Backup and Recovery

With regard to backup and recovery, there are two elements that need to be addressed:

- Backup of data
- Backup of the system

Backup of data is often the easier element because you deal with a known set of files that can be backed up with traditional UNIX commands such as `tar`, `cpio`, `dd`; or traditional AIX commands such as `backup`. There are commercially available applications which can handle cross platform backup: ADSM and Sysback. ADSM is discussed in 6.1, “ADSTAR Distributed Storage Manager” on page 280. Once ADSM is correctly set up and configured, it will take care of backing up your specific files onto tape, and even optical disk if required.

Backing up of the system is more involved because there are:

- Boot devices that need to be backed up in such a way that a node or a Control Workstation can be replaced and still be able to boot from the media upon which the backup was originally made.
- System Data Repositories which keep a record of all the other nodes and their Ethernet addresses.
- Control Workstations, which are the key to the management of the entire system.
- Nodes, of which there could be hundreds. Traditional backup and restore techniques are easily applicable to a three-node system, but when there are three hundred nodes, the system administrator is going to be exhausted before he has individually restored each node. There has to be a better way to back up the system.

In addition to hardware failure (which is actually rare), another reason to have a backup policy is because of upgrades and PTFs. Before you upgrade your system, you should check the upgrade on a node, and before you upgrade a node, you should back it up. There is a book about this with interesting and detailed topics such as:

- mksysb install of nodes
 - Enter node configuration data
 - Verify installation settings
 - Run setup_server to configure the changes
 - Refresh system partition-sensitive subsystems
 - Disable node from the switch
 - Shut down the node
 - Network boot the node
 - Run verification tests

This book is *IBM Parallel System Support Programs for AIX Installation and Migration Guide*, GC23-3898.

5.10.1 mksysb and savevg

The traditional AIX backup command, `mksysb` is an integral part of the backup strategy for an RS/6000 SP. This command is an AIX-specific backup command that makes a bootable image of the mounted file systems on the root volume group of a machine. People who grew up on AIX take this for granted, but it an amazing concept worth exploring.

mksysb makes a bootable image on a boot device. The boot device is usually a tape, although it can be a file. If the mksysb image were created in a file, then the file would not contain the boot image and would not be bootable. This tape bootable image contains four parts, as shown in Figure 126 on page 270.

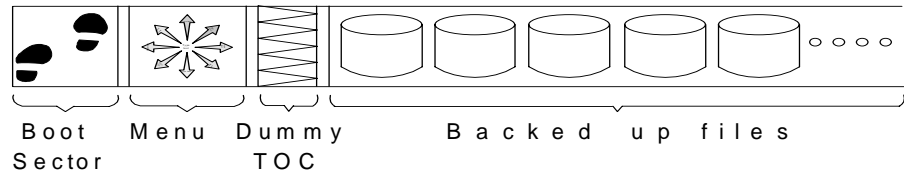


Figure 126. Contents of an mksysb Backup Image

The first part is the boot sector from which the node does a bootstrap boot, building the basic kernel in memory.

The next part is a menu that is displayed on the console of the node that is booting. This menu can be overridden with commands at the time of creation of the bootable image, or at the time of the restoration by the presence of a certain file with a specific internal format.

The third part is a dummy table of contents, which is not used.

The last part is the data and the files. This is the largest part. One of the first files that is in this data section is a file that contains the size, location and details about the various logical volumes and file systems that were mounted on root volume group at the time of the mksysb. This file is looked at by the booting process to reconstruct the logical volume manager part of the system as it was before so that the rest of the files on the tape can be restored into the same file systems that they were in before.

The `mksysb` command only backs up mounted file systems in the root volume group. Therefore:

- If there was a database in a raw logical volume on the root volume group, it would not get backed up by the `mksysb` command.
- If a particular root volume group file system was not mounted at the time of the backup, it would not be backed up.
- If there were other volume groups on a node, `mksysb` would not pick them up, and would not back them up, but there is a `savevg` command designed specifically for non-root volume groups.

On a traditional RS/6000, the `mksysb` image is used for backup of systems, not cloning of systems - the `mksysb` image created on one type of machine will generally not be restorable on another type of machine. This is not true in the RS/6000 SP system. An `mksysb` image made on one node will restore on any other node.

`savevg` is like `mksysb` in that it makes an image of the volume group so that at restoration time, the original logical volume policies are restored, except that the `savevg` image is not bootable.

To restore a node where there are three volume groups - `rootvg`, `datavg` and `salesvg` - you must first boot off and restore the `rootvg` with the `mksysb` image created, and then you use the `restvg` command to restore the non-root volume group data that was originally backed up using the `savevg` command.

5.10.2 Using `mksysb` and `savevg`

With traditional AIX, one is encouraged to keep one's data and operating systems separate - only AIX on the root volume group and only the data on separate non-root volume groups. This way, the `mksysb` image only contains the operating system and the applications installed on the `rootvg` volume group. The RS/6000 SP system exploits this in the backup scenario.

During installation, `mksysb` images are created on the Control Workstation. These generic images cover all the possible types of nodes that one could attach to the system.

There are two kinds of systems when talking about backups:

- Those with a small number of different nodes. This shall be known as a simple system.
- Those with a large number of nodes collected together into like groups for easy backup manageability. This shall be known as a complex system.

5.10.2.1 Backing Up a Simple System

In this scenario, assume that there is the Control Workstation and three other nodes: a database node, and two different computation nodes.

When generating the tapes for a backup, the generic `mksysb` images of the nodes already exist on the Control Workstation. The following remains to be done:

- Create a specific `mksysb` image for each node on the Control Workstation. It would be convenient to put these `mksysb` images into the root volume group of the Control Workstation, but as the system grows, the root

volume group can become unwieldy, so it is better to put the specific mksysb images into a separate non-root volume group on the Control Workstation.

- Back up the data of the various nodes. This can be done using `savevg` to back up each non-root volume group on each node into a file onto the Control Workstation, or commercial backup applications like ADSM could be used.
- Back up the Control Workstation using `mksysb` for the root volume group and either `savevg` or some other application to back up the non-root volume groups.

This simple backup strategy is shown in Figure 127 on page 272.

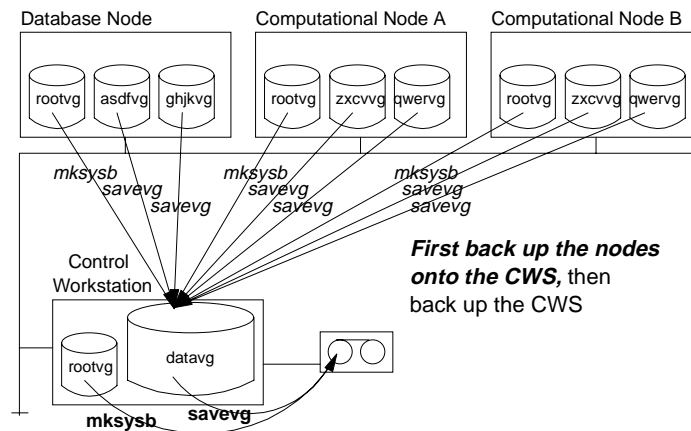


Figure 127. Backing Up a Simple System Using `mksysb` and `savevg`

5.10.2.2 Restoring a Simple System

To restore a simple system, you first have to restore the Control Workstation using the `mksysb` image and either the `restvg` command or an application like ADSM to restore the non-root volume group data.

Then it is a relatively simple procedure to restore each node's root volume group, and then the data specific to each node.

Once you have the Control Workstation up in a working state, there is a SMIT panel, or a command line interface, which allows the administrator at the Control Workstation to tell each node where to boot from. Given that the `mksysb` images for the nodes are on the Control Workstation, each node boots from the Control Workstation and restores its `mksysb` image.

Once each node has its operating system, the node's data can then be restored using the `restvg` command or ADSM.

This simple restore strategy is shown in Figure 128 on page 273.

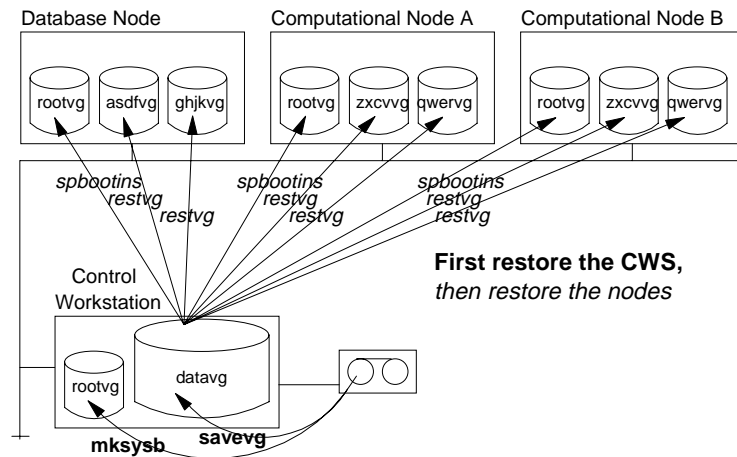


Figure 128. Restoring a Simple System Using `mksysb` and `restvg`

5.10.2.3 Backing Up a Complex System

Looking at the picture of the backup and restore of a simple system, it is easy to see that this procedure is definitely not scalable. It is simple to restore two or three nodes in this manner, but to restore a large system requires another mechanism.

When the system has a large number of nodes, they should be grouped together into like groups of that have the same root volume group. Each group of nodes will then have one `mksysb` image stored on the Control Workstation. The data for each of the nodes will be backed up separately using an application like ADSM.

This is shown in Figure 129 on page 274.

5.10.2.4 Restoring a Complex System

To restore a complex system, first the restoration of the Control Workstation is addressed, then each node has its generic `mksysb` image restored off the file on the Control Workstation, and then node-specific data restoration scripts can be executed.

5.10.3 SP-Specific Data That Needs to Be Backed Up Frequently

In the ideal world, one creates a regular mksysb image of the Control Workstation, and a regular mksysb image of each different node, and one also has some way of regularly backing up the data on the nodes. Unfortunately, mksysb takes time, so it is not practical to make a mksysb images often. Therefore, the next best scenario is to create an mksysb image before any major system configuration changes and then do incremental backups of regularly changing system information daily.

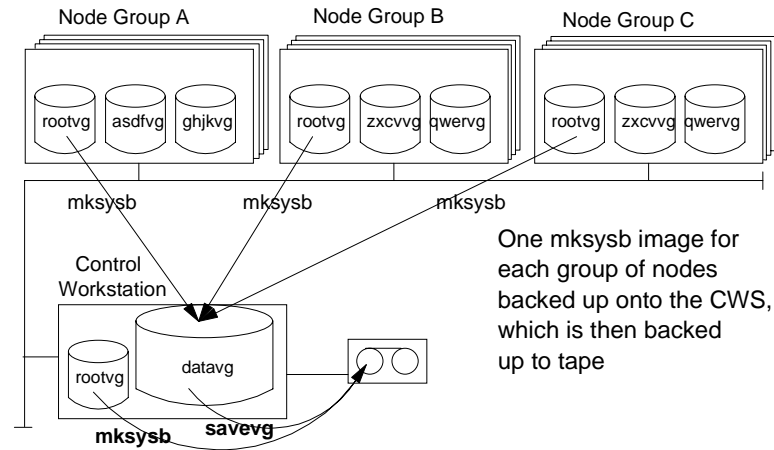


Figure 129. Scalable Backup Policy

In addition to following the regular AIX backup policies for the Control Workstation, one should back up the Kerberos database and the SDR database before and after any configuration changes.

5.10.3.1 SDR Backup and Recovery Strategies

To back up the SDR database, you should use the `SDRarchive` command which archives all the files that comprise the SDR database into the tar file located in the `/spdata/sys1/sdr/archive` directory on the Control Workstation. This file can then be saved onto a tape or be part of the daily backup procedure. You can create a cron job to run the `SDRarchive` command just before the daily backup procedure runs.

There is also an `SDRrestore` command which will restore the SDR to the state that it was in when the last SDR archive command was run. After one has restored the SDR, the SDR daemon must be refreshed to allow it to recognize the new files.

It is technically possible to do a partial restore of specific classes of SDR data. This is not recommended at all. There are dependencies between the classes in the SDR. These classes must be archived and restored together. Partially restoring the SDR can produce unpredictable results.

The system administrator should back up the SDR before doing any tasks that reconfigure frames, nodes, switches or VSDs. In addition, the SDR should be backed up periodically, such as once a week. The SDR should only be restored when it is known to be corrupted. If a reconfiguration command does not work, the SDR is not necessarily corrupted; it may be sufficient to rerun the command to have it succeed.

5.10.3.2 Kerberos Backup and Recovery Strategies

The Kerberos database on the Control Workstation is stored in the `/var/kerberos/database` directory. This directory needs to be backed up before and after any Kerberos changes.

The user-specific Kerberos information is stored in `~/.klogin` for each user that uses kerberos, so all these files in the users' home directories need to be backed up daily, too.

To restore the Kerberos database, the `/var/kerberos/database` directory is restored, along with the `~/.klogin` files that were backed up at the same time as the `/var/kerberos/database` directory.

Part 3. Solutions

Part 3 discusses the commercial and technical solutions that are available on the IBM RS/6000 SP. It also provides information regarding the strengths of the SP in different environments and describes solutions that exploit the parallel architecture.

Chapter 6. Enterprise System Management

AIX and PSSP form the foundation of SP-specific system management, but how can the SP be managed within the enterprise? This chapter briefly presents popular IBM enterprise systems management applications that can encompass the SP.

Many people around the world only know the SP as "Deep Blue" computer associated with the world of chess. Figure 130 illustrates this marketing picture.

May 3-10, 1997

Kasparov		Deep Blue	
5' 10"	height	6' 5"	
176 Pounds	weight	1.4 Tons	
32 Years	age	4 Years	
Azerbaijan	birthplace	USA	
50 B Neurons	# of processors	32 RISC Chips	
3-5	processing capacity (board positions/sec)	200,000,000	
Electrical/Chemical	power supply	Electrical/AC	
World Chess Champion	future duties	Business & Technical	




Figure 130. Is SP a solution?

However, for more practical applications, Figure 131 on page 280 shows the solution domains for the SP.

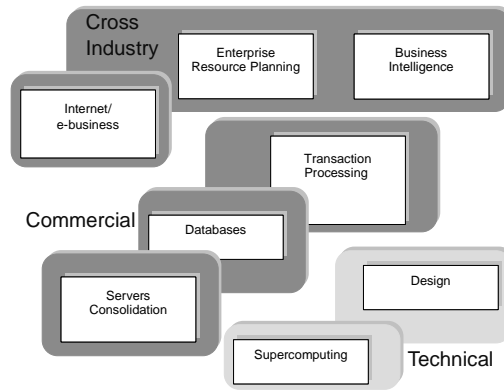


Figure 131. Solution Domains

The next three chapters discuss each one of these domains in detail.

6.1 ADSTAR Distributed Storage Manager

The ADSTAR Distributed Storage Manager (ADSM) family of software products is an enterprise-wide solution integrating unattended network backup and archive with storage management and powerful disaster recovery planning functions. ADSM's structure is presented in Figure 132 on page 280.

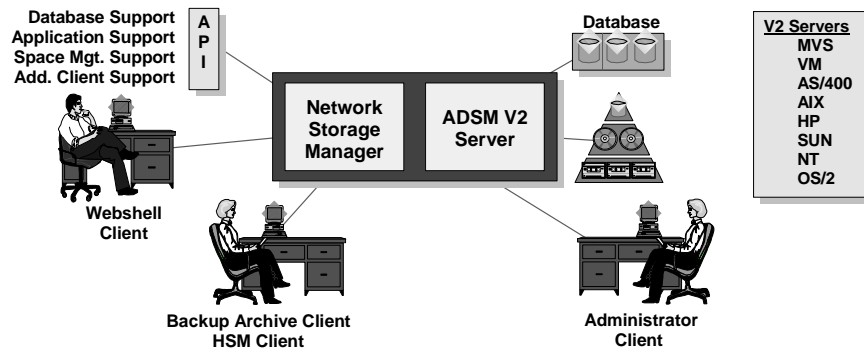


Figure 132. ADSM Structure

ADSM provides backup and archive services to more than 25 multi-vendor client platforms, including industry-leading databases and applications such as SAP/R3. Data is stored on an ADSM storage management server running on a variety of IBM and non-IBM desktop, midrange, UNIX, and large systems

platforms. In addition to multiple client and server platforms, ADSM supports a variety of storage devices and communication protocols. This broad range of support makes ADSM a comprehensive storage management solution for heterogeneous networks.

6.1.0.1 SP Implementation

You can implement ADSM in three ways on the SP:

- **Server only.** One or more nodes on the SP act as ADSM servers. They directly attach the storage devices (tape, optical, disk, libraries). The nodes connect to the clients in the enterprise either directly via network adapters in the nodes, or through a gateway node (standard node or GRF). This configuration is not common; customers placing the ADSM server on the SP typically use it to back up the SP as well.
- **Clients only.** Any or all SP nodes, including the CWS, could have the ADSM client software installed. The ADSM server is external to the SP and could be an IBM host system (MVS, VM, or VSE), an RS/6000 server, or another manufacturer's server (such as HP or Sun). The SP clients connect to the server either directly via network adapters in the nodes, or through a gateway node (standard node or GRF). This configuration is common where customers have large investments in ADSM servers and storage regimes before they buy an SP. A popular way to connect the SP clients to a host-based ADSM server is via a gateway node containing an Enterprise System Connection (ESCON) adapter.
- **Server and clients.** ADSM server(s) and clients reside in the SP. In this scenario, the tremendous bandwidth of the SP switch is used to full advantage. Networks are a common bottleneck in enterprise backup/archive processes. The SP switch removes this bottleneck and allows other key system resources (such as ADSM server node disks supporting buffer pools) to be driven at higher utilizations.

Node system backup files, created by the `mksysb` command, can be large and numerous, depending on the number of nodes in the SP with unique configuration and the configuration management policy of the organization. If a node needs to be reinstalled, its `mksysb` image must reside on the node's boot/install server. Managing disk space for the node images on the boot/install servers across the SP can be simplified by using ADSM.

ADSM allows files backed up on one node to be restored by another. Let ADSM store all the node images. If you need to recover a node, you will retrieve the `mksysb` image from ADSM to the boot/install server node for that node, and then reinstall the failed node from that `mksysb` image. If minimizing recovery time is critical, then the delay in retrieving the node image from the

ADSM server may not be tolerable; often, however, particularly with hardware failures, the image can be retrieved while the node is being serviced.

For detailed information on implementing ADSM on the SP, refer to *ADSM/6000 on SP2*, GG24-4499 and *Backup, Recovery and Availability on RS/6000 SP*, SG24-4695.

6.2 High Availability Cluster Multiprocessing (HACMP)

Clustering servers or nodes helps provide the performance, redundancy and fault resilience required for business-critical applications. HACMP is a highly-regarded and successful IBM clustering offering for the RS/6000 family. Over 10,000 licenses have been sold worldwide to date.

Strictly speaking, today's HACMP falls somewhat short of a true enterprise-class offering, as it works only on the RS/6000 product family; however, HACMP is evolving with IBM's broader clustering technology, which will encompass diverse IBM, and potentially other manufacturers', platforms.

6.2.1 HACMP for AIX

HACMP is a control application that can link up to eight RS/6000 servers or SP nodes into highly available clusters. HACMP is flexible in configuration and use. Uniprocessors, symmetric multiprocessors (SMPs), and SP nodes can all participate in highly available clusters. You can mix and match system sizes and performance levels as well as network adapters and disk subsystems to satisfy your application, network, and disk performance needs. A typical HACMP cluster is shown in Figure 133 on page 282.

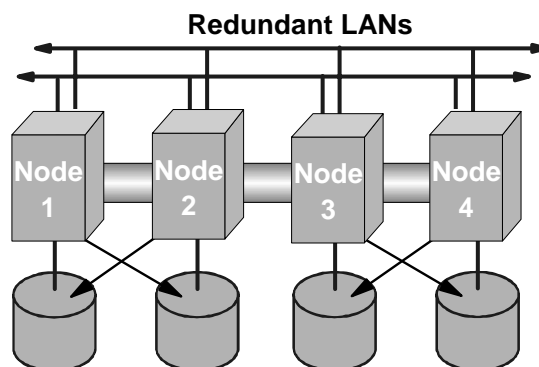


Figure 133. Generic HACMP Cluster

HACMP clusters can be configured in several modes for different types of processing requirements. Concurrent access mode suits environments where all of the processors must work on the same workload and share the same data at the same time. In mutual takeover mode, the processors share the workload and back each other up. Idle standby allows one node to back up any of the other nodes in the cluster.

For NFS environments, HACMP provides the HANFS for AIX feature, which defines two AIX systems as a single highly available NFS server, helping to eliminate single points of failure. This server can survive hardware and software outages, and many planned outages, that can make a single system NFS server unavailable.

With the HAView function, you can use Tivoli's TME 10 NetView for AIX graphical network management interface to monitor clusters and their components across the network from a single node.

HACMP is a separately priced and orderable LPP.

6.2.1.1 High Availability Geographic Cluster for AIX

Combined with HACMP for AIX, High Availability Geographic Cluster for AIX (HAGEO) extends HACMP's loosely-coupled cluster technology. HAGEO provides real-time data mirroring between HACMP clusters, providing disaster recovery capability to clusters placed in two widely separated geographic locations. HAGEO eliminates the site itself as a single point of failure. Data mirroring with HAGEO can be synchronous, ensuring real-time data consistency, or asynchronous for maximum performance while maintaining sequential data consistency.

HAGEO is a separately priced and orderable LPP, and requires HACMP for AIX to be installed.

6.2.1.2 SP Implementation

In designing a cluster for availability, you consider all single points of failure in the system. The SP introduces additional single points of failure beyond those of a cluster of individual RS/6000 machines:

- **SP Switch**
Currently, there is only one switch fabric in an SP. Each node can only have one switch adapter. The switch is designed to be very robust and has redundant internal components, but should it fail, a fast network, such as FDDI or ATM, could act as an alternate.
- **SP Administrative Ethernet**
In most cases, this is not a critical production component, but applications

relying on VSD/RVSD/GPFS require it. There is no failover function for this network. As a standard thin Ethernet, a loose connector or terminator can bring down the whole network. This can be contained by using a routed Ethernet network, where node groups are on their own physical segment but in the same subnet.

- **Control Workstation**
The failure of the CWS does not necessarily affect the execution of applications on the nodes, even though the CWS is the single point of control for the SP. Still, control functions such as starting the switch are performed only at the CWS, and if you exploit Kerberos authentication for system management applications on the nodes and elected the default Kerberos implementation (CWS as primary authentication server, no secondary servers), then your authenticated applications will fail. The CWS can be protected with High Availability Control Workstation (HACWS), described later.
- **SP Frame**
An SP frame has redundant power supplies for its nodes, but also a number of other components (such as the RS232 link, frame supervisor card, and power cord) that can be single points of failure. Redundant frames, housing redundant nodes, offer the only way to protect against failures in these components.

For detailed information on implementing HACMP on the SP, refer to *High Availability on RS/6000 SP*, SG24-4742.

6.2.1.3 HACWS

HACWS is an optional collection of components that implement a backup CWS for an SP. The backup CWS takes over when the primary control workstation requires upgrade service or fails. The HACWS components are:

- A second RS/6000 machine supported for CWS use.
- The HACWS connectivity feature (#1245) ordered against each frame in the system. This furnishes a twin-tail for the RS-232 connection, so that both the primary and backup CWSs can be physically connected to the frames.
- HACMP for AIX installed on each CWS. HACWS is configured as a two-node rotating HACMP cluster.
- The HACWS feature of PSSP. This software provides SP-specific cluster definitions and recovery scripts for CWS failover. This feature is separately orderable and priced and does not come standard with PSSP.

- Twin-tailed external disk, physically attached to each CWS, to allow access to data in the /spdata file system.

An HACWS cluster is depicted in Figure 134 on page 285.

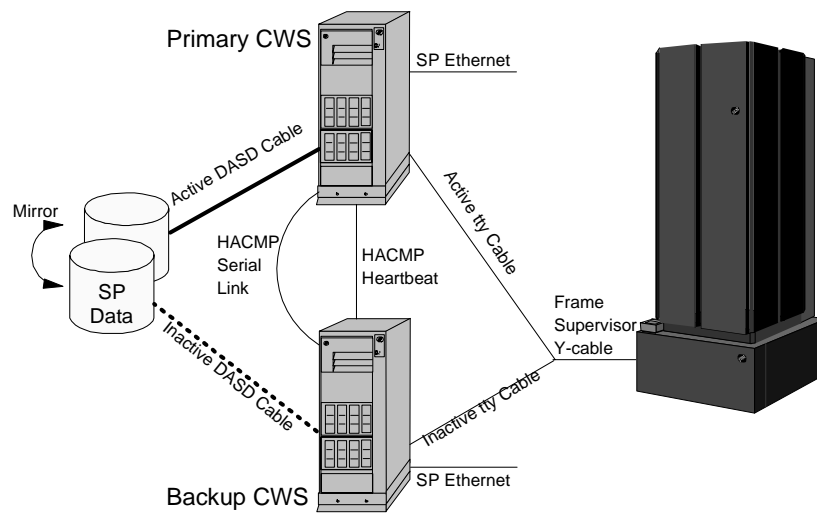


Figure 134. HACWS Cluster

If the primary CWS fails, the backup CWS can assume all CWS functions with the following exceptions:

- Updating passwords (if SP User Management is in use)
- Adding or changing SP users
- Changing Kerberos keys (the backup CWS is typically configured as a secondary authentication server)
- Adding nodes to the system
- Changing site environment information

HACWS implementation and system management considerations are fully described in *IBM Parallel System Support Programs for AIX Administration Guide*, GC23-3897.

6.2.2 HACMP Enhanced Scalability (HACMP ES)

HACMP ES provides the same functions of HACMP for AIX, except that it can link up to 32 SP nodes in a cluster (HACMP for AIX supports 8). HACMP ES is architected to support 128 nodes in future releases. A customer with

HACMP for AIX installed on the SP can reuse the hardware configuration, cluster definitions, event scripts, and skills when moving to HACMP ES.

HACMP ES is a separately priced and orderable feature of the HACMP for AIX LPP.

