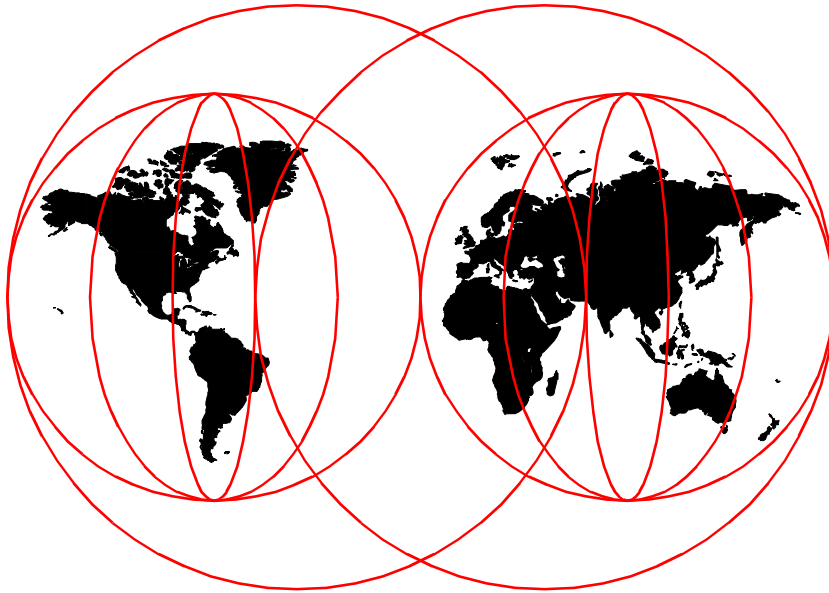


Implementing SAS on the RS/6000 Family

Ole Conradsen, Colin Cunningham, Jim West



International Technical Support Organization

www.redbooks.ibm.com

SG24-5513-00



International Technical Support Organization

Implementing SAS on the RS/6000 Family

February 2000

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix H, "Special notices" on page 181.

First Edition (February 2000)

This edition applies to base SAS software Version 6.12, for use with the AIX 4.3.2 Operating System.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. JN9B Building 003 Internal Zip 2834
11400 Burnet Road
Austin, Texas 78758-3493

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 2000. 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

Figures	vii
Tables	ix
Preface	xi
The team that wrote this redbook	xi
Comments welcome	xii
Chapter 1. RS/6000 hardware overview	1
1.1 RS/6000 history	1
1.2 RS/6000 design overview	1
1.2.1 RS/6000 system blocks	2
1.2.2 Processor technologies	2
1.3 RS/6000 new technologies and directions	3
1.3.1 64-bit technology	3
1.3.2 Storage technologies	3
1.4 RS/6000 facts and features summary	4
1.5 Workstations and workgroup servers	12
1.6 Midrange enterprise servers	12
1.7 High-end enterprise servers	12
1.8 RS/6000 SP systems	13
1.9 PCI storage adapters	14
1.9.1 SCSI-2 fast/wide RAID adapter	14
1.9.2 PCI single-ended Ultra SCSI adapter	14
1.9.3 PCI differential Ultra SCSI adapter	14
1.9.4 SCSI-2 fast/wide adapter 4-A	14
1.9.5 SCSI-2 differential fast/wide adapter 4-B	14
1.9.6 PCI 3-Channel Ultra SCSI RAID adapter	15
1.9.7 SSA RAID 5 adapter, SSA fast-write	15
1.9.8 Advanced serialRAID, SSA fast-write	15
Chapter 2. SAS installation	17
2.1 SAS preinstallation file system planning	19
2.2 Installing the base SAS software on AIX	19
Chapter 3. Application test environment	23
3.1 AIX system configuration	23
3.2 SAS applications	26
3.2.1 The test environment	27
3.3 Application data and tests	27
3.3.1 U.S. Census data	27

3.3.2	Computational tests and data	28
3.3.3	Test diagnostics	29
Chapter 4. SAS tuning parameters		31
4.1	Base SAS software options	32
4.1.1	Inclusion of system options.	32
4.2	Other considerations	34
4.2.1	Work space	34
4.2.2	Use SAS compression wisely	35
4.2.3	Indexing	36
4.2.4	Enabling very large file (> 2 GB) access	36
4.3	Data processing performance test results.	37
4.3.1	Reading and writing, CPU-bound	37
4.3.2	Sorting processes.	40
4.3.3	Sortsize and memory usage	45
4.3.4	Last thoughts on sorting	46
4.4	Computational test results	47
4.4.1	PROC REG	47
4.4.2	PROC LOGISTIC	48
4.4.3	Taming PHREG	49
Chapter 5. AIX performance tools		51
5.1	Performance tools description	51
5.1.1	vmstat command	51
5.1.2	iostat command	53
5.1.3	svmon command	54
5.1.4	lsps command	56
5.1.5	vmtune command	57
5.1.6	rmss command	58
Chapter 6. Optimizing AIX parameters		59
6.1	CPU bound load	60
6.2	Tuning memory bound load	61
6.3	Tuning disk I/O bound load	64
Chapter 7. AIX file system, SAS system, and performance		67
7.1	Testing performance of SAS workspace	68
7.2	Testing disk striping	70
Chapter 8. The user community and performance considerations		77
8.1	Community assessment	77
8.1.1	General usage	77
8.1.2	User profiles	77
8.1.3	Expected SAS application	78

8.1.4	Access and interface of choice	79
8.1.5	Disk access	79
8.1.6	Data sources	80
8.1.7	SAS files on disk	81
8.1.8	Job scheduling	82
8.1.9	Other software	83
8.1.10	System objectives	83
8.2	Remote access setup	83
Appendix A. The SAS System on an IBM RS/6000 SP		85
A.1	SP configuration for cluster operation	85
A.2	Load balancing	86
A.3	Data sharing	87
Appendix B. Disk space and RAM requirements		89
Appendix C. Optimizing systems performance		93
C.1	Techniques for optimizing I/O	93
C.2	Techniques for optimizing memory usage	94
C.3	Techniques for optimizing CPU performance	94
C.4	SAS system options	95
C.5	SAS procedures that use extra resources	100
C.6	Performance considerations of DATA step views	104
Appendix D. Disk controller cache test results		107
D.1	EXECUTION time with write cache is disabled	108
D.2	EXECUTION time when disk adapter write cache is enabled	114
Appendix E. The SAS System and DB2 partitioned databases		121
E.1	SAS on a cluster of SP nodes	122
E.1.1	SP configuration for cluster operation	122
E.1.2	Load balancing	123
E.1.3	Data sharing	123
E.1.4	Practical flexibility	127
E.1.5	SAS and DB2 partitioned databases on the RS/6000 SP	128
E.1.6	SAS cleansing/transforming of data	128
E.2	Extracting large amounts of data from DB2 partitioned databases	130
E.3	SAS to DB2 partitioned database parallel extract	131
E.3.1	CAE for AIX configuration	132
E.3.2	SQL modifications and table view definitions	133
E.3.3	Query restrictions	134
E.3.4	Parallel extract processing flow	134
E.3.5	Parallel extract SAS node alternatives	135
E.3.6	Single node parallel extract processing	136

E.3.7 Multiple node parallel extract processing	137
E.3.8 Multiple SAS nodes parallel extract implementation.	138
E.4 Hybrid parallel extract	139
E.5 Multiple DB2 logical nodes implementation	140
E.5.1 Hybrid parallel extract	141
E.5.2 True parallel extract	142
E.5.3 Summary	143
Appendix F. The SAS system and GPFS	145
F.1 GPFS overview	145
F.1.1 Hardware configuration	146
F.1.2 Software configuration	147
F.1.3 Physical disk configuration	147
F.1.4 System configuration	147
F.1.5 Local disk configuration	148
F.1.6 NFS disk configuration	148
F.1.7 GPFS disk configuration	148
F.2 Tests	149
F.2.1 Test results	150
F.2.2 Notes on testing	153
F.3 Conclusions.	155
Appendix G. Scalable performance.	157
Appendix H. Special notices	181
Appendix I. Related publications	185
I.1 IBM Redbooks publications	185
I.2 IBM Redbooks collections	185
I.3 Other resources	185
I.4 Referenced Web sites	186
How to get IBM Redbooks	187
IBM Redbooks fax order form	188
Glossary	189
Index	191
IBM Redbooks evaluation	197

Figures

1. RS/6000 systems handbook products in focus	5
2. Disk adapters	23
3. Disk drives in the test system	23
4. Paging space	24
5. Volume group rootvg	24
6. Volume group mfs01 holding SAS Input dataset	25
7. Volume group mfs02 holding SAS output dataset	25
8. Volume group dbvg holding SAS workspace.	26
9. Volume group sasvg	26
10. SAS system diagnostics with FULLSTIMER	30
11. Typical output from the vmstat command	52
12. Typical output from the iostat command	53
13. Typical output from the svmon command	54
14. svmon -Pa	56
15. Typical output from an lsps -a command	57
16. Optimizing procedure	59
17. CPU usage for SAS process	60
18. Memory usage for SAS process	61
19. SAS job with finite memory usage	62
20. SAS job using more memory than physical available	62
21. Time to run a SAS program as function of memory.	63
22. Free pages during program execution	64
23. Memory usage during program run	65
24. Execution times for various SAS workspace configurations	69
25. Non striped disk configuration	70
26. Striped disk configuration.	71
27. Average execution time without write cache	73
28. Average execution time with write cache enabled	74
29. Non-striped test runs with relative performance change	75
30. Striped test runs with relative performance change	75
31. SAS cluster configuration on SP, LoadLeveler ISS Pool -spsas	125
32. Data cleansing/transforming with SAS on SP	129
33. SAS to DB2 partitioned database - Default access	131
34. SAS to DB2 partitioned database parallel extract access	132
35. SAS to DB2 partitioned database access	135
36. Single SAS node parallel extract implementation	137
37. Multiple SAS node parallel extract implementation	138
38. SAS to DB2 partitioned database hybrid parallel extract	140
39. Multiple logical DB2 nodes per SMP node.	141
40. Hybrid parallel extract	142

41. True parallel extract	143
42. Node configuration	147
43. Local and NFS configurations	148
44. Single node GPFS configuration	149
45. GPFS configuration	149
46. Test 1 results	150
47. Test 2 results	151
48. Test 3 results	152
49. Test 4 results	153
50. Parallel extraction from DB2UDB	158
51. Database extraction into sequential SAS data	165
52. Parallel extraction using OFS	166
53. Parallel extraction from DB2 creating a sequential dataset on a thinnode	168
54. Scalability of parallel extraction from DB2 to equential SAS dataset	168
55. Parallel SAS application	171
56. SAS application run time versus number of processors	173
57. Superlinear speedup of SAS forecasting model	174

Tables

1. Facts and features for Models 140, 150, and 260	6
2. Facts and features for Models F40 and F50	7
3. Facts and features for Models H50 and H70	8
4. Facts and features for Models S70 and S70 Advanced.	9
5. Facts and features for SP 332 MHz SMP Thin and SMP Wide Nodes . . .	10
6. SP 160 MHz and POWER3 SMP Thin and POWER3 SMP Wide Nodes .	11
7. Performance benchmark using default settings.	38
8. Test 1 SAS data file sizes	38
9. Buffer versus performance in Test 1	39
10. Best case sort for Test 2	41
11. Test 2 under reduced memory.	42
12. Sorting data file larger than physical memory	43
13. Unleashing SAS's access to memory	43
14. Overcommitting memory to SAS in Test 2.	44
15. Four simultaneous executions of Test 2 runs with varying SORTSIZE . . .	45
16. PROC REG declining memory in Test 3	48
17. Logistic regression and Test 4.	48
18. Staggered gains with memory with PHREG	49
19. CPU time as function of MEMSIZE parameter	65
20. .Table 20File system usage.	67
21. Node name and rremote host	133
22. Catalogued databases.	133
23. VSD parameters	154
24. Census data.	167
25. Results of parallel SAS application test.	172

x Implementing SAS on the RS/6000 Family

Preface

We wrote this redbook to provide information for business partners and IBM employees installing SAS on IBM RS/6000 systems. This redbook will help you install, tailor, and configure the base SAS software on the IBM RISC System 6000 platform. Throughout the book, we will describe the software installation and how various configuration parameters influence function and performance. We will also examine RS/6000 system configurations to help you determine the best configuration.

It is our goal to describe the installation procedure, SAS configuration parameters, and RS/6000 and AIX parameters. This information is the basis for preinstallation planning and sizing as well as a guide during system setup.

After reading this redbook, you should be able to:

- Interview users/customers about their requirements and transform the information into performance requirements
- Compare RS/6000 HW models and components (DASD, Memory, and so on) with the type of load and requirements for the specific installation
- Install SAS on the IBM RS/6000 (AIX)
- Monitor performance and pinpoint bottlenecks
- Optimize SAS and AIX based on the information gathered from monitoring the system

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, Austin Center.

Ole Conradsen is a Project Leader at the International Technical Support Organization, Austin Center. Before joining the ITSO, he worked with AIX projects in IBM Denmark for 11 years. Ole holds a master of science degree in Electrical Engineering from the Technical University of Denmark.

Colin Cunningham joined the IBM Worldwide Database Marketing department in 1998 after four years of statistical consulting roles for industry. His statistical consulting has focused on predictive and segmentation modeling applications to database marketing. Colin holds a degree in Actuarial Mathematics from the University of Michigan and a master's degree in Statistics from Pennsylvania State University. Colin lives in Seattle, Washington.

Jim West joined the IBM Federal Systems Division to work with the Superconducting Super Collider Laboratory and continued with IBM Federal Systems performing scientific technical benchmarking on the SP. In 1995, he joined the RS/6000 Division to work with large commercial software vendors in software enablement and benchmarking. Mr. West has a bachelor of arts degree in Physics from Emory University and a master of science degree in Nuclear Engineering from the Georgia Institute of Technology.

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

Margaret Crevar, Leigh Ihnen
SAS Institute Inc. Cary, North Carolina

Jack Rivers
IBM Raleigh, North Carolina

Harry T. Seifert
IBM Louisville, Kentucky

P.W. Leathem, Keith F. Olsen
IBM Corporation

Scott Vetter
IBM ITSO, Austin Center

Milos Radosavljevic
IBM ITSO, Austin 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 “IBM Redbooks evaluation” on page 197 to the fax number shown on the form.
- Use the online evaluation form found at <http://www.redbooks.ibm.com/>
- Send your comments in an Internet note to redbook@us.ibm.com

Chapter 1. RS/6000 hardware overview

The RS/6000 family combines the benefits of UNIX computing with leading-edge IBM RISC technology in a broad product line that includes powerful desktop workstations ideal for mechanical design, workgroup servers for departments small businesses, enterprise servers for medium to large companies, and massively-parallel RS/6000 SP systems that can handle demanding technical computing, business intelligence, and Web-serving tasks. Along with AIX, the award-winning IBM UNIX operating system, and HACMP (the leading high-availability clustering solution), the RS/6000 platform provides the power to create change and the flexibility to manage it with a wide variety of applications that provide real value.