6.2.2.1 Relationship with HAI

HACMP ES differs from HACMP for AIX by exploiting HAI on the SP. As discussed in 4.5, “High Availability Infrastructure (HAI)” on page 112, HACMP for AIX has scalability issues that HAI technology was designed to solve. Refer to Figure 135 on page 286. It is important to understand that HACMP ES uses its *own* versions of PSSP’s Topology Services and Group Services, Topology Services/ES and Group Services/ES, respectively, which have been enhanced to provide support for classic HACMP functions such as IP address takeover and support for non-IP point-to-point networks (such as RS-232). The PSSP and HACMP ES instances of Topology Services and Group Services do not communicate. The Cluster Manager exploits PSSP’s Event Management subsystem for event notification.

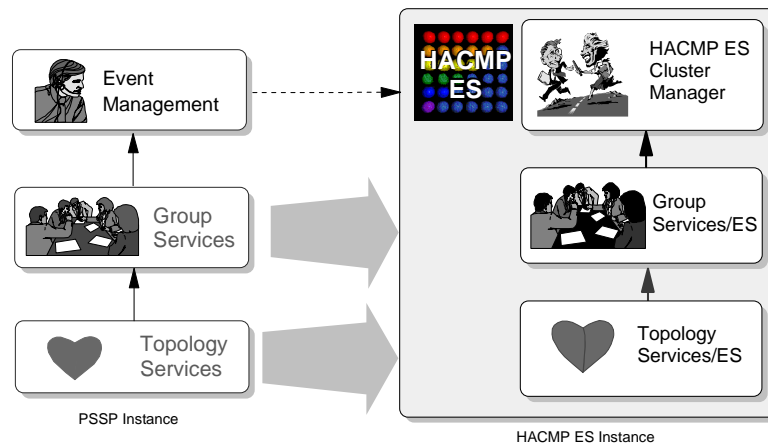


Figure 135. HACMP ES Relationship to HAI Components

HACMP ES also benefits from HAI technology by exploiting the synchronization and coordination facilities of Group Services/ES. You can add barrier commands to your event-handling scripts to ensure that all nodes in a cluster reach the same point together in recovery processing.

For detailed information on implementing HACMP ES, refer to *HACMP Enhanced Scalability*, SG24-2081.

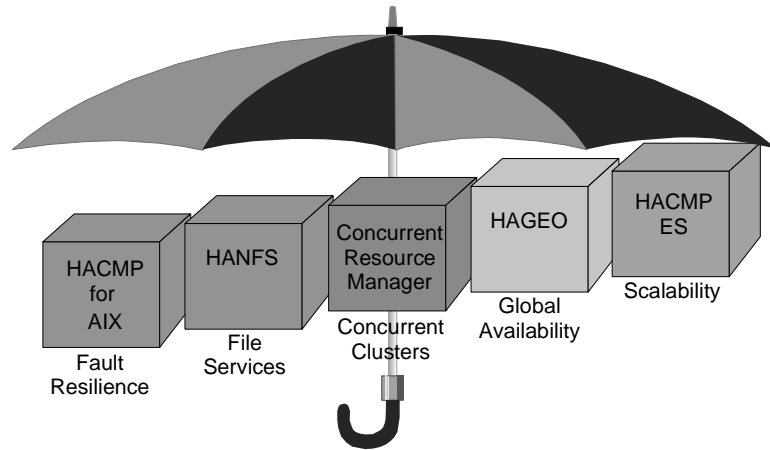
6.2.2.2 Positioning HACMP and HACMP ES for the SP

How do you choose between using HACMP for AIX and HACMP ES on an SP? Keeping in mind that the products offer virtually the same function, here are some points to consider:

Consideration	HACMP for AIX	HACMP ES	Comments
Number of nodes per cluster	8	16	HACMP ES will support more in future.
Node types in a cluster	SP and RS/6000 stand-alone	SP only	HACMP ES relies on HAI's Event Management, currently only on SP.
Partition sensitivity	None	Cluster cannot span partitions. Multiple clusters per partition allowed.	HACMP ES relies on HAI technology, the daemons of which are partition-sensitive.
Works with HAGEO?	Yes	No	
Network support limitations	No	ATM not supported yet	
Concurrent disk access configuration support	Yes	No	
Ready to exploit IBM's cluster technology?	No, not as currently architected	Yes	Important for cluster size scaling.
Price	Less expensive	More expensive	

6.2.3 Summary of HACMP Products

Figure 136 on page 288 summarizes the HACMP family of products, all of which can be implemented on the SP. HACMP for AIX tends to meet most customers' high availability requirements on the SP, but as cluster technology matures, HACMP ES will be a more attractive product.



This release of HACMP ES is recommended for new SP installations

Figure 136. High Availability Family

6.3 Tivoli

The Tivoli Management Environment (TME) provides a management platform and integrated applications to manage many facets of an enterprise computing environment. TME supports a broad range of heterogeneous platforms with a consistent interface to commonly used administrative functions. Areas of system management, with sample TME components:

- Deployment (TME 10 Software Distribution)
- Availability (TME 10 Enterprise Console and Netview)
- Security (TME 10 User Administration)
- Operations (TME 10 ADSM and Remote Control)

6.3.1 SP Implementation

One of the challenges of integrating the SP into a Tivoli environment is to decide what management tasks are best performed by the SP-specific function of PSSP, and which should be done using Tivoli. You must consider the frequency of the task, domain of administration, and skills of the personnel making these choices.

Tivoli applications can reside external to the SP, or be installed on SP nodes. Tivoli can treat nodes as separately managed entities, or group nodes under

common policy regions and profile managers. The SP can be viewed and managed with a single system image

Integration Points

Figure 137 on page 290 depicts three areas of SP and Tivoli integration.

- Tivoli Application Actions

At the lowest level of integration, TME could use its own resource monitors, via TME Distributed Monitoring, to forward events and alerts to the central managing TME application, such as TME 10 Enterprise Console (T/EC). This bypasses the HAI infrastructure. Monitors would need to be configured node by node.

- Event Notification and Monitoring

A T/EC module has been developed to interface to Event Manager on the SP. Any Event Manager event can be forwarded to T/EC, allowing you to efficiently monitor many SP-unique and AIX resources. You can use T/EC as a centralized point for event correlation, notification, and automated reaction.

Via PMAN, the SP allows any SNMP-based network manager (such as TME 10 Netview) to access the SP-specific Management Information Base (MIB), SNMP traps, SP configuration information, the last trap in any node's AIX error log, the last subscribed event generated in EM for any node, and current values of EM resource variables, as well as the list of defined resource variables.

- SP-specific Action

Many SP-specific functions, such as switch management, node power control, and parallel commands, are foreign to Tivoli. Instead of using SMIT or Perspectives, you could define a task library in Tivoli consisting of SP commands and scripts. An administrator could execute these tasks against individual nodes and the CWS.

The highest degree of integration is achieved by configuring the TME 10 Desktop so that the managed node objects correspond to SP nodes and can directly access the additional SP-specific functions defined in the task library.

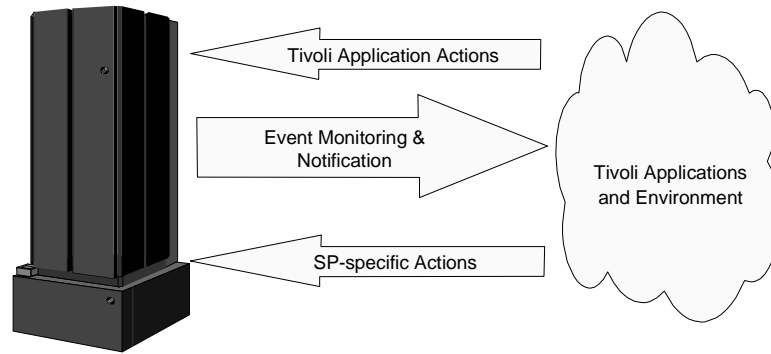


Figure 137. Tivoli and SP Integration

For detailed information on implementing and integrating Tivoli with the SP, refer to *Integrating TME 10 on the RS/6000 SP*, SG24-2071. This reference includes a listing of all SP MIB objects. Further information on the SP MIB and configuring the SP for monitoring by an enterprise network manager is found in *IBM Parallel System Support Programs for AIX Administration Guide*, GC23-3897.

Chapter 7. Cross-Industry Applications

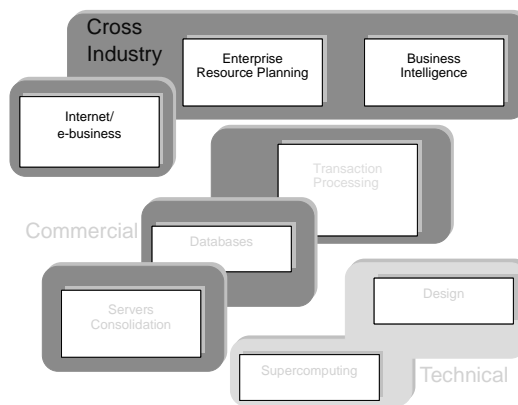


Figure 138. Cross-Industry Applications

This chapter presents the following topics:

Enterprise Resource Planning (ERP) Solutions and Lotus Notes:

We describe the concepts of ERP solutions based on SP, which permit people to work together more efficiently.

Business Intelligence Solutions:

We briefly describe what business intelligence and Decision Support Systems (DSS) mean and why the SP is a key component in this approach.

Internet Solutions:

The Internet has become a key domain, and the need for Web servers and applications is growing fast. We describe why the SP fits so well for:

- Internet Service Providers
- Content Hosting
- Mega Web-site management

7.1 Enterprise Resource Planning (ERP)

Organizations of all sizes are investing millions of dollars and thousands of hours implementing ERP applications. This is because ERP systems help create a seamless system that permits employees to work together more efficiently. ERP solutions offer benefits in such areas as:

Business Reengineering

ERP solutions provide uniform information management to help support process management throughout the organization.

Year 2000 Time Constraints

It is often more efficient to implement a new “best-of-breed” ERP solution than to rewrite existing code.

European Monetary Union

Organizations worldwide must adjust to the introduction of the Euro monetary unit, including the need to maintain parallel accounts, on a pre-specified timetable.

There are several ERP products available for the SP. The ones that have the lion’s share of the international ERP marketplace are:

1. SAP R/3
2. Oracle Applications
3. Peoplesoft Applications
4. JD Edwards’ OneWorld

7.1.1 SAP R/3

SAP is the largest standard application software company in the world, serving more than 4,000 customers in 41 different countries. Its R/3 system, a client/server set of applications for distributed enterprises based on a flexible, tiered architecture, is designed to integrate and consolidate the financial, sales, distribution, production and other functions within a company.

In the class of ERP applications, SAP R/3 exemplifies all the aspects of this type of client/server products. For this reason, the remainder of this section focusses on SAP R/3, with the understanding that similar comments can be made for all products that fall into this category.

SAP R/3’s suite of client/server data processing products relies on combining all the business activities and technical processes of a company into a single integrated software solution. Users do not have to access different databases

for data created from different divisions of a company. Instead, data from a wide variety of sources, such as financial, asset management, controlling, production planning, project system, quality assurance, and human resources, can coexist in the same database, and be accessed in real-time from a variety of applications.

7.1.2 SAP R/3 Design

The design of SAP R/3 is based on three categories of platform-independent software services (see Figure 139 on page 293):

Presentation services - These consist of graphical user interfaces on PC, X-terminal, or workstation machines.

Application services - These services provide the application logic and they may run on one or more UNIX or Windows NT-based systems. Application services may run in batch or interactive mode and are responsible for gathering monitoring data, to be dispersed to the presentation services.

Database services - These are the services that a typical database engine would provide. SAP R/3 can use several database engines, depending on the particular needs of the client. These services would typically run on a mainframe, or on a cluster of UNIX or Windows NT machines.

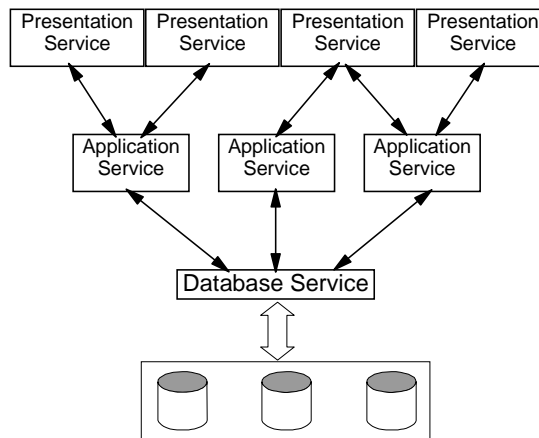


Figure 139. Conceptual Model of SAP R/3 Structure

An application service is designed and implemented in a layered way, isolating the SAP R/3 application logic from the operating system-dependent services. A middleware layer, called the “basis layer”, communicates with the

operating system and the network. Figure 139 on page 293 shows the layering of an application service.

Services are provided by administrative units called *instances* that group together components of SAP R/3. When first installed, the SAP R/3 system has a central instance, which has services such as dialog, update, enqueue, batch, message, gateway, and spool. After installation, these services can be moved to other application servers in order to balance workloads.

7.1.3 RS/6000 SP Implementation

Although an SAP R/3 solution is inherently hardware-independent, its power and versatility is complemented very well on an RS/6000 SP implementation. The fundamental design principles that make the SP so successful in other areas also create almost countless possibilities for SAP implementations.

Conceivably, one can host a complete SAP solution in a single SP system. Certain nodes can act as the database services, others as the application level services, and yet others as the presentation services. The flexible and dynamic allocation and clustering of nodes in the SP let system administrators create configurations that adapt to the current needs, and expand equally easily. SAP R/3 customers can use its utilities to add more machines for database, application or presentation services to the existing SAP R/3 system.

The SAP R/3 implementation on the SP follows a three-tier model. Contrary to most multi-tier configurations that use TCP/IP for network communication, the SP can take advantage of its internal fast switch to expedite network traffic, and do so in a highly available manner. Because of its HACMP/ES capabilities, a cluster of nodes that acts as a database service would sense a node that fails, and take appropriate steps to compensate for the loss.

7.1.4 SAP R/3 Examples on SP

There are several examples of companies who have transformed their business by using SAP R/3 on an SP. Here we will mention one from the industry and one from higher education.

7.1.4.1 Business

Thyssen Informatik Co. Ltd (TIC) is one of 323 companies that make up the Thyssen conglomerate. Located in Krefeld, Germany, TIC provides a full range of data processing services, including complete, turnkey outsourcing projects. Its multi-platform central computing center supports 60 customers, some 25,000 end users, conducting millions of transactions a day.

To support its many SAP R/3 users, TIC has installed three RS/6000 SPs and Serial Storage Architecture (SSA) technology. A key characteristic of the SP is flexibility in adding small, low-cost increments as new applications are developed or data processing priorities shift. TIC's rapid expansion required that an SP system be installed during each of the company's first three years.

7.1.4.2 Higher Education

The University of Toronto is the largest university in Canada. It prides itself on being an international center of innovation, along with having a deep commitment to the arts and basic sciences.

To establish better control over running the business side of this huge enterprise, the university recently installed an RS/6000 SP running a number of SAP R/3 client/server software applications. The major reason why IBM won the hardware contract was (in the words of Eugene Siciunas, the University Director of Computing and Networking Services) "... in part because they came up with a more competitive, very creative growth path for our future".

The initial SP configuration was for five nodes, expected to grow to thirteen in the course of a year. Among the advantages of running SAP applications on the SP are flexibility and scalability, according to Siciunas. "We like the fact that, as we grow, we can mix and match--by adding more CPUs to the SP or replacing them with more powerful ones," he says. "And the SP's single point of control is certainly important."

7.2 Lotus Notes

Lotus Notes is a software application that promotes an effective workgroup computing environment. Notes provides the technology to allow people to collaborate regardless of software or hardware platform, technical, organizational or geographical boundaries. There is a lot of literature in the market for implementing, configuring and using Lotus Notes (refer to the Lotus Notes official web site at <http://www.lotus.com>). In this section we do not attempt to introduce or reproduce these works, rather we summarize the basic concepts and design principles that Notes is implementing.

To this end we need to explain the concept of groupware. Groupware is used as a concept to characterize software whose objective is to improve the sharing of information among members of a group or a team. Lotus Notes exemplifies the meaning of the term "groupware product", and it is central to IBM's intranet/internet strategy. In 1995 the company recognized the potential of Lotus Notes and proceeded to introduce it as the new platform for

collaborating over the network, sharing access to databases, and communicating with people outside IBM.

Lotus Notes is best understood through examples of it in action. Consider a team that writes a review of a new technology. Through Lotus they can share resources, literature, communicate with developers, and create a dynamic database that contains the sum of individuals' efforts to researching a subject. When compiling the information, and writing the documentation, team members can draw resources from the common database, update the sections they are working on, for everyone else to see, and update the to-do list for management to see.

One of the main advantages of Notes is its ability to embed rich text documents. *Rich text* describes a document that can contain text, graphics, scanned images, audio, and full-motion video data. This feature alone can improve the communication quality of shared information dramatically. Notes incorporates a shared, distributed document database which resides on a server, an easy-to-use graphical user interface, a built-in e-mail function, and built-in security to allow access control for critical information. Furthermore, Notes provides a rapid application development environment so that users can create custom applications that reside on top of Notes, enabling Notes to conform to particular business needs. The improved coordination that Notes offers provides a competitive advantage for organizations that choose to utilize the groupware capabilities of the program. IBM is but one in a list of companies that saw spectacular results from the use of Lotus Notes.

7.2.1 Lotus Domino

Domino was the code name of a Lotus component program that was designed to allow standard Web browsers to connect to a Lotus server. This new functionality allowed Notes servers to do much more than provide groupware and electronic mail services. Lotus then decided to "rebrand" the Notes server and the new Domino program as one complete package called Lotus Domino Server 4.5, Powered by Notes, or simply Domino. Clients that connect to this Domino server are Notes clients, Web browsers, and alternate mail clients. See *Domino Integration Guide for IBM Netfinity and PC Servers*, SG24-2102, for further details on Lotus Domino servers.

Lotus Domino 4.6 is an application and messaging server that enables a broad range of secure, interactive business solutions for the Internet and intranet. With Domino, developers can rapidly build, deploy and manage applications that engage co-workers, partners, and customers in on-line collaboration, cooperation, and coordination of critical business activities.

The Web server component of Domino comprises of two subcomponents. The first is a standard Web server. The second is an interface between the Notes subsystem and the Web server. This interface translates Notes elements such as documents, forms, views and action buttons into HTML format for a Web browser dynamically. This means that data in a Notes database is viewed, modified and created using a standard Web browser on-the-fly, and there is no need for any HTML programming. All of Notes strengths such as security, replication, full text searching, agents and application development now apply to the Web site.

7.2.2 RS/6000 SP Implementation

With the introduction of Lotus Domino Release 4, customers can now realistically configure systems (or networks of systems) to support thousands of users from a single location. As the capacity of your Domino servers increase, the degree to which your business depends on the availability of the Domino server complex will also increase dramatically, bringing with it additional requirements and expectations. The RS/6000 SP system provides a scalable, complete solution for the Enterprise Domino Server.

A typical Domino configuration consists of mail, database, and replication servers. These servers are grouped into centrally administered domains which make up an enterprise-wide Domino environment.

Figure 140 on page 297 illustrates a hub-and-spoke topology, which is common in large sites and is the most adaptable for system growth and change.

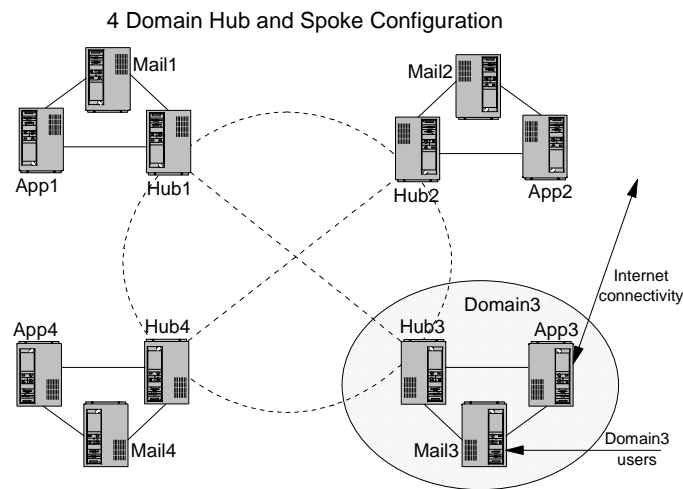


Figure 140. Enterprise-Wide Domino Configuration

Within each domain, a dedicated hub or multiple hub servers are used to replicate information and route mail across domains. Within each domain, there are typically three types of servers: one type dedicated to data replication and/or mail routing; another as user mail servers; and the third for serving data and applications.

Figure 141 on page 298 illustrates the SP Domino configuration. The Lotus/Domino model is very popular for groupware tools, but introduces a lot of traffic into the network. The clusters have to replicate information, as well as manage new data.

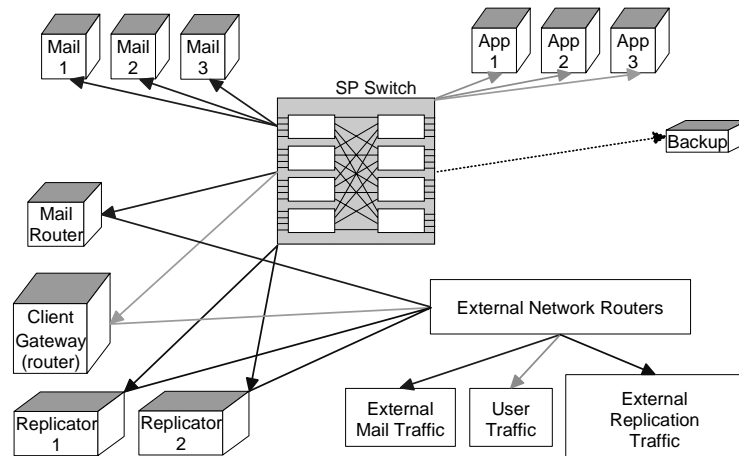


Figure 141. Logical model of Domino on the SP

The high speed of the SP switch alleviates bottleneck problems and offers a Domino server with a single point of control. These benefits, along with the high availability infrastructure and scalability potential of the RS/6000 SP, make it an excellent choice for Domino implementations. The success stories of the companies that followed this strategy are testimony to the effectiveness of this solution.

7.2.3 Lotus Domino Examples on SP

7.2.3.1 IBM

With 194,000 entries in its name and address book, IBM has one of the largest electronic directories in the world. Through an aggressive deployment process, this directory has become a key component of the infrastructure of its Notes and Domino 4.5 system which serves some 130,000 employees in 131 countries around the world. The system allows IBM employees to

communicate and collaborate on business processes on a daily basis. The details of the project are described in a case study entitled: "Enterprise Deployment of Lotus Notes: an IBM Case Study." It states: "In mid-1995 we accepted the mission to migrate 130,000 employees to a client/server-based environment anchored by Lotus Notes over a twelve-month period beginning January 1996." And they were successful: "By the fourth quarter of 1996, we had exceeded IBM's Notes seat deployment objectives worldwide...[and] had achieved 99.6% availability for all Lotus Notes servers."

Previously IBM had worked for years with a VM-based office system. The primary reason for the migration to Notes and Domino was to take advantage of its superior electronic mail and groupware capability. But Notes' scalability offered even more benefits, allowing IBM to centralize administration in order to maintain the necessary coherence for the entire system and reduce costs.

For example, IBM is deploying Notes on one, large SP in each major European country. This type of scalability enabled IBM to move to only nine major Control Centers and 16 Satellite Centers connected in a hub-and-spoke topology. The system configuration supports 1500 users on 2-way SMP servers, and up to 1800 users in 4-way SMP machines. The new system now carries up to 3.9 million messages per day.

7.2.3.2 Lotus Virtual Lotusphere Site

How do you support a Web site that gets from 800,000 to 1.2 million hits/day? And up to 360 hits/second? A site that provides access to 30 information databases with over 20 gigabytes of data, creating a cyberspace experience of interactive conference participation, software downloads, chat rooms, distance learning, real time streamed video and audio and secure electronic commerce?

With numbers like that, you need a highly scalable platform. The Domino Web server is up to the job, and IBM hardware equipped with AIX can provide the power to use it all. The Virtual Lotusphere site ran on 22 POWERparallel SP uniservers, using AIX V. 4.1.2, 256MB of RAM, with an average of 3GB of disk space on each server. IBM Network Dispatcher provided load balancing and ensured operation without failure. The system used a maximum of 12 servers at any single point in time, with each of 30 information databases mirrored on 12 other SP servers. Even with all that workload, CPU utilization ran at an average of 9%. In addition, worldwide access was available via desktop PCs using Microsoft Internet Explorer or Netscape Navigator.

7.2.3.3 Wake Forest University

Today university IT organizations, like their counterparts in the corporate world, are being pulled in two directions regarding requirements: to provide greater capacity and to reduce costs. To balance these demands, Wake Forest University selected Lotus Notes. Why? As Jay Dominick explains:

“We looked at Exchange and Eudora as well as Notes. Scalability was an important consideration as were security measures like authentication, encryption, and certification. SMTP compatibility was a must. The system also needed to support POP, MIME, and it had to leverage our IBM UNIX server investment. Notes was the winner.”

“Our ROI is measured in terms of satisfied students and faculty. What makes them happy is stable and reliable email that has a very high functionality and is completely compatible with the Internet. Notes delivers that. And its scalability clearly reduces the impact on our most constrained resource: people.”

To satisfy users and to reduce demand on maintenance and administration, Wake Forest uses IBM SP servers with an SMP node for e-mail and an MTA server on a thin node for academic/administration purposes. Both nodes run Domino on AIX 4.1.4. The e-mail server uses Symmetrical Multiprocessing supporting 6 processors. Wake Forest is also using the Shared Mail (that is, single copy message store) feature. “In an academic setting messages often have multiple recipients - such as an entire class,” says Dominick, “having a single copy of the message saves space.” Wake Forest also puts a 7.5MB cap on each student's hard disk space. But even with all of these efforts, Notes Mail's large storage capacity comes in handy: Dominick has a 100 gigabyte capacity on the Notes Mail server.

By the fall of 1997, Wake Forest will support over 5000 users on a single server. “I think Notes allows us much room for growth considering that we are running Notes on a highly scalable platform. To date, we have not yet reached the maximum number of concurrent users on any of our servers,” says Dominick. “In the future, we'll implement server clustering to ensure non-interrupted service for everyone.”

7.3 Decision Support Systems (DSS)

Every day, huge amounts of data pour into your business -- sales transactions, inventory figures, billing information, marketing campaign responses, technical and engineering data and much more. More data, in

fact, than anyone can be expected to look at, which is why the average business uses only 7% of the data it collects.

Decision Support Systems (DSS) provide *business intelligence* to explore this data and to enhance business decisions. Several terms used by DSS are summarized in Figure 142 on page 301.

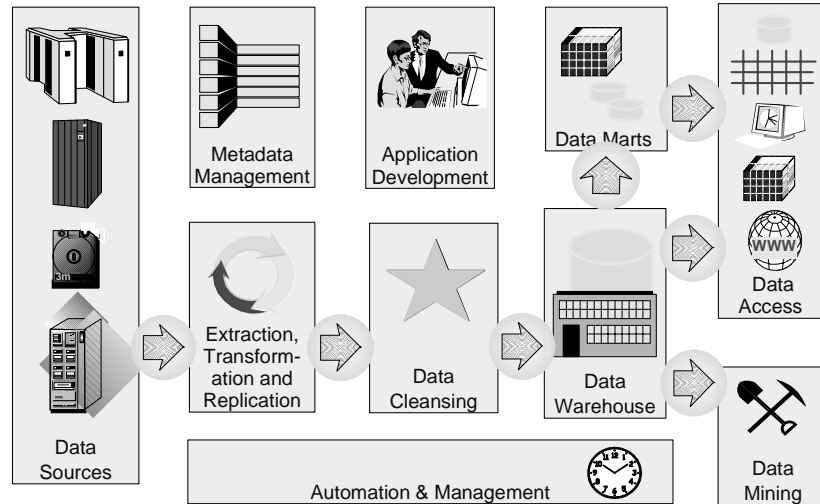


Figure 142. Business Intelligence Framework

A Data Warehouse defines a collection of data designed to support management decision making. The core database is a central repository for enterprise-wide data. A warehouse is normally a large relational database that comes with its own set of tools for loading, creating indexes, and performing other standard utility operations.

Data Marts are designed to focus on the information requirements of a single department, function or group.

Ideally, data marts are populated with data drawn from the data warehouse. Sometimes these marts employ special techniques to improve efficiency when dealing with queries in a single object context. These techniques include OnLine Analytical Processing (OLAP).

OLAP tools enable users to analyze different dimensions of multidimensional data. For example, they provide time series and trend analysis views.

Data access is provided by a mixed set of tools on the end user desktop. These tools range from personal databases to spreadsheets and desktop OLAP software, and also extend to intranet or Internet access.

A unique set of tools is categorized as *data mining*. Data mining is generally a batch process where computer algorithms sift through data to detect previously unknown patterns and correlations. For example, data mining software can help retail companies find customers with common interests. The term is commonly misused to describe software that presents data in new ways. True data mining software discovers previously unknown relationships among the data.

Data can be "discovered" with data mining tools like Intelligent Miner (IM). IM is a suite of tool functions that support data mining operations and deploy a variety of techniques to:

- Create classification and prediction models.
- Discover associations and sequential patterns in large databases.
- Automatically segment databases into groups of related records.

Warehouse data generally comes from a diverse set of operational systems or from purchased data. These sources vary widely in their operating systems and hardware implementations. Clearly, mainframe-based systems contribute in some fashion to most warehouses. So there have to be tools that can get at this mainframe data and feed it to the warehouse. These tools generally use extraction or replication techniques, and mainframe-based job scheduling systems generally trigger the associated processes.

Typically, data must be transformed in some way as it travels from operational systems to the warehouse. A fully populated column in a warehouse table may come from several different operational systems and logic needs to be maintained about what the selection criteria was from each source and what the transformation logic is to map that to the warehouse target. This mapping is stored by *metadata* tools.

Sometimes business logic is imbedded into data fields of systems as a way of implementing a quick change to a legacy system. If the intelligence in this logic is to be retained in the warehouse, special *data cleansing* tools are often necessary.

Applications are written to customize data presentation to end users, and sometimes to deal with customized ways of processing data on the way into the warehouse. Sometimes, it is desirable to have these applications

themselves be enabled to exploit shared nothing parallel architectures and there are tool providers to help support this.

7.3.1 Why a Parallel Approach is Key for DSS

Example 1 - In this example, the database is a table of 100 bytes x 600 million of rows = 60 GB of data. Assume we need 10 instructions per byte of data. The total number of calculations becomes $10 \times 60 \text{ MB} = 600 \text{ MB}$ of data.

With a processor running 50 million instructions per second, we need 12,000 seconds (200 minutes) to scan the table.

10 processors will reduce this time to 20 minutes; 100 will perform the job in 2 minutes.

Example 2 - To illustrate this concept in another way, a good example would be taking five decks of playing cards and passing them out evenly to 100 people, asking everyone to find their kings and pass them to you. Compare this with how long it would take you to search through all the cards to find the kings by yourself.

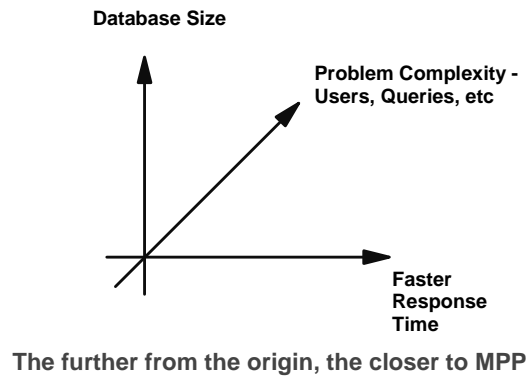
Three factors/axes increase everyday and drive the DSS market:

- Database size
- Problem complexity
- Faster response times

For example, forecasts from Gartner or Palo Alto Management Group show that database sizes will be roughly multiplied by 30 in five years.

Figure 143 on page 304 illustrates these three trends.

Clearly the reasons for using multi-processing processors (MPP) machines will become more and more compelling as we move away from the origin of this diagram.



Source: Ken Rudin - Emergent Technologies

Figure 143. Three DSS Axes/Trends

7.3.2 Why Use the SP for DSS

For large data warehouse applications, the SP offers a broad range of scalability.

Linear scalability with DSS has been demonstrated many times with different Relational Data Base Management Systems (RDBMS). An example is shown in Figure 144 on page 305. For identical data in the database, response times to requests are inversely proportional to the number of nodes.

Scalability means that you get constant response times as the database grows. You just need to increase the number of nodes. For example, the response time for N nodes and D data in the base will remain the same for $2N$ nodes and $2D$ data.

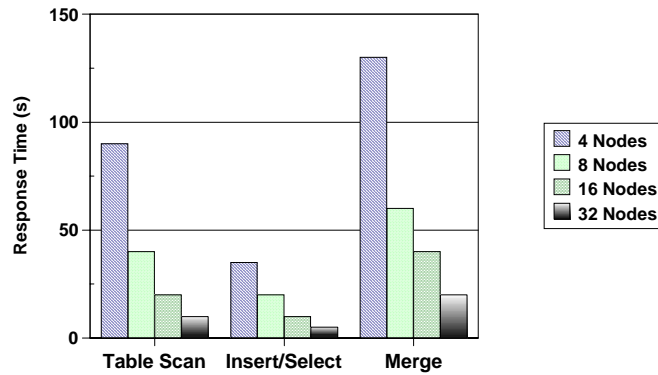


Figure 144. Scalability with DSS using DB2 PE

Scalability is not the only reason for choosing an SP. Other reasons are:

- The SP allows linear scalability with more than 90% efficiency.
DSS is both CPU- and I/O-intensive and the SP architecture supports lots of both.
- The High Performance Switch is effectively utilized.
This enables fast movement of many rows of table data between nodes.
- Relational Database Management Systems can support essentially all operations in parallel.
This is needed for very large database (VLDB) and/or complex queries.
- High volume parallel batch processing is possible.
The SP helps with loading, updates, printing, valuation, backups, data cleansing, and so on.
- Interoperability with mainframes allows bulk data transfer over channels, as well as interoperability with legacy systems and networks.

7.3.3 Examples:

7.3.3.1 John Alden Insurance Company

This insurance company needs to perform analyses of millions of medical claims forms to measure outcomes, quality and costs.

To meet this requirement, a 26 node SP with 36GB of serial disk per node (936GB total) was installed, running Oracle Parallel Server 7.3. 250GB of raw

data, including 65 different tables, and files over 10GB in size with 37 million very wide rows receive common queries with up to six multi-table joins.

The result has been that the SP takes only minutes to perform tasks that used to take a mainframe all night.

7.3.3.2 Aetna

Aetna is one of the largest insurance and financial services organizations in the United States. Its core business is health care insurance and retirement services.

Aetna needed to understand the continuum of care being provided to their insured population in order to positively influence the quality of overall care being given. Prior stovepipe applications for areas like Pharmacy dealt with incidents of care independently and did not provide the ability to view diagnosis, procedures and results effectively.

To meet this requirements the SP configuration installed in mid-1997 had 119 processors in 50 nodes, housed in 5 frames. Roughly 4.2TB of disk were also installed and all data was mirrored. The core database is DB2 PE, and testing in preparation for migration to UDB is underway.

Tools such as HOLOS (an OLAP tool) and Statistical Analysis (SAS) are also being used.

The node configuration is summarized in Table 12 on page 306.

Each high node has (53) 4.5 GB 7133 disk drives, which totals $9 \times 53 = 477$ drives. Each thin node has (12) 4.5 GB 7133 disk drives, which totals $38 \times 12 = 456$ drives. Collectively, these 933 drives provide $933 \times 4.5 = 4.2$ terabytes of disk space.

Table 12. Node Allocation

Node Type	Purpose
8 thin nodes	Pharmacy (database)
20 thin nodes	HMO (database)
5 thin nodes	UDB (database)
1 thin nodes	SAS (statistical analysis)
2 thin nodes	Testing

Node Type	Purpose
2 high nodes	HOLOS (OLAP tool)
3 wide nodes	Communications gateway - CLIO/S
6 high nodes	HMO conversion
1 high nodes	Spare

AIX 4.1.5 is installed on the thin and wide nodes and 4.2.1 is used on the high nodes. High Availability Control Workstation (HACWS) is currently being implemented.

7.3.3.3 An example of datamining: ShopKo Stores Mines

ShopKo Stores is a major regional discount chain with annual gross sales of more than \$2 billion and stores located in 15 midwestern and northeastern states. Each of its 129 stores offers more than 300,000 UPCs, distributed among hard lines, soft lines, home, in-house pharmacy and in-house optical departments.

In 1994, ShopKo began a major undertaking to reengineer and rebuild many aspects of its business by using the most advanced technology available. In addition to installing IBM RS/6000 computers as in-store processors throughout the chain, the company also began researching alternative corporate platforms. After a rigorous benchmark analysis, ShopKo installed IBM's Scalable POWERparallel System (SP) to meet its current and future needs.

"We decided to replace our mission-critical systems with client/server applications. By going with IBM's open systems architecture with parallel processing, we gave ourselves long-term choices, for example, flexibility in hardware and software selection," said Jim Tucker, senior vice president of ShopKo.

ShopKo was the first customer in the world to do transaction processing using an SP. The SP uses processor nodes based on IBM's POWER2 RISC technology. The SP is using 25 nodes in a production environment. IBM's Client Input Output/Sockets (CLIO/S) product allows connection and fast data transfer between an external host and the SP.

This massive computational power enabled ShopKo to use IBM's data mining tools to analyze data for their advertising and merchandising groups. The company recently performed a Market Basket Analysis project to test the

effectiveness of their promotional campaigns, including its direct mail weekend circulars. As the senior vice president noted:

“Retailers build huge mounds of data and never seem to ask what information might be available. With IBM's advanced data mining tools and the tremendous horsepower of the SP, we can easily manipulate data warehouses in the hundreds of gigabytes, and even terabytes, without disrupting our routine processing operations,” said Tucker.

ShopKo's Market Basket system is now running two sets of applications:

- The first is an advertising planning system used to produce financial information. This system allocates the necessary ad space and placement dollars to the particular products, and then calculates the margins and product sales volume expected from the promotional campaign. This system is also used for ad-hoc analysis to test buying trends and other consumer patterns.
- A second system uses information obtained from the first application to plan and construct advertising campaigns, including artwork, graphics and copy. This system even downloads an ad's color separations to be delivered to the appropriate printer, generates signs for stores, captures sales generated at stores as a result of the campaign, and feeds this information back into the Market Basket Analysis system. The system also builds a database of campaign results that can be pulled up later for future planning.

7.4 Internet/Web Servers

The SP and the Interactive Session Support/Network Dispatcher (ISS/ND) software is a key Web-serving platform offering many advantages over alternative solutions.

There are three main selling points for using the SP in a Web-server environment:

- Scalability
- Cost of ownership
- Flexibility

7.4.1 Scalability

The scalability is possible because:

- The nature of a Web request is that it does not generally require lots of resources to be completed.
- The SP is viewed as any number of needed IP addresses. The number of servers applied to each of those IP address can either map to a single or many servers.

We can described three aspects of scalability:

- Bandwidth

Assuming you have sufficient network capacity to accept the data coming out of the SP, you can achieve a system bandwidth that is an aggregate of the bandwidth on each of the servers individually. It is this *scalable bandwidth* that makes the SP an excellent choice for a proxy-server environment where response time is a very important component of the solution.

- Processing Power

As with the bandwidth discussion, the processing power of the system is equivalent to the aggregate of the processing power that is available on each of the servers. This fact is particularly important to application areas where lots of dynamic content is being served up. *Dynamic content* is, by definition, content that is created on demand in response to a Web request. The origin of the dynamic content could be a program that runs a query against a database, or a single script that executes on the server to return some random data to the user. In both cases, CPU cycles are not needed to return the data, but rather to create the content that will be returned to the client.

Secure Socket Layer (SSL) encryption is another CPU cycle burner that often drives the need for the scalable processing power available in the SP. This ability to apply as much processing power as desired at a Web site is particularly attractive to a solution area such as electronic commerce where there is a high percentage of dynamic content and where almost all of the Web conversation occurs with SSL encryption enabled.

- Active Users

The number of possible active users follows almost directly from the scalable bandwidth and scalable processing power. The SP's ability to support a large number of active users makes it the perfect choice in an environment where you need to serve up lots of static content to a very large user community. An 8-node SP was installed at Netscape Communications where it served up 1000 HTML documents per second, with more than 20000 active connections during the peak period.

Of course, the bandwidth, processing power and active users scale linearly to the number of SP nodes applied to a solution.

7.4.1.1 Cost of Ownership

The cost of ownership of an SP system used as a Web server follows the same rules as for the consolidation of servers discussed in Chapter 8.3, “Server Consolidation” on page 355.

7.4.1.2 Flexibility

In addition to the flexibility offered by the SP itself, Interactive Session Support/Network Dispatcher (ISS/ND) adds even more flexibility.

ISS/ND allows requests to be routed to back-end servers based upon the IP address (which is in the destination field of the TCP/IP packet) and also based upon the service or port number to which the incoming packet is directed. Further, with very simple-to-use commands the administrator can instruct the ISS/ND to send/not send traffic to any back-end server.

These functions open up a wonderful world of opportunities for the administrator. The content of several domains can be housed in a single node or, for very busy domains, spread across multiple nodes. In a multinode domain, any of the nodes can be removed for service without removing that domain from the user community.

Likewise, the administrator can segregate servers by the services they provide. It might make good sense to place your FTP servers on a different network from your Web servers, in addition to separating the functions across different nodes.

Both tasks are easily accomplished with ISS/ND. Administrators who decide to house their Web traffic on the SP benefit from the economy of scale that comes with the SP. In particular, highly redundant disk arrays like the IBM 7135 can be attached to an SP to insure that the customer data continues to be available in case of disk or controller failures (such an I/O attachment is generally very tough to justify with a single SMP or a couple of node clusters).

Likewise, SP nodes can be assigned primary and additional secondary roles in the implementation of an HA solution. For example, in addition to being the Web server for a particular URL, a single SP node could also be given the responsibility of serving as the HACMP backup node in case of ISS/ND node failover. Built into ISS/ND is the ability to detect lack of progress/activity in back-end servers.

In such a situation, work will stop being routed to that back-end server and will be shared among the remaining nodes of that domain's pool. This solution was built with both high availability and high scalability in mind.

7.4.2 Applications

There are three main application areas for which the SP-based Web server is a good solution:

- Internet service providers
- Content hosting
- Mega Web site management

In addition to these, a few details of other well-known applications are discussed in the following sections.

7.4.3 Internet Service Provider (ISP)

An Internet Service Provider connects end users to the Internet. The services offered with a subscription include access to news, chat room and bulletin board services, some authority checking and service, a home page, the ability to send and receive mail and Web access via a browser such as Netscape's Navigator.

The SP with ISS/ND is an ideal solution because of the ease with which these various services can be split across SP nodes, the asymmetry of configuration that is possible, and the High Availability solutions with HACMP.

For example, Frontier Communications purchased a 3-node SP to provide Internet service to their 25,000 clients in the Rochester, NY area. But since Frontier is a holding company for many other regional phone companies, they plan to grow the customer base as they offer the service in their other regions. Frontier implemented their solution on the SP using public domain software.

7.4.4 Content Hosting

Customers in either the Internet or intranet world who have a lot of content to be delivered would also be well-served with an SP platform. These types of customers can fall into two categories:

- Those that house and serve up their own content
- Those that provide the content hosting service to smaller companies with the resulting aggregate bandwidth being quite large

In the first case, an SP-based solution would make sense if there were a large number of accesses to the content requiring the scalable performance of the SP for acceptable Web response time; but it would also be a good fit for a solution where the quantity of data is huge and response time is not a critical factor. In this case, a single node could serve as the Web server, but you might employ several back-end database servers that each have access to the desired data, which could be retrieved by any of these servers.

As an example, suppose you have four clients and each one has a relatively inactive Web site. You can place all four of those Web sites on a single workstation. They can each have their own IP address and their own hostname URL, but they would all electronically get mapped to the single network adapter on that workstation. On this workstation, four instances of our Web server software would be running, each with its own list of documents and each binding to its own IP address.

Now combine this concept with ISS/ND and the SP. ISS/ND has a network adapter that could be aliased to all of the IP addresses for all of the hosts whose contents are being housed on the server nodes:

- Node 1 would house the content for companies A, B, C,D.
- Node 2 would house the content for companies E, F, G.
- Nodes 3 and 4 would house the content for company H.

We could continue this for any number of combinations and permutations. The client request would come into the ISS/ND node and be routed to the back-end servers who are responsible for that IP Web-site. Then the Web server that is listening for that IP address would handle the Web request.

7.4.5 Mega Web Site Management

Because of its excellent scalability, ISS/ND and the SP provide a winning combination for problems that are large in scope. One such example is socks servers. Almost all companies that allow their employees access to the Internet from their intranet do so with the use of a socks server. The client's browsers point to the socks server, which then takes the client's Web requests and sends them out on the Internet. For a large corporation, a large number of socks servers would be required to implement this service.

A solution to this socks server problem is to house all of them in an SP and allow one of the nodes on the SP to run the ISS/ND software. All the corporation browsers would point to this node, which would then route the requests to the heaviest-loaded socks server. This solution to the socks server problem insures that the load is adequately balanced across the

servers, and it also reduces the support costs compared to a regular cluster of workstations.

Likewise, if an Internet Service Provider wants to provide excellent response time to these customers, they may elect to place some proxy servers between the clients and the Internet. Of course, in such a solution, we do not want our proxy server to become a bottleneck. A natural response is to replicate the proxy servers and then balance the client requests across each of the proxy servers. The advantages described in the socks server example also apply to the scalable proxy server solution: balanced work across the servers and lower support costs.

The SP is a perfect solution for extremely busy Web sites, especially those serving up static content. The number of responses per second that the administrator wants to be able to support scales directly with the number of back-end servers. With only a slight reduction in overall performance, the administrator can also employ a shared file system (AFS, DFS, GPFS, and so on) so that changes to documents need only occur in one place. Also, as previously mentioned, the solution is equally applicable to customer problems that are not only constrained by their ability to serve up documents, but also by the inability of the back-end servers to do the processing necessary for completing transactions.

7.4.6 Other Well-Known Applications

Many well-known examples of Web servers powered by an SP exist around the world.

Winter Olympics 98 (Nagano)

To provide data processing capability for the 1998 Winter Olympic games more than 100 nodes spread across five SPs around the world handled schedules, calculating results, generating reports, and so on.

More than six million transactions were processed on an intranet-based system, which delivered news and results to more than 80,000 accredited members of the Olympic family (athletes, judges, coaches and reporters).

In the IBM Surf Shack, more than a quarter of a million e-mail messages were received by over 2,000 athletes.

Chapter 8. Commercial Applications

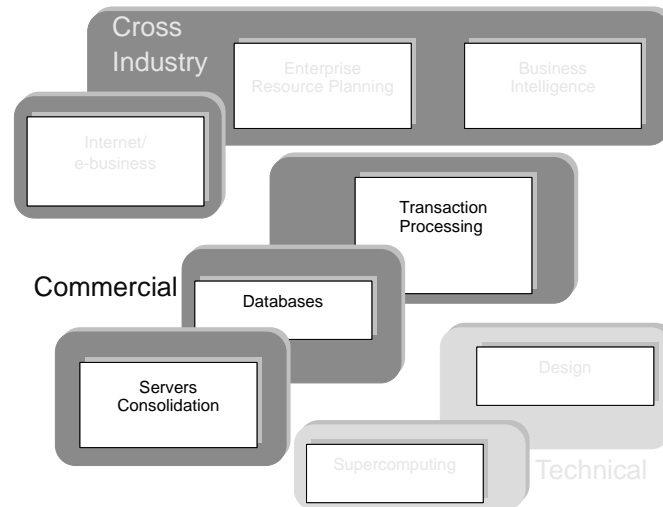


Figure 145. Commercial Applications

This chapter describes:

OnLine Transaction Processing (OLTP)

We present the OLTP concept and describe why SP is the key element of this high availability solution.

Databases

We describe how databases work in a parallel environment and how to use SP with databases. We focus on two products:

- DB2 PE
- Oracle

Server consolidation

One of the key reasons for using an SP is to consolidate servers. In this section, we discuss the advantages of server consolidation and give examples of how you can use an SP to achieve this goal.

8.1 OLTP Mission-Critical Applications

OnLine Transaction Processing (OLTP) is the solution for any high end business where the systems are highly available utilizing the UNIX operating system. This is consistent with IBM's traditional business, where large scale mission-critical applications is a requirement.

Many customers are moving to an open environment or are developing leading-edge applications, and they require greater flexibility, scalable performance and systems management.

In this competitive environment, industries including finance, telecommunications, insurance, health, transportation and travel have the following requirements:

- Transaction integrity
- Security
- Consistent response times
- Continuous system availability

They also have the following system requirements:

- Continuous availability from the user point of view
- Backup and recovery
- Manageability with single point of control
- Scalability
- Interoperability
 - Distributed/network computing
 - Web interoperability

They also need service and support, including consulting, implementation services, support line and training.

To summarize, the mission-critical OLTP client/server used by these industries and others requires a combination of technology and support factors to get mainframe-like service in a UNIX environment.

8.1.1 Distributed System

A *distributed system* consists of multiple software components that run in separate independent processes on different machines in a network. For example, a distributed debit-credit operation can require applications that run

on UNIX workstations and Windows NT machines, and also have data residing in a database on a mainframe. Even though the software is distributed across a network, it can be accessed reliably by multiple users as if it were running on a single system. Bank representatives in diverse locations can process a debit or credit request, accessing the programs and data as if they were local.

A *transaction* is a set of related operations that must be executed as a unit (though each operation may run in a different process).

The debit-credit transaction can consist of multiple related operations; for example, updating several databases, displaying results on a user screen, and forwarding information to a message queue. All related operations define a transaction—the transaction is not considered complete if any one of its operations is not carried out. *Transactional systems* coordinate multiple related operations being carried out in different processes in many network locations. They coordinate the work so that it is completed successfully and accurately in all locations. When a debit-credit transaction successfully completes, all affected bank accounts have correct balances, and the correct balance is seen from all locations of the system.

8.1.2 2-Tier and 3-Tier Concepts

Distributed architectures take many forms. One form is the three-tiered client/server model.

In this model, a client/server system organizes software into two separate parts, clients and servers. *Clients* interact with users and make requests to servers. *Servers* perform services, such as updating or retrieving data in response to requests from clients.

Servers communicate with resource managers. A *resource manager* is an application that manages shared data, such as an Oracle or DB2 database used to hold account information. Servers can draw on a wide range of resources, including mainframes and relational database management systems (RDBMS), to perform their work. Figure 146 on page 318 shows an example of three-tiered design (the 3-tier concept).

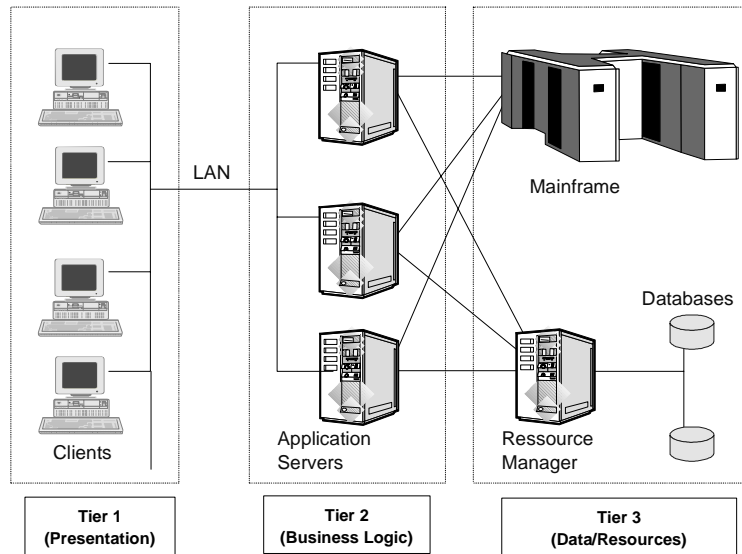


Figure 146. 3-Tier Architecture

The first tier contains presentation software, which consists of client applications that interact with users via screens or command-line interfaces. The client applications send requests to server applications in the second tier.

Second-tier applications contain the business logic, while the third tier contains data and resources, separating them from processing logic.

For smaller applications, other solutions exist such as the 2-tier approach, where the presentation and business logic are merged. Figure 147 on page 319 shows the 2-tier and 3-tier architectural concepts.