1.1 RS/6000 history

The first RS/6000 was announced February 1990 and shipped June 1990. Since then, over 800,000 systems have shipped to over 125,000 customers.

- RS/6000 systems use POWER2, POWER3, and POWERPC CPU chips. All CPUs are compatible and run the same AIX versions. However, different CPUs have different performance characteristics.
- In the past, RS/6000 I/O buses were based on the Micro Channel Architecture (MCA). Today, RS/6000 I/O buses are based on the industry-standard Peripheral Component Interface (PCI) architecture.
- Processor speed, a key element of RS/6000 system performance, has increased dramatically over time.
- There have been many machine types over the entire RS/6000 history. However, in recent years, there has been considerable effort to reduce the complexity of the model offerings without creating gaps in market coverage.

1.2 RS/6000 design overview

This section provides information on the following elements, which are important in the design of RS/6000 machines:

- An explanation of general system blocks
- RS/6000 processor architectures

1.2.1 RS/6000 system blocks

All platforms (from workstations to high-end servers) consist of one or more processors, a volatile system memory separate from other subsystems, and a number of I/O devices that may initiate transactions to system memory.

In general, I/O devices do not connect to the primary processor bus/switch. The host bridges connect to secondary buses that have I/O devices connected to them. Most commonly, the adapters use the PCI architecture.

1.2.1.1 PCI slots

The PCI architecture provides an industry-standard specification and protocol that allows multiple adapters access to system resources through a set of adapter slots. Each PCI bus has a limit on the number of slots (adapters) it can support. Typically, this can range from two to six. To overcome this limit, the system design can implement multiple PCI buses.

1.2.1.2 Integrated adapters

A number of devices are now integrated onto the main processor board, but they physically connect to one of the PCI buses. For this reason, some of the buses may only have two or three slots available to install adapters. Examples of integrated PCI adapters are SCSI adapters and Ethernet adapters.

1.2.2 Processor technologies

IBM has developed industry-leading processor fabrication technologies. These technologies are copper circuitry and silicon-on-insulator (SOI) on complimentary metal oxide semiconductor (CMOS) chips.

These technologies, which contribute to higher performance and reduced power requirements, are the basis for enhancements to the current IBM POWER3 processors and for the upcoming IBM POWER4 Gigahertz processor. It is likely that these technologies will benefit many areas of system development.

1.2.2.1 POWER3 processor

The POWER3 processor introduces a new generation of 64-bit processors especially designed for high-performance and visual computing applications. POWER3 processors replaces the POWER2 and the POWER2 Super Chips (P2SC) in high-end RS/6000 workstations and SP nodes. The RS/6000 43P 7043 Model 260 workstation features the POWER3 processor as well as the POWER3 wide and thin nodes.

1.2.2.2 RS64 and RS64-II processors

The RS64 processor, based on the PowerPC Architecture, was designed for leading-edge performance in OLTP, e-business, BI, server consolidation, SAS, SAP, Notesbench, and Web serving for the commercial and server markets.

The RS64 processor focuses on commercial performance. It has 64 KB of L1 instruction and data caches, one cycle load support, four superscalar fixed point pipelines, and one floating point pipeline.

1.3 RS/6000 new technologies and directions

This section gives a brief overview of the RS/6000 leading-edge technology and directions.

1.3.1 64-bit technology

64-bit computing is the direction for all RS/6000 products. The essence of the RS/6000 64-bit computing strategy can be summed up in three themes:

- Complementing the established scalability of the existing 32-bit product set, 64-bit technology is the enabler for scaling enterprise SMP servers to higher capacity, making high-end system performance one of the primary customer benefits of 64-bit computing.
- 64-bit computing is complementary to 32-bit computing. Customers want the benefits of 64-bit technology available to them, but know their 32-bit systems and 32-bit applications will be important investments for a long time.
- The transition from 32-bit computing to a future in which 64-bit and 32-bit computing coexist will be, for RS/6000 customers, a very smooth evolution.

These themes were also the design principles behind the product implementation of AIX Version 4.3 as a 64-bit operating environment. AIX Version 4.3 introduces significant functional and scalability enhancements that benefit all RS/6000 customers. A single AIX product supporting both 64-bit computing as well as broad general improvements is a prime example of the RS/6000 evolutionary vision at work.

1.3.2 Storage technologies

Although Serial Storage Architecture (SSA) has been IBM's major focus for the disk storage technology for the past three years, other technologies, namely Ultra SCSI and Fibre Channel, have been advancing throughout the

industry. IBM has been deploying all these technologies across its product sets. Ultra SCSI, Fibre Channel, and Serial Storage Architecture (SSA) are all variations of SCSI-3 standard. These variations of SCSI-3 all support the same command set, while differing in the physical cabling and low-level protocols that are transparent to software. Both SSA and Fibre Channel (including Fibre Channel Arbitrated Loop, FC-AL) offer a number of benefits beyond Ultra SCSI, such as increased speed and distance. In terms of bandwidth, Fibre Channel's 1 GBps will outperform Ultra SCSI's 80 MBps.

1.4 RS/6000 facts and features summary

The following section, taken from the *RS/6000 Facts and Features Brochure*, G320-9878, outlines the important characteristics of the featured RS/6000 models. Figure 1 on page 5 shows the RS/6000 Model family. Tables 1 through 6 give a detailed description of the features and options for each model.




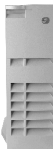






Large Scale Servers	<p>9076-550 SP 332 MHz PowerPC 604e, 160 MHz P2SC, and 200 MHz POWER3 Nodes 1 to 512 Nodes</p>			
High-End Enterprise Servers	<p>7017-S70 125 MHz RS64/ 262 MHz RS64-II 4- to 12-way SMP</p>		<p>7017-S7A 262 MHz RS64-II 4- to 12-way SMP</p>	
Midrange Enterprise Servers	<p>7025-F50 166/332 MHz PowerPC 604e 1-to 4-way SMP</p>		<p>7026-H50 332 MHz PowerPC 604e 1- to 4-way SMP</p>	
			<p>7026-H70 340 MHz RS64-II 1- to 4-way SMP</p>	
Workstations and Workgroup Servers	<p>7043-140 233/332 MHz PowerPC 604e Uniprocessor</p>		<p>43P Model 150 375 MHz PowerPC 604e Uniprocessor</p>	
			<p>43P Model 260 200 MHz POWER3 1- to 2-way SMP</p>	
			<p>7025-F40 233 MHz PowerPC 604e 1-to 2-way SMP</p>	

Figure 1. RS/6000 systems handbook products in focus

Table 1. Facts and features for Models 140, 150, and 260

RS/6000 Model Machine Type	140 7043	150 7043	260 7043
Processor type	PowerPC 604e	PowerPC 604e	POWER3
#processors/system	1	1	1 or 2
Clock rates available (standard/option)	233/332 MHz	375 MHz	200 MHz
System memory (standard/maximum)	64 MB/768 MB	128 MB/1 GB	256 MB/4 GB ¹
Memory type	64-bit ECC DIMM	64-bit ECC SDRAM	64-bit ECC SDRAM
Data/instruction (L1) cache	32 KB/32 KB	32 KB/32 KB	64 KB/32 KB ²
Level 2 (L2) cache	1 MB	1 MB	4 MB ²
Memory slots	6	4	2
Capacity			
Slots available	3 PCI (32-bit) + 2 PCI/ISA (32-bit)	5 PCI (32-bit)	2 PCI (64-bit) + 3 PCI (32-bit)
PCI bus speed	33 MHz	33 MHz	33/50 MHz
Disk/media bays	5	5	5
Standard/maximum internal disk	4.5 GB/27.3 GB	4.5 GB/27.3 GB	4.5 GB/27.3 GB
Storage interfaces			
SCSI-2 Fast/Wide SE and SCSI-2 Fast/Wide Differential	X	X	X
SCSI-2 Fast/Wide RAID-5	-	-	-
Ultra SCSI SE and Ultra SCSI Differential	X	X ³	X
SSA 8-way JBOD/2-way RAID EL (PCI and MCA)	X	X ³	X
PCI 3-channel Ultra SCSI RAID	-	X	X
Fibre Channel	-	-	-
Communications and connectivity			
EIA RS232D/EIA RS422A	X	X	X
Token-ring 4/16 Mbps	X	X	X
Ethernet 10 Mbps	X	X	X
Ethernet 10/100 Mbps	X	X	X
Gigabit Ethernet	-	-	X
FDDI 100 Mbps	X	X	X
ATM 25 Mbps	X	X	X
ATM 155 Mbps	X	X	X
ISDN	X	X	X
X.25	X	X	X
SDLC	X	X	X
BSC	X	-	X
SP system attachment	-	-	-
3270 connection	-	-	-
ESCON	-	-	-
Block multiplexer	-	-	-
HIPPI 100 Mbps	-	-	-
T1/E1	X	-	X
Telephony	X	-	-
Graphics accelerators available	GXT120P, 250/255P, 550P, 800P; 2000P	GXT120P, 250/255P, 550P, 2000P, 3000P	GXT120P, 250/255P, 550P, 2000P, 3000P

6 Implementing SAS on the RS/6000 Family

Table 2. Facts and features for Models F40 and F50

RS/6000 Model Machine Type	F40 7025	F50 7025
Processor type	PowerPC 604e	PowerPC 604e
#processors/system	1 or 2	1, 2, 3, or 4
Clock rates available (standard/option)	233 MHz	166/332 MHz
System memory (standard/maximum)	64 MB/1 GB ¹	128 MB/3 GB ¹
Memory type	64-bit ECC DIMM	64-bit ECC SDRAM
Data/instruction (L1) cache	32 KB/32 KB ²	32 KB/32 KB ²
Level 2 (L2) cache	1 MB ²	256 KB ²
Memory slots	8	2
Capacity		
Slots available	7 PCI (5 32-bit + 2 64-bit) + 2 PCI/ISA (32-bit)	7 PCI (5 32-bit + 2 64-bit) + 2 PCI/ISA (32-bit)
PCI bus speed	33/50 MHz	33/50 MHz
Disk/media bays	22	18/4
Standard/maximum internal disk	4.5 GB/172.8 GB	4.5 GB/172.8 GB
Storage interfaces		
SCSI-2 Fast/wide SE and SCSI-2 Fast/Wide Differential	X	X
SCSI-2 Fast/wide RAID-5	X	X
Ultra SCSI SE and Ultra SCSI Differential	X	X
SSA 8-way JBOD/2-way RAID EL (PCI and MCA)	X	X
PCI 3-channel Ultra SCSI RAID	-	X
Fibre Channel	-	X ³
Communications and connectivity		
EIA RS232D/EIA RS422A	X	X
Token-ring 4/16 Mbps	X	X
Ethernet 10 Mbps	X	X
Ethernet 10/100 Mbps	X	X
Gigabit Ethernet	-	X
FDDI 100 Mbps	X	X
ATM 25 Mbps	X	X
ATM 155 Mbps	X	X
ISDN	X	X
X.25	X	X
SDLC	X	X
BSC	X	X
SP system attachment	-	-
3270 connection	-	-
ESCON	-	X
Block multiplexer	-	-
HIPPI 100 Mbps	-	-
T1/E1 ⁴	X	X
Telephony ⁴	X	X
Graphics accelerators available	GXT120P, 250/255P, 550P, 800P	GXT120P, 800P ⁶

Table 3. Facts and features for Models H50 and H70

RS/6000 Model Machine Type	H50 7026	H70 7026
Processor type	PowerPC 604e	PowerPC RS64-II
#processors/system	1, 2, 3, or 4	1, 2, 3, or 4
Clock rates available (standard/option)	332 MHz	340 MHz
System memory (standard/maximum)	128 MB/3 GB ¹	128 MB/8 GB ¹
Memory type	64-bit ECC SDRAM	64-bit ECC SDRAM
Data/instruction (L1) cache	32 KB/32 KB ²	64 KB/64 KB ²
Level 2 (L2) cache	256 KB ²	4 MB ²
Memory slots	2	2
Capacity		
Slots available	7 PCI (5 32-bit + 2 64-bit) + 2 PCI/ISA (32-bit)	8 PCI (4 32-bit + 4 64-bit)
PCI bus speed	33/50 MHz	33/50 MHz
Disk/media bays	13/3	13/3
Standard/maximum internal disk	4.5 GB/118.2 GB	4.5 GB/127.4 GB
Storage interfaces		
SCSI-2 Fast/Wide SE and SCSI-2 Fast/Wide Differential	X	X
SCSI-2 Fast/Wide RAID-5	-	X
Ultra SCSI SE and Ultra SCSI Differential	X	X
SSA 8-way JBOD/2-way RAID EL (PCI and MCA)	X	X
PCI 3-channel Ultra SCSI RAID	X	X
Fibre Channel	X ³	X ³
Communications and connectivity		
EIA RS232D/EIA RS422A	X	X
Token-ring 4/16 Mbps	X	X
Ethernet 10 Mbps	X	X
Ethernet 10/100 Mbps	X	X
Gigabit Ethernet	X	X
FDDI 100 Mbps	X	X
ATM 25 Mbps	X	X
ATM 155 Mbps	X	X
ISDN	X	X
X.25	X	X
SDLC	X	X
BSC	X	X
SP system attachment	-	-
3270 connection	-	-
ESCON	X	X
Block multiplexer	-	-
HIPPI 100 Mbps	-	-
T1/E1	X	X
Telephony	X	X
Graphics accelerators available	GXT120P	GXT120P

Table 4. Facts and features for Models S70 and S70 Advanced

RS/6000 Model Machine Type	S70 7017	S70 Advanced 7017	S80 7017
Processor type	PowerPC RS64/RS64-II ⁸	PowerPC RS64-II	PowerPC RS64-III
#processors/system	4, 8, or 12	4, 8, or 12	6, 12, 18 or 24
Clock rates available (standard/option)	125 MHz/262 MHz	262 MHz	450 MHz
System memory (standard/maximum)	512 MB/32 GB ¹	1 GB/ 32 GB ¹	2 GB/
Memory type	Card-based ECC SDRAM	Card-based SDRAM	ECC Card-based SDRAM
Data/instruction (L1) cache	64 KB/64 KB ²	64 KB/64 KB ²	128KB / 128 KB
Level 2 (L2) cache	4 MB (125 MHz)/8 MB ² (262 MHz)	8 MB (262 MHz)	8 MB
Memory slots	20	20	
Capacity			
Slots available	53 PCI (33 32-bit + 20 64-bit)	53 PCI (33 32-bit + 20 64-bit)	
PCI bus speed	33 MHz	33 MHz	
Disk/media bays	48/12	48/8	
Standard/maximum internal disk	4.5 GB/436.8 GB	9.0 GB/436.8 GB	
Storage interfaces			
SCSI-2 Fast/Wide SE	X		
SCSI-2 Fast/Wide Differential	X	X	X
SCSI-2 Fast/Wide RAID-5	-	-	-
UltraSCSI SE		X	X
Ultra SCSI Differential	X	X	X
SSA 8-way JBOD/2-way RAID EL (PCI and MCA)	X	X	X
PCI 3-channel Ultra SCSI RAID	-	-	-
Fibre Channel	X	X	X
Communications and connectivity			
EIA RS232D/EIA RS422A	X	X	X
Token-ring 4/16 Mbps	X	X	X
Ethernet 10/100 Mbps	X	X	X
Gigabit Ethernet	X	X	X
FDDI 100 Mbps	X	X	X
ATM 25 Mbps	-	-	-
ATM 155 Mbps	X	X	X
ISDN	X	X	X
X.25, SDLC and BSC	X	X	X
SP system attachment	X	X	X
3270 connection	-	-	-
ESCON	X	X	X
Block multiplexer	-	-	-
HIPPI 100 Mbps	-	-	-
T1/E1	X	X	X
Graphics accelerators available	GXT120P	GXT120P	