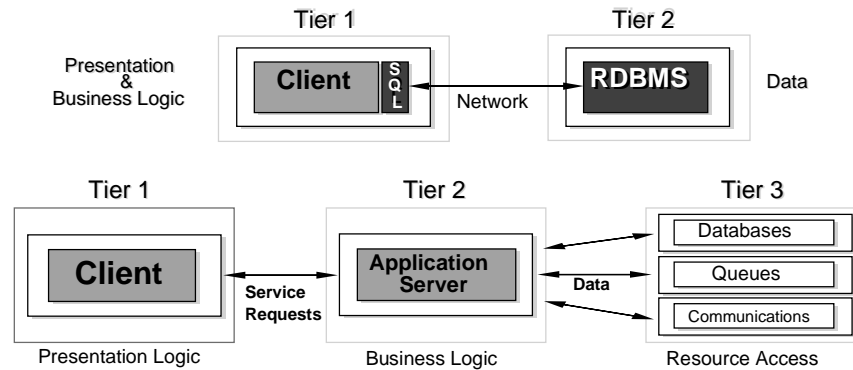


Figure 147. 2-Tier and 3-Tier Architecture

The difference between these two architectures are as follows:

The 2-tier concept solution is database-oriented, with a single source of data. This approach is ideal for:

- Applications supporting under approximately 100 users
- Applications with low security requirements

The 3-tier concept solution is ideal for domains with the following requirements:

- Flexibility
- Scalability with large numbers of users
- Multiple data sources support
- Transaction processing monitors for manageability, flexibility and security

8.1.3 OLTP Solution Components

An OLTP solution is built with three main components, as illustrated in Figure 148 on page 320:

- The server itself, including SSA and RAID storage subsystems
- Databases like DB2, Informix Oracle or Sybase
- Transaction Processing monitors like CICS, Encina or Tuxedo

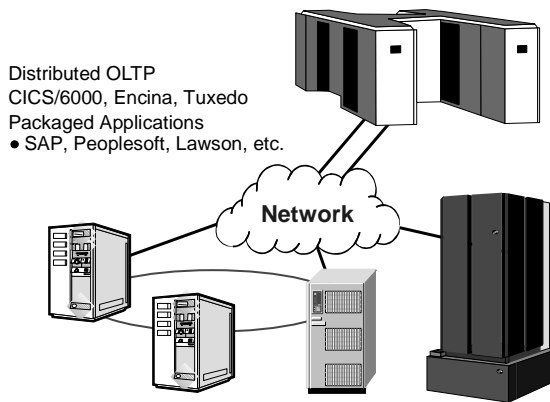


Figure 148. OLTP Scheme with SP

Of course additional components like CLIO/s or ESCON may be used.

8.1.4 Why Use the SP for OLTP

The main reasons to use the SP for OLTP are:

- The SP allows linear scalability with more than 75% efficiency.
 - OLTP typically is memory- and I/O-intensive and the SP architecture supports large amounts of both.
 - There is a need to support concurrent users far beyond the support provided by SMP machines.
- The High Performance Switch is effectively utilized.

This provides a network path from the gateway node to the servers, and between the application and database nodes, in the 3-tier client/server model.

- Application servers are in the system and so easily manageable.

This is illustrated in Figure 149 on page 321.

- Parallel RDBMS can be larger and support more users than a SMP.
- Many highly available configurations are possible:
 - Recoverable VSD and Fault Tolerant DLM for Oracle Parallel Server
 - HACMP is available for other vendors to address the OLTP market.
- TP monitors are designed for distributed computing.

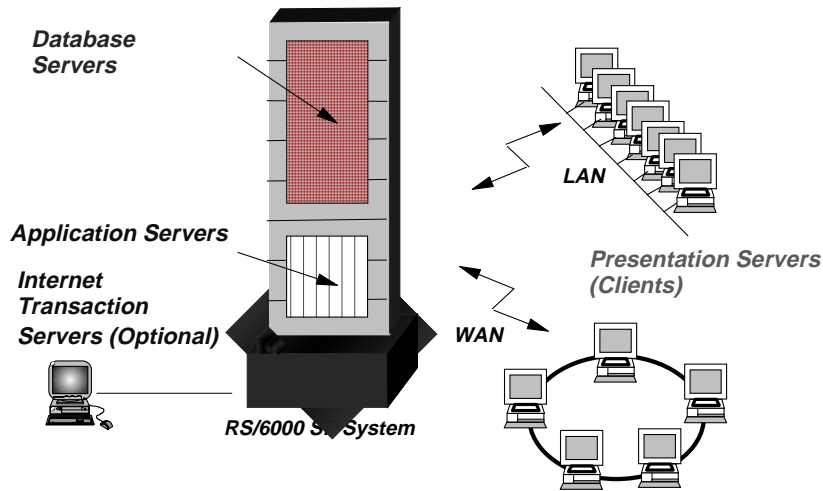


Figure 149. Typical SP OLTP 3-Tier Environment

8.1.5 Transaction Processing Monitors: CICS, Encina and Tuxedo

The basic idea of transaction monitors is to allow different parts of an application to communicate. These products are used for developing, executing, and administering distributed transaction processing systems.

Whichever form its architecture takes, a distributed transactional system requires the following key features:

- **Scalability:** It must support multiple servers, clients, and users and allow for expansion as more servers, clients, and users are added.
- **Fault-tolerance:** It must tolerate system and network failures without loss of data, and it must be able to recover from physical (media) failure and restore data to a previous consistent state.
- **Reliability:** It must be able to track and coordinate the work of many processes so that a failure in one process does not compromise the overall accuracy of data.
- **Security:** It must prevent unauthorized operations on data and shut out unverified parties.
- **Interoperability:** It must work with many different communication protocols, operating systems, and hardware platforms.

Note that due to the fact CICS and Encina are closely tied to DCE, the previous scheme needs to be modified to include this connection. Figure 150 on page 322 illustrates this concept when using CICS or Encina.

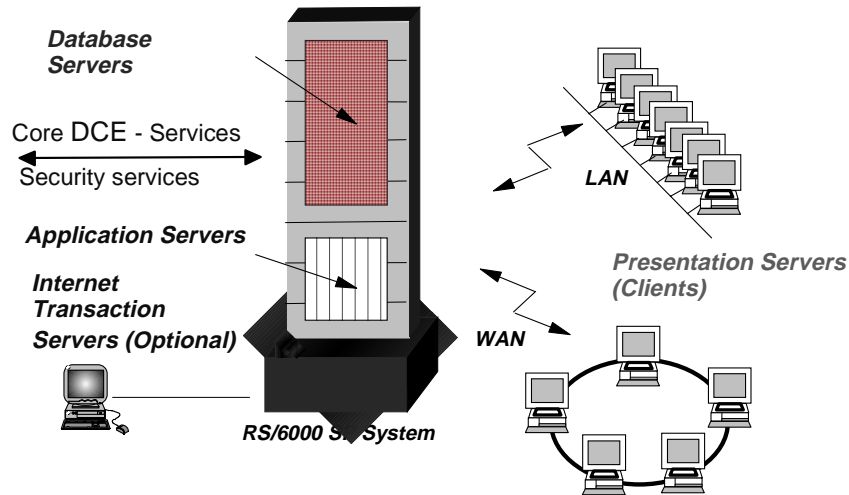


Figure 150. Typical SP DCE OLTP 3-Tier Environment

8.1.5.1 CICS

CICS is IBM's general-purpose on-line transaction processing (OLTP) software. It is a powerful application server that runs on a range of operating systems from the smallest desktop to the largest mainframe.

CICS seamlessly integrates all the basic software services required by OLTP applications, and provides a business application server to meet information-processing needs of today and the future.

Typical transaction processing applications include:

- Retail distribution systems
- Finance--banking, insurance, and stockbroker systems
- Order entry and processing systems
- General ledger systems
- Payroll systems
- Automatic teller machines
- Airline reservation systems
- Process control systems

Each product in the CICS family is designed to run on a particular operating system and hardware platform, and each product has powerful functions to allow interproduct communication with other members of the CICS family. CICS provides a cost-effective and manageable transaction processing system, allowing you to write your own applications or to choose from many existing vendor-written business products.

Transactions are familiar to us in everyday life when we exchange money for goods and services, such as buying a train ticket or paying for medicine. Such transactions involve a short conversation (for example, requesting availability and cost), and then making the payment. The processing of any one of these items is a business transaction, of the kind that is handled by CICS.

Viewed simply, a typical OLTP transaction consists of many computing and data-access tasks to be executed in one or more machines; the tasks may include handling the user interface, data retrieval and modification, and communications. In CICS terms, these operations are grouped together as a unit of work or a transaction.

A transaction management system (sometimes called a transaction monitor) such as CICS:

- Handles the start, running, and completion of units of work for many concurrent users.
- Enables the application (when started by an end user) to run efficiently, to access a number of protected resources in a database or file system, and then to terminate--normally returning an output screen to the user.
- Isolates many concurrent users from each other so that two users cannot update the same resource at the same time.

CICS is a layer of middleware that shields applications from the need to take account of exactly what terminals and printers are being used, while providing a rich set of resources and management services for those applications.

In particular, CICS provides an easy-to-use application programming interface (API) which allows a rich set of services (including file access and presentation services) to be used in the application and to be ported to and from a wide variety of SP environments.

8.1.5.2 Encina

Encina provides the infrastructure, development environment, and monitoring capabilities necessary for building and managing distributed transaction

processing systems. The following list summarizes the services and features provided by each Encina product:

- **Core Distributed Services:** The Encina Toolkit Executive and the Toolkit Server Core, together with Distributed Computing Environment (DCE) services, provide an infrastructure that ensures security, recoverability, and reliability for the distributed system.
- **Transaction Processing Monitor:** The Encina Monitor orchestrates activities in the distributed system by keeping server processes running, routing transactions across the system, and making sure that distributed work is completed successfully and accurately.
- **Application Development Environment:** Encina provides a rich set of interfaces and language extensions for developing custom client and server applications. These include the Monitor Application Programming Interface (API), the TX interface, Encina++, and Transactional-C.
- **Recoverable Servers:** The Recoverable Queueing Service (RQS) and Structured File Server (SFS) manage data stored in queues and record-oriented files, respectively. Both RQS and SFS are built using the Encina Toolkit services. The Toolkit Services can also be used to build custom-recoverable servers. Custom-recoverable servers can interact with Encina native resource managers (RQS and SFS) and with other resource managers. Encina also provides APIs for writing RQS and SFS clients.
- **Central Administration:** The Enconsole graphical user interface (GUI) permits central administration and monitoring of Encina.
- **Monitor Systems:** The Encina Control Program (enccp) is a command-line and scripting interface for administering Encina.
- **Monitor Cells:** XA Distributed Transaction Processing (DTP) Integration. The Transaction Manager-XA (TM-XA) interface is used to develop applications that can interact with XA-compliant resource managers. XA-compliant resource managers follow the X/Open standard which is a blueprint for how resource managers must behave in order to support transactional access. The TM-XA interface coordinates Encina's two-phase commit protocol with RDBMS.
- **Mainframe Connectivity:** The Peer-to-Peer Communications (PPC) Services permit connectivity between Encina and mainframe systems using a standard Systems Network Architecture (SNA) LU 6.2 protocol. The PPC Services coordinate Encina's two-phase commit protocol with mainframe systems.

8.1.5.3 Tuxedo

BEA Tuxedo is one of the most widely deployed transaction middleware products for building high-performance and reliable distributed applications. BEA Tuxedo provides an industry-leading middleware framework for building scalable 3-tier client/server applications in heterogeneous, distributed environments. With this transaction processor, developers can successfully develop, manage and deploy transactional applications completely independent of the underlying communications, hardware and database environment.

BEA Tuxedo provides an API that not only offers a wide variety of communication methods, but it also makes writing distributed applications simpler by supporting, among other features, transactional communication and the hiding of hardware and networking details from programmers.

Classic 3-tier partitioning separates human interaction (called presentation logic), business logic, and data access. In a BEA Tuxedo-based application on the SP, these functions can be partitioned onto various sets of processors. For example, several SP nodes might be dedicated to run the BEA Tuxedo Workstation Gateway process. This process receives input from PC-type devices, and allocating 1000 PCs to each node is entirely feasible. Ten such nodes might thus accommodate a total population of 10,000 PCs. If for example, each PC was an Automated Teller Machine (ATM), you could hook 10,000 ATMs to one SP.

The business logic (for example banking operations) that these PCs access would exist as application servers running on other nodes dedicated to them. For example, assume that the system has an input every 25 seconds from each ATM. This requires 400 complex transactions per second for the SP to handle. Assuming 40 complex transactions per second per application server node, we could use ten such nodes to handle the application's business logic. The business logic will usually require database access. This access could either be directly on the application servers, or could itself be offloaded to special database servers. Let us assume the latter, with four such servers each giving 100 complex transactions per second.

So we have ten nodes accepting ATM input. Within the SP, nodes communicate with ten application server nodes using ATMI, the BEA Tuxedo client/server programming interface. These nodes in turn generate SQL requests to four database server nodes. Therefore, the total configuration processes 400 complex transactions per second.

The hardware and software throughout the SP complex can be centrally managed. The SP console can collect hardware statistics and the BEA

Tuxedo administration GUI can do likewise for the various clients and servers throughout the system. We might want to dedicate one such node as our administrative hub. In addition to classical PC input, we could have nodes accepting Internet requests or requests for dumb terminals.

Next, we might want to dedicate some nodes for communications to other computers, including MVS systems. To do this, you could just configure additional SP nodes to run the BEA Connect software. Communication with those servers is also via ATMI.

Finally, we might want to configure some nodes as spare nodes reserved for high availability. For example, we might have one spare node as a reserve for the application servers. Should one of those nodes fail, we could use BEA Tuxedo facilities to migrate the servers which had been working on it over to the backup node.

In summary, the SP system and BEA Tuxedo are ideal for each other. BEA Tuxedo was designed to allow business applications to be distributed, and the SP provides a great distributed hardware environment to host such a partitioned system.

8.1.6 Mission-Critical Power Solutions Examples

In this section we provide examples of SP implementation for OLTP.

8.1.6.1 Transportation Application

Problem - This customer needed to move from batch-oriented package tracking to near real time tracking. They also wanted to significantly increase the number of tracking scans per package to improve their tracking ability.

Solution - This customer developed an integrated package tracking application. This application is a realtime application running on an RS/6000 SP with the Oracle Parallel Server database and the BEA Tuxedo transaction processing monitor (this application replaces an older batch application which ran on a proprietary system). The new application is linked to both the mainframe and to the Web and supports 10000 users.

The application architecture is a 3-tier client/server chosen for its ability to scale in an effective manner and provide the required high availability characteristics and flexibility. The customer's availability target is 99.99%.

The customer chose the RS/6000 SP because it would scale better than an UNIX SMP or the older proprietary system (which was also more expensive).

Benefits - This new system allows the customer to more accurately track packages and dynamically determine how to reroute shipments around bottlenecks. They view it as a key infrastructure change and a huge competitive advantage. The system architecture provides the required high availability characteristics and flexibility.

Summary

- 80 SP nodes
- Oracle Parallel Server (on 45 nodes)
- BEA Tuxedo
- 10,000 users
- 3-tier architecture
- Both Web and mainframe connections
- Availability target: 99.99%

8.1.6.2 Health Care Industry (Hospital Management Application)

Problem - This customer was running on a proprietary (Tandem) UNIX system which was expensive to maintain and difficult to develop applications for. They were not able to leverage mainstream databases, middleware, system management, and so on. They needed to re-engineer their application (hospital management) for new functions and to fuel their business need to integrate/merge hospitals. The merge aspect meant that easy scalability was key. They also needed to interoperate with their existing systems.

Solution - They designed a system based on a 3-tier architecture utilizing the RS/6000 SP, Oracle Parallel Server database and BEA Tuxedo transaction processing monitor.

Benefits - The application architecture is a 3-tier client/server chosen for its ability to scale in an effective manner and provide the required high availability characteristics and flexibility. Their availability target is 99.99%. They have achieved over 99.4% and are approaching 99.9%. The application connects with an existing Tandem system. This application was developed to give the customer a competitive advantage.

The customer chose the RS/6000 SP because it would scale better than a UNIX SMP and had better availability. They moved from the proprietary UNIX system because of "openness" and better application availability.

Summary

- 12 SP nodes
- 3-tier architecture
- Oracle Parallel Server
- BEA Tuxedo
- Availability target: 99.99%
- Currently approaching 99.9%
- Only one scheduled outage in first four months

8.1.6.3 Telco Industry (Billing Application)

Problem - The market share leader in the United States for service bureau billing saw the need for an in-house billing solution for certain clients.

The convergence of voice, video and data is driving the development of a billing and customer care solution specifically for convergence. The customer needed a solution that went well beyond cable billing. They needed a system that could bill for many of the other services that cable companies and other communications companies were trying to package together-a convergence billing system. They needed to build a system (based on the client/server model) that could get to a level of scalability which could support tens of millions of customers.

They needed to build a "world-class, next-generation customer management solution" that increases flexibility to accommodate the rapidly changing marketplace.

Solution - The customer decided that building on their existing systems was only asking for trouble. The company decided to start from scratch. The architects quickly adopted a 3-tier architecture, with the database as the bottom tier, the desktop and end user on the top tier, and the middle tier using the transaction processing monitor BEA Tuxedo, along with application software. The three tiers enable the scalability that is needed for a world-class system. The RS/6000 SP system and the Oracle Parallel Server database complete the solution.

Benefits - The system is built on top of IBM's RS/6000 SP system, which is a scalable parallel system. You can attach hundreds of processing nodes into this system. For example, with a 14-node system, the customer can support over 1 million subscribers. The customer has designed an architecture so that

as you add new nodes, the number of users you can add goes up linearly as well. The solution has built into it the ability to send out bills and statements by several methods, and also to accept payments by several methods.

In fact, the first installation in Europe can bill via direct debit in some of the special ways that European banking systems work. The customer plans to use billing and payment method indicators to allow billing across the World Wide Web and on-line service providers, or even directly to the home and present it on the TV as needed.

The solution enables clients that use it to move into new service areas, to be able to enact new business models, to go after new customer types if they wish, to expand overseas, and so on. Much of the history of the billing industry has been of billing systems holding clients back from doing things differently or from going after services or customer types. This world-class billing system relaxes most of those constraints and allows the billing system to evolve and support the new business direction of the client.

Summary:

- 8 SP nodes
- 3-tier architecture
- Availability target: 99.99%
- Oracle Parallel Server
- BEA Tuxedo

8.1.6.4 Finance (Order entry, back-office)

Problem - This customer needed to develop an order entry application for mutual funds. Additional functions required included other back-office applications. The customer's key consideration was high availability.

Solution - The solution is a custom-written application running on an RS/6000 SP with the Oracle Parallel Server database and the BEA Tuxedo transaction processing monitor. This system was chosen for its high availability and replaced a competitive UNIX system which was not able to achieve the required levels of availability. The application architecture is a 3-tier client/server chosen for its ability to scale in an effective manner and provide the required high availability characteristics and flexibility. Their availability target is 99.99%.

Benefits - This application was developed to give the customer a competitive advantage. The customer chose the RS/6000 SP because it would scale better than an UNIX SMP and had better availability.

Proprietary systems were not considered as it is not their strategic direction.

Summary

- 12 SP nodes
- Growing to 6 frames/multi-terabytes
- 3-tier architecture
- Availability target: 99.99+%
- Mainframe attach planned
- Currently approaching 99.9%
- Oracle Parallel Server
- BEA Tuxedo

8.2 Databases: DB2, Oracle, Informix and Sybase

The development of parallel database systems resulted from an increase in the use of Relational Database Management Systems (RDBMS). There is also the expectation that responses should be received faster than ever before. Users may want their queries answered either by directly entering input from a terminal screen or through an application program. High-level query languages, such as Structured Query Language (SQL), were developed to access an RDBMS. Queries are structured to generate complex reports, searching gigabytes or more of data. These queries have grown not only in the amount of data searched, but also in complexity.

Based on the current rate of increase in the complexity of queries and volumes of data, it is not expected that the processing capacity of any single processor, or even a closely-coupled multiprocessor, will be sufficient to provide acceptable response times. It may be more cost effective to have more machines with less individual CPU power accessing data than to have one large CPU. Consider the example of scanning a table in a serial RDBMS. At present, a table is scanned at a rate of 2.25 MB/second. If the table size is 10 GB, it would take one processor over an hour to scan a table. If that processing could be distributed evenly among 128 processors, the time required to scan that same table would drop to 35 seconds. There are also architectural limits in terms of the number of disks that can be attached to a

single processor. This limits the size of a database held on a single processor.

In today's parallel database market, there are two basic database architectural approaches: the shared-nothing approach and the shared-disk approach.

In a shared-nothing database architecture, the database is divided into partitions based on partition keys (one key per table). Each node has exclusive access to a single partition. Examples for this architecture are Informix XPS, Sybase MPP, DB2/6000 PE, Tandem Nonstop SQL and Teradata.

A shared-disk database architecture is characterized by the fact that all nodes share all of the data in the database. An example of a database using this approach is Oracle Parallel Server.

Function shipping and Data shipping

Depending on the hardware architecture type, a parallel database can perform an operation in two ways:

- Function shipping
- Data shipping

Function shipping

Function shipping means that relational operators are executed on the processor containing the data whenever possible, so the operation (or the SQL) is moved to where the data resides. Function shipping is well suited to the shared-nothing architecture. Figure 151 on page 332 illustrates function shipping.

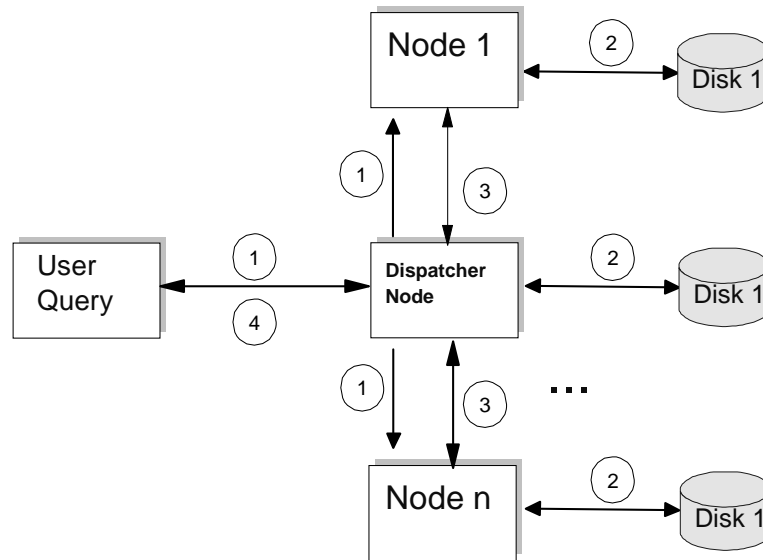


Figure 151. Function Shipping

The numbers in the figure are explained as follows:

1. First a relational operator is invoked. In Figure 151, an SQL select is issued. Every processor receives the operation from one processor, which works as a dispatcher. In DB2 Parallel Edition, this is called the coordinator node.
2. Every processor executes the operation on its own set of data.
3. An exchange of information among the nodes may occur.
4. The result from the operation is sent back to the dispatcher node. The dispatcher node assembles the data and returns it to the requestor.

This flow control architecture has two advantages:

- It minimizes data communication.
- No central lock manager is needed. Only a global deadlock detector, checking at regular intervals, is needed.

Data shipping

I/O shipping is implemented by data shipping to one or more processors and then executing the database operation. I/O shipping is particularly suited to a

shared disk architecture. Figure 152 on page 333 illustrates the steps in I/O shipping:

1. A relational operator is invoked; for example, an SQL select is invoked.
2. A processor is selected to run the operation (Figure 152 only shows one processor as being selected).
3. Every processor owning data involved in the operation sends it to the executing processor.
4. The executing processor performs the operation and returns the result to the requestor.

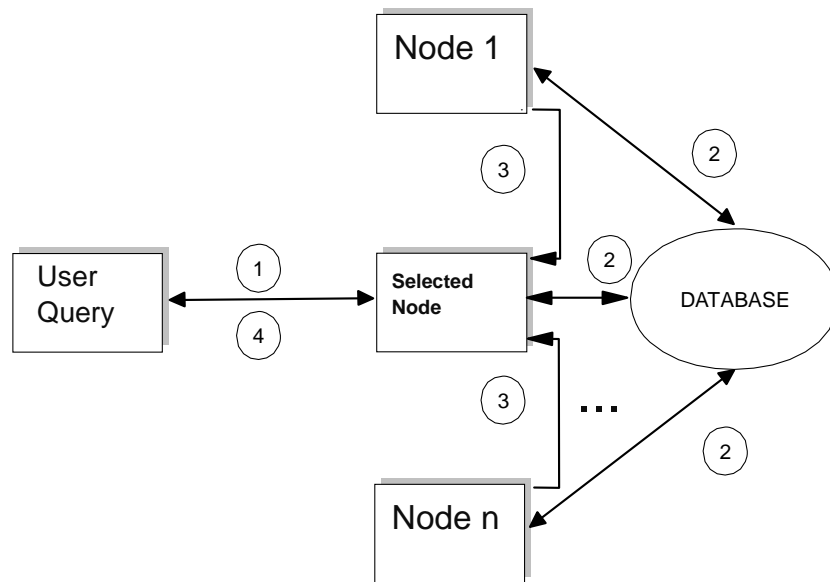


Figure 152. Data Shipping

This flow control architecture fits well in a heterogeneous environment. However, there are two disadvantages:

- Data movement is increased.
- A Central lock manager is required.

The advantage of I/O shipping is that a system may be configured with specialized machines:

- I/O machines with a large number of disks
- Compute machines with limited I/O capacity

Architecture comparison

Both database architectures have strengths and weaknesses, as summarized in Table 13 on page 334.

Table 13. Parallel Query Processing - Architecture Comparison

Shared-Nothing Architecture, Function Shipping (for example, DB2/6000 Parallel Edition)	Shared Disk, I/O Shipping (for example, Oracle Parallel Query)
More function shipping because database knows where the data is physically located	More I/O shipping because database does not know where data is physically located
Parallel optimization	Serial optimization followed by parallelization
Number of nodes of execution cannot be varied	Number of nodes of execution can be varied
Nodes cannot be separated between OLTP and DSS	Nodes can be separated between OLTP and DSS
All queries executed in parallel	Query should have at least one table scan for parallel execution

8.2.1 DB2

Architecture

DB2 Parallel Edition, an extension of the DB2/6000 Version 1 product, is a parallel database product that operates on AIX-based parallel processing systems such as the IBM RS/6000 SP.

DB2 Parallel Edition supports a shared-nothing architecture and uses the function-shipping execution model for its parallel process flow.

The data is managed by a database manager. The database manager controls CPU, memory, disk and communications resources. The database manager also provides users with the ability to store and access the data. The collection of data and system resources that is managed by a single database manager, together with its database manager software, is referred to collectively as a *node*. The node resides on a processor and one or more nodes can be assigned to each processor. In DB2 Parallel Edition, a node is called a Parallel Database Node or PDB node.

Parallel Database Nodes

The mapping of logical database nodes to physical machines is accomplished by the node configuration file, `db2nodes.cfg`.

Figure 153 on page 335 shows an example configuration with four parallel database nodes.

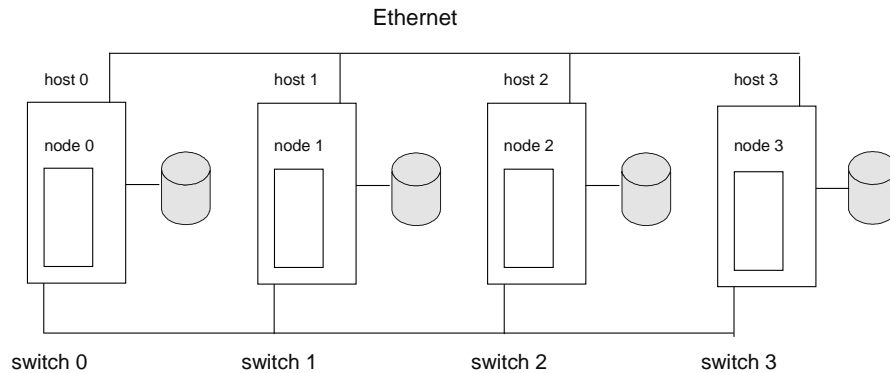


Figure 153. Four Physical Nodes Using DB2 and the SP switch

Database Instance

A DB2 Parallel Edition instance is defined as a logical database manager environment. Every DB2 Parallel Edition instance is owned by an instance owner, and is distinct from other instances. The AIX login name of the instance owner is also the name of the DB2 Parallel Edition instance. The instance owner must be unique for each DB2 Parallel Edition instance. Each instance can manage multiple databases; however, a single database can only belong to one instance. It is possible to have multiple DB2 Parallel Edition instances on the same group of machines. There are several reasons why you may wish to generate more than one instance on a group of machines:

- To maintain distinct test and production environments
- To use different software levels
- To keep separate access on different databases
- To keep separate sets of database manager configuration parameters for performance considerations

Each instance is related to an AIX user, the instance owner. The instance owner owns the files of the instance and has authority over the databases belonging to that instance.

Instance Owner

The instance owner has complete control over the instance. The instance owner can start or stop the database manager, modify the database manager configuration or modify parameters specific to a database.

Nodegroups

In DB2 Parallel Edition, data placement is one of the more challenging tasks. It determines the best placement strategy for all tables defined in a parallel database system. The rows of a table can be distributed across all the nodes (fully declustered), or a subset of the nodes (partially declustered). Tables can be assigned to a particular set of nodes. Two tables in a parallel database system can share exactly the same set of nodes (fully overlapped), or at least one node (partially overlapped), or no common nodes (non-overlapped). Parallel Edition supports the hash partitioning technique to assign a row of a table to a table partition.

Data Partitioning

DB2 Parallel Edition supports the hash partitioning technique to assign each row of a table to the node to which the row is hashed. You need to define a partitioning key before applying the hashing algorithm. The hashing algorithm uses the partitioning key as an input to generate a partition number. The partition number then is used as an index into the partitioning map. The partitioning map contains the node number(s) of the nodegroup. There are three major steps to perform the partitioning:

- Partitioning key selection
- Partitioning map creation
- Partitioning rows

Partitioning Key

A partitioning key is a set of one or more columns of a given table. It is used by the hashing algorithm to determine on which node the row is placed.

Partitioning Map

A partitioning map is an array of 4096 node numbers. Each nodegroup has a partitioning map. The content of the partitioning map consists of the node numbers which are defined for that nodegroup. The hashing algorithm uses the partitioning key on input to generate a partition number. The partition number has a value between 0 and 4095. It is then used as an index into the partitioning map for the nodegroup. The node number in the partitioning map

is used to indicate to which node the operation should be sent. Figure 154 on page 337 summarizes this concept.

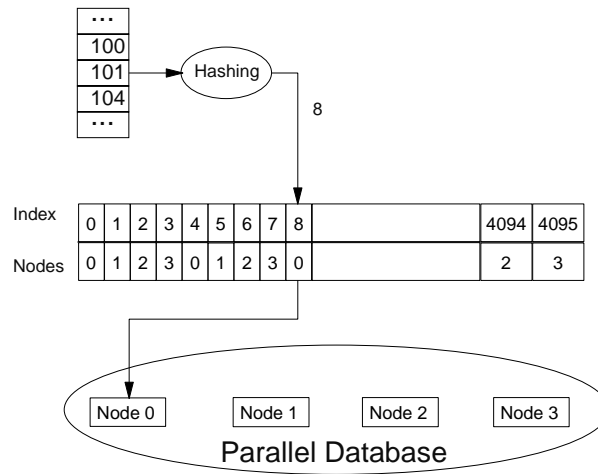


Figure 154. Partitioning Map

8.2.1.1 DB2 and High Availability

Continuous access to data (disk recover) and continuous access to application is done through High Availability Cluster Multi-Processing (HACMP).

Figure 155 on page 337 illustrates an example configuration with four nodes.

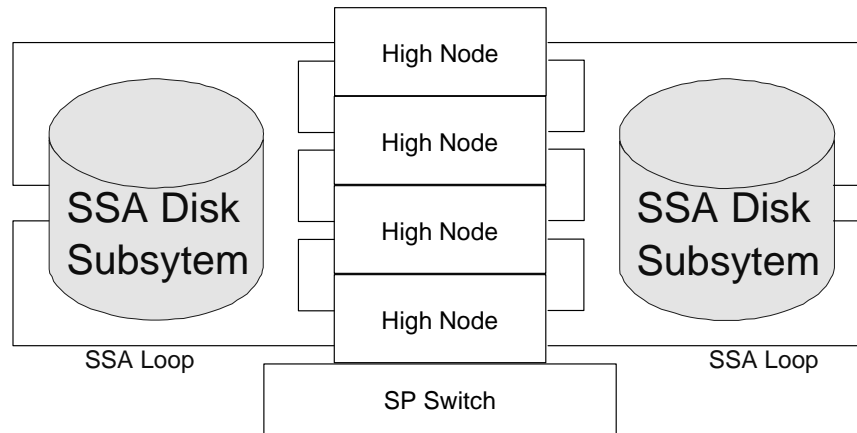


Figure 155. RS/6000 SP Configuration with HACMP

Each of the four high nodes contains:

- 1 SP Switch adapter
- 1 Ethernet adapter

The nodes are set up four clusters of two nodes. Each cluster is a mutual takeover cluster. After a failure, the backup node may vary on the volume groups of the failed node and start the failed DB2 PE partition.

AETNA U.S. HEALTHCARE EXECUTIVES DELIGHTED WITH TERAPLEX EXPERIENCE

Recent work done in the IBM RS/6000 Teraplex Integration Center has enabled Aetna U.S. Healthcare to go into production earlier than expected. The customer's objectives were met and the results will enable U.S. Quality Algorithms, Inc. (USQA), an affiliate of Aetna U.S. Healthcare, to deliver more comprehensive and complex analyses, faster. Sullivan McConnell, Director of Data Warehouse Development, USQA, says, "We are very pleased with the multi-user performance on the workload that we've put through the DB2 Universal Database on the RS/6000 SP. This speaks volumes for both the DB2 UDB and the RS/6000."

The nodes are set up four clusters of two nodes. Each cluster is a mutual takeover cluster. After a failure, the backup node may vary on the volume groups of the failed node and start the failed DB2 PE partition.

8.2.1.2 Aetna U.S. Healthcare Example

Recent work done in the IBM RS/6000 Teraplex Integration Center has enabled Aetna U.S. Healthcare to go into production earlier than expected. The customer's objectives were met and the results will enable U.S. Quality Algorithms, Inc. (USQA), an affiliate of Aetna U.S. Healthcare, to deliver more comprehensive and complex analyses, faster. Sullivan McConnell, Director of Data Warehouse Development, USQA, says, "We are very pleased with the multi-user performance on the workload that we've put through the DB2 Universal Database on the RS/6000 SP. This speaks volumes for both the DB2 UDB and the RS/6000."

Aetna U.S. Healthcare's research and development group delivers sophisticated information products and services to managed healthcare organizations, hospitals, physician groups, pharmaceutical manufacturers, government organizations and employers. In order to provide their customers the information they want--medical diagnostic trends, medical service costs and healthcare provider performance measurements, they needed an information warehouse that could integrate many operational datasets into a

single decision support environment. With the continually increasing volume of data and users that would need to access this warehouse, Aetna needed assurance for reliability and scalability.

Aetna believed that this data warehouse would be a key factor in their ability to provide the highest quality health care with excellent cost performance.

These requirements led Aetna to the IBM RS/6000 Teraplex Integration Center in Poughkeepsie, NY, where they could combine the dynamic RS/6000 SP in a parallel environment with DB2 Universal Database to fulfill their needs. The IBM Teraplex Integration Center provided the complex hardware and software configurations, as well as the technical expertise to enable Aetna to conduct performance tests on 100GB, 500GB and terabyte-sized databases.

The hardware configurations for the three databases were as follows:

- 100GB Database
 - 3 high nodes (8-way SMP 604 PowerPC 112MHz) w/2GB memory
 - 9 drawers of 4.5 GB SSA disks
- 500GB database
 - 10 high nodes (8-way SMP 604 PowerPC 112MHz) w/2GB memory
 - 48 drawers of 4.5 GB SSA disks
- 1 TB Database
 - 20 High nodes (8-way SMP 604 PowerPC 112MHz) w/2GB memory
 - 96 Drawers of 4.5 GB SSA disks

The following software configuration was used for all three hardware configurations:

- AIX Base Operating System, Version 4.1.5
- DB2 Universal Database, Enterprise - Extended Edition, Version 5 Beta
- Performance Agent Daemons and Utilities
- Version 2.1 SP System Support Package

One of the initial challenges of this project was to duplicate Aetna U.S. Healthcare's data warehouse in the Teraplex Center, building tables and indices to prepare for query analysis against their data. Another objective was to test DB2 UDB on a large database with real customer data and queries to understand its stability and functionality. Scalability testing was done from many different view points; does the code scale with many data

partitions, how does the hardware scale, and does DB2 UDB and TCPIP scale for large numbers of SMP nodes and data partitions. The results of these objectives were met and exceeded in many cases. Performance tests were conducted on a terabyte-sized database with up to 33 concurrent queries. Scalability testing was extremely successful, which made the Aetna executives very happy.

Aetna provided several complex queries, of which many had between 11 and 17 table joins and represented a star or snowflake pattern. Results for the query execution was several times faster than expected and demonstrated excellent execution times for concurrent users.

The largest table, which was over 400 GB, had 935 million rows, with some single rows over 400 bytes long. The table scan rate against this table was in excess of 30 MB/second, which was on a single 8-way 604 SMP high node. A job executing scans on two of the biggest tables concurrently had a scan rate in excess of 40 MB/second on a single high node.

8.2.2 Oracle

Oracle Version 7 includes two different softwares:

- Oracle 7 Single Instance: This version runs in independent mode (single instance) on different nodes of the SP (like a standard RS/6000).
- Oracle 7 Parallel Server: This version runs in parallel mode server (OPS) and parallel query (OPQ) on multiple nodes. This software allows multiple instances of Oracle to run on different nodes simultaneously while maintaining the look of a single image to the end user. Oracle Parallel Server consists of the normal Oracle executables, a component called the Distributed Lock Manager (DLM), and special I/O routines that use IBM's Virtual Shared Disk (VSD). The distributed lock manager coordinates the resources between Oracle instances on nodes of the IBM RS/6000 SP. One instance of Oracle runs on each node of the RS/6000 SP. Oracle uses IBM's VSD software to allow instances to access disks that are physically attached to other nodes. Data that needs to be exchanged between nodes is passed through the RS/6000 SP's high performance switch for increased system throughput.

This is illustrated in Figure 156 on page 341.

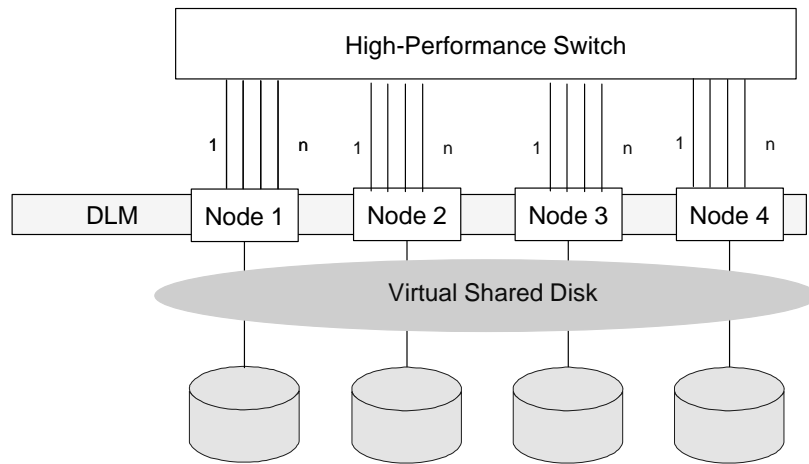


Figure 156. Oracle Parallel Server

Architecture

In an MPP system like the IBM RS/6000 SP, each node has its own memory. Main memory cannot be used to synchronize data access as it possible on a symmetric multiprocessor. Therefore, an additional component must be added to coordinate data access between nodes. This additional component is the Distributed Lock Manager or DLM.

Each node belonging to an Oracle Parallel Server system maintains its own cache in main memory. The cache holds the data blocks needed for transaction or query processing. The blocks are held in memory as long as possible to be available for the following transactions. If space is needed for new data blocks, old datablocks are removed from the cache according to a LRU (least recently used) algorithm. In an MPP system the contents of the caches on the different nodes must be kept consistent across node boundaries. This is the task of the DLM.

The DLM consists of multiple processes which are distributed across the nodes. The number of processes are not limited. The DLM manages the access rights for the nodes using global locks called PCM (parallel cache management) locks. PCM locks are not transaction oriented. PCM locks refer to all data blocks stored in any of the caches of the different nodes in the parallel server system. If one node needs a block, which is not stored in the local cache, it asks the appropriate DLM process, which can be located on a different node. if this block is already cached on another node. If it is already

cached, the block and the corresponding lock are transferred to the requesting node.

The DLM knows at each point in time which node has access rights to which blocks of data. With this knowledge, it can ensure consistency and avoid read or write conflicts. The PCM locks employed by the DLM are not to be confused with transaction blocks. PCM locks are “owned” by nodes, the transaction locks belong to individual transactions.

On the IBM RS/6000 SP, the DLM uses the high-performance switch for communication. The DLM uses the high-performance and scalability of the interconnect to efficiently handle distributed cache management.

The DLM is also the crucial component in database high availability. It detects the failure of an instance and coordinates instance recovery to restore database consistency allowing the surviving instances to access the database. For more details see the Chapter 8.2.2.2, “Oracle and High Availability” on page 349.

True and False Pinging

Figure 157 on page 343 illustrates how Oracle Parallel Server works to ensure data consistency. Assume block B1 is held in instance ORA1’s cache. Instance ORA2 wants to modify block B1. Instance ORA2 has to request the lock for B1 from the DLM. The DLM locates B1 on instance ORA1 and requests instance ORA1 to release the lock on B1 so that it can be granted to instance ORA2. Instance ORA1 writes B1 to disk and releases the lock, which can then be granted to instance ORA2. This procedure is known as “ping-pong” or simply as “ping”.

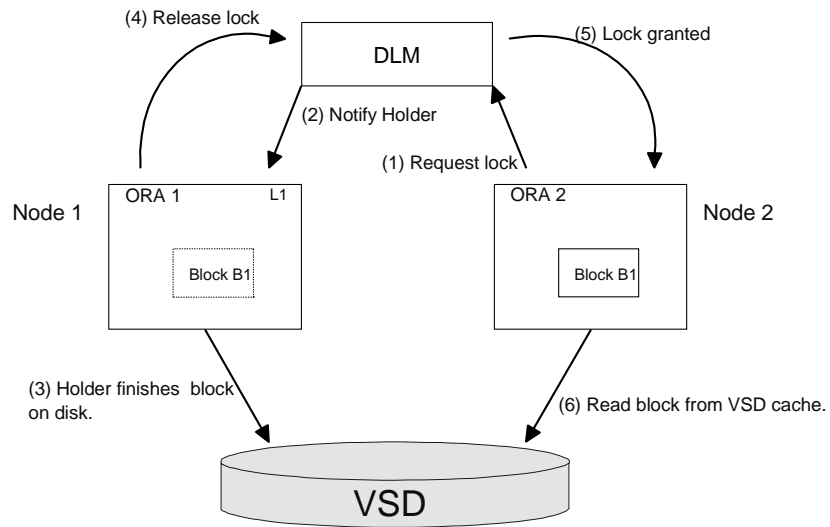


Figure 157. Events During a "ping"

Now assume that we have two blocks B1 and B2, and assume that they are covered by the same lock L. Instance ORA1 holds blocks B1 and B2 in its cache. Instance ORA2 requests the lock on block B1. In this case, instance ORA1 has to write both blocks B1 and B2 to disk before releasing the lock, although only block B1 has been requested. This is because both blocks are covered by the same lock. This is called a false ping (see Figure 158 on page 344).

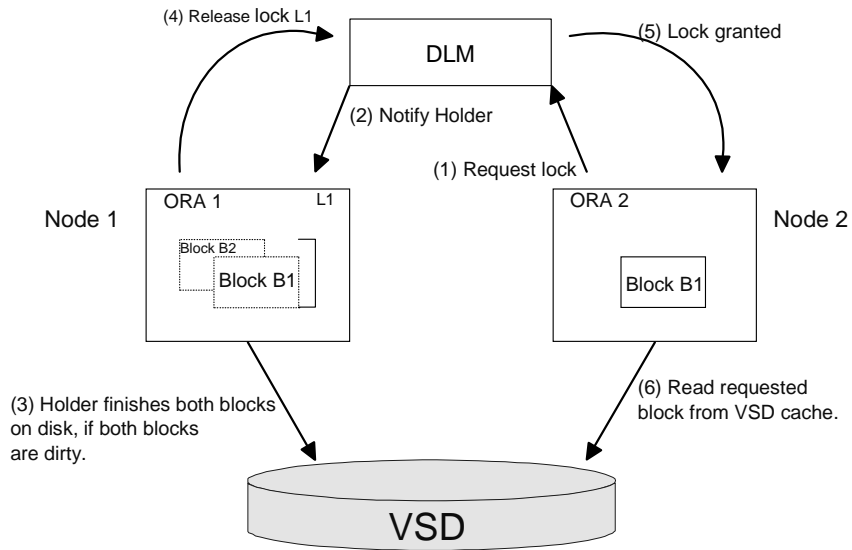


Figure 158. Events During a False "ping"

While all OPS systems have to deal with pinging, the RS/6000 SP and the VSD layer provide special enhancements to reduce its overhead. In particular, the VSD servers have an in-memory write-through cache. When a ping occurs, the write to the disk occurs, but the read back from disk to the second instance is eliminated. Instead, the data is read from the VSD cache which is several orders of magnitude faster.

Hashed locking and fine-grained locking (dba locking)

There are two different locking schemes in Oracle Parallel Server: hashed locking and fine-grained locking (also known as dba locking). When hash locking is used, there is a fixed assignment of PCM locks to data files. Locks are allocated statically at instance start-up time. When fine-grained locking is used, locks are dynamically allocated when needed, that is, at block-access time. The advantage of hash locking is that there is no overhead for lock allocation at the time when a data block is accessed. However, since all the locks have to be allocated at start-up time and each lock requires a small amount of memory, the total number of locks which can be allocated is limited by system resources. This usually means that multiple blocks have to be covered by the same lock. In other words, there is a low lock granularity. This might result in false pinging, as previously described. The start-up of the instance will also require more time, since all the lock resources have to be allocated at start-up time.

With fine-grained locking, locks are dynamically allocated at block-access time. The resources for the lock are only allocated during the time the lock is needed and are released when the lock is released. This makes it possible to achieve very high lock granularity, as far as covering each block with a separate lock. If resource minimization is the goal, fine-grained locks can also cover multiple blocks, but are still allocated dynamically.

Which type of lock to use depends heavily on the type of application. For example, if, in an OLTP application, a table is heavily accessed by multiple instances, dba locking eliminates false pinging and is probably the best solution. In contrast, in a pure query environment, few locks are needed since data is mainly accessed for reading. In this case, hash locking provides the better performance. Both locking types can be used within the same database.

Virtual Shared Disks (VSD) and Hashed Shared Disks (HSD)

As seen before, Oracle Parallel Server is a shared-disk architecture and requires that each node on the system is able to access every disk in the system. Traditionally this data sharing is accomplished by physically attaching each disk to each processing node. This is not a very scalable solution because wiring complexity becomes burdensome. The number of nodes is limited by the number of ports on the disks. This also means that only expensive multiport disk controllers can be used.

By leveraging its High Performance Switch, the RS/6000 SP is able to deliver a more elegant solution to the problem. In an RS/6000 SP system, each disk is attached to a single node (or to two nodes for high availability reasons). Access to the disks from other nodes is enabled by a layer known as the Virtual Shared Disk (VSD). Each node that needs to communicate with VSD controlled disks must also have a VSD layer on this node.

This VSD interface is a thin layer that sits above the Logical Volume Manager and provides a raw device interface to Oracle. Access to remote disks is transparent to Oracle. The local VSD layer communicates through a low-overhead protocol on the SP Switch to the VSD layer on the node which controls the disks. The bandwidth of the switch is much higher than the bandwidth of an individual disk, and the latency is several orders of magnitude smaller than the disk latency. Consequently, the switch is not a bottleneck in the data transfer, and the data transfer from a remote node is very close to the same rate at which data is accessed locally.

With standard VSD, there is no automated way of striping data across nodes. Striping with VSDs is limited to the disks attached to one node. There is no

striping at the VSD level, but within one node, the underlying logical volume can be used for striping. With IBM's Hashed Shared Disk (HSD), the capability to stripe across nodes has been added. This eliminates the manual effort required to spread large tables across multiple nodes and enhances disk usage (and therefore performance) by allowing simultaneous disk access.

When data is written to HSD, it is put on disks in stripes of configurable size. When HSD has written one stripe on one disk, it moves on the next disk for the next stripe, repeating the process until all disks have a stripe on them. It then starts over again at the first disk.

When Oracle reads from an HSD, the HSD can be accessing multiple stripe in parallel since they reside on different disks. HSD therefore increases the I/O throughput dramatically.

Parallel Query and Data Warehousing Optimizations

This section briefly lists parallel query features, data warehousing features and optimizations in Oracle 7.3. For a detailed description, refer to the Oracle 7 Server Release 7.3 documentation set.

- **Parallel Queries:** Oracle's parallel query option provides scalability for most common query operations. This dramatically improves response times for very large complex queries. Scans, sorts, joins, distinct and group can all be performed in parallel. The *asynchronous read-ahead* feature overlaps CPU and I/O time for each parallel query server, further increasing performance.
- **Parallel Data Load:** Parallel data load makes it possible to load huge amounts of data into an Oracle database in a very fast and efficient manner. For example, data located on different tapes can be read in parallel and then distributed over a large number of disks.
- **Parallel Create Index:** Oracle Parallel Server exploits the benefits of parallel hardware by distributing the workload generated by a large index create among a large number of processors. The time needed to create a large index can be diminished linearly with an increasing number of processors. This makes it possible to generate indexes for single applications and remove them afterwards. By specifying the "unrecoverable" option, the logging overhead for building indexes can be removed.
- **Parallel Create Table As Select:** This feature makes it possible to create aggregated tables in parallel. This is especially useful in big data warehousing environments, where smaller data marts are created out of

huge amounts of data coming from the data warehouse. This feature makes it possible to create aggregate tables for individual users in a timely manner. By specifying the “unrecoverable” option, the logging overhead during a create table as select can be removed.

- **Partitioned Views:** This feature makes it possible to partition huge tables that are often encountered in data warehousing environments and still access the data in a global way. Partitioned views improve the manageability and availability for large tables and they can improve performance dramatically on certain queries of these tables.
- **Star Queries:** A “star schema” is a natural representation for many data warehousing applications. A star schema contains one very large table (often referred as a fact table) and multiple smaller tables (commonly called dimension tables). The cost-based optimizer is able to generate very efficient execution plans for star queries and dramatically improve performance for this type of queries.
- **Parallel Cost Model for the Optimizer:** The parallel cost model allows the optimizer to avoid non-parallelizable execution plans. This feature exploits the parallel capabilities of the underlying hardware as much as possible.
- **Dedicated Temporary Tablespaces:** Tablespaces can be explicitly defined to be dedicated to temporary objects only. This allows for optimized space management. On a shared-nothing hardware architecture, this feature ensures that while processing a parallel query, sort operations which require temporary space will allocate this space on a local disk on each node. This minimizes I/O traffic over the interconnect and improves overall performance.

Shared-Nothing Optimization: Disk Affinity

The disk affinity feature is an important shared-nothing optimization for the Oracle Parallel Server. As discussed earlier, the Oracle Parallel Server requires all disks in the system to be accessible from all the nodes. When a query is processed in parallel, with disk affinity, Oracle knows which data are on the local disks of a given node. The nodes which are local to the requested data are primarily used for query processing to increase performance.

Parallel Query processing in Oracle 7.3 can be subdivided into three phases:

1. The first phase can be called a shared-everything phase. Using a divide-and-conquer approach, the query is divided up into subtasks which are sent to the nodes where the requested data resides.

2. The second phase is a pure shared-nothing phase. Each of the processing nodes does purely local processing to retrieve the requested data.
3. The third phase is a shared-disk phase. Nodes which have finished early with their processing can be borrowed to take over tasks from nodes which have not finished processing yet. The workload can be redistributed dynamically since all nodes have access to all the data. In this way, the processing power of all nodes can be used to process the query as effectively as possible.

8.2.2.1 System Administration

Oracle Parallel Server for the RS/6000 SP provides a number of advanced system administration tools. The goal in providing these tools is to make system management of an MPP system similar to that of any other system. In particular, system administration tasks can be performed from a single point of control. The following paragraphs describe the tools that make this possible. For more detailed information, see the Oracle Server for IBM RS/6000 Scalable POWERparallel systems - Tools and Utilities Guide.