Table 5. Facts and features for SP 332 MHz SMP Thin and SMP Wide Nodes

RS/6000 Model Machine Type	SP System (9076)⁹	
	332 MHz SMP Thin	332 MHz SMP Wide
Processor type	PowerPC 604e	PowerPC 604e
Min/max of each node type per system	1/128 ⁴	1/128 ⁴
Number of processors per node	2 or 4	2 or 4
Clock rates available (standard/option)	332 MHz	332 MHz
System memory per node (standard/maximum)	256 MB/3 GB ¹	256 MB/3 GB ¹
Memory type	64-bit ECC SDRAM ¹	64-bit ECC SDRAM ¹
Data/instruction (L1) cache	32 KB/32 KB ²	32 KB/32 KB ²
Level 2 (L2) cache	256 KB ²	256 KB ²
Memory slots	2	2
Capacity		
Slots available	2 PCI (32-bit)	10 PCI (7 32-bit + 3 64-bit)
PCI bus speed	33 MHz	33/50 MHz
Disk/media bays	2	4
Standard/maximum internal disk	0/36.4 GB	0/72.8 GB
Storage interfaces		
SCSI-2 Fast/Wide SE and SCSI-2 Fast/Wide Differential	X	X
SCSI-2 Fast/Wide RAID-5	-	-
Ultra SCSI SE and Ultra SCSI Differential	X	X
SSA 8-way JBOD/2-way RAID EL (PCI and MCA)	X	X
PCI 3-channel Ultra SCSI RAID	-	-
Fibre Channel	X ³	X ³
Communications and connectivity		
EIA RS232D/EIA RS422A	X	X
Token-ring 4/16 Mbps	X	X
Ethernet 10 Mbps	X	X
Ethernet 10/100 Mbps	X	X
Gigabit Ethernet	X	X
FDDI 100 Mbps	X	X
ATM 25 Mbps	-	-
ATM 155 Mbps	X	X
ISDN	-	-
X.25	X	X
SDLC	X	X
BSC	X	X
SP system attachment	-	-
3270 connection	-	-
ESCON	X	X
Block multiplexer	-	-
HIPPI 100 Mbps	-	-
T1/E1	-	-
Telephony	-	-

Table 6. SP 160 MHz and POWER3 SMP Thin and POWER3 SMP Wide Nodes

<i>RS/6000 model</i>	<i>SP System (9076)</i>		
Machine type	160 MHz Thin	POWER3 SMP Thin	POWER3 SMP Wide
Processor type	POWER2SC	POWER3	POWER3
Min/max of each node type per system	1/128 ⁴	1/128 ⁴	1/128 ⁴
Number of processors per node	1	1 or 2	1 or 2
Clock rates available (standard/option)	160 MHz	200 MHz	200 MHz
System memory per node (standard/maximum)	64 MB/1 GB	256 MB/4 GB ¹	256 MB/4 GB ¹
Memory type	40-bit ECC SIMM	64-bit ECC SDRAM	64-bit ECC SDRAM
Data/instruction (L1) cache	128 KB/32 KB	64 KB/32 KB ²	64 KB/32 KB ²
Level 2 (L2) cache	-	4 MB ²	4 MB ²
Memory slots	4	2	2
Capacity			
Slots available	4 Micro Channel	2 PCI (32-bit)	10 PCI (2 32-bit + 8 64-bit)
PCI bus speed	N/A	33 MHz	33/50 MHz
Disk/media bays	2	2	4
Standard/maximum internal disk	4.5 GB/18.2 GB	0 GB/36.4 GB	0 GB/72.8 GB
Storage interfaces			
SCSI-2 Fast/Wide SE and SCSI-2 Fast/Wide Differential	X	X	X
SCSI-2 Fast/Wide RAID-5	-	-	-
Ultra SCSI SE and Ultra SCSI Differential	-	X	X
SSA 8-way JBOD/2-way RAID EL (PCI and MCA)	X	X	X
PCI 3-channel Ultra SCSI RAID	-	-	-
Fibre Channel	-	X ³	X ³
Communications and connectivity			
EIA RS232D/EIA RS422A	X	X	X
Token-Ring 4/16 Mbps	X	X	X
Ethernet 10 Mbps	X	X	X
Ethernet 10/100 Mbps	X	X	X
Gigabit Ethernet	-	X	X
FDDI 100 Mbps	X	X	X
ATM 25 Mbps	-	-	-
ATM 155 Mbps	X	X	X
ISDN	-	-	-
X.25	X	X	X
SDLC	X	X	X
BSC	X	X	X
SP system attachment	-	-	-
3270 connection	-	-	-
ESCON	X	X	X
Block multiplexer	X	-	-
HIPPI 100 Mbps	X	-	-
T1/E1	-	-	-
Telephony	-	-	-
Notes: 1 shared memory 2 per processor 3 statement of direction 4 up to 512 available in special order cases			

1.5 Workstations and workgroup servers

Workgroup servers covers the following models:

- 7043-140 - Entry-Level Workstation or Entry Workgroup Server
- 7043-150 - Price/Performance Workstation or Entry Workgroup Server
- 7043-260 - High Performance 3D Workstation or 64-Bit Workgroup Server
- 7025-F40 - Expandable Workgroup Server
- 7046-B50 - Rack mounted 2U Server, up to 20 B50s in one rack

Workgroup Servers can be defined as desktide machines that contain a substantial amount of storage in order to support the clients a midsize company or a medium- to large-size department has, depending on the chosen application. All RS/6000 workstations can be equipped with features that allow them to adopt to a workgroup server role.

Note

The 7043 Models 140, 150, and 260 are referred to as 43P machines because they are the follow-on products to the 7248 Model 43P.

1.6 Midrange enterprise servers

Midrange servers are the following models:

- The Desktide Servers:
 - 7025-F50 - Enterprise Server
- The Rack-Mounted Servers:
 - 7026-H50 - Enterprise Server
 - 7026-HA50 - Enterprise Server Solution
 - 7026-H70 - 64-Bit Enterprise Server
 - 7026-HA-H70 - Enterprise Server Solution

1.7 High-end enterprise servers

High-end servers are the following models:

RS/6000 Model S70

- 7013-S70 - Enterprise Server

- 7015-S70 - Enterprise Server
- 7017-S70 - Enterprise Server

RS/6000 Model S70 Advanced

- 7013-S7A - Enterprise Server
- 7015-S7A - Enterprise Server
- 7017-S7A - Enterprise Server

RS/6000 Model S80

- 7013-S80 - Enterprise Server
- 7015-S80 - Enterprise Server
- 7017-S80 - Enterprise Server

1.8 RS/6000 SP systems

This section provides information about the IBM RS/6000 large scale servers. The servers that fall into this category are called the RS/6000 SP (Scalable POWERparallel) systems.

The RS/6000 SP high-performance system uses the power of parallel processing. Designed for performance and scalability, this system makes feasible the processing of applications characterized by large-scale data handling and computational intensity.

Customer uses include: Mission-critical commercial computing solutions to address business intelligence applications, server consolidation, and collaborative computing comprised of Lotus Domino Server, Internet, intranet, extranet, and groupware application solutions. The SP database and computation scalability, critical for business intelligence applications including data warehousing, has led to many installations of more than a terabyte of data.

Recognized in the industry as a high-capacity and reliable Web server, the SP system is an ideal base for e-business applications. Numerous companies and organizations worldwide use it to handle their Web sites. Scientific and technical computing users, including corporations, universities, and research laboratories, use the SP system for leading-edge applications ranging from seismic processing, computational fluid dynamics, engineering analysis/design, and medical simulation.

1.9 PCI storage adapters

The purpose of this section is to provide some *rules of thumb* for the performance of various PCI disk adapters.

Throughput information is provided; the information can be used to approximate throughput with multiple adapters and for other systems. The reason for including storage information here is that the disk performance is essential for overall system performance.

1.9.1 SCSI-2 fast/wide RAID adapter

The SCSI-2 Fast/Wide RAID Adapter implements RAID level 0, 1, and 5 support for SCSI-2 attached disks.

Configuration management, RAID algorithms, and error recovery are handled by an on-board 403 PowerPC. The maximum number of addressable device IDs enabled by the adapter is 15 per bus or 45 devices per adapter.

1.9.2 PCI single-ended Ultra SCSI adapter

The PCI Single-Ended Ultra SCSI Adapter provides a single-ended SCSI-2 Ultra/Wide interface that can burst data between devices on the SCSI bus at 40 MBs (twice the fast/wide rate) using block sizes greater than 64 KB. It conforms to SCSI-2 standards and Fast-20 (Ultra) documentation.

1.9.3 PCI differential Ultra SCSI adapter

The PCI Differential Ultra SCSI Adapter provides a differential SCSI-2 Ultra/Wide interface that can burst data between devices on the SCSI bus at 40 MBs. This adapter conforms to SCSI-2 standards and the Fast-20 (Ultra) documentation.

1.9.4 SCSI-2 fast/wide adapter 4-A

The PCI SCSI-2 Fast/Wide Single Ended Adapter provides a single-ended SCSI-2 Fast/Wide interface. It conforms to SCSI-2 standards and supports fast/wide synchronous data rates of up to 10 MHz.

1.9.5 SCSI-2 differential fast/wide adapter 4-B

The PCI SCSI-2 Fast/Wide Differential Adapter provides a differential SCSI-2 Fast/Wide interface that can burst data between devices on the SCSI bus at 20 MB/s. It conforms to SCSI-2 standards and supports fast/wide synchronous data rates of up to 10 MHz.

1.9.6 PCI 3-Channel Ultra SCSI RAID adapter

The RS/6000 PCI 3-Channel Ultra SCSI RAID Adapter is a non-bootable high performance Ultra SCSI RAID Adapter providing RAID 0,1,or 5 capability and can address up to forty-five 16-bit SCSI-2 physical disk drives on three independent SCSI buses.

1.9.7 SSA RAID 5 adapter, SSA fast-write

The PCI SSA RAID 5 Adapter supports RAID 5 SSA disk arrays and can be used to access non-RAID disks (JBOD) between multiple hosts. It has the capability to improve write response time in the single initiator mode for both RAID and non-RAID disks by the addition of the Fast-Write Cache Option.

1.9.8 Advanced serialRAID, SSA fast-write

The Advanced SerialRAID Adapter is a Serial Storage Architecture (SSA) adapter providing a data transfer rate of up to 160 MBs per loop. The adapter supports Hot Spare drives in RAID 5 mode.

Note

- There is NO SSA boot support on PCI based RS/6000 for any SSA adapter without Open Firmware. This means only the 6225 is bootable.
- The current 6225 will only boot from a JBOD disk. There is no boot support at present for either RAID or Fast-write disks.
- The 6225 should be bootable on any machine which supports Open Firmware boot in its IPL ROS. Check with your IBM support representative for the most current information.

Chapter 2. SAS installation

The installation of SAS on AIX is well described in the documentation delivered with the software package, and the same information is also available online from the installation media. The documentation can be found in the files `Readme`, `alert_notes`, `install_instructions`, and `sys_req`. All files, except the `Readme` file, come in two formats: Postscript (`.ps`) for printout and plain ascii text format (`.txt`). Before you start the installation, some preparations should be made. Read the installation documentation, or look in the file, `sys_req.txt`, and check your installation to see whether the system requirements are fulfilled.

Plan disk space and disk layout; you will need disk space for the SAS executables (SAS code) and SAS data disks. The installation of SAS 6.12 requires about 400 MB for the programs. This disk will not have traffic as heavy as the data disks, and the SAS program can be placed almost anywhere, possibly in your root volume group at the same physical drives as AIX. You will also need space for your SAS data; the disk layout for the data space depends on the usage, but you will need performance to be as high as possible, and this means you want as many fast disk drives as possible. Depending on the typical usage, you can choose different strategies. As shown in the examples in this book, one strategy is to create one volume group for the SAS work space, one for input datasets, and one for output datasets. Before describing disk layout further, let us define some of the terms used in the AIX disk system.

A hierarchy of structures is used to manage AIX fixed-disk storage. Each individual fixed-disk drive, called a physical volume (PV), has a device name, such as `/dev/hdisk0`. The disk connection can make use of different technologies, such as SCSI (I, II, or III) or SSA; however, the name is independent of the physical disk type. The disk numbering starts at 0 and increases consecutively up to the number of disks in the system.

Every physical volume in use belongs to a volume group (VG). All of the physical volumes in a volume group are divided into physical partitions (PPs) of the same size. The PP size depends on the disk size; the larger the disk space in a volume group, the bigger PP size. The PP size is a permanent attribute for a volume group. Once the volume group is created, this size cannot be changed, and a small PP size can limit the total amount of disk space in the volume group. For space-allocation purposes, each physical volume is divided into five regions (`outer_edge`, `inner_edge`, `outer_middle`, `inner_middle`, and `center`). The number of physical partitions in each region varies depending on the total capacity of the disk drive. If the volume group is

created with the -B option in the `mkvg` command, the above limits increase to 128 physical volumes and 512 logical volumes.

Within each volume group, one or more logical volumes (LVs) are defined. Logical volumes are placeholders for information located on physical volumes. Logical volumes typically hold a file system. The usage of a logical volume is similar to a partition in other systems. Data space on logical volumes appears to be contiguous to the user but can be discontinuous on the physical volume. This allows file systems, paging space, and other logical volumes to be resized or relocated, span multiple physical volumes, and have their contents replicated for greater flexibility and availability in the storage of data.

Each logical volume consists of one or more logical partitions (LPs). Each logical partition corresponds to at least one physical partition. If mirroring is specified for the logical volume, additional physical partitions are allocated to store the additional copies of each logical partition. Although the logical partitions are numbered consecutively, the underlying physical partitions are not necessarily consecutive or contiguous, if mirroring is specified, it is recommended to place each copy on separate physical volumes.