The opsconf Tool

The opsconf tool generates and distributes the network-related files needed to run an Oracle parallel server database.

The opsctl Tool

The opsctl tool provides a single point for starting and stopping operations on Oracle 7 parallel servers.

The parallel server operations handled by the opsctl tool are categorized into stages. The opsctl tool defines dependencies between all stages of parallel server operations. It ensures that a stage with dependencies on other stages is started or stopped in the correct order.

The Oracle Trace Logging Utility

The trace logging utility for the Oracle Parallel Server, oratrc, generates one trace file for multiple Oracle instances running on different nodes. A single output log file helps users find messages quickly. They only need to look through one file instead of searching for many files produced by different processes on different nodes.

CLB - Load Balancing in an MPP environment

The connection load balancing listener (CLB) provides a load-balancing mechanism by managing connections to instances in the Oracle Parallel Server. The CLB listener lets a user connect to any instance of the Oracle Parallel Server on any node of a parallel machine, without having to specify the instance and the node.

The CLB process accepts connections from Oracle clients and assigns the connections to instances running on the parallel machine. The CLB listener runs on a node of the parallel machine.

For high availability, a backup CLB can be configured which takes over in case of a failure of the primary CLB.

Enterprise Parallel Backup

The Oracle parallel backup/restore utility provides the ability to backup and restore large amount of data in a much shorter time frame. Backups can be completed in a fraction of the time because the utility can use multiple fast peripheral devices and can scaleup backup speeds as the number of nodes increases.

8.2.2.2 Oracle and High Availability

Oracle on the RS/6000 SP offers advanced fault tolerance features to exploit the inherent redundancy of an MPP system in order to provide maximum reliability and availability.

In a fault-tolerant configuration, twin-tailed disks are used so that each disk is connected to two nodes. The twin-tailed connection is designed such that in the case of a node failure, the connection to the failed node is automatically deactivated and the connection to the backup node is activated.

At VSD level, there is a product called Recoverable VSD. Each VSD corresponds to one or more disks on a node and is mastered by the node the disks are attached to. If a node failure is detected, all the VSDs mastered by the failed node are automatically remastered by the backup node. In this case, the backup node is the node which has the second connection to the disks belonging to the VSDs. Therefore these disks can now both physically, and on a VSD level, be accessed through the backup node. The whole process is completely transparent to Oracle.

At a database level, the fault-tolerant lock manager plays the crucial role in providing fault tolerance. If a node fails, the instance on that node goes down, but all other instances in the system can still access the database. The lock manager is the component which detects the failure and alerts the other

instances. The instance which first detects the failure performs instance recovery for the failed instance. The instance performing the recovery rolls uncompleted transactions back and rolls completed transactions forward.

This brings the database back to a consistent state. Once the database is in consistent state, it can be accessed as before. Since the additional workload can be evenly distributed across all nodes, the performance loss in such a case is only $1/n$, where n is the number of nodes.

HACMP software mode 1 and mode 2 are also supported on the SP to provide features like IP address swapping or takeover. The HACMP mode 3 functionality (concurrent access) is completely replaced by the Oracle FT DLM and the recoverable VSDs on the SP. Therefore, the traditional HACMP mode 3 is not supported by Oracle Parallel Server on the SP.

Performance:

Typical measurements were performed in 1996 on a 46-node SP system using Oracle 7.3.

Overall conclusions:

Speedup: The SP/Oracle 7.3 database system has near-perfect (97%) speedup characteristics between 2 and 46 nodes.

Scaleup: Scaleup measurements also show near-perfect linearity.

Data Affinity: Oracle 7.3 has introduced *data affinity* which means that each node attempts to process only data which is local to itself, thereby avoiding internode data transfers. This is demonstrated by lack of switch traffic during queries.

Oracle 7.3 Performance: The combination of data affinity, the asynchronous-read-ahead feature and the direct read feature lets Oracle 7.3 scan tables approximately three times faster than Oracle 7.1.

Figure 159 on page 351 and Figure 160 on page 351 show speedup and scaleup results for a query on a 600 million row table. Figure 160 on page 351 shows how the execution time for the query decreases linearly with an increasing number of nodes.

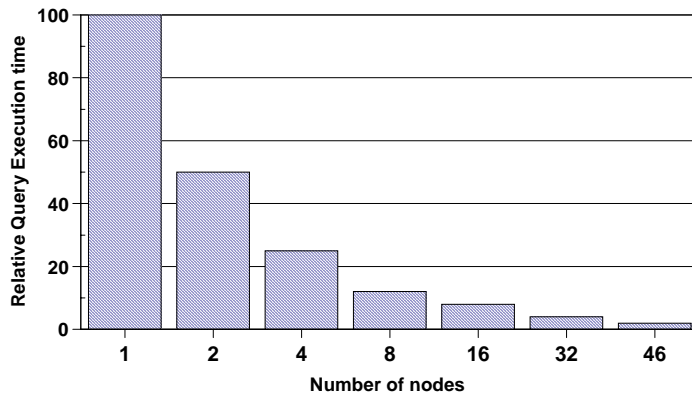


Figure 159. Query Execution Time

Figure 160 on page 351 shows the speedup for the same query. The speedup is defined as the execution time for a query run on N nodes, divided by the execution time of the same query run on 1 node and is a very good measure to show scalability. As stated before, the measured speedup proved to be nearly perfect (97%).

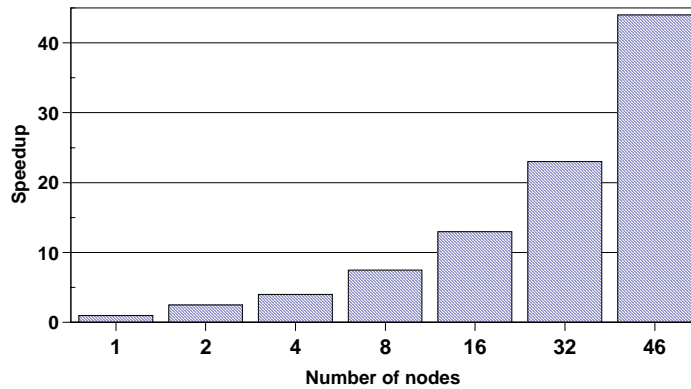


Figure 160. Speedup Numbers

8.2.3 Informix and Sybase

Informix On-line 8.0 eXtended Parallel Server (XPS) and Sybase MPP have adopted a “shared nothing” and “function shipping” architecture type that is

equivalent to the DB2 PE one. The main differences between these two databases are:

- Different data partitioning (hash-coding by table, by column or by circular replacement)
- Different capabilities to associate nodes to create sub-assemblies of functions (join, sort and so on). This concept is termed *pipelining*.
- Different approaches to parallelize SQL.

8.2.4 Informix

Informix-OnLine eXtended Parallel Server (XPS) for RS/6000 SP leverages the shared-nothing nature of the SP architecture to provide a highly available fault-tolerant database environment, including automatic switchover, dataskip, and log and database mirroring capabilities:

- This product is designed for data-intensive tasks associated with data warehousing, decision support, and other very large database (VLDB) applications. It is able to run multiple database operations simultaneously across multiple nodes in a clustered symmetric multiprocessing or massively parallel processing environment.
- It is able to break each database operation down into multiple tasks, which are spread across all available nodes for independent parallel execution.
- It provides a single system view, enabling database administrators to manage multiple database objects residing on multiple nodes from a single console.
- It is tightly integrated with Tivoli Management Environment (TME) to significantly ease database administration.
- It supports application transparency to facilitate migration of databases across a range of hardware environments without requiring recoding.

8.2.4.1 OnLine XPS Architectural Overview

OnLine XPS is the third in an ongoing series of compatible database products based on Informix's Dynamic Scalable Architecture (DSA). DSA provides parallel processing capabilities at the core of Informix's relational database management system (RDBMS) kernel, thereby exploiting a range of multiprocessor environments. OnLine XPS extends DSA to loosely coupled or shared-nothing computing architectures, including clusters of symmetric multiprocessor (SMP) and massively parallel processor (MPP) systems, such as SP systems.

OnLine XPS is particularly well-suited for data-intensive tasks associated with data warehousing, decision support, and other very large database (VLDB) applications. Its unique parallel processing and management capabilities provide an order-of-magnitude improvement in performance by spreading out database operations across all available hardware resources.

OnLine XPS provides a set of features directly associated with improving VLDB application performance, availability, and manageability, including enhanced parallel SQL operations, high-availability capabilities, and a suite of systems management tools. It also provides support for relevant open systems standards. The result is a next generation parallel database architecture that delivers outstanding scalability, manageability, and performance, minimal operation system overhead, and automatic distribution of the workload.

8.2.5 Sybase

8.2.5.1 Sybase MPP for the IBM RS/6000 SP

The SP, coupled with Sybase MPP, is an open systems platform that allows businesses to invest in a hardware and software solution that will scale with their business requirements. The shared-nothing architecture, common to both Sybase MPP and the SP, delivers near-linear speedup and scaleup of users, transactions, and data. This linear speedup and scaleup protects customers' investments in hardware, software, and skills. The SP's High Speed Switch, which provides high-speed communication between the nodes in an SP system, scales as you add nodes, with data transfer rates unmatched in the industry.

Sybase MPP is an extension of the Sybase SQL Server technology designed and optimized for massively parallel processing environments. The open parallel architecture of Sybase MPP furnishes the scalability, flexibility, and manageability to support very large database (VLDB) operations, such as maintaining and querying a centralized data warehouse. Using powerful and proven SQL Server technology, Sybase MPP implements a high-performance parallel database system.

The key to the performance and scalability of Sybase MPP is its data partitioning. Partitioning divides the data up physically so that it can be accessed and managed separately, minimizing contention. A big table becomes a series of smaller tables, each accessed independently with its own processing resources. Because each partition is explicitly defined by table and key value, partitioning can be maintained by the system, at data load and as the data changes, relieving administrators of time-consuming I/O tuning.

Each Sybase SQL Server actually controls its own data partition and can work independently on its assigned piece of the request. There are no costly delays while waiting for resources controlled by other systems. Each table is divided into partitions by the key values of its rows. The parallel optimizer uses system-wide schema information to ensure that each request is only sent to partitions with data involved. Partitions are completely transparent to the application and can be changed as needed for optimum performance.

Sybase MPP supplies three different partitioning methods, which are suitable for a variety of tables and data access types. Each table in the system can have its own unique partitioning scheme. Partitioning analysis is automated through the use of the configurator, which recommends partitioning keys and partitioning methods for each table to achieve optimal system performance.

Hash partitioning assigns each record to a physical partition according to a calculation based on the partitioning key. It is designed to spread data evenly across all partitions. Hash partitioning minimizes data or processing imbalance, while providing intelligent access to any row.

Range partitioning allows related rows of information to be stored together in the same partition. Range partitioning provides additional performance where data access is tied to ranges of the partitioning key.

Range partitioning may require more planning to distribute data evenly across partitions. Schema partitioning assigns all rows of a table to a single partition. This can provide more efficient use of space and data access for small tables that may not benefit from partitioning. All three partitioning methods can operate in concert with table indexes, combining the benefits of parallelism and indexing for even faster results.

Processing a request clients log into Sybase MPP as if it were a single Sybase SQL Server RDBMS. An individual request for information (SQL query) is analyzed by a full-function parallel optimizer, using comprehensive location and partitioning information. Parallel SQL is generated and sent to each processing partition involved and executed concurrently, reducing overall access time. Working in parallel, the independent processing partitions retrieve results only from their own portion of the data. Sybase MPP merges the results from all partitions and returns the results to the requester.

The shared-nothing architectures of Sybase MPP and the IBMSP provide nearly linear performance improvements and scalability as you add capacity to your system. You can take full advantage of each new processor and disk without being concerned about operating system or I/O system bottlenecks.

Only questions and answers get passed through the high-speed interconnect, minimizing interference.

8.3 Server Consolidation

In technical and commercial environments, the proliferation of servers increases the support cost and the number of licenses.

The way to reduce these costs is to consolidate all the servers in one system. In this way you will:

- Reduce your system management
- Have single system maintenance
- Control licenses
- Have a single point for backup and archiving

8.3.1 The Costs of Distributed Computing

For the last decade, the emphasis has been on distributed computing. "Escape the glass house" was the battle cry. Free the common man to compute for himself. Most people believe that PC/LAN computing is cheaper than mainframes.

Now reality is setting in. In fact, as demonstrated in the following studies, the distributed world is much more expensive than expected. This is not a failure of PC technology; it is the nature of a technology in the early stages of growth. Very few people advocate taking away the advantages of personal computing, but it is time to address efficiency along with this function, including the out-of-pocket cost of support and the loss of end-user productivity caused by ineffective service.

Briefly, the SP is a solution because:

- It scales well beyond customer needs.
- It has excellent reliability and availability characteristics.
- It allows for higher utilization and, therefore, a better return on investment.
- It is flexible enough to support many services at once.

8.3.2 Cost of Computing Study

Early in 1994, the International Technology Group produced a report entitled "Cost of Computing, comparative Study of Mainframe and PC/LAN Installations". The group surveyed more than 250 companies and studied 24

installations in detail. They found that the mainframe average annual support cost per user was \$2,282. The comparable PC/LAN cost was \$6,445 which is 2.8 times that of mainframes. Further analysis indicates 58% (\$3,738) of the PC/LAN costs are for labor to provide ongoing service maintenance. Another 40% of that cost (\$1,540) was foreshore Management', compared to \$230 for support of a mainframe user. Thus, the ongoing resource management of PC/LAN systems is 6.7 times that of mainframes. The SP provides a means to reduce these costs.

8.3.3 Total Cost of Ownership

Despite the challenges of managing the costs of distributed computing, the overall productivity potential is enormous. Businesses need to realign the functions of computing and put the right work in the right hands. Peer-to-peer support cost is high because data processing people cannot be everywhere at once. One powerful alternative is to put the technology in one place along with the support personnel. The SP provides a platform where one can create a reliable central service in a distributed open environment. Given this reliability, almost all end users are happy to relinquish their peer support tasks and get on with their work.

8.3.4 Scalability

Single workstations or PC servers are very powerful, but they soon reach a limit. One server can comfortably support one department, but departments do not work alone. The trend is to integrate with the enterprise to share information and applications. The SP provides a platform that can support any level of enterprise server. Frequently, we hear the number of 50 to 75 users per server. An SP starts at four nodes which, depending on the application, could easily support an organization of 800 end users, as could four workstations. However, the SP scales to 512 nodes. It could support, at the high end, 102,400 end users. Very few companies would need a server of that size, but by investing in an SP, companies have plenty of room to grow.

8.3.5 Reliability and Availability

Failures are costly in support personnel time and the loss of user time. A central system that supports thousands of end users must be capable of minimizing outages and, in the case of an outage, recover quickly. These characteristics are a major advantage of the SP.

Distributed servers usually lack redundant hardware. Having a spare for every distributed server is expensive and requires someone to visit the server to activate the spare. A platform like SP makes it affordable to have redundancy built in. The power supplies are one example. The frame comes

with four, but needs only three to run. If one fails, the processors continue to run while someone replaces the hot pluggable defective power supply. For example, if one owned an SP with ten nodes, it would be reasonable to have one extra processor as a spare. If one node fails, the spare can take its place. One final option for the SP is to use HACMP technology.

8.3.6 Utilization

One problem with distributed servers is mismatch of capacity with demand. Different groups have different computing needs, and their needs change independently and frequently at unpredictable times. If all servers are sized for the largest possible demand, they will sit idle most of the time. As time passes, some servers will develop bottlenecks and provide poor service. Avoiding this situation requires the constant monitoring of capacity versus demand.

SP provides an effective solution because servers in a SP can share resources. Work can be allocated dynamically across several servers. New servers can be added to the pool of servers as needed so that service remains consistent and capacity grows smoothly.

Most end-user servers will sit idle at night and on weekends. With shared resources available to the SP, it becomes possible to mix LAN services with other workloads. The enterprise may need batch service and online applications. Users queue batch jobs during the day, then the jobs are released at night when the end-user work slows down. This sharing of capacity translates into a reduction in total cost.

8.3.7 Flexibility

End users expect and need good, reliable and dependable service. Good service takes good planning, but nobody is perfect at planning. People's needs change over time. Changing a distributed system is difficult because capacity, software and data are disbursed. Moving the pieces of a distributed system around is disruptive and error-prone. Distributed systems tend to be inflexible.

SP is well positioned as a flexible server. Since SP functions like one platform under a single point of control and it scales to very high levels of capacity, it is feasible to merge several services. For example, the file service might have four nodes, the data base six, and the batch service has six, for a sixteen-node SP. If, at year-end, the data base service needs more capacity to perform year end planning to reduce the company inventories, then one or two of the batch servers can be easily transferred to the data base pool.

When the peak is over, the two extra data base nodes go back as batch job servers. Of course, if the load changes permanently, the new configuration can be left intact. This is illustrated with the scheme in Figure 161 on page 358.

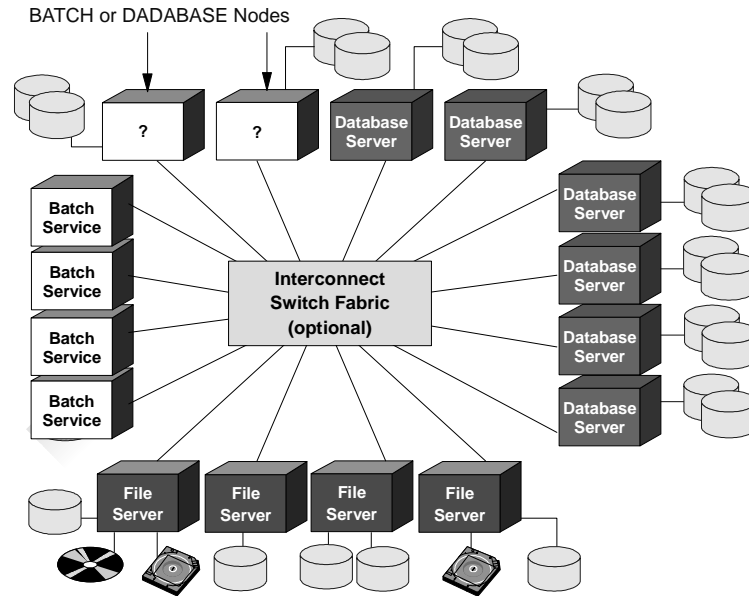


Figure 161. SP As a Flexible Server

8.3.8 A Server Consolitation Example

In New York City, there is a 500 person firm that uses a distributed e-mail system. They now have three people on their staff just to keep the e-mail servers functioning. The administrators are frequently in at night or on the weekends. If we assume that this company spends \$75,000 per year, fully burdened, for each administrator, then their labor cost alone is \$225,000.

An SP of an approximate cost of \$250,000 could provide the capacity for this e-mail system. The SP has superior availability, operations, and support characteristics. Pulling the servers into one place and onto one footprint would reduce the administrative costs to half of one administrator. There are several reasons of this:

- Centralized rather than distributed computing eliminates running around to separate servers.

- One point of control means all servers can be monitored at once.
- Remote control means the administrator can be updating the servers from his or her office while doing other tasks.
- Remote access also means that the administrator can work from home on weekends or in an emergency.
- Redundancy means that there are fewer outages to deal with.
- Systems management tools make software maintenance more efficient.
- The capacity of the SP nodes means there are fewer servers to deal with and fewer administrative chores to balance the work.

The bottom line is that, in this case, administration of the SP costs \$37,500 instead of \$225,000. The break-even point for using a SP is less than two years, and this does not even take into account the improved end-user productivity. A validation of these savings can be found at the IBM Watson Research Center where one system administrator supports a 128-processor SP.

Chapter 9. Scientific and Technical

This chapter describes the benefits of using an SP to solve problems in a scientific/technical setting. We present different applications that exploit the structure of the SP in order to model processes in science and engineering disciplines.

The computational power of the SP allows researchers to attack problems that would otherwise be impossible to solve. Intel is using an SP to design its next generation microprocessors. Deep Blue can face the most formidable chess players in the world.

The flexible and scalable structure of the SP can be used in various ways in science and technology. One may view the whole system as an SIMD machine and ask all nodes to execute the same piece of code on different data, or MIMD machine, where each node can execute different code.

Several scientific computer implementations fall into this category. Pharmaceutical research, hydrogeology, turbulence modeling, electromagnetics, astronomy, and meteorology are some of the disciplines that greatly benefit from the increased power of the SP.

The RS/6000 SP has several strong points that can be exploited in scientific/technical applications:

- Flexibility. A customer with multiple frames or systems can reconfigure them and treat the machines as a pool of nodes that can be installed on several systems as the need arises.
- Scalability. The SP allows inherently parallel applications to benefit from the scalability potential of the hardware. In other words, the realizable speedup of serial code is typically linearly proportional to the number of nodes in the machine.
- Performance. The P2SC processor has the industry-leading balance of floating-point performance and memory bandwidth, plus large memory capacity per node compared to competitors
- Reliability. Compared to SMP and other NUMA-based machines, the RS/6000 SP is much more reliable, a concept that in IBM terms is often called High Availability. Unlike most other MPP machines, the SP stays up and operational even when a node fails, whereas other MPP systems would become unusable.

There are several ways one can categorize the different applications that run on the SP. The scheme we use here is by no means the best or most

comprehensive, but it serves the purpose of presenting them in an organized fashion.

Research. These are applications in science and engineering that are developed primarily in higher education institutions and government agencies.

Process. These are applications that have to do with designing manufacturing plant processes, manipulating large amounts of data, creating models of drugs and other chemicals, and expediting and streamlining the testing phases of drug development.

Petroleum exploration and processing. These applications are geared towards analyzing vast amounts of data in order to predict the most favorable locations for petroleum recovery, and forecast the revenue produced from such recovery processes.

Engineering. These applications focus on solutions that are specifically targeted to various sectors of the engineering community. Examples are the electronic design and analysis of circuits, computer-aided design, and CATIA systems.

We discuss these application in more detail in the following sections.

9.1 Research Applications

The research sector is not a traditional industry. The term is indicative of any agency that performs research as a major part of each operation.

The RS/6000 SP is an ideal platform for research. Several academic institutions and government laboratories are using SPs to solve Grand Challenge problems. Over 36% of all RS/6000 SP nodes are installed in scientific research settings in higher education and government systems. Many high profile researchers and scientists are using SPs to solve complex and time-consuming problems.

There are over 300 SPs in universities worldwide. The main reason for the SP's success in this industry is its stability, scalability, and performance in computation- and data-intensive applications.

The disciplines where SPs are utilized in a scientific context include: computational fluid dynamics, atmospheric and ocean modeling, reservoir simulation, seismic modeling, structural analysis, electromagnetic analysis, biotechnology, astronomy, and mathematical sciences.

9.1.1 RS/6000 SP Applications to Research

The range of applicability of an MPP computer in research seems unending. The main reason is that whenever scientists and engineers try to model complex phenomena, they cannot solve them analytically. This makes the use of computers inevitable, and the more powerful the computer, the larger and more demanding the programs can be. Although not all applications need to be large and demanding to produce meaningful answers, there is a growing list of problems whose answers elude us only because our computers are not fast enough, or cannot store enough data.

In this section we outline a few problems that interest researchers in various fields, and show why the use of powerful computers (especially massively parallel ones) could help them solve their problems. Although the physical, chemical or biological processes that the scientists and engineers model are distinct and share almost nothing, they lead to computer implementations that share similar problems, and face similar computational obstacles.

Consider the problem of weather forecasting at a global level. To accomplish this, scientists need to model ocean current movement and atmospheric air movement, and the interaction of the two. This problem is far from solved, and to get meaningful predictions one needs to go to fine levels of detail. Consequently, storage and computational requirements grow dramatically, and the only hope of solving the problem is by using a supercomputer. The appeal of a distributed memory, massively parallel system like the RS/6000 SP is that some nodes can perform the ocean modeling, other nodes can perform the atmospheric modeling, and still another set of nodes can combine the results to produce the final answer. This type of computing is very effective and scalable in that it efficiently utilizes not only the current resources, but will also be valid ten years from now when a faster and larger system is available.

9.1.2 Research Solutions Examples

Following are several examples that demonstrate the potential and strength of the SP in solving research problems:

- In 1996, Lawrence Livermore National Laboratory (LLNL) awarded IBM a \$93 million contract to build the world's fastest supercomputer. In its final form, the system will consist of several hundred SMP nodes for a peak performance of 3.2 teraflops, aggregate total memory of 2.5 terabytes, and disk space of 75 terabytes. The machine will allow a flexible partitioning of the system that will be mutually exclusive. The system is primarily be used for studying complex fluid dynamics problems. Figure 162 on page 365 shows a three-dimensional turbulence calculation done

at LLNL. This simulation required 130 million grid points and 10^{15} (quadrillion) operations to complete. On the then-current installation of ASCI Blue, this task used 10,000 node-hours. Figure 162 on page 365 shows the results of a turbulence calculation performed at LLNL.

- National Institutes of Health and the National Cancer Institute are using a 56-node RS/6000 SP to perform bioscience research, computational chemistry and chemotherapy drug design.
- The primary goal of the world-class Maui High Performance Computing Center (MHPCC) is to provide Research and Development supercomputing for U.S. Air Force and Department of Defense labs and their contractors.

Other goals are: fostering technology exchange programs in high-performance computing among the governmental, academic and industrial communities; stimulating economic development contributing to Research and Development in software for scalable parallel processors supporting collaboration to address Grand Challenge problems.

To address these research requirements, the MHPCC installed as its central computing complex one of the most powerful supercomputers in the world--an IBM 400-processor RS/6000 SP. Scientists and engineers across the United States can harness its raw power, with a peak capability of over 100 billion floating point operations (flops) per second, through high-speed networks, primarily the widely used Internet.

Recently MHPCC announced that it will increase its computational capabilities by 50% with the purchase of 192 nodes of IBM's latest RS/6000 SP technology. This enhancement will provide MHPCC's base of 1,500 users with a more effective environment for migrating projects from research through development to production.