A number of attributes must be defined at logical volume creation. Some attributes can be changed later and some cannot. The most used/important are:

- Logical volume NAME - Any name given to the LV
- VOLUME GROUP name - The VG where the LV belongs
- Number of LOGICAL PARTITIONS - The size wanted for this LV
- PHYSICAL VOLUME names - Restrict the use of PVs
- POSITION on physical volume
- Number of COPIES of each LV - One for standard, two for mirrored LV; this determines if an LV is mirrored or not. Notice that the mirroring is on an LV level.
- Stripe Size - If a stripe size is defined, the LV will be striped across all volumes in the VG; here, it is assumed that the VG holds more physical disks.

Logical volumes can serve a number of system purposes, such as paging space or file system; each logical volume that holds ordinary system or user data or programs contains a single journaled file system (JFS). Each JFS consists of a pool of page-size (4KB) blocks. When data is to be written to a file, one or more additional blocks are allocated to that file. These blocks may

or may not be contiguous with one another or with other blocks previously allocated to the file.

After installation, the AIX system has one volume group defined: The rootvg volume group consisting of a base set of logical volumes required to start the system and any other volume groups you specified during the AIX installation.

2.1 SAS preinstallation file system planning

File system configuration has a large effect on overall system performance and is time-consuming to change after installation. Deciding on the number and types of hard disks and the sizes and placements of paging spaces and logical volumes on those hard disks is, therefore, a critical pre-installation process. As mentioned before, the I/O traffic to the SAS programs is not very intensive and can be placed on a LV in rootvg.

SAS data on the other hand, will be accessed heavily, and the placement must be on dedicated disks and VGs. The SAS data can be subdivided into three classes: SAS input data, SAS output data, and SAS work space. The approach used for the tests in this book was to create separate VGs for each type of data. The reason for doing this was to make it easier to track the I/O load, and it gives excellent performance when the system has many physical drives. On the other hand, in a production environment, one will expect multiple users running multiple programs on different data sets, and, in this case, it might be a better strategy to spread the data randomly across the physical drives. That is, create one SAS data VG, and then create one or more LVs for SAS data.

2.2 Installing the base SAS software on AIX

This section provides instructions for installing the SAS System on AIX. Follow the steps outlined in this section, and then proceed as described in the product documentation.

Extracting SAS Manager from the Tape or CD-ROM

You must extract the SAS Manager application from your distribution media before proceeding with the installation. This section explains how to extract the SAS Manager to begin the installation of the SAS System on AIX. You can install the SAS System in any location on the system with sufficient space. The installation requires that all SAS files exist in a directory named sas612, which is created for you and contains all files associated with the SAS

System. SASROOT is the pathname for the location of the SAS files. SASROOT is /sas in this description. It is not necessary to have root privileges to install the SAS System. If the software is not installed as root, the SAS system administrator should install the software using either the administrator's user ID or a user ID created for the SAS System, such as SAS. To complete future installs, the system administrator requires read and write privileges on the SASROOT directory and its contents.

If you are installing the SAS System from an 8 mm tape, you must change the block size parameters on the 8 mm tape drive to 0 as described below. To determine the status of the block size, issue the following command:

```
lsattr -l rmt0 -E
```

Record the block size information that is displayed. If the block size is not set to 0, you must change it for the installation. To change the block size, issue the following command:

```
chdev -l rmt0 -a block_size=0
```

Note

You must have root privilege to issue this command.

You can also use the `smit` command to change these parameters. Use the `smit chgtpe` command.

Note

You must restore the original system tape parameters after you have completed the installation, other system jobs as backup routine might depend on these settings.

The following steps show you how to install SAS. The instructions use /sas/sas612 as the pathname for the SASROOT directory. This pathname is an example; substitute your installation directory:

1. Insert the media into the appropriate drive.

For tape drives, rewind the tape with the `tctl -f /dev/rmtX.1 rewind` command where `x` corresponds to the actual tapedrive.

For CD-ROM drives, if there is no existing CD-ROM file system, create a CD-ROM file system. Use the `smit makcdr` command.

To mount the CD-ROM, use the following command:

`mount /dev/cdX` where `X` corresponds to the actual CD-ROM drive.

2. Change to the directory where you want the SAS System installed by issuing the following command:

```
cd /sas
```

3. Enter one of the following commands to extract the installation programs. When you issue the `tar` command, the SAS System creates or appends to a `./sas612` subdirectory.

For tape media, use:

```
tar xf /dev/rmt0.1
```

For CD-ROM media, use:

```
tar xf /cdrom/sas_inst
```

4. Change directories to the SASROOT subdirectory by issuing a command similar to the following:

```
cd /sas/sas612
```

5. Invoke the SAS Manager by typing `./sasmanager` at the prompt.

Note

- If you use Control C (^C) to exit the installation script, SAS Manager does not retain the information you supplied about media type and location.
- If you use Control C (^C) to exit the installation script, SAS Manager does not retain the information you supplied about media type and location.

6. From the SAS Manager Primary Menu, select Option 1, **Load Software From Media**. You are prompted to indicate your installation media. Select **1** for tape or **2** for CD-ROM.
7. You are prompted for the location of the tape drive or CD-ROM.
If you are installing from a local tape drive or CD-ROM, enter `local`.
If you are installing from tape, you are prompted for a non-rewinding device name. If you are installing from CD-ROM, you are prompted for the pathname of the CD-ROM.
8. You are prompted for the correct pathname for your SETINIT program. The default path should be correct for most installations.

Note

For Non-U.S. Customers Only:

Installations outside of the United States must manually update the SETINIT information with the paper SETINIT included with your installation materials before continuing with the installation.

You are given the chance to view the contents of the installation media. Type `y` at the prompt to view the contents or `n` to continue without viewing the contents.

You are asked if you want to continue the installation. Press **Enter** to continue, or type `n` at the prompt to stop the installation.

If you continue with the installation, the installation process completes the following tasks:

1. Installs the SAS System on the selected file system with adequate space
2. Creates the SAS configuration files `config.sas612` and `autoexec.sas`
3. Applies new SETINIT information.
4. Applies Technical Support fixes supplied with the Usage Notes.
5. Patches the SASROOT directory to the SAS executable.
6. Executes the installation test streams. You will receive messages upon completion of the test as to the validity of the installation.

The installation is now complete except for product-specific configuration. If your installation includes products that need post-installation configuration, the Product Configuration menu appears. Be sure to complete any necessary post-installation configuration that is described in the corresponding product documentation.

Chapter 3. Application test environment

The test environment consists of an RS/6000 system, AIX Version 4.3.2, and base SAS software Version 6.12. In this chapter, the environment will be described in more detail.

3.1 AIX system configuration

The test system used for all tests in this book was an IBM RS/6000 43P model 260. The system was configured with two CPUs and 3 GB of memory.

The disk system is connected with an SCSI subsystem and an SSA disk subsystem; as you can tell from the following configuration listings, there was one SCSI adapter and two SSA adapters. See Figure 2.

```
scsi0 Available 10-60 Wide/Fast-20 SCSI I/O Controller
ssa0 Available 10-70 IBM SSA Enhanced RAID Adapter (14104500)
ssa1 Available 10-78 IBM SSA Enhanced RAID Adapter (14104500)
```

Figure 2. Disk adapters

The System was installed with 18 disk drives: Two connected with SCSI and 16 connected via the SSA adapter subsystem as shown in Figure 3.

```
hdisk0 Available 10-60-00-8,0 16 Bit SCSI Disk Drive
hdisk1 Available 10-60-00-9,0 16 Bit SCSI Disk Drive
hdisk2 Available 10-70-L SSA Logical Disk Drive
hdisk3 Available 10-70-L SSA Logical Disk Drive
hdisk4 Available 10-70-L SSA Logical Disk Drive
hdisk5 Available 10-70-L SSA Logical Disk Drive
hdisk6 Available 10-70-L SSA Logical Disk Drive
hdisk7 Available 10-70-L SSA Logical Disk Drive
hdisk8 Available 10-70-L SSA Logical Disk Drive
hdisk9 Available 10-70-L SSA Logical Disk Drive
hdisk10 Available 10-78-L SSA Logical Disk Drive
hdisk11 Available 10-78-L SSA Logical Disk Drive
hdisk12 Available 10-78-L SSA Logical Disk Drive
hdisk13 Available 10-78-L SSA Logical Disk Drive
hdisk14 Available 10-78-L SSA Logical Disk Drive
hdisk15 Available 10-78-L SSA Logical Disk Drive
hdisk16 Available 10-78-L SSA Logical Disk Drive
hdisk17 Available 10-78-L SSA Logical Disk Drive
```

Figure 3. Disk drives in the test system

The system was configured with three logical paging spaces placed on hdisks 0 and 1. The paging space was placed on hdisks 0 and 1 to minimize the impact on other file systems.

The file systems, /mfs01 and /mfs02, are file systems striped across three drives. SAS workspace is placed in /db/work on a non-striped disk.

The reason for this disk layout is that we wanted to place input (/mfs01), output (/mfs02) and work (/db) datasets on different physical disk volumes because this makes it easier to identify the disk traffic. The file systems, /mfs01 and /mfs02, were placed on striped logical volumes to improve performance because we expect the most data reads or writes to and from these file systems. The paging space is placed in rootvg, and, in a separate volume group, ps, the small paging space, hd6 in Figure 4, is generated by the system in rootvg at installation time and is left there because we wanted paging space on hdisk0 and hdisk1, and the root filesystem at hdisk0 is not busy.

Page Space	Physical Volume	Volume Group	Size	%Used	Active	Auto	Type
paging03	hdisk1	ps	800MB	3	yes	yes	lv
paging02	hdisk1	ps	800MB	3	yes	yes	lv
hd6	hdisk0	rootvg	512MB	5	yes	no	lv

Figure 4. Paging space

The system volume group, rootvg on hdisk0, contains all default file systems.

```

VOLUME GROUP:  rootvg
Physical Volumes:
hdisk0          active      537          0          00..00..00..00..00
rootvg:
LV NAME         TYPE      LPs  PPs  PVs  LV STATE    MOUNT POINT
hd5             boot      1    1    1    closed/syncd  N/A
hd6             paging    64   64   1    open/syncd    N/A
hd8             jfslog    1    1    1    open/syncd    N/A
hd4             jfs       3    3    1    open/syncd    /
hd2             jfs      270  270  1    open/syncd    /usr
hd9var          jfs       10   10   1    open/syncd    /var
hd3             jfs       10   10   1    open/syncd    /tmp
hd1             jfs       55   55   1    open/syncd    /home

```

Figure 5. Volume group rootvg

The file system, /mfs01, is used for SAS programs and input data.

```
VOLUME GROUP:  mfs01vg
Physical Volumes:
PV_NAME        PV STATE    TOTAL PPs   FREE PPs   FREE DISTRIBUTION
hdisk4         active      268         0           00..00..00..00..00
hdisk5         active      268         1           00..00..00..00..01
hdisk6         active      268         1           00..00..00..00..01

Logical volumes:
LV NAME        TYPE        LPs   PPs   PVs  LV STATE    MOUNT POINT
mfs01lv       jfs         801   801   3    open/syncd  /mfs01
mfs01log      jfslog      1     1     1    open/syncd  N/A
```

Figure 6. Volume group mfs01 holding SAS Input dataset

The file system, /mfs02, is used for SAS programs and output data.

```
VOLUME GROUP:  mfs02vg
Physical Volumes:
PV_NAME        PV STATE    TOTAL PPs   FREE PPs   FREE DISTRIBUTION
hdisk7         active      268         4           00..00..00..00..04
hdisk8         active      268         4           00..00..00..00..04
hdisk9         active      268         0           00..00..00..00..00

Logical volumes:
LV NAME        TYPE        LPs   PPs   PVs  LV STATE    MOUNT POINT
mfs02lv       jfs         792   792   3    open/syncd  /mfs02
mfs02log      jfslog      1     1     1    open/syncd  N/A
```

Figure 7. Volume group mfs02 holding SAS output dataset

The file system /db is used for SAS workspace or temporary space. The default installation path for SAS work space is /tmp; however, in a production environment, it is important to create a large high-performance filesystem to hold the SAS workspace because it is heavily used.

```

VOLUME GROUP:  dbvg
Physical Volumes:
PV_NAME        PV STATE    TOTAL PPs   FREE PPs   FREE DISTRIBUTION
hdisk10        active      268         1          00..00..00..00..01
hdisk11        active      268         1          00..00..00..00..01
hdisk12        active      268         2          00..00..00..00..02
hdisk13        active      268         264        54..50..53..53..54

Logical volumes:
LV_NAME        TYPE        LPs   PPs   PVs  LV STATE    MOUNT POINT
dblv           jfs         800   800   3    open/syncd  /db
dblvlog        jfslog      1     1     1    open/syncd  N/A

```

Figure 8. Volume group dbvg holding SAS workspace

The SAS binaries, programs, and libraries are placed in sasvg. The I/O traffic to this volume is small, and /sas could have been placed in another volume group as rootvg. However, the I/O traffic is more easily identified when the file system is isolated on a physical disk drive.

```

VOLUME GROUP:  sasvg
Physical Volumes:
sasvg:
PV_NAME        PV STATE    TOTAL PPs   FREE PPs   FREE DISTRIBUTION
hdisk17        active      537         12         00..00..00..00..12

Logical volumes:
LV_NAME        TYPE        LPs   PPs   PVs  LV STATE    MOUNT POINT
saslv          jfs         524   524   1    open/syncd  /sas
saslvlog       jfslog      1     1     1    open/syncd  N/A

```

Figure 9. Volume group sasvg

3.2 SAS applications

We hope to entertain examples relevant to practices common among the SAS user community on UNIX servers. Because our focus remains on system performance, we will attempt to both document performance capabilities and intentionally stress the AIX system at its boundaries. Having read this text, the reader should be better prepared to both live within the constraints of a fixed environment and adjust techniques for improved performance under extreme conditions.

Usages of SAS vary so widely that it is not possible to define normal use. Still, some functions of SAS are commonly and frequently applied across customer installations. Some applications and tests to consider carefully are:

- Data input - Reading files into SAS data files
- Data processing - Editing, sub-setting, joining, and sorting SAS data files
- Analytic - Computationally heavy uses

Our applications will touch on these three very broad areas. The speed with which the system completes the task will always be constrained by, at least, one of: Available memory, I/O throughput, or CPU speed. That is, your job would finish sooner if one of these components was somehow upgraded.

Improvements in this dynamic system environment ease each constraint in separate, staggered increments. It has become very difficult for the SAS programmer to assess which area is holding performance back.

What intuitively appears to a SAS user as a computationally-intensive, I/O, or memory exhaustive task may surprise some users. In fact, the same program running against the same data on the same system may test one constraint at one moment and another on the next submission. This would most often occur when other users come on-line with tasks demanding a share of the processor(s), physical memory, and paging space. Since the possible combinations of events are infinite, it will be best to write your code and set your parameters for the general case while minimizing the likelihood of incurring catastrophic conditions.

3.2.1 The test environment

SAS Version 6.12 is still the most commonly used production quality environment available to UNIX. We placed SASWORK on a local drive with 13.1 GB of free disk space. Our installation of SAS included most of the SAS software family offerings, but only Base SAS and SAS/STAT components were needed to conduct the tests written up in the this redbook.