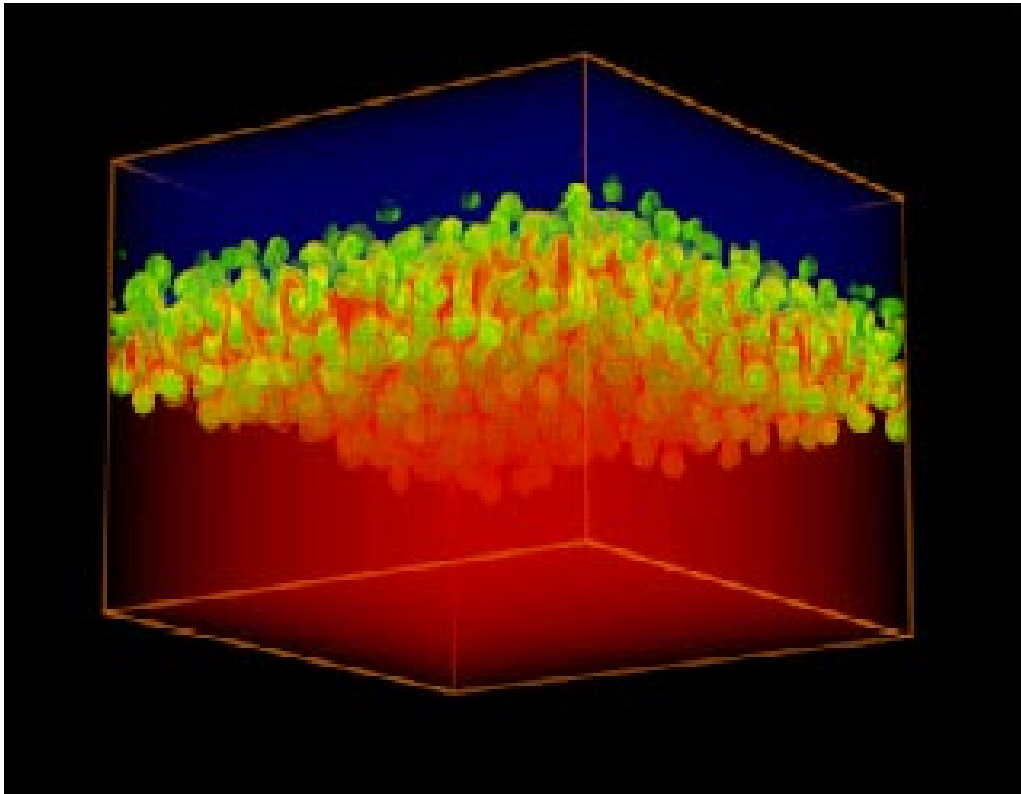


Figure 162. Three-Dimensional Turbulence Calculation Done at LLNL

- The Cornell Theory Center (CTC) is a high-performance computing center funded by the National Science Foundation. Cornell was the first U.S. installation of an SP and for a while had the largest SP installation in the world--512 nodes. CTC specializes in biomedical, CFD, environmental, molecular and astrophysical modeling.
- The U.S. Department of Energy's Argonne National Laboratory has one of the largest SP installations with 128 nodes. They have also acquired three smaller systems, including a 12-node SP for advanced multimedia research, including virtual reality.
- More than 50 percent of the world's high-energy physicists carry out their research at CERN, the European Laboratory for Particle Physics in Geneva, Switzerland. Funded by 19 European member states, it is the world's largest scientific research center. More than 8,000 scientists, engineers and computer specialists from 75 nations collaborate to do

research into the fundamental building blocks of matter and the forces that hold them together. The laboratory has installed a 64-node RISC-based RS/6000 SP for access by the scientists at CERN, and researchers worldwide via the Internet.

IBM has provided several development tools to create applications that take advantage of the SP architecture. A brief discussion of these tools is given in 4.11, "Parallel Environment (PE)" on page 204. Furthermore, Independent Software Vendors (ISV) provide several programming tools specifically tailored for the SP. These tools are mostly highly efficient compiler implementations, as well as code optimizers. They are designed to analyze source code and restructure it so that the full potential of the SP can be realized. A partial list of available tools is given in figure 163 on page 366.

APPLICATION	VENDOR	AVAILABILITY
FORGExplorer	Visual Numerics	Available
TotalView	Dolphin	Available
Trapper	Genias	Available
KAP/Pro	Kuck & Associates	Available
PAMS	Myrias	Available
VAST-parallel	Pacific Sierra Research	Available
Data Parallel C	Pacific Sierra Research	Available
VAMPIR	Pallas	Available
Treadmarks VSM	Parallel Tools/Rice University	Available
Bert77	Paralogic Inc	Available
Linda	Scientific Computing Associates	Available
Paradise	Scientific Computing Associates	Available
Piranha	Scientific Computing Associates	Available

Figure 163. Parallel Programming Tools Offered for the RS/6000 SP

9.2 Process Applications

Applications in this category help scientists and engineers to experiment, model and design new methods and processes. This research is aimed at creating efficient and cost-effective alternatives to current processes currently used for production, for example in the pharmaceutical industry.

Because of the high level of invested capital involved in these research projects, it is important to have a streamlined, effective, fast and reliable method of testing a particular process. Several pharmaceutical companies spend millions of dollars in perfecting their testing methods, reducing their drug discovery costs, and improving their time-to-market.

9.2.1 RS/6000 SP Implementations to Process Applications

The RS/6000 SP presents a very attractive solution for computational chemistry, drug selection, process control, evaluation and trial administration markets. Most of the customers in this category are pharmaceutical, chemical, or petroleum companies.

The sheer speed of the SP makes it very appealing in an industry where speed is important, and algorithms can easily be parallelized. Several ISVs have either ported their codes to the SP or developed new applications to explicitly use the SP architecture. Some of the most popular applications are listed in Figure 164.

At the same time, the same HA considerations that make the SP attractive to other industries apply here as well. When a company is basing their product cycle around the performance of a computer, they cannot afford to lose time due to single points of failure.

9.2.2 Process Solutions Examples

Successful implementations of the SP include:

- A 16-node RS/6000 SP with which Sumitomo Chemical which is a major chemical company was able to achieve processing capabilities seven times greater than, and at one-third the cost of previous solutions. The areas where the SP system was used were computational chemistry, chemical engineering, process design, and process plant production management.
- Keiper Recaro, a leading international producer of seating, seat structures and components for the motor vehicle and airline industries performs primary research at the company's Technical Center in Kaiserslautern which focuses on exploring new horizons in ergonomics, weight reduction, protection of natural resources and recycling. Relying on state-of-the-art equipment in CAD, engineering, prototype development and testing, the company recently installed a 4-node RS/6000 SP system.
- Research at the Environmental Molecular Sciences Laboratory (EMSL) at Pacific Northwest Laboratories (PNL) is focused principally on developing a molecular-level understanding of the physical, chemical and biological processes that underlie the most critical environmental issues facing the Department of Energy: contaminant fate and transport, waste processing, cellular response to environmental contaminants, and atmospheric chemistry. To provide the advanced computing capability needed by EMSL, PNL maintains and operates the Molecular Science Computing Facility (MSCF) in the EMSL. The MSCF contains an RS/6000 SP with

512 processors and a peak theoretical performance of 247 gigaflops, 67 gigabytes of memory and 2.9 terabytes of disk storage.

A next generation SP system based on SMP nodes is expected to be installed at the Experimental Computing Laboratory at the time of this writing. The new system will help EMSL extend the range and reduce the cost of molecular or reactive chemical transport simulations.

APPLICATION	VENDOR	AVAILABILITY
BATCHMIN	Columbia University	Available
GAUSSIAN94	Gaussian, Inc.	Available
GAMESS	Iowa State University	Available
MULLIKEN	IBM Almaden	Available
CHARMm	Molecular Simulation, Inc.	Available
DISCOVER	Molecular Simulation, Inc.	Available
DMOL	Molecular Simulation, Inc.	Available
XPLOR	Molecular Simulation, Inc.	Available
AMBER	Oxford Molecular Group	Available
AMPAC	QCPE	Available
BIGSTRN-3	QCPE	Available
HONDO	QCPE	Available
MOPAC	QCPE	Available
SIMBIG	QCPE	Available
SPARTAN	Wavefunction, Inc.	Available
WESDYN	Wesleyan University	Available

Figure 164. Biological/Chemistry/Process Parallel SP Applications

9.3 Petroleum Research Applications

The petroleum industry's main focus is to find new oil deposits. In some cases these reservoirs are several miles in the subsurface, or even worse, undersea. Locating and extracting oil from those reserves becomes a very expensive task. The cost of an error in the drilling location can cost millions of dollars. To pinpoint the location of oil reserves, one needs to map the subsurface using elaborate, time consuming, and very computation-intensive techniques. Nevertheless, the computing task is highly parallelizable, and this is where the power of the SP shines.

9.3.1 Seismic Processing Applications

The Seismic processing sector is utilizing the largest number of SPs in this market. The sheer amount of computation required to process the terabytes of data (thousands of 3590 tape cartridges may be acquired in a single survey) produced in the field, necessitates the use of high performance computers.

Typically, a company that works in exploration for hydrocarbon deposits would use seismic waves to map out the region of interest. The technique is in principal analogous to the ultrasound method used to view the fetus in a woman's womb. The sound waves produced by the explosion travel with different speeds and reflect differently in different types of rocks. By recording the intensity and speed of those reflected waves at different listening stations, scientists are able to reconstruct the shape and structure of the subsurface region that created such reflections.

A very important property of the algorithms used for the surface reconstruction is that they lend themselves to data parallelism. This means that data gathered from one listening post can be analyzed seperately from other posts' data. This makes the use of a massively parallel computer a prime candidate for performing the computational task. Moreover, there are other reasons that make the RS/6000 SP an excellent choice for petroleum exploration:

- Although parallelizing the task reduces the "wall clock" time required to solve the problem, it can take several days or weeks to complete a single job. During this time it is essential that the system does not fail either in hardware or software. The high availability solution of the SP addresses this issue.
- The machine is scalable, so that a customer with several frames (and nodes) can create custom configurations to fit particular needs. The status of the survey requirements at a given location is reevaluated after completion of the task, and "spare" nodes are shipped to different locations.
- The physical dimensions of an SP frame permit easy loading and installation in small quarters, making the RS/6000 SP an ideal choice for field work on board ships. This ensures high-quality survey data prior to processing at the on shore central computing facility, and eliminates costly re-shoots of the survey. In addition, full 3D processing can now be accomplished on board, which results in faster delivery of results to the client. With this combination of higher quality and faster, lower-cost results, the RS/6000 SP is the leader in seismic processing today.

9.3.2 Reservoir Modeling Applications

Reservoir simulation logically follows the seismic data acquisition and processing phase. When planning the oil extraction phase, petroleum engineers need to take into account the difficulty in pumping the oil into the surface. There are three categories of oil extraction:

- Primary extraction is used when the oil pressure in the subsurface is so great, the oil reaches the surface on its own.

- Secondary extraction is used when the oil must be pumped from the subsurface.
- Tertiary extraction involves using chemical or steam injection to “wash” oil from the porous rocks in the subsurface.

Because the cost of oil extraction increases as we move down the list, computer models can be used to approximate the time and effort needed for the task. The same models can be used for modeling flow and transport phenomena for other hazardous chemicals (BTEX, TCE, PCE, PCBs) and radioactive materials in the environmental engineering industry. Regardless of the particular application, all subsurface flow models are very complex and time-consuming, and impose large memory requirements.

Nevertheless, the industry trends towards increasing the complexity and detail of reservoir models have resulted in an order-of-magnitude increase in the need for memory and computational requirements for reservoir simulation. Only several tens of thousands of grid blocks could be used in the model due to limitations of current computer CPU speed and performance, while the real need is for several millions of grid blocks to obtain the necessary simulation detail. However, reservoir simulation codes are so complex that it has taken several years of re-engineering the applications to allow for scalable parallel processing.

Recently, these new scalable, parallel versions of the code are emerging as commercially available applications. Now re-structured for distributed domain processing, these applications make feasible the overnight simulation of fully-characterized reservoir models, using millions of grid blocks, which provides detailed views of the reservoir that have never been obtained before. Figure 165 on page 371 shows an Upscaled model (a coarse, averaged model with fewer than necessary cells) with 36,000 grid cells, where the Full model has 1,000,000 grid cells. It is obvious even to the casual observer that the full model offers a crisper, more information-rich picture than the averaged one of the Upscaled model.

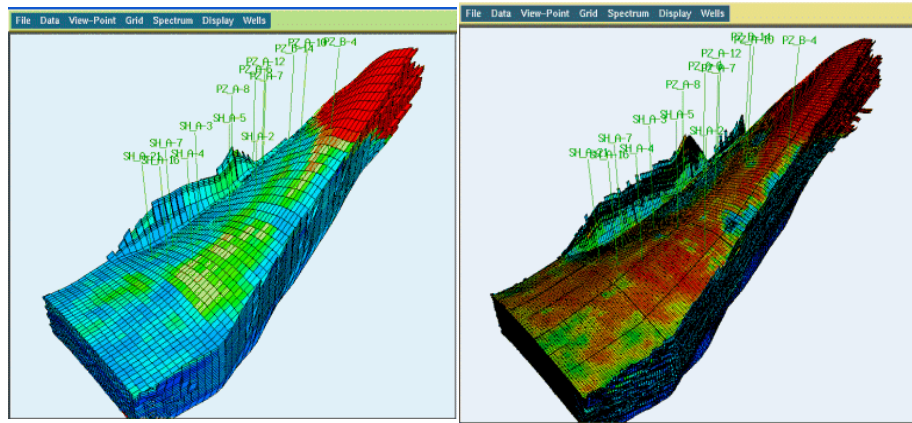


Figure 165. Upscaled vs Full Model Using Landmark's Parallel-VIP

The RS/6000 POWER2 and P2SC processors provide the balanced combination of floating-point performance with industry-leading memory bandwidth, which makes the RS/6000 the processor of choice for reservoir simulation. The scalable design of the RS/6000 SP, using P2SC processor nodes, combined with these new production parallel reservoir simulators, provides the petroleum industry with the ideal solution for simulation into the 21st century. Indeed, judging from its performance and market share, the SP is the recognized leader in this field.

9.3.3 SP Implementations to Reservoir Modeling Applications

There are currently over 250 SP systems with 2900+ nodes deployed to clients in the petroleum industry. Of those companies, 51% are contractors, 34% are international oil firms, and 15% are national oil firms. Almost half (48.5%) of the SP installations are used for data processing, while one quarter (27.9%) is used for data acquisition. The remainder percentage is used for reservoir simulation, geostatistical, environmental, hydrogeological and SAP applications. Figure 166 on page 372 lists some major petroleum exploration and production applications ported to the SP platform.

APPLICATION	VENDOR	AVAILABILITY
Parallel Geo Vecteur Plus	CGG	Available
FOCUS 2D/3D	CogniSeis	Available
SEISMOS	GECO-PRAKLA	Available / service contract
SeisUp	GeoCenter	Available
MIGPACK (SIRIUS)	GX Technology	Available
PROMAX 3D	Landmark Graphics	Available
PGS Seismic Processing Tools (Cube Mgr)	PGS Exploration	Available / service contract (AIX V4.2.0)
OMEGA	Western Geophysical	Available
Parallel-VIP	Landmark Graphics	Available
ECLIPSE Family	GeoQuest	Planned for 1998
MPBLITZ/CBLITZ	Landmark Graphics	Available
More Family	Smedvig Technologies	To be determined
SSI Simulators	Scientific SW - Intercomp	To be determined

Figure 166. Petroleum Exploration and Production Parallel SP Applications

9.3.4 Seismic Research Examples

Some firms that are using SP systems in seismic exploration include:

- Ensign Geophysics, which specializes in supplying the full range of marine seismic data processing services to the oil and gas exploration industry.

In 1991, Ensign installed the world's first parallel production processing system. During 1994, given the seismic industry's greater awareness of the benefits of parallel computing and the consequent wider hardware availability, Ensign adopted IBM parallel technology, in order to offer cost-effective pre-stack 3D imaging solutions.

By 1996, the company was operating the largest single IBM SP system in use in the petroleum industry. Ensign recently set a record by processing a survey of 115,500 CMP kilometers through a demanding sequence, which included pre-stack time migration, in 13 weeks from completion of the acquisition phase.

The computer system for this type of application needs to be powerful, scalable, flexible and very cost-effective, in order to handle both the large volumes and the computation-intensive applications, such as 3D pre-stack depth migration. The 48-node IBM SP provides the bulk of the power requirements at the UK head office, while an upgraded 16-node system is operated at the US branch office.

- Petroleum Geo-Services (PGS) is using RS/6000 SPs to do marine seismic data acquisition, onboard seismic data processing, and data quality control.

PGS has over 500 nodes in the field and is able to match the computing power to the size of the survey by simply increasing or decreasing the number of nodes onboard their boats. This practice enhances flexibility so they can respond in a cost-effective manner to the demands of their oil company clients.

A list of clients that use SPs for reservoir simulation includes BP, Chevron, Mobil, PDVSA (Corpoven, Lagoven, INTEVEP), Phillips, Saudi Aramco, Statoil, and Texaco. The companies that develop the software packages often offer consulting services, effectively becoming users themselves.

9.4 Engineering Applications

The engineering solutions that the RS/6000 SP offers include the structural design and analysis of complex structures, electronic design, analysis and packaging of computer circuit components.

9.4.1 Mechanical Engineering

In the mechanical engineering arena, there are several areas that require high performance computing power, as well as large storage capabilities. Examples are the design of automobiles and airplanes, analysis of the flow characteristics around the fuselage and the wings, and automobile crash modeling.

Several large automobile and airplane manufacturers have chosen the RS/6000 SP as their preferred platform to conduct these sophisticated and complex design and modeling computations. At the same time, several ISV partners of IBM have developed applications that take advantage of the RS/6000 SP's unique architecture and performance characteristics. IBM, along with the US DOD Advanced Research Projects Agency (ARPA), is sponsoring application development initiatives that promote the use of the SP. The list of ISV partners includes: MacNeil Schwendler, Fluent, Inc., Molecular Simulations Inc., and Centric Engineering. Early versions of the developing applications are expected in 1998.

A list of available software packages is given in figure 168 on page 376.

9.4.1.1 CATIA

An application developed by Dassault Systemes called Computer Aided Three-dimensional Interactive Application (CATIA) plays a prominent role in engineering design and analysis. CATIA is comprised of a series of programs that address the CAD-CAM needs of the engineering industry. The packages that make up the latest version of CATIA are:

Mechanical Design. Designers can select from 2D and 3D wireframe, exact and mock-up solids, feature-based part design, 3D parametric/variational modeling, drafting and annotation, and more. Specialized applications assist with assembly management, sheetmetal work and dimensioning and tolerancing.

Shape Design and Analysis. These offer tools to design parts where innovative forms or complex shapes are key factors. Included are functions to easily create, import, modify, analyze, and manage surfaces. Tools help stylists and designers working on shapes requiring aesthetic, aerodynamic, or other design constraints. Specialized programs facilitate working with clouds of points, engraving/embossing and mapping patterns onto curved surfaces.

Analysis and Simulation. CATIA provides analysis solutions for the designer and for the analysis specialist. Design engineers can analyze solids and surfaces transparently with automatic meshing, solving, and results-visualization tools. Solutions are available which fully automate design optimization to achieve stress targets. Specialists can use CATIA's proven tools for static and dynamic structural, stress, vibration and other types of analyses.

Manufacturing. CATIA provides a full complement of tools for 3- and 5-axis milling. Automated solutions are provided for the NC programmer, as well as specialized applications for the NC specialist. These applications help companies share knowledge on stock, tools, techniques and other manufacturing parameters. Tools are also provided for nesting, defining and programming robots, and for stereolithographic rapid prototyping.

Equipment and Systems Engineering. These packages facilitate design, modification and analysis of electrical and fluid systems. Electrical systems design is handled from wire and cable content definition through manufacturing. These solutions include piping and tubing, schematics, electrical equipment, harness routing and cable formboard design. Space reservation, fitting simulation and interference and clearance analysis, provide for comprehensive space management.

Architecture, Engineering and Construction (AEC) Plant Design (CCPlant). CCPlant applications exploit engineering and design data across the business and throughout a plant's life cycle. Owner/operators can bring products to market faster and achieve high quality/low cost production while meeting regulatory requirements. Engineering and construction firms can capitalize

on know-how to win projects and deliver them on time, within cost, at the highest quality while maximizing profitability.

CATIA Network Computing. CATweb Navigator is the first product from the new line of CATIA Network Computing Solutions. CATweb Navigator lets anyone access CATIA data from anywhere, over normal networks, using ordinary PCs and workstations. Clients require only a Java-enabled browser such as Netscape Navigator. Users can quickly view multiple models, drafts and reports remotely with no prior training, and 10 or even 10 100MB models can be viewed in just seconds. Figure 167 on page 375 shows a screen dump of CATweb in action.

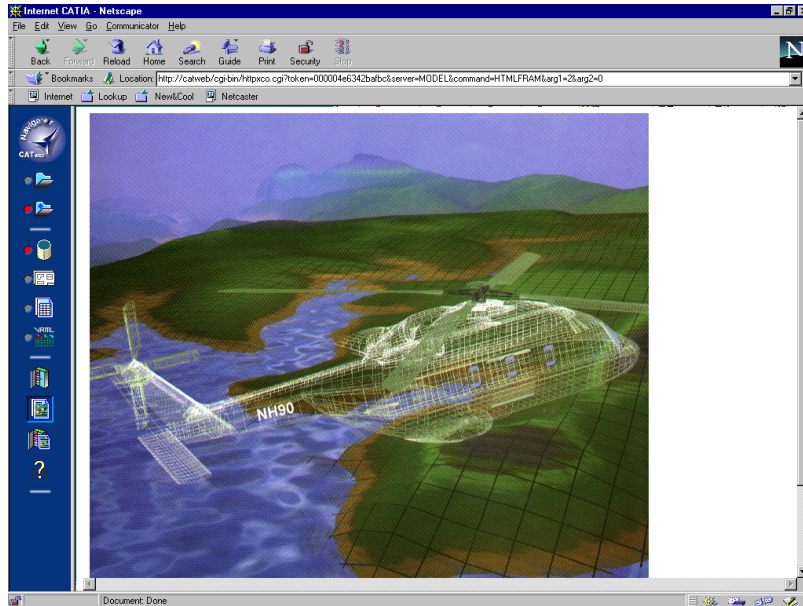


Figure 167. A Sample View of CATweb

CATIA has been extensively used to design large objects, and has proven to lower batch costs and times to completion. Boeing's 777 series design effort is the most publicized success story of a CATIA-based project.

A CATIA port of the RS/6000 SP should be available shortly after the publication of this book. Once the port is available, one can use a cluster of SP nodes to hold the database engine, and another set for applications accessing the database. The high network speeds inside the SP ensure that even with the largest sets of data, all clients would get response for the engine in a timely manner. And when even the speed of the SP and its

network is not enough, one can always add extra nodes to boost performance. Figure 168 lists mechanical engineering applications available for the RS/6000 SP.

APPLICATION	VENDOR	AVAILABILITY
AIRPLANE	AESOP, Inc.	Available
FIRE	AVL	Available
SPECTRUM V2.0	Centric	Available
CFX4	AEA Technologies	Available
STAR-HPC V3.0	Computational Dynamics	Available
CSA/NASTRAN	CSAR Corporation	Test version available
SESTRA	DNVSA (Veritas)	To be confirmed
PAM-CRASH	ESI	Available
RAMPANT	Fluent, Inc.	Available
FLUENT/UNS V4.0	Fluent, Inc.	Available
FIDAP	Fluent, Inc.	In Progress
GTNS2D **	Georgia Tech	Available
ABAQUS/Standard	HKS	Available
FLOW3P	Indiana U - Purdue U at Ind	Available
LS-DYNA3D V940MP	LSTC	Available
MARC	MARC Analysis	Available
RADIOSS	Mecalog Sarl	Available
MSC NASTRAN	MSC	Available
PCG Solver **	NASA Langley	Available
PVSOLVE **	NASA Langely	Available
SYSNOISE 5.3	LMS	Available
POLYFLOW	POLYFLOW	Available
SAMCEF	SAMTECH	Available
RAYON	Straco	Available
FORGE3	Transvalor	Available
NPARC3D V1.0 **	US Air Force	Available

Figure 168. Mechanical Engineering Parallel SP Applications

9.4.2 Electronic Design and Analysis

In the Electronic Design and Analysis (EDA) industry as well, there are several companies that are utilizing SPs to solve the ever more complex and convoluted problems associated with new chip and board designs. Companies that develop applications for this industry include Mentor Graphics, Synopsis, Avant!, and Silvaco. The applications are used for circuit analysis, device analysis and simulation, design simulation, packaging evaluation and placement and routing.

A partial list of applications currently available for the EDA industry is given in figure 169 on page 377.

APPLICATION	VENDOR	AVAILABILITY
AURIGA	FTL Systems	Beta Test
GAMBIT GA and SC	Gambit Automated Design	Available
ASX-p	IBM Microelectronics	Available
POWER Spice	IBM Microelectronics	Available
Quick Fault	Mentor Graphics Corp.	Beta test
PWORDS	MIT	Available
RADIATION	MIT	Available
OPTUM	Optum Engineering	Available
P-SPICE	Pacific Numerix	Available
Quest	Quickturn Design Systems	Beta test
ProperCAD	Sierra Vista Research	Available
VWF	Silvaco, Int.	Available
FIESTA **	Stanford University	Available
DAVINCI	Avant! Corporation (TMA)	Available
LITHO-WORKBENCH	Avant! Corporation (TMA)	Available
MEDICI	Avant! Corporation (TMA)	Available
TIMBERWOLF	Timberwolf Systems, Inc.	Available
FAIM	Vector Tech. Inc.	Available
MCP2D	Vector Tech. Inc.	Available
MCP3D	Vector Tech. Inc.	Available
OPC	Vector Tech. Inc.	Available
PROCLAT	Vector Tech. Inc.	Available
PROCPHASE	Vector Tech. Inc.	Available
BEBOP	University of Bologna	Available
TEMPEST **	UC Berkeley	Available
ProperSYN	University of Illinois	Available

Figure 169. Electronic Design and Analysis Parallel SP Applications

9.4.3 RS/6000 SP Implementations

All applications listed in figure 169 on page 377 take advantage of the massively parallel architecture of the SP to increase overall performance. Because of the programming and system tools that are available on the SP, applications may exploit parallelism by code or data parallelism, or by combinations of code and data parallelism techniques.

Since each application has different requirements and specifications, we will not describe the particular optimizations. A more appropriate reference would be any book that describes porting of a serial application to a parallel environment, along with *RS/6000 Scalable POWERparallel System: Scientific and Technical Overview*, SG24-4541.