3.3 Application data and tests

In order to push the test environment, the test data should be sufficiently large and realistic in content to enable the application of multiple tasks. We chose to use some U.S. Census Bureau files for our data processing examples. We then borrowed from two additional sources to complete our computational problems.

3.3.1 U.S. Census data

The United States Census Bureau makes available its Public Use Microdata Sample (PUMS). This data comes from actual responses to the American

Community Survey, which is conducted to record most participant population and housing characteristics.

This data includes two types of record: A household record with corresponding information and a personal record with details for each known individual within a household. From a 5 percent sample of the 1990 survey, the states of Alaska, Alabama, Arkansas, and California were used. These five flat text files used 650 MB of storage.

These data sets will serve to test the majority of data input and processing aspects of the SAS System.

3.3.1.1 Test 1 - Read Census Bureau data

In this case, the SAS DATA step reads from the five raw Census Bureau data files and creates two SAS data files. The newly-created household records file (HRECS) and the personal records file (PRECS) occupy roughly 170 megabytes and 475 megabytes, respectively.

The resulting HRECS data file is used in Test 2.

3.3.1.2 Test 2 - Sort the household data

PROC SORT is employed to sort the Census Bureau household data file. This common task was expected to be very CPU intensive.

3.3.2 Computational tests and data

We acquired some examples that were previously used to benchmark performance in statistical modeling.

The Boardroom data is a 183,000 record SAS data file with 297 fields used in modeling response to some historical campaign. One field records the campaign response while the rest were considered for their explanatory power. The SAS file measured 221 megabytes.

The Home data set ultimately comprised 40,000 records that was the consolidation of data on Home Price Index to that following unemployment. This data, a 8.1 megabyte file, was used in Test 5.

The drink data set contains survey data on 601 records. This file contained consumer data linking indicated consumer preferences to product and consumer attributes. This 0.6 MB data file was used in Test 6.

3.3.2.1 Test 3 - Linear regression

Using PROC REG, a linear regression model was fit using the Boardroom data. The stepwise selection method was used but was stopped after the first variable was brought into the model.

3.3.2.2 Test 4 - Logistic regression

Using PROC LOGISTIC, a logistic regression model was fit to the Boardroom data. Again, we employed stepwise selection to choose a single variable model. Logistic regression is known to consume more CPU and memory than linear regression models.

3.3.2.3 Test 5 - Survival/Event history analysis

PROC PHREG was chosen to exemplify a more computationally demanding modeling application using the Home data file. Test 5 required PHREG to fit proportional hazard rates using partial likelihood in a rather complex econometric model. The example was said to be a case where performance would increase were SAS given more memory by way of MEMSIZE.

3.3.2.4 Test 6 - Extreme modeling

This case uses actual consumer choice survey data to fit a multivariate model to the Drink data file. PROC GLM was used to fit the multivariate regression. This was chosen as an extreme example of how an unbalanced design might tax memory requirements in SAS.

3.3.3 Test diagnostics

In Chapter 5, “AIX performance tools” on page 51, we begin to compile the results of these tests under a variety of conditions. In that chapter, the focus will be on the SAS System Options.

Our interest will focus on requirements and performance as they relate to I/O and CPU utilization. SAS can produce some useful diagnostics that we will closely follow while we run the tests under an assortment of system settings.

SAS always sends some basic system performance statistics to the SAS log. However, detail on real, user CPU and system CPU times, along with full memory use statistics are only available if the FULLSTIMER System Option is turned on. This can be activated from either the command line or from the Options statement in any SAS program.

Performance statistics are produced both for each PROC or DATA STEP, as well as for the entire SAS process. Figure 10 below shows the summary information for a Test 5 run.

```

NOTE: The SAS System used:
      time:
      real      0:03:50.26
      user cpu  0:03:15.84
      system cpu 0:00:32.03
      block I/O operations:
      input      0
      output     0
      memory:
      page faults 36
      page reclaims 18968
      usage      18.18 M
      context switches:
      voluntary  2
      involuntary 21

```

Figure 10. SAS system diagnostics with FULLSTIMER

These statistics were formatted nicely (*time11.*) by including an undocumented System Option, `-stimefmt <format>`, where the format value may be `z` for *time11.* or `s` for seconds output.

The real, user, and system times and memory-used values will be stored and analyzed to make performance assessments. We will especially look for changes in these values as we vary the available SAS and AIX tuning parameters.

Chapter 4. SAS tuning parameters

When a user executes a SAS job, the goal is, obviously, to finish the execution as quickly as possible. This can be achieved by letting the SAS software take more system resources. The SAS software can, for example, improve performance by using memory for sorting rather than disk. SAS software supports such system tuning with various parameters with which you can force SAS to use more resources, especially memory; however, the performance improvement is most often insignificant since the system will allocate a fair amount of resources to each process. Another much more important reason to avoid changing the default (low) values for the SAS parameters is that, if you start more concurrent instances of SAS with high values for memory sizes, the system can easily run out of physical memory. The insignificant performance improvement then turns into a situation in which the system can't handle all the system requests, and we see a significant performance degradation. This chapter will describe the most important system parameters and show the impact of changing the values or setting the values too high.

Whenever a system is to be optimized and you are considering changing parameters, it should be kept in mind that the most important decisions impacting system performance stem from the SAS programmer. Efficient code and limited reads and writes to disk (especially remote disk) will save resources across the board.

Despite the warnings, the parameter settings can have a substantial effect on execution time. However, settings appropriate for a single-user single-process environment will often prove disastrous as the numbers of simultaneous users and processes increases. Because changes impact the entire system, these decisions are best left to the SAS administrator. It should also be said that the initial option values arriving with the SAS distribution were set for SAS jobs doing query and reporting tasks.

That being said, the System Options may be changed by any user with the submission of any SAS process. Individuals should refrain from changing these options. While some parameters will result in performance gains, some alterations to these parameters may negatively impact individual and collective user performance, sometimes catastrophically. Parameter settings should be overridden only with a full understanding of their implications. This chapter, along with practical experience within a UNIX environment, should provide a SAS administrator or sophisticated user with the necessary level of understanding.

This chapter will offer background information and usage instructions for the SAS System Options, which particularly address the system's Input/Output (I/O) processing, memory allocation, and CPU utilization.

4.1 Base SAS software options

Base SAS software options define the session interaction with system resources, note that each SAS user starts their own SAS session and parameter values may vary from session to session. Discussion of system performance optimization, as affected by the SAS System Options, should be limited to the subset of four options: MEMSIZE, SORTSIZE, BUFSIZE, and BUFNO. A full description of SAS System Options may be found in the *SAS Companion for UNIX Environments: Language*, Version 6, Cary, NC: SAS Institute Inc., 1993, ISBN 1-55544-565-9.

4.1.1 Inclusion of system options

At the start of each SAS session, the session will use the system options in the configuration file(s) if not overruled by command line parameters. The baseline and default configuration file, named *config.sas612* in SAS Version 6.12, will be found within the directory where the SAS System is installed (sasroot). Generally only system and/or SAS administrators will have write permissions in this directory.

On execution, SAS looks first to sasroot, the user's home directory, the local directory, and the command line for gathering pertinent configurations. Those coming later in the above list take precedence. Any user may include a file, such as *config.sas12*, in one's home directory, which would add or change the options defined by the configuration file in sasroot.

All system option settings take the format - sasoption <value>, whether in the configuration file or following the SAS executable at the command line. For a complete explanation of the SAS System Options, see the *SAS Language* manual.

The majority of SAS System Options deal with input and output formats and file locations, but few directly impact physical and virtual memory usage and, thus, system performance. Beyond improvements in application programming, SAS and its user community have identified four options that have been found most relevant in tuning for different system environments. Our plan is to discuss and test these SAS options extensively, but a bit of history should be kept in mind. Many of these options were originally found useful for tuning SAS for use on a mainframe (IBM System 3xx) or on earlier implementations of the UNIX environment. We will pursue, confirm, or rebut

these parameters' continued relevance to the modern RS/6000 family running AIX.

4.1.1.1 MEMSIZE = memory specification

The MEMSIZE option specifies a limit on the total amount of system memory, physical or virtual, the SAS System may use at any moment. The operating system will also have a fixed amount of memory available. It is preferable that not all of it be allocated to supporting SAS applications. Clearly, the system memory defines the upper bound of MEMSIZE. This parameter specifies only the amount of memory available to SAS in any single process. But, in a multi-user environment, all jobs must share a fixed amount of system memory at any point in time.

The sub-statement below gives the syntax following the SAS executable with the values:

```
-MEMSIZE n | nK | nM | nG | MAX
```

where

- n - specifies the amount of memory in bytes
- nK - specifies the amount of memory in kilobytes
- nM - specifies the amount of memory in megabytes
- nG - specifies the amount of memory in gigabytes

The default setting as supplied in config.sas612 for our AIX release is 32 MB.

4.1.1.2 SORTSIZE memory specification

The SORTSIZE option specifies a limit on the total amount of memory the SAS System may allocate to the SORT procedure. This has an upward bound per the MEMSIZE specification.

```
-SORTSIZE n | nK | nM | nG | MAX
```

The syntax for SORTSIZE follows that given for MEMSIZE. The default setting in our installation is 16 MB.

4.1.1.3 BUFSIZE memory specification

The BUFSIZE option is a permanent characteristic of any SAS data set and defines the I/O buffer size to be used when addressing its data. If you choose a BUFSIZE value of 0 or leave it out, the SAS System aims to choose a host system default value that is optimal for the SAS data set. In AIX, this is 4096 bytes.

```
-BUFSIZE n | nK | nM | nG | MAX
```

Here again, the syntax follows the pattern set for MEMSIZE. An increased buffer size can reduce the number of I/O operations required in working with a particular SAS data set, yet this will increase memory utilization. The allowable range spans one kilobyte to two gigabytes.

4.1.1.4 BUFNO value

This number directs SAS as to how many buffers of size BUFSIZE to allocate for any one move between physical and virtual memory. As with BUFSIZE, higher parameter values assigned to BUFNO may also limit I/O operations and, similarly, push memory utilization upwards.

-BUFNO n

The number of buffers of size BUFSIZE to utilize in paging was not explicitly set in config.sas612; it defaults to one.

4.2 Other considerations

In addition to the system options provided by SAS, users may improve performance by incorporating other data management techniques provided by SAS. How data sets are defined and where file systems reside will have an impact. Still, it is hard to recommend a rule of thumb when user requirements vary so widely, both from company to company and from user to user.

4.2.1 Work space

Perhaps, more than any other disk, those associated with SASWORK will see the most read and write activity. SASWORK is a directory to which all temporary SAS data files are directed. Within SASWORK, every SAS process creates a directory (named partially by the process ID). This will hold all files beyond those elsewhere on disk or in memory that SAS needs to execute its tasks. Upon the successful completion of any SAS process, the SASWORK process directory and its contents are deleted.

4.2.1.1 Location and size

The best results will be found if SASWORK is placed on a disk optimized for intense writing activity. It is safest to isolate SASWORK from other system applications to protect both the SAS sessions and the those initiated by another application. The default location setting for SASWORK is /tmp; this is, most likely, an inappropriate choice for a production system and should be changed immediately. If SASWORK were to fill up entirely (and it will on occasion), all users would lose write access to /tmp, and much system functionality would be lost with it.

4.2.1.2 Cleaning up after yourselves

If a SAS session terminates abnormally, the files and directories of the SASWORK directory are not deleted. Because of this SASWORK can become littered with process remains (both large and small files). It is wise to clean this up at regular intervals. You may write your own shell cron script to regularly (nightly) test for SASWORK subdirectories tied to dead processes and, if any are found, to discard the directory and contents. However, the SAS Institute has recognized this need and has provided such a script. The simplest option is then to take this *cleanwork* script, found in \$SASROOT/utilities/bin, and tie it to a cron job. If SASWORK fills, all SAS processes are killed, and more users will be affected.

To avoid the likelihood of SASWORK overflowing under normal circumstances, SASWORK should be a large enough partition to simultaneously handle all users' temporary data files. A recommended amount of space to reserve for SASWORK cannot be given without knowing the exact use of SAS; larger jobs and big data sets will require more workspace. Users with one-off work space requirements greater than SASWORK could consider rethinking their task, or they could be advised to redirect SASWORK to a disk with larger space using the `-saswork <directory>` system option.

4.2.2 Use SAS compression wisely

SAS offers a compression option that differs dramatically from UNIX compression tools. UNIX tools substantially reduce the size of files but are not usable again until they are uncompressed. Thus, great savings are made in file space, but processing costs may add up due to execution of the compression and decompression commands themselves.

The SAS compress option has two driving motivations:

- Reducing the size of SAS data sets
- Reducing the time needed to process data

While users will see file size reductions varying by up to 50 percent (sometimes, they will see a modest increase in size), our focus is on reducing processing time and, thereby, improving system performance. The reduction in storage space may lead to reduced processing time if the applications executing on a particular data set primarily consist of input and output operations. However, if your applications are primarily computational or otherwise field specific, your processing costs will rise because SAS will need to decompress variables in order to operate.

An increase in processing occurs because SAS operates on numeric variables at a full width of 8 bytes and must widen compressed or otherwise reduced-format numeric data to 8 bytes. SAS moves data in 8-byte boundaries, and your AIX system will struggle to properly align numeric or character data. Therefore, the best performance will occur when operating on character fields whose length is maintained in 8-byte multiples. (see Appendix C, “Optimizing systems performance” on page 93).

Like the BUFSIZE system option discussed earlier, compression is a fixed attribute of a SAS data set. One engages (or negates) SAS compression by setting the compress=YES (or =NO) either as a system option or as a data set option.

4.2.2.1 AIX compressed file system

We also note here that with AIX4, AIX will now support a compressed file system. The compression is transparent to the user and SAS. What you gain in reduced I/O and disk space may more than offset the additional CPU required to process the compressed data.

4.2.3 Indexing

A user who routinely joins or merges individual SAS data sets would benefit from utilizing the SAS indexing capabilities. The index is stored in a data structure separate from the data itself and enables direct read access to observations when selections are performed on the values of indexed variables. (Indexes are leveraged with the use of WHERE or BY statements in SAS when followed by indexed variables).

The price of indexing is the processing time required in creating the index along with slight storage and administrative costs for the indexed structure files. The processing costs are similar to those for sorting a data set. Indexing will begin to pay for itself as repeated selections and merges on indexed data sets are made.

4.2.4 Enabling very large file (> 2 GB) access

It is increasingly common to access or create very large files. On a 32-bit operating system, such as AIX, it is natural for a 2 GB (2^{32}) threshold to mark our entry into very large files.

SAS users may have years of practical experience before a log error message states that large files are not supported. A user may not even be aware that users are trying to read, write, or compose a SAS work space file

of such magnitude. Fortunately, getting around this natural restriction is a simple process.

Since Release 6.12, SAS has been able to support native files in excess of 2 GB. This is enabled with the inclusion in the system of the - LARGEFILE sasvlf System Option or user SAS configuration file, or it is passed as a parameter with the SAS executable. The SAS administrator may choose to selectively disseminate this knowledge among the SAS users, if such activity is judged inappropriate for any reason. Users might then be encouraged to partition files manually or take advantage of advances with Release 6.12 that enable the logical grouping of SAS data files that, in the aggregate, exceed 2 GB. Please contact your SAS consultant or reference the appropriate manuals for complete details on file partitioning within SAS.

4.3 Data processing performance test results

Examples and test results will follow. We chose the programs of our test plan for their ability to push on one or more of our three system performance boundaries. We then submitted and resubmitted the test programs under altered System Option and memory settings.

We hoped to highlight cases where a program would execute more efficiently if the SAS parameters were optimally set. This was rarely the case. Aside from some modeling exercises where SAS needs to cache the entire design matrix in memory for it to run at all, programmers will rarely see benefit over the default parameters.

4.3.1 Reading and writing, CPU-bound

Our first test seeks to look closer into the performance of the first task SAS users are likely to encounter: Reading data into SAS. Reading data into SAS and writing data into a SAS data file is, inherently, an input/output process. Though the process was an I/O task, the constraint is not found there.

4.3.1.1 The benchmark

Test 1 was run using the default settings. These settings and the performance output gathered from the SAS program log fullstimer option. When left alone on the system, this reading and writing process is processing at 100% capacity for almost the entirety of the process. This is evident in the lack of a gap between the user time and the real time.

Table 7 on page 38 summarizes our parameter settings and performance results for our initial run. Not only did this job run at full capacity, but it was an essentially memory-free task. Just launching SAS consumes about 1

megabyte of physical memory, and our log below shows a total consumption by SAS of less than 2 MB. We know memory is not a constraint here.

Table 7. Performance benchmark using default settings

memsize	32 MB		
bufsize	24576 KB	real time	568.88 s
bufno	1	user time	542.73 s
file size (HRECS)	169.2 MB	system time	19.54 s
file size (PRECS)	467.0 MB	memory used	1.40 MB

Because the input files fit easily within physical memory, there is no paging to inhibit our job. Nor is I/O contention an issue. If it were, the real time-to-user time gap would have been larger. With less than 20 seconds of system time in a 9 minute task, surely the I/O movements themselves are not diverting much of our processing resource. This is perhaps a best-case scenario, but only a faster processor could lead to a reduction in completion time.

4.3.1.2 Buffer size: getting it right the first time

Whether you allow SAS to set the buffer size or override that decision, the buffer size will remain a fixed attribute of that data set unless it is wiped out and created anew. The effects of any sub-optimal decision will be amplified as the data set's size and shelf life increase. A bad decision here will waste disk space when creating a new data set. The system will then have to supply additional memory and CPU time each time that data is brought into SAS.

Setting the buffer size too low will have many knock-on effects. Minimally, the buffer size should exceed the record's length, or your system will need to access multiple pages to operate on a single record. Moreover, never choose a page size below that used by the operating system (4096 bytes on AIX V4.x), or some space on every page will be wasted. The result will be additional paging along with additional storage and memory. The additional storage needed was evident from the beginning of testing. Table 8 shows the size of the PRECS data set under a variety of buffer size settings.

Table 8. Test 1 SAS data file sizes

Buffer size in kilobytes	1	4	8	24*	64	5242
PRECS size in megabytes	505.8	476.1	472.0	467.0	467.5	472.0
HRECS size in megabytes	209.7	176.6	172.1	169.2	169.4	168.9

The results in Table 8 on page 38 suggest that you could unnecessarily produce a data set larger than the default by up to 15 percent, but, at the

same time, provide little hope of attaining a more compact data structure. SAS employs an algorithm that attempts to optimize CPU and I/O performance with its choice of BUFSIZE. It seeks the smallest multiple of 8 KB capable of holding 80 observations subject to a maximum of 64 KB.

Since the SAS algorithm is proprietary, you cannot know in advance what SAS will select as an optimal buffer size. Nonetheless, this solution delivers data in structures that appeared at or near local minima with regard to both space and time.

These tests considered quite a few alternatives to bufsize and none succeeded in either lowering the data set size or enhancing the speed with which the SAS data set was created. It is unlikely that most sites have time to experiment, and the costs of experimentation grow with the magnitude of your data. The only recommendation we offer is to give the SAS algorithm a chance to perform. Only if this data set sees routine access and performance diagnostics indicate that a change in buffer size is warranted should anybody consider changing this parameter.

4.3.1.3 Protecting your I/O will cost you memory

Do you still think you should be saving I/O operations? Some users expect better performance if fewer and larger memory chunks are employed. We can demand fewer I/O operations, but only by agreeing to an increase in memory usage. But, our RS/6000 carried out its paging and other I/O activity so effortlessly that attempts to simply limit the number of movements between physical memory and external storage will, generally, see slim (if even measurable) decrements in execution time.

Nonetheless, increasing the number of pages to exchange can be done by moving up the SAS BUFNO parameter from its initial value of one. One could multiply this effect by simultaneously increasing BUFSIZE. Table 9 compiles the results of such an exercise on Test 1.

Table 9 should bolster our confidence in the SAS default settings (seen in the second row). Neither in this test, nor in any other, was there evidence suggesting that reduced paging (partially evident in the form of reduced system CPU) delivers any reduction in overall processing time.

Table 9. Buffer versus performance in Test 1

Bufno / no. of cases	Bufsize (KB)	Mean real time (s)	Mean user time (s)	Mean system time (s)	Maximum memory used (MB)
1 / 3	4	574.88	548.31	21.63	1.36

Bufno / no. of cases	BuFSIZE (KB)	Mean real time (s)	Mean user time (s)	Mean system time (s)	Maximum memory used (MB)
1 / 3	24	565.93	544.87	16.17	1.40
1 / 3	64	565.98	546.85	14.20	1.48
1 / 1	512	563.72	545.58	14.02	15.18
30 / 3	4	579.13	550.44	23.43	1.61
30 / 3	24	567.08	546.33	16.20	2.79
30 / 3	64	565.58	546.76	15.10	5.09
30 / 1	5242	565.32	547.46	15.76	29.96

If it is the only active significant process, using a larger buffer should reduce the paging activity and utilize more physical memory. While not a concern on a single-user system, increasing bufno and taking more RAM in the process may push physical memory requirements up against system limits in a loaded system. It also increases the risk of reaching that state where thrashing between physical and virtual memory begins.

The only reward to increasing buffer size seems to be reduced system time use (primarily kernel processes), but this comprises merely 15 to 20 seconds of a 9 minute process. Exaggerated cases employing one and thirty 512 MB buffers saw no significant gains to completion time. However, these jobs utilized significantly more physical memory than all the others combined. Such settings will needlessly reduce the entire system's capacities.

4.3.2 Sorting processes

Sorting a data set is an extremely common event for most SAS users. It is routinely a prerequisite that SAS data files be sorted on variables for BY processing, whether this occur in the DATA step or another SAS PROC. Data files to be sorted will clearly vary in size.

The absolute size of the data set will not be our primary concern. It will help, however, to understand the interdependency of the data set size, the amount of physical memory available on the system, the subset of memory allocated to SAS, and the portion of that allocated to PROC SORT.

Note

In total, SAS will use an amount of memory just a few megabytes above the SORTSIZE setting, if not constrained immediately by MEMSIZE. Remember this, since it is the SORTSIZE parameter's allocation of memory to PROC SORT that drives the results in the following examples.

Scenarios of these possible memory conditions will be discussed in the subsections to follow. In each, Test 2 is carried out under revealing settings of SORTSIZE and available physical memory. Among the relative size scenarios we found interesting were:

- SORTSIZE < 2 x data < physical memory
- SORTSIZE < 1 x data < physical memory < 2 x data
- SORTSIZE < physical memory < data
- 1 x data < SORTSIZE < physical memory
- SORTSIZE <= physical memory < 1 x data

During the sort, SAS is creating and reshuffling the output (sorted) data file within its work space. Until completion of the sort, this data file is maintained in the process' /saswork directory; its size is equal to that of the input data file. The system would prefer immediate access to both of these data sets. Therefore, we can expect significant performance implications when physical memory availability passes through thresholds defined at one and two times the data file's size.

After moving through examples, we will return with an overview of memory use by PROC SORT, and the system that made this example set valuable.

4.3.2.1 Unlimited uncontested physical memory

Any process with essentially free access to RAM would be expected to run at its best. This is the case with PROC SORT when the RS/6000 is able to cache the entire data set into physical memory. In fact, we can cache both the input (unsorted) and output (sorted) data sets.

Furthermore, if no other processes are consuming memory, no processing will be consumed by paging, and no time will be wasted waiting for the return of those I/O operations. The CPU is free to concentrate on the sort.

Table 10. Best case sort for Test 2

Real time(s)	User time(s)	System time(s)	Memory used (MB)
60.92	20.67	21.52	17.47

Notice that system time consumes a far greater share of the real time in PROC SORT than in our previous example due to the large number of changes to the output data set as the sort is processed. Not only that, but there is a sizeable gap between real time and the sum of real and user times. This 20 second bridge suggests that nearly one-third of the real time is in an I/O wait state. This is a gap we should keep an eye on because it will only grow larger when we open up resources for competition. In Chapter 7, “AIX file system, SAS system, and performance” on page 67, AIX tuning for I/O bound processes will be addressed.

Still, not many users would object were their one-half gigabyte file to sort in a single minute as did ours in the default case. This machine has fully three gigabytes of physical memory allowing SAS to simultaneously hold the input data and the output data set as it is sorted. The following tests exemplify the performance declines as various resources are taxed.

4.3.2.2 Uncontested but limited

We now introduce the sort of file that is between one and two times physical memory. Leaving the SAS parameters alone, the RS/6000 requires both physical and virtual memory in handling such a sort.

Rather than searching for a 1.5-plus gigabyte file, we had our system run with a simulated single gigabyte of RAM. Logging on as *root*, this can be done with the `rms -c 1024` command. Now that we are unable to hold input and output data in physical memory, we force the system to page a bit.

Table 11. Test 2 under reduced memory

Real time(s)	User time(s)	System time(s)	Memory used (MB)
92.95	20.22	25.79	17.47

The process's increased reliance on paging and I/O to */saswork* led to a 50 percent increase in real time, while user time and memory requirements saw no effect. The few seconds increase in system time accounts for some of the expansion in real time. However, the increase in time waiting for I/O operations to complete is the key factor.

In the coming examples, watch for the gap between user time and real time to grow as increasingly more time is spent waiting on I/O (in the form of paging) and less time is spent on the actual sort.

4.3.2.3 Memory below input size

In cases where either the input data set is truly large or competing processes reduce available physical memory, not even a single copy of the data can be

held in RAM. This example simulated a system having only 256 MB of physical memory, half the size of the object to be sorted. The same amount of sorting will be required, so user time should be relatively static, but system time and I/O waiting time will rise.

Table 12. *Sorting data file larger than physical memory*

Real time(s)	User time(s)	System time(s)	RAM used(MB)
126.38	20.97	28.89	17.47

In the examples given so far, the memory used has been constant at 17.47 MB. The examples thus far have employed the SAS system defaults of 16 MB of SORTSIZE and 32 MB of MEMSIZE.

These results should show that not having unlimited RAM on the system will not destroy performance. At least in PROC SORT, SAS does not require much physical memory to process a sort. Furthermore, the execution time only doubled in this example where the data set could not be cached into memory from that with unlimited RAM. It would be easy to recommend purchasing twice the amount of physical memory relative to one's largest data set, but it is hardly practical.

4.3.2.4 Giving SAS everything

Having read the SAS System Options materials, one could come away with the expectation that a sort would run fastest should you be able to fit the entire data set within its SORTSIZE parameter. We looked into this by allotting a full 768 MB to SORTSIZE and 1 GB to MEMSIZE. Table 13 shows the processing results. Another case was run with outrageous SAS limits but displayed simply to show that PROC SORT will not use much more physical memory than that needed to contain the data set size.

In all three of the Table 13 cases, SAS is able to cache the entire output data set eliminating the need to read or write to SASWORK entirely.

Table 13. *Unleashing SAS's access to memory*

Sortsize/mem size	Real time(s)	User time(s)	System time(s)	Memory used (MB)
512 MB / 600 MB	60.01	17.06	29.89	484.11
768 MB / 1GB	60.42	17.76	28.12	484.11
1 GB / 2 GB	61.13	17.97	28.15	484.11

The results are equivalent. In fact, allowing SAS enough sorting space through the SORTSIZE parameter led to no reduction in sort time. Provided the data has already been cached into physical memory, SAS takes about one minute to sort our Census data regardless of MEMSIZE and SORTSIZE settings.

However, by moving up the SORTSIZE, SAS will consume more RAM for the sort and leave less for other applications. Allocating more memory (in our case, over 400 MB more) to a job without measurable gains in performance is clearly wasteful and could needlessly put other users' applications at risk.

4.3.2.5 Let AIX have it

Above that which is absolutely required by SAS to perform normal tasks, MEMSIZE should not be increased. AIX is better positioned to properly allocate physical memory. Individual and, more importantly, collective performance will benefit if AIX is allowed to maximize the amount of data kept in memory. With data more likely to be resident in memory, less disk access and paging will be required.

Excessive paging is perhaps our greatest enemy. We moved towards this state by reserving increasingly more memory into SORTSIZE. Table 14 illustrates how performance of a SAS sort can deteriorate on a system with 256 MB of accessible physical memory.

Table 14. Overcommitting memory to SAS in Test 2

Sortsize / memsize (MB)	Real time(s)	User time(s)	System time(s)	Memory used (MB)
16 / 32	126.38	20.97	28.89	17.47
32 / 64	138.79	20.71	29.35	33.73
64 / 128	148.27	20.87	30.63	66.22
128 / 256	799.05	21.30	41.64	131.31
256 / 256	1661.26	22.25	61.56	255.10

As the RAM available to AIX falls, the elapsed time before job completion ballooned. In the last row, not enough memory is left to comfortably run the system; a system that actually had to run an increased number of paging operations. These examples were run in an otherwise process-free environment. If other applications were also calling for memory, these and others would have seen their real times multiply.

AIX utilizes memory much more efficiently than SAS. Handing memory resources over to AIX, versus internalizing memory inside a SAS job allows a greater amount of the data set to sit in memory. This results in a greater proportion of CPU use going to the user, a much better alternative to idling or awaiting I/O. Confirm this with the `vmstat` command.

4.3.3 Sortsize and memory usage

Remaining with the Census sorting example, we now consider testing a multi-user scenario on a system with finite memory. By running four processes simultaneously, each of our two CPUs will share the load. By setting our physical memory below the size of any data set, physical memory and virtual memory will also be economized among the four tasks.

Specifically, we are concerned with memory allocation and paging activity under SORTSIZE settings (on individual tasks) from eight to 64 megabytes. Four runs of Test 2 were run for each value of the SORTSIZE parameter. The results follow.