9.4.4 Engineering Application Examples

This section presents the stories of some companies that used engineering applications on the SP to become more effective and competitive.

The companies that use such software tools include:

- Volvo Olofstrom is using a 28-node RS/6000 SP running LS-DYNA3D to design stamping equipment and perform sheet metal component engineering. The realizable speedup in a fender-forming simulation demonstrated the scalability power of the SP (see Figure 170).
- Bose is using a 4-node RS/6000 SP running SYSNOISE, TOSCA, and ABAQUS to do structural mechanics analysis, and acoustics and electromagnetics research.
- LG Electronics Inc. is designing semiconductor memory using SILVACO and TMA (Avant!) applications on an 8-node RS/6000 SP.

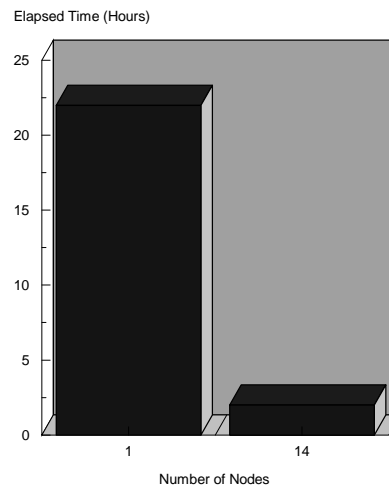


Figure 170. Speedup on RS/6000 SP using Parallel LS-DYNA3D

Appendix A. Special Notices

This publication is intended to help both RS/6000 SP specialists and general users who want to learn about the SP. The information in this publication is not intended as the specification of any programming interfaces that are provided by RS/6000 SP. See the PUBLICATIONS section of the IBM Programming Announcement for RS/6000 SP for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM ("vendor") products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no

guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

IBM ®	AIX
POWERparallel	RS/6000
NetView	LoadLeveler
SP	Deep Blue
AS/400	ES/9000
ESCON	SP1
AIX/ESA	SP2
POWER Architecture	Micro Channel
Current	PowerPC 604
Scalable POWERparallel Systems	Service Director
MVS/ESA	ADSTAR
MVS	Netfinity
Intelligent Miner	DB2
CICS	DB2 Universal Database
BookManager	

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.

Java and HotJava are trademarks of Sun Microsystems, Incorporated.

Microsoft, Windows, Windows NT, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

Pentium, MMX, ProShare, LANDesk, and ActionMedia are trademarks or registered trademarks of Intel Corporation in the U.S. and other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Other company, product, and service names may be trademarks or service marks of others.

Appendix B. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

B.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see “How to Get ITSO Redbooks” on page 385.

- *RS/6000 SP PSSP 2.4 Technical Presentation, SG24-5173*
- *RS/6000 SP PSSP 2.3 Technical Presentation, SG24-2080*
- *RS/6000 SP 2.2 Technical Presentation, SG24-4868*
- *RS/6000 SP High Availability Infrastructure, SG24-4838*
- *RS/6000 SP Monitoring: Keeping it Alive, SG24-4873*
- *GPFS: A Parallel File System, SG24-5165*

B.2 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

CD-ROM Title	Subscription Number	Collection Kit Number
System/390 Redbooks Collection	SBOF-7201	SK2T-2177
Networking and Systems Management Redbooks Collection	SBOF-7370	SK2T-6022
Transaction Processing and Data Management Redbook	SBOF-7240	SK2T-8038
Lotus Redbooks Collection	SBOF-6899	SK2T-8039
Tivoli Redbooks Collection	SBOF-6898	SK2T-8044
AS/400 Redbooks Collection	SBOF-7270	SK2T-2849
RS/6000 Redbooks Collection (HTML, BkMgr)	SBOF-7230	SK2T-8040
RS/6000 Redbooks Collection (PostScript)	SBOF-7205	SK2T-8041
RS/6000 Redbooks Collection (PDF Format)	SBOF-8700	SK2T-8043
Application Development Redbooks Collection	SBOF-7290	SK2T-8037

B.3 Other Publications

These publications are also relevant as further information sources:

- *IBM Parallel System Support Programs for AIX: Administration Guide, GC23-3897*
- *IBM Parallel System Support Programs for AIX: Installation and Migration Guide, GC23-3898*
- *IBM Parallel System Support Programs for AIX: Command and Technical Reference, GC23-3900*

How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at <http://www.redbooks.ibm.com/>.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Redbooks Web Site on the World Wide Web**

<http://w3.itso.ibm.com/>

- **PUBORDER** – to order hardcopies in the United States

- **Tools Disks**

To get LIST3820s of redbooks, type one of the following commands:

```
TOOLCAT REDPRINT
TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
```

To get BokkManager BOOKs of redbooks, type the following command:

```
TOOLCAT REDBOOKS
```

To get lists of redbooks, type the following command:

```
TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
```

To register for information on workshops, residencies, and redbooks, type the following command:

```
TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1998
```

- **REDBOOKS Category on INEWS**

- **Online** – send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL

Redpieces

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Web Site (<http://www.redbooks.ibm.com/redpieces.html>). Redpieces are redbooks in progress; not all redbooks become redpieces, and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** – send orders to:

	IBMMAIL	Internet
In United States	usib6fpl at ibmmail	usib6fpl@ibmmail.com
In Canada	caibmbkz at ibmmail	lmannix@vnet.ibm.com
Outside North America	dkibmbsh at ibmmail	bookshop@dk.ibm.com

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	(long distance charges apply)
(+45) 4810-1320 - Danish	(+45) 4810-1020 - German
(+45) 4810-1420 - Dutch	(+45) 4810-1620 - Italian
(+45) 4810-1540 - English	(+45) 4810-1270 - Norwegian
(+45) 4810-1670 - Finnish	(+45) 4810-1120 - Spanish
(+45) 4810-1220 - French	(+45) 4810-1170 - Swedish

- **Mail Orders** – send orders to:

IBM Publications Publications Customer Support P.O. Box 29570 Raleigh, NC 27626-0570 USA	IBM Publications 144-4th Avenue, S.W. Calgary, Alberta T2P 3N5 Canada	IBM Direct Services Sortemosevej 21 DK-3450 Allerød Denmark
--	--	--

- **Fax** – send orders to:

United States (toll free)	1-800-445-9269
Canada	1-800-267-4455
Outside North America	(+45) 48 14 2207 (long distance charge)

- **1-800-IBM-4FAX (United States) or (+1) 408 256 5422 (Outside USA)** – ask for:

Index # 4421 Abstracts of new redbooks
Index # 4422 IBM redbooks
Index # 4420 Redbooks for last six months

- **On the World Wide Web**

Redbooks Web Site	http://www.redbooks.ibm.com
IBM Direct Publications Catalog	http://www.elink.ibm.com/pbl/pbl

Redpieces

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Web Site (<http://www.redbooks.ibm.com/redpieces.html>). Redpieces are redbooks in progress; not all redbooks become redpieces, and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

IBM Redbook Order Form

Please send me the following:

Title	Order Number	Quantity
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____

First name _____ Last name _____

Company _____

Address _____

City _____ Postal code _____ Country _____

Telephone number _____ Telefax number _____ VAT number _____

Invoice to customer number _____

Credit card number _____

Credit card expiration date _____ Card issued to _____ Signature _____

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

List of Abbreviations

IBM	International Business Machines Corporation	FIFO	First-In First-Out
ITSO	International Technical Support Organization	GB	Gigabytes
ACL	Access Control List	GL	Group Leader
AIX	Advanced Interactive Executive	GPFS	General Purposes File System
AMG	Adapter Membership Group	GS	Group Services
ANS	Abstract Notation Syntax	GSAPI	Group Services Application Programming Interface
APA	All Points Addressable	GVG	Global Volume Group
API	Application Programming Interface	hb	heart beat
BIS	Boot/Install Server	HiPS	High Performance Switch
BSD	Berkeley Software Distribution	hrd	host respond daemon
BUMP	Bring-Up Microprocessor	HSD	Hashed Shared Disk
CP	Crown Prince	IP	Internet Protocol
CPU	Central Processing Unit	ISB	Intermediate Switch Board
CSS	Communication Subsystem	ISC	Intermediate Switch Chip
CW	Control Workstation	JFS	Journaled File System
DB	Database	LAN	Local Area Network
EM	Event Management	LCD	Liquid Crystal Display
EMAPI	Event Management Application Programming Interface	LED	Light Emitter Diode
EMCDB	Event Management Configuration Database	LRU	Last Recently Used
EMD	Event Manager Daemon	LSC	Link Switch Chip
EPROM	Erasable Programmable Read-Only Memory	LVM	Logical Volume Manager
		MB	Megabytes
		MIB	Management Information Base
		MPI	Message Passing Interface
		MPL	Message Passing Library

<i>MPP</i>	Massive Parallel Processors
<i>NIM</i>	Network Installation Manager
<i>NSB</i>	Node Switch Board
<i>NSC</i>	Node Switch Chip
<i>OID</i>	Object ID
<i>ODM</i>	Object Data Manager
<i>PE</i>	Parallel Environment
<i>PID</i>	Process ID
<i>PSSP</i>	Parallel System Support Programs
<i>PTC</i>	Prepare To Commit
<i>PTPE</i>	Performance Toolbox Parallel Extensions
<i>PTX/6000</i>	Performance Toolbox/6000
<i>RAM</i>	Random Access Memory
<i>RCP</i>	Remote Copy Protocol
<i>RM</i>	Resource Monitor
<i>RMAPI</i>	Resource Monitor Application Programming Interface
<i>RPQ</i>	Request For Product Quotation
<i>RSI</i>	Remote Statistics Interface
<i>RVSD</i>	Recoverable Virtual Shared Disk
<i>SBS</i>	Structure Byte String
<i>SDR</i>	System Data Repository

Symbols

/etc/krb.conf 154
/etc/krb.realms 155
/etc/krb-srvtab 152, 154
/etc/sysctl.acl 169
/etc/sysctl.conf 169
/var/kerberos/database 155

Numerics

2-tier and 3-tier concepts 317
2-tier concept solution 319
3-tier concept solution 319

A

abbreviations 390
Accelerated Strategic Computing Initiative 3
Access Control List 168
Accounting 256
 Exclusive-use of resources 256
 LoadLeveler parallel job 257
 record consolidation 256
 user name space considerations 257
ACL 213
acronyms 390
Active Users 309
Add an SP user 230
add_new_key 156
admin_acl.add 157
admin_acl.get 157
admin_acl.mod 157
administration service 35
ADSM 7, 268
Adstar Distributed Storage Manager 7, 280, 281, 282
AFS 163
afsd 164
Agora 5
AIX Automounter 199
AIX Security 144
allow 233
Aluminum 13
AMD 199
Andrew File System 172
Andrew Filesystem 163
Applications
 Cross Industry
 ERP 292
 JD Edwards' OneWorld 292

Oracle Applications 292
Peoplesoft Applications 292
R/3 292
 Design 293

Lotus

Domino 296
Notes 295
architecture 9
ASCII 3, 6
at 202
Athena project 145
Attributes 82
authentication 142, 152
Authentication Servers 218
authenticator 148
authorization 142
Authorization Callback 167
Automounter 197
automounter 228
Automounter daemon 199
Availability 356

B

Backup 89
backup, complex system 273
backup, Kerberos database 275
backup, SDR database 274
backup, simple system 271
Bandwidth 309
Base Operating System 199
batch 202
Batch Node 260
batch work 202
block 233
boot device 270
boot image 270
Boot Processing 223
boot sector 270
boot/install server 35, 47, 220
Boot/Install Servers 218
bootable image 270
bootp 220
BOP 101
buddy buffer 174
Business Intelligence Solutions 291

C

- C shell 201
- cache, level one 16
- cache, level two 16
- callbacks 168
- CATIA see Solutions
- Central lock manager 333
- Central Manager 263
- Central Manager, alternate 203, 267
- Central Manager, primary 203, 267
- Change SP user 231
- Checkpointing 268
- chkp 157
- CICS 321, 322
- Class 82
- Classes 87
- CLB 349
- Coexistence 226
- complex instruction 11
- components coexistence 227
- connection load balancing 349
- Connection Machine 14
- Content Hosting 291, 311
- Control Workstation 203, 261
- coordinator node 332
- copper 13
- Cost of Ownership 308, 356
- Costs of Distributed Computing 355
- cron 202
- css.snap 105
- customize nodes 220
- cw_allowed 231
- cw_restrict_login 231
- CWS 47
- CWS AIX Upgrade 225
- CWS PSSP Upgrade 225

D

- daemon, collector 264
- daemon, kbdd 267
- daemon, master 263
- daemon, negotiator 264
- daemon, schedd 264, 265
- daemon, SDR 274
- daemon, startd 264
- Daemons 84
- Data Bytes 101
- data cleansing 302

- Data Marts 301
- Data mining 302
- Data Packet 101
- Data Partitioning 336
- Data Shipping 331
- Data Warehouse 301
- Data Warehousing 346
- Database Master Password 159
- database propagation 151
- Databases 315
- DB2 330, 334
- DB2 PE 315
- DCE 322, 324
- Dedicated Temporary Tablespaces 347
- Delete an SP user 231
- deny 233
- DES 144
- device database 104
- Digital Time Service 110
- Disk Affinity 347
- dispatcher node 332
- Distributed File System 172
- Distributed Lock Manager 340, 341
- Distributed system 316
- DLM 340, 341, 342
- dog coffins 5
- DSS 300
- dump logical volume 222

E

- Eannotator 97
- EDA see Solutions
- EMAPI 130
- Encina 321, 323
- Enterprise Parallel Backup 349
- Enterprise Resource Planning Solutions and Lotus Notes 291
- Enterprise System Management 279
- Envoy 5
- EOP 101
- error logging 91
- Event Management 127
 - client 129
 - EMAPI 131
 - EMCDB 131, 132
 - event registration 130
 - ha_em_peers 131
 - haemd 128, 131

- instance vector 130
- predicate 130
- rearm predicate 130
- resource class 130, 133
- Resource Monitor Application Programming Interface 129
- resource variable 130
- RMAPI 131
- Event Manager 114
 - EMAPI 114
- Examples
 - Engineering 377
 - Lotus Notes 298
 - Petroleum 372
 - R/3 294
 - Research 363
- executing processor 333
- ext_srvtab 154
- Extended Tc 166

F

- Fault-tolerance 321
- Fencing 109
- File Collection 187
- file collection 199
- File Collections 188
- Flexibility 308, 357, 361
- frame_info 91
- fsck 180
- Function shipping 331

G

- General 260
- General Node 260
- General Purpose File System 180, 181
 - Tiger Shark 181
- Giga-processor 9
- GPFS, see General Purpose Filesystem 180
- Graphics 47
- Group Services 114
 - barrier synchronization 126
 - clients 120
 - external or user groups 121
 - group 120
 - Group State Value 121
 - Membership List 121
 - Name 121
 - Group Leader 124, 126

- Group Services Application Programming Interface 121
- hagsd 120, 123
- hagsglcmd 123
- internal components 122
- internal groups 121
- join request 123
- meta-group 122
- nameserver 122
- namespace 122
- Protocols 125
- providers 120
- Source-target facility 126
- subscriber 120
- sundered namespace 127
- Voting 125
- voting
 - 1-phase 125
 - n phase 125
- GSAPI 121
- GUI 213

H

- HACMP 113, 337
- HACMP Enhanced Scalability 285, 286
 - Positioning versus HACMP for AIX 287
 - Relationship with HAI 286
- HACMP ES, see HACMP Enhanced Scalability 285
- HACMP, see High Availability Cluster Multiprocessing 282
- HACWS, see High Availability Control Workstation 284
- HAGEO 283
- HAI, see also High Availability Infrastructure 112
- hardmon 90
- Hardware Control Subsystem 89
- Hardware Perspectives 215
- Hashed Shared Disk 175, 176
- Hashed Shared-Disks 345
- hatsd 115
- high availability 36
- High Availability Cluster Multi Processing 337
- High Availability Cluster Multiprocessing 113, 282
 - SP Implementation considerations 283
- High Availability Control Workstation 284
- High Availability Geographic Cluster for AIX 283
- High Availability Infrastructure 113

High Performance Supercomputer Systems Development Laboratory 4
High Performance Switch 305
high-performance switch 342
high-speed switch 19
HiPS 46
HiPS LC-8 46
hmacs 92
hmcmds 91, 93
hmno 93
hmmon 91
Host Response daemon 134, 135
host, sched public 265
HPSSDL 4
HPSSL 5
hrd, see host responds daemon 134
hwevents 91

I

I/O slots 47
identification 142
Impersonation 143
Implementing SPAC 233
Informix 330, 352
Informix and Sybase 351
Initialization 104
inside security 142
Installation 218
Instance Owner 336
instances 151
Institute of Advanced Studies 10
insulation 13
Intel 9
Intelligent Miner IM 302
Interactive Node 260
Internet Service Providers 291
Internet Solutions 291
Irving Wladawsky-Berger 5
ISB 43

J

job management 202

K

kadmin 155
kadmin 154, 160
kas 164, 165

kdb_util 157, 159
KDC 144
kdestroy 154, 159
Kerberos 143, 144, 284, 285
kerberos 110, 160
Kerberos Daemons 160
Kerberos Database 159
Kerberos database, restore 275
Kerberos Directories and Files 153
Kerberos primary server 35
Kerberos Principals 155
Kerberos secondary server 35
Kerberos secondary servers 151
Kerberos Tickets 158
kerberos, database backup 275
Key Distribution Center 144
kinit 154, 158
klist 154
klog.krb 164, 165
kpasswd 157
kpropd 160, 161
KRBTKFILE 153
kstash 154

L

least recently used 341
List SP users 231
Load Leveler 258
LoadLeveler 202
LoadLeveler Central Manager 263
lock manager 332
Locking 86
Log 89
Log Files 105
Logical Volume Manager 173
Login Control 232
LRU 341
lskp 157
LVM, see Logical Volume Manager 173

M

master database 35
Mega Web-site management 291, 311
memory bus 16
memory cross bar 16
Message Passing Interface 19
metadata 302
Michael Flynn 9

- migration 223
- Migration, automatic 268
- Migration, job 268
- MIMD 13, 15
- mksysb 269, 281
- MPI 93
- MPL 93
- MPP 303
- Multi processing processors 303
- Multiple Instruction, Multiple Data 15
- Multiple Instruction, Single Data 15
- multiprocessor 15
- MUSPPA 94

N

- N+1 26
- n2 problem 113
- NEF 61
- Network Connectivity Table 115
- Network File System 172
- network filesystem 197
- Network Information Services 197
- Network Installation 220
- Network Support Facilities 199
- Network Time Protocol 110
- new frame 26
- NFS 181, 197
- NFS Automounter 197
- NIS 197, 227
- NIS domain 198
- node configuration 219
- Node Customization 221
- node, submit only 265
- node_info 91
- Nodegroups 336
- NSB 43
- nternet Service Providers 311
- nteroperability 321
- NTP 110
 - consensus 111
 - duration of synchronization 111
 - internet 112
 - peers 111
 - primary 111
 - secondary 112
 - stratum 111
 - timemaster 112

O

- Objects 82
- OLAP 301
- OLTP 315
- On Line Analytical Processing (OLAP) 301
- OPQ 340
- OPS 340
- opsconf Tool 348
- opsctl Tool 348
- Oracle 173, 315, 330, 340
- Oracle 7 Parallel Server 340
- Oracle 7 Single Instance 340
- Oracle Parallel Server 173, 175, 176
- Oracle Trace 348

P

- Parallel Cost Model for the Optimizer 347
- Parallel Create Index 346
- Parallel Data Load 346
- Parallel Input/Output File System 178, 179, 180
 - advantages and disadvantages 180
 - Basic Striping Unit 179
 - Checkpointed file versions 180
 - partitionable files and views 180
- parallel jobs 202
- parallel mode server 340
- Parallel Operating Environment 258
- Parallel Queries 346
- Parallel Query 346
- parallel query 340
- Partition 240
- Partitioned Views 347
- Partitioning Key 336
- Partitioning Map 336
- partition-sensitive 246
- passwd 229, 231
- Performance 361
- Performance Agent 114
- Performance Optimization with Enhanced 4
- Peripheral device 47
- Phoenix 113
- Phoenix, see also High Availability Infrastructure 113
- ping 342
- planning tasks 218
- POE 93
- poll 92
- Pool 260

POSIX 180
post upgrade tasks 226
POWER 4
POWER Architecture 12
Primary 95
primary control workstation 36
primary name 152
Princeton University 10
principal password 145
principals 145
Problem Management 135
 pmanrmd 135
Processing Power 309
protocol 93
PSSP 198
pssp_script 221
pts 164

R

rc.boot 221
RDBMS 330
realm 151
Recoverable Virtual Shared Disk 176
 hc 177
 rvsd 177
Reduced Instruction Set Computer 12
Relational Database Management Systems 330
Reliability 321, 356, 361
Reliable Messaging 115
Reservoir simulation 369
resistance 13
Resource Manager 204, 258
Resource Manager, Access Control 260
Resource Manager, accounting 266
Resource Manager, backup 259
Resource Manager, primary 259
Resource Monitor 132
Resource Monitors 128
 external 132
 harmld 135
 harmpd 135
 hmrmd 134
 pmanrmd 135
 internal 132
 aixos 135
 Membership 135
 Response 135
 resource variables 133

 observation interval 133
 reporting interval 133
Restore 89
restore, complex system 273
restore, Kerberos database 275
restore, simple system 272
RMAPI, see also Resource Monitor Application Programming Interface in Event Management 129
rmkp 157
root.admin 156
Route Bytes 101
Routing 101
RS/6000 SP Implementation
 Engineering 377
 Lotus Domino 297
 Petroleum 371
 Process 367
 R/3 294
RSCT 113

S

SAP see Applications
 Cross Industry
 ERP
 R/3
savevg 271
Scalability 308, 321, 356, 361
scan 192
script.cust 222
SDR 82, 116, 119, 131
SDR database 274
SDR, archive 274
SDR, restore 274
Secondary 96
Secondary Authentication Servers 161
Secure Socket Layer 309
Security 141, 321
Security Management 152
security perimeter 141
SEF 58
Seismic processing see Solutions
serial job 204
serial ports 47
Servers Consolidation 355
Servers consolidation 315
Service Control Character 102
Service Packet 101
service ticket 149
session key 146

- setup_authent 152, 155, 219
- setup_logd 91
- setup_server 220
- Shared Memory 15
- Shared Nothing 15
- shared-nothing architecture 47
- SIMD 13, 14
- simple instruction 11
- Single Program, Multiple Data 22
- SISD 10
- SISD uniprocessor 10
- SMPI, see also System Performance Measurement Interface 115
- Software Components 81
- Software Upgrade 223
- Solution Domains 280
- Solutions
 - Scientific and Technical 361
 - Engineering 373
 - EDA 376
 - Mechanical 373
 - CATIA 373
 - Petroleum 368
 - Reservoir 369
 - Seismic Processing 368
 - Research 362
 - SP Access Control 232
 - SP System Monitor 217
 - SP System Partitioning 219
 - sp_passwd 229
 - SP1 5
 - SP2 6
 - SPAC 232
 - spacs.log 234
 - spacs_cntrl 232
 - spchuser 231
 - SPdaemon.log 91
 - sphardware 93
 - splogd 91
 - splsuser 231
 - SPMD 22
 - SPMI Library 136
 - spmuser 229, 230
 - spmon 93
 - SPOT 221
 - spruser 231
 - SPS 46
 - SPS-8 46
 - SPUM 227
 - SQL 330
 - SSL 309
 - Stanford University 9
 - Star Queries 347
 - starter process 267
 - Structured Query Language 330
 - submit only node 203
 - SunOS 4.x Automounter 199
 - supercomputer 4
 - supper 188, 195
 - supper daemon 199
 - support processor 36
 - SVC See Service Control Character Switch 93
 - Switch Clocks 105
 - Sybase 330, 353
 - Sysback 268
 - Sysctl 165
 - sysctl scripts 171
 - sysctl server 166
 - sysctld 166
 - Syspar 245
 - system characteristics 6
 - system clock 110
 - System Data Repository 260
 - System Monitor GUI 217
 - System Performance Measurement Interface 115, 132

T

- Tcl 166
- Thinking Machines 14
- Ticket 145
- ticket cache file 153
- ticket-granting service 35
- Ticket-Granting Ticket 146
- ticket-granting ticket 146
- time management 110
- time zones 110
- timed 110
- timestamp 110
- Tivoli 7, 283, 288
- TOD 104
- token.krb 164
- tokens.krb 165
- Topology Services
 - Adapter Membership 116
 - Reliable Messaging 115
- Topology 246

Topology File 96
Topology Services 114
 Adapter Membership 116
 Adapter Membership Group 116, 117, 119
 Crown Prince 117
 DEATH_IN_FAMILY 118
 Group Leader 117
 hatsd 115
 heartbeat 118, 119
 frequency 118
 sensitivity 118
 Machine list 117
 machine list 116, 119
 machine lists 117
 Neighbor 118
 Network Connectivity Table 115, 118, 119
 Node Membership 116
 Prepare To Commit 118
 Proclaim Packet 117
 Reliable Messaging 128
 Singleton group 116, 117
 topology table 119
TP Monitors 321
Trailblazer 6
TTY 73
tuning.cust 222
Tuxedo 321, 325

U

UDP 115
UDP/IP 111, 120, 179
unblock 233
Unfencing 109
UNIX domain sockets 129
UNIX Domain Stream 115
unlog 165
upgrade 223
User Interface 84
User Interfaces 92
User Management 227
Using sysctl 170
Utilization 357

V

Virtual Memory Manager 173
Virtual Shared Disk 172, 340
 buddy buffer 174
Virtual Shared-Disks 345

VLDB 305
VMM, see also Virtual Memory Manager 173
von Neumann, Bottleneck 13
von Neumann, John 10
VSD 173, 213, 241, 340
Vulcan 5

W

Winter Olympics 98 (Nagano) 313
Worm 94

X

X_RUNS_HERE 267
xntpd 111

ITSO Redbook Evaluation

Inside the RS/6000 SP
SG24-5145-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at <http://www.redbooks.ibm.com>
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Which of the following best describes you?

Customer **Business Partner** **Solution Developer** **IBM employee**
 None of the above

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction _____

Please answer the following questions:

Was this redbook published in time for your needs? Yes___ No___

If no, please explain:

What other redbooks would you like to see published?

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

Inside the RS/6000 SP

SG24-5145-00

**SG24-5145-00
Printed in the U.S.A.**