Table 15. Four simultaneous executions of Test 2 runs with varying SORTSIZE

Sortsize	Mean real time(s)	Mean user time(s)	Mean system time(s)	Total memory used (MB)
8	612.63	24.21	33.83	38.07
12	628.70	23.57	34.53	54.34
16	595.80	23.77	34.13	70.59
20	640.55	23.34	34.39	86.86
24	700.13	23.39	34.47	103.12
28	746.08	23.94	36.00	119.38
32	771.00	23.95	36.81	135.65
36	854.25	24.08	36.71	151.90
40	1248.05	24.19	37.69	168.18
44	1403.33	23.86	39.24	184.46
48	993.55	23.64	38.70	199.76
52	5109.48	25.05	46.84	215.95
56	5662.58	25.37	48.50	233.08
64	6467.90	25.66	51.11	265.64

Again, the SAS default proved best. The minimum real time was observed for the 16 MB SORTSIZE case, although there was little penalty for choosing a still lower value. The amount of sorting was identical in each of the 14 cases; so, we expected and saw similar user times. The system time did not rise much until higher SORTSIZE values cut severely into the system's memory.

Severe performance deterioration was observed initially with the 52 MB SORTSIZE. When asking for up to 200 MB of total memory SAS, we observed a relatively scalar increase in real time. This threshold was pierced with the 52 MB case where real times spiked upward four or five-fold.

At face value, four SAS processes demanding 65 MB of memory each would consume more than 100 percent of the available 256 MB of RAM. Actually, SAS does not differentiate between physical and virtual memory, but the reliance on virtual memory to access SAS program data will dramatically slow down any process. At this point, we are expecting and encouraging the system to begin thrashing about among its paging space. And it does. The average time to completion, at 108 minutes, was ten times that seen under the default settings.

Actually, AIX needs some RAM to efficiently carry out its paging and kernel operations while supporting AIXwindows. Leaving room for I/O in memory is critical. These results indicate that AIX would prefer to keep aside at least 50 MB of physical memory for itself. That is, the sum of memory usage by SAS processes should not be allowed to exceed your system's available RAM by less than 50 MB.

Further gains will result by offering AIX even more memory resources. The test results convincingly show that SAS jobs run optimally when AIX, rather than the SAS program, is given the power to allocate memory among competing processes.

4.3.4 Last thoughts on sorting

PROC SORT will store as much of the input data into SORTSIZE as allowed. That, plus about two megabytes, is as much memory as SAS will seek in any one sorting process. Until it finishes, SAS is also working on another, increasingly sorted, copy of the data set. As physical memory is available, one or both of these copies might be maintained in memory. If held in memory, no paging is required, and the process, or processes, will quickly move along to completion.

Large data sets, limited physical memory, or a busy system will often disallow sorting to take place entirely within memory. Paging is then a necessary

activity. Under normal conditions, an RS/6000 can effectively carry out SAS sort processes with little difficulty. We have shown, however, that difficulties will arrive should SAS processes collectively occupy too much of the physical memory. Performance scales downward as memory is used but plummets as the cumulative amount given to SAS approaches the system's physical memory.

In all but rare cases, we think the worst-case scenarios can be avoided if the SAS community does not engage in altering individual process memory assignments. Our testing has shown that individual sort processes do not benefit from individual reserves of memory by way of MEMSIZE. Instead, they work just as well with the standard allotment.

4.4 Computational test results

SAS's most demanding user community includes those who call on SAS for its statistical analysis capabilities. SAS offers an extensive suite of statistical algorithms. The most complex of these are often those procedures designed for the construction of statistical models. Statistical modeling often involves matrix algebra. While some SAS users are aware of the matrix algebra involved and its computational implications, many are not.

Some SAS procedures will try to fit part or all of the matrices into the memory offered to the SAS program. Part of this section's directive is to determine whether that memory should necessarily be allocated to SAS by way of the MEMSIZE System Option. And if not required, does the amount allocated impact performance times?

In this section, we will examine a couple of computational examples and try to measure performance relative to the SAS System Options that have already been introduced. Most importantly, minimal requirements for the settings will be sought. The section will also offer an attempt to introduce some guidelines for anticipating these requirements. Since we cannot examine every SAS procedure, we recommend you examine indexes with the appropriate SAS manual under *memory requirements*.

4.4.1 PROC REG

Many users receive their practical introduction to simple and multiple linear regression with the SAS PROC REG. Its versatility and frequency of use demand our attention. Fortunately, for the user and the system, the equations needed to fit a regression model have received much attention from computational statisticians. The computations have successfully been

partitioned so finely that very little of the data needs to be in memory at any one time.

The example we chose fit models on 5, 25, and 100 percent samples of a 297 field data set. The results are for the stepwise selection allowing only one step. Allowing more steps and using other selection methods did not affect the memory usage, nor would it impact our conclusions.

Table 16. PROC REG declining memory in Test 3

Size of data (MB)	Memsiz (MB)	Real time(s)	Memory used (MB)
11.0	32	8.13	2.77
55.5	32	39.75	2.76
222.1	32	156.64	2.78

the SAS system has clearly implemented many of the advances in computational statistics allowing the program to concentrate on the mathematics with very limited resource requirements. No matter how much memory we offered it, PROC REG used a constant amount. The increase in execution time also increased linearly with the size of the data set.

4.4.2 PROC LOGISTIC

Like linear regression, logistic regression also receives heavy use. Unfortunately, the computations are not as simple. We shall see this reflected in the amount of time SAS takes to produce results with PROC LOGISTIC on the same data used in the prior section.

Computationally, the matrix algebra required by logistic regression cannot be partitioned to the degree that it is in linear regression. Therefore, the matrices must be carried around in memory. Given enough memory with MEMSIZE setting, SAS will internalize these matrices. Otherwise, they can be kept in the physical or virtual memory space of the RS/6000. The test data indicates that it makes little difference in computing time until SAS returns with the model.

In each case, a logistic regression model was fit using a single step of a stepwise procedure.

Table 17. Logistic regression and Test 4

Size of data (MB)	Memsiz (MB)	Real time(s)	Memory used (MB)	System RAM (MB)
11.0	8	68.30	3.09	3072

Size of data (MB)	Memsizes (MB)	Real time(s)	Memory used (MB)	System RAM (MB)
11.0	16	68.08	3.09	3072
11.0	32	67.49	23.56	3072
222.1	32	1164.39	9.09	3072
222.1	32	1390.67	9.09	256
222.1	32	1182.89	3.11	512
222.1	512	1944.60	425.09	512
222.1	512	1197.69	425.09	3072

In the small 11 MB data set examples, the MEMSIZE setting had no effect on the real time results. In all cases, the memory used was extremely limited unless the MEMSIZE setting allowed space for about two times the data's size.

This turns out to be a frivolous use of memory that can lead to trouble. This trouble can arise even if no other users or processes are competing for resources. Look to the full data set cases in Table 17. Whether we allotted 0.5 or 3 GB of RAM, or a MEMSIZE of 32 or 512 MB, the resulting real times were similar. Only in the case where we allowed SAS to cache the working data file without adequate physical memory (row 7) did the execution time cause concern. With 425 of 512 MB of available memory taken by SAS, the system was forced to excessively page.

PROC LOGISTIC will not benefit by topping off the MEMSIZE parameter. The best results would again be obtained by staying with the defaults.

4.4.3 Taming PHREG

PHREG is the SAS survival analysis procedure, and it offers a more interesting performance curve with respect to MEMSIZE. Our example data set measured 41 MB in size. Memory usage climbs with memory allocation up to the size of the data set. At that point, we see a step down in execution time including a significant fall in system time required.

Table 18. Staggered gains with memory with PHREG

Memsizes (MB)	Real time(s)	User time(s)	System time(s)	Memory used (MB)
4	230.51	196.30	32.52	3.10

Memsize (MB)	Real time(s)	User time(s)	System time(s)	Memory used (MB)
32	230.26	195.84	32.03	18.18
64	195.12	184.34	8.29	39.09
512	192.55	184.04	7.83	41.64

PHREG is different for a couple of reasons. First, an implicit sort is called by the program and, thus, attempts to bring the entire design matrix into the sort. It then takes a bit more for the program and other data. (In our tests, SORTSIZE was set to half the value of MEMSIZE). Next, PHREG, using partial likelihood, essentially fits a logistic regression model for every unique event time. With each fit, the data set is somewhat altered, which requires additional I/O for the modeling data. This may partially explain why PHREG, more than other SAS Procedures, may actually benefit from having a large enough MEMSIZE to hold the design matrix.

This test is run optimally when it is able to bring the data set into the SAS allocated memory space. The SORTSIZE should then be set a bit above the data set size. However, you should watch your memory consumption.

What then is the penalty for setting the memory parameters a bit low? We observed 10 to 15 percent more execution time even in the case where MEMSIZE held a value of 2 MB. It is not a large sacrifice to make considering the potential problems that might unfold should a user choose to be too greedy with SAS memory consumption.

More tests should be run with differing numbers of observations, independent variables, and observed (time) intervals. This would enable a full assessment of the scalability of performance against parameter settings. Few of our test cases exhibited dramatic changes in system time as did our PHREG example. It would be interesting to see how it behaves in other situations. In short, PHREG may warrant increased attention to the SAS System Options settings.

Chapter 5. AIX performance tools

In this chapter, we will describe some AIX tools and methods to determine and monitor work loads. It is necessary to use some tools (either these or others) if you want to optimize the utilization of AIX and the RS/6000 system.

In a given system, when you are running a workload, there will be a bottleneck. The system runs out of a resource, and performance is then limited by this resource. The limitation is most likely to one of the four system resources:

- Physical memory
- Virtual memory (paging space)
- Disk I/O
- CPU

The optimization task is, basically, to find the resource that caused the limitation and either add some more resources or reconfigure the system to use less of the limited resource.

5.1 Performance tools description

Five of the most important and used tools are `vmstat`, `iostat`, `svmon`, `vmtune`, and `rmss`. In the following sections, we will give a short description of each. For a complete description including all parameters, see the *AIX Commands Reference*, Volumes 1 through 6, SBOF-1851. This section describes only a subset of the parameters and the output.

5.1.1 `vmstat` command

The syntax of the `vmstat` command is as follows:

```
vmstat [ Interval [ Count ]
```

The `vmstat` command reports statistics about virtual memory, disks, interrupts, and CPU activity. Output from `vmstat` command gives an overview of system activity. The Interval parameter specifies the amount of time in seconds between each report. The first report contains statistics for the time since system startup and is rarely used. Subsequent reports contain statistics collected during the interval since the previous report. If no Interval parameter is specified, the `vmstat` command generates a single report and then exits. If the Count parameter is specified, its value determines the number of reports

generated and the number of seconds apart. If the Interval parameter is specified without the Count parameter, reports are continuously generated.

Figure 11 show the results from three vmstat samples:

kthr		memory			page				faults			cpu				
r	b	avm	fre	re	pi	po	fr	sr	cy	in	sy	cs	us	sy	id	wa
0	1	46712	2059	0	0	0	0	0	0	1406	232	101	3	3	94	0
0	1	46712	2058	0	0	0	0	0	0	1392	2075	2037	6	3	91	1
0	1	46712	2058	0	0	0	0	0	0	1395	2077	2076	2	4	94	0

Figure 11. Typical output from the vmstat command

Output description:

kthr: kernel thread state changes per second over the sampling interval.

- r - Number of kernel threads placed in run queue
- b - Number of kernel threads placed in wait queue (awaiting resource, awaiting input/output)

Memory: information about the usage of virtual and real memory. Virtual pages are considered active if they have been accessed. A page is 4K bytes.

avm Active 4K virtual pages. This is not physical memory.

fre Size of the free list. The number of pages on the free list is usually small; it is controlled by the vmtune parameters maxfree and minfree. If the number of pages on the free list drops below minfree, some pages will be made free. If a page is marked dirty or used, it will be written to page space on disk before it's made free. Note, this disk transfer appears only in the **po** column for working-segment pages. If the page was a dirty permanent-segment page, that I/O does not appear in the **po** column.

Note

A large portion of real memory is utilized as a cache for file system data.

Page: information about page faults and paging activity. These are averaged over the interval and given in units per second.

- re - Pager input/output list.
- pi - Pages paged in from paging space.
- po - Pages paged out to paging space.

- `fr` - Pages freed (page replacement).

CPU: breakdown of percentage usage of CPU time.

- `us` - User time.
- `sy` - System time.
- `id` - CPU idle time.
- `wa` - CPU cycles to determine that the current process is wait and there is pending disk input/output.

5.1.2 iostat command

The syntax of the `iostat` command is as follows:

```
iostat [ -d | -t ] [ PhysicalVolume ... ] [ Interval [ Count ] ]
```

The `iostat` command reports statistics about virtual memory, disks, interrupts and CPU activity. Output from the `iostat` command gives an overview of system activity. The Interval parameters specify the amount of time in seconds between each report. The first report contains statistics for the time since system startup and is rarely used. Subsequent reports contain statistics collected during the interval since the previous report. If no Interval parameter is specified, the `iostat` command generates a single report and then exits. If the Count parameter is specified, its value determines the number of reports generated and the number of seconds apart. If the Interval parameter is specified without the Count parameter, reports are continuously generated.

Figure 11 show the results from one sample with the `vmstat` command:

tty:	tin	tout	avg-cpu:	% user	% sys	% idle	% iowait
	0.0	6.8		2.7	2.7	94.2	0.4
Disks:	% tm_act	Kbps	tps	Kb_read	Kb_wrtn		
hdisk0	0.5	3.8	0.4	780813	7387862		
hdisk1	0.0	0.0	0.0	0	0		
cd0	0.0	0.1	0.0	320396	0		

Figure 12. Typical output from the `iostat` command

Output Description

- `% user` - Shows the percentage of CPU utilization that occurred while executing at the user level (application).
- `% sys` - Shows the percentage of CPU utilization that occurred while executing at the system level (kernel).

- % idle - Shows the percentage of time that the CPU or CPUs were idle and the system did not have an outstanding disk I/O request.
- % iowait -Shows the percentage of time that the CPU or CPUs were idle during which the system had an outstanding disk I/O request. This value may be slightly inflated (in SMP systems with AIX 4.3.2 and before) if several processors are idling at the same time, an unusual occurrence.
- % tm_act- Indicates the percentage of time the physical disk was active (bandwidth utilization for the drive).
- Kbps - Indicates the amount of data transferred (read or written) to the drive in KB per second.
- tps - Indicates the number of transfers per second that were issued to the physical disk. A transfer is an I/O request to the physical disk. Multiple logical requests can be combined into a single I/O request to the disk. A transfer is of indeterminate size.
- Kb_read - the total number of KB read.
- Kb_wrtn - the total number of KB written.

5.1.3 svmon command

The `svmon` command reports information about the current state of memory. The displayed information does not constitute a true snapshot of memory, because the `svmon` command runs at user level with interrupts enabled.

```
svmon [ -G ] [ -P [ n | s | a ] [ pid1 ... pidN ] ] [ -P [ n | s | a ] { u | p | g | r } [ Count ] ] [ -S sid1 ... sidN ] [ -S { [ n | s | a ] [ u | p | g | r ] } [ Count ] ] [ -D sid1 ... sidN ] [ -i Interval [ NumIntervals ] ] [ -r ]
```

Figure 13 show the output from the `svmon` command:

memory		in use			pin			pg space			
size	inuse	free	pin	work	pers	clnt	work	pers	clnt	size	inuse
98304	94727	1911	4288	45031	49696	0	4288	0	0	196608	10055

Figure 13. Typical output from the `svmon` command

The global report is printed when the `-G` flag is specified. The column headings in a global report are as follows:

- `memory` - Specifies statistics describing the use of real memory, including:
 - `size` - The number of real memory frames (the size of real memory)

Note

This includes any free frames that have been made unusable by the memory sizing tool, the `rmss` command.

- `inuse` - The number of frames containing pages
- `free` - The number of frames free

Note

This includes any free frames that have been made unusable by the memory sizing tool, the `rmss` command.

- `pin` - The number of frames containing pinned pages
- `in use` - Specifies statistics on the subset of real memory in use, including:
 - `work` - Number of frames containing pages from working segments. These are pages holding program data and shared memory segments.
 - `pers` - Number of frames containing pages from persistent segments, persistent storage Segments are used to manipulate files and directories. When a persistent storage segment is accessed, the pages are read and written from its file system.
 - `clnt` - Number of frames containing pages from client segments. These are file buffers for nfs file systems and CD-ROM file systems
- `pin` - Specifies statistics on the subset of real memory containing pinned pages (pages that cannot be paged out to disk) including:
 - `work` - The number of frames containing pinned pages from working segments
 - `pers` - The number of frames containing pinned pages from persistent segments
 - `clnt` - The number of frames containing pinned pages from client segments
- `pg space` - Specifies statistics describing the use of paging space. This data is reported only if the `-r` flag is not used.
 - `size` - The size of paging space
 - `inuse` -The number of paging space pages in use
- `ref` - Specifies the number of real memory frames that have been recently referenced (reference bits are set). This data is reported only if the `-r` flag is used.

- `inuse` - The number of frames in use that have recently been referenced
- `pin` - The number of pinned frames that have recently been referenced

In the example, we have a system with $786432 * 4 \text{ K pages} = 3 \text{ GB}$ of physical memory, of which $567278 * 4 \text{ pages} = 2215 \text{ MB}$ are in use, and the remaining 219154 is free.

The `Inuse` column shows $26492 * 4 \text{ K pages}$ in work segments (program and data) and $540786 * 4 \text{ K pages}$ in persistent segments (file I/O buffers).

$15510 * 4 \text{ K pages}$ are pinned from programs.

The paging space on disk is $540672 \text{ pages} = 2 \text{ GB}$, whereas 20862 pages are in use.

```

Pid: 10506
Command: cp

Segid  Type  Description          Inuse   Pin   Pgspace  Address Range
10004  work  kernel extension    22      22     0  0..24581
64019  work  sreg[10]            0        0     0  0..-1
60018  work  sreg[9]             0        0     0  0..-1
5c017  work  sreg[8]             0        0     0  0..-1
2c00b  work  sreg[7]             508     508     0  0..507
3800e  work  sreg[5]             3200    1526    5501 0..65535
155095 pers  /dev/mfs011v:2073  42260   0        0  0..123495
3400d  work  sreg[1]             6744     0     7820 0..20374
 6701  work  lib data             19        0     0  0..600
54015  work  shared library text 464     0    1275 0..65535
59b36  work  private              75        1     0  0..2 : 65310..65535
 340  pers  code,/dev/hd2:5491  4        0     0  0..3
 0     work  kernel               1448    1193    1137 0..32767 : 32768..65535

```

Figure 14. `svmon -Pa`

In Figure 14, we show the output from the `svmon -Pa <pid>` command while copying a big file. Because we expect a lot of memory, a 42260 page I/O buffer is used as the I/O buffer.

5.1.4 `lsps` command

The `lsps -a` command shows all paging spaces; in the example, we have three paging spaces in the system, placed on `hdisk0` and `hdisk1`.

Page Space	Physical Volume	Volume Group	Size	%Used	Active	Auto	Type
paging03	hdisk1	ps	800MB	4	yes	yes	lv
paging02	hdisk1	ps	800MB	4	yes	yes	lv
hd6	hdisk0	rootvg	512MB	6	yes	no	lv

Figure 15. Typical output from an `lspv -a` command

- Page Space** - This describes the logical volume name.
- Physical Volume** - The name of the physical DASD(s) where the logical volume is placed.
- Volume Group** - This describes the volume group name.
- Size** - This describes the size of the logical volume.
- %Used** - This describes how much of this paging space is actually used now.
- Active** - This field is Yes or No and tells if this paging space is active now.
- Auto** - This field tells if this paging space is activated automatically during boot.
- Type** - This describes the type of the logical volume name, always lv.

The two next tools `vmtune` and `rmss` are not only for monitoring parameter values. You can alter system settings; so, misuse of this command can cause performance degradation or operating-system failure. Before experimenting with `vmtune`, you should be thoroughly familiar with both "Performance Overview of the Virtual Memory Manager (VMM)" and "Tuning VMM Page Replacement" in the *AIX Version 3.2 & 4 Performance Tuning Guide*, SC23-2365.

5.1.5 vmtune command

The `vmtune` command is described in the *AIX Version 3.2 & 4 Performance Tuning Guide*, SC23-2365, and we will only describe the four parameters related to large data I/O traffic here.

- `-p minperm` - Specifies the point below which file pages are protected from the repage algorithm. This value is a percentage of the total real-memory page frames in the system. The specified value must be greater than or equal to 5.
- `-P maxperm` - Specifies the point above which the page stealing algorithm steals only file pages. This value is expressed as a percentage of the total real-memory page frames in the system. The specified value must be greater than or equal to 5.

- `-r minpgahead` - Specifies the number of 4K pages with which sequential read-ahead starts. This value can range from 0 through 4096. It should be a power of 2.
- `-R maxpgahead` - Specifies the maximum number of 4K pages to be read ahead. This value can range from 0 through 4096. It should be a power of 2 and should be greater than or equal to `minpgahead`.

5.1.6 `rmss` command

The `rmss` command is a tool to simulate RS/6000s with different sizes of real memories that are smaller than your actual machine. Moreover, the `rmss` command provides a facility to run an application over a range of memory sizes, `rmss` is designed to help you answer the question: "How many megabytes of real memory does a RS/6000 need to run AIX and a given application?"--or in the multiuser context--"How many users can run this application simultaneously in a machine with *x* megabytes of real memory?" `Rmss` can operate in two ways with a number of parameters. One mode is to issue the `rmss` command with another command as parameter, this will execute the parameter command, but simulating a system with less memory. The second mode is to issue a `rmss` command with one of 3 flags We will only describe the second method and the three flags here. The `rmss` command has no output.

- `-c` flag `Memsiz` - Changes the memory size to the specified value; the `memsiz` value must be smaller than the actual memory size.
- `-p` flag - Displays the current (emulated) memory size, that is, if the effective memory size is reduced with the `-c` flag, `rmss` with the `-p` flag will show the reduced memory size.
- `-r` flag - Resets the memory size to the real installed size.

Chapter 6. Optimizing AIX parameters

In this chapter, we will look at some SAS programs and monitor the usage of system resources with the tools described in the previous chapter.

In real-life optimization, you would use the procedure described in Figure 16 to examine a workload and determine whether it is CPU-bound, I/O-bound, or memory-bound. Then, when the bottleneck is known, we will know how to improve the system and performance.

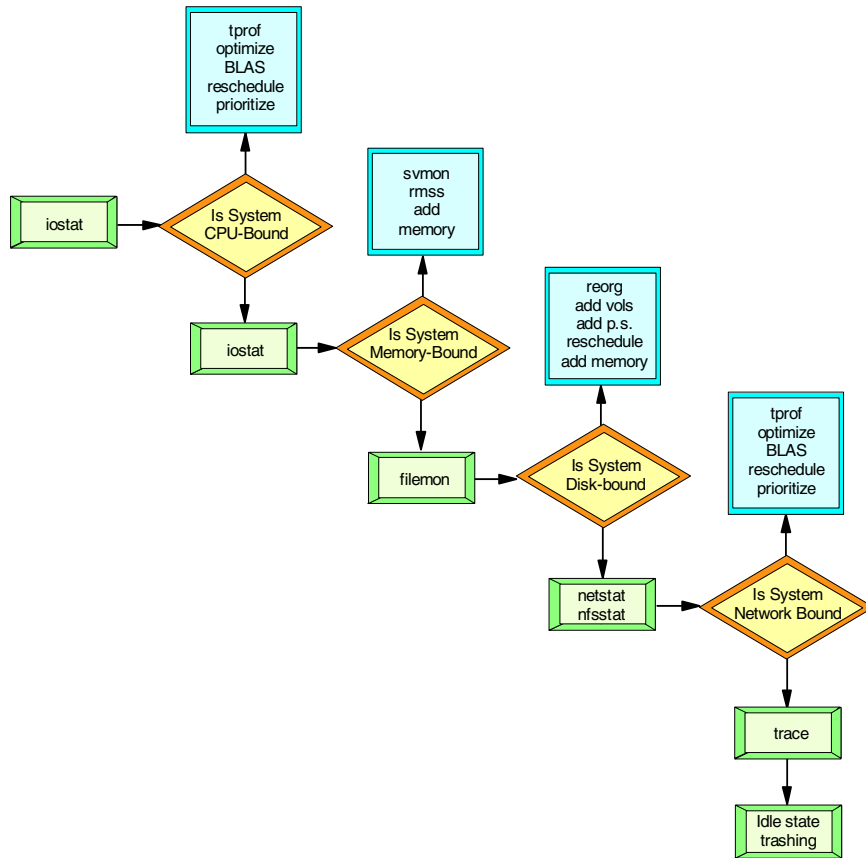


Figure 16. Optimizing procedure

However, in this chapter, we will use some known workloads and check the effect of changing AIX system parameters.

6.1 CPU bound load

The screen shown in Figure 17 is taken from a very CPU-intensive SAS job running alone on a two-CPU F50. In Figure 17, we see the output from a `vmstat` command while the SAS program makes some calculations on two large data sets. The single SAS job takes 50 percent of the two CPUs in the system, that is, it is equal to one CPU. There is no waiting for I/O or paging, and all the CPU time is counted as user time. We can conclude that this job, with this system, is CPU bound since we have no paging activity and no wait for I/O. The reason the SAS program only gets 50 percent CPU and not 100 percent, which would complete the job faster, is that SAS is a single-threaded task, and when the SAS program is started on one CPU, it remains there. To take advantage of our second CPU or any multi-CPU system, one must run more SAS programs in parallel.

kthr		memory				page				faults				cpu			
r	b	avm	fre	re	pi	po	fr	sr	cy	in	sy	cs	us	sy	id	wa	
1	1	76333	650357	0	0	0	0	0	0	206	209	29	50	0	50	0	
1	1	76333	650357	0	0	0	0	0	0	205	173	28	50	0	50	0	
1	1	76333	650357	0	0	0	0	0	0	205	173	29	50	0	50	0	
1	1	76333	650357	0	0	0	0	0	0	205	173	28	50	0	50	0	
1	1	76333	650357	0	0	0	0	0	0	205	179	29	50	0	50	0	
1	1	76333	650357	0	0	0	0	0	0	205	172	28	50	0	50	0	
.	

Figure 17. CPU usage for SAS process

In Figure 18 on page 61, there is a memory status sampled with the `svmon` command for process 12736. The real memory consumption is 3074 4K pages, and the paging space usage is 2620 4K pages.

Pid	Command	Inuse	Pin	Pgspace		
12736	sas	3074	1195	262		
Pid: 12736						
Command: sas						
Segid	Type	Description	Inuse	Pin	Pgspace	Address Range
4c013	work	sreg[0]	5	1	0	0..20798
2f2ab	work	lib data	16	0	0	0..358
54015	work	shared library text	1092	0	1483	0..65535
6829b	mmap	mapped to segment id 0				
1d6a7	work	private	125	1	0	0..96 : 65304..65535
6ca9b	pers	code,/dev/saslv:79908	385	0	0	0..511
0	work	kernel	1451	1193	1137	0..32767 : 32768..65535

Figure 18. Memory usage for SAS process

CPU-bound programs, such as the one above, can neither be optimized by parameter changes in AIX nor, most likely, by SAS parameters; the job is spending all CPU cycles in the user program, and optimization must be done in the program part where the most time is spent. In other words, to optimize a program that is spending all its CPU time in user time, the only way is to change the program code. In general, one can expect a 60/40 ratio between user time and system time. If the user time rises significantly to more than 60 percent, one may consider optimizing the user program, and, if the system time rises to more than 40 percent, one may consider optimizing the system environment, that is, adding more memory disks and a larger paging space.

6.2 Tuning memory bound load

A load is memory-bound if the process or load require a large amount of memory to run or to run fast. As the memory requirements increase, one expects the process to run faster because it has more memory and can make use of it. As the memory requirements increase, the system must free some memory by paging out to paging space on disk, and this slows down the system. The task in a memory-bound system is to find this balance. With SAS, however, we have seen neither a dramatic performance increase nor, even, a moderate performance increase as a function of more memory (see Chapter 5, "AIX performance tools" on page 51). Therefore, we conclude, contrary to what one might expect, that, in general, SAS jobs are not memory-bound. However, if the SAS parameter specifies a large value for the MEMSIZE and/or SORTSIZE parameters, SAS will try to allocate more memory. The performance gain, however, will not match the increase in memory usage. Listings of SAS results running the same job on a 256 MB system are shown in figures 19 and 20 on page 62. In the memory range of

19 MB to 200 MB, there are no significant performance increases, but, as the memory usage increases to all physical memory, the system starts to trash or, in other words, spend all the CPU time in paging.

```
NOTE: The SAS System used:
time:
  real      0:02:01.76
  user cpu  0:00:20.91
  system cpu 0:00:29.14
block I/O operations:
  input     0
  output    0
memory:
  page faults  31944
  page reclaims 24119
  usage        19.100 M
context switches:
  voluntary    2
  involuntary  316
```

Figure 19. SAS job with finite memory usage

We see in the listings that when the memory usage hits 255 MB, the real time for the job is increased from 2 min. 01 to 30 min. 24. The system time increases from 29 seconds to one minute due to increased paging administration. The user time is the time spent in the user part of the program and is constant around 20 seconds. One could expect this to decrease due to the larger virtual memory for the process, but the user time actually increased by 1.88 seconds.

```
time:
  real      0:30:24.57
  user cpu  0:00:22.79
  system cpu 0:01:02.86
block I/O operations:
  input     0
  output    0
memory:
  page faults  189326
  page reclaims 421807
  usage        255.102 M
context switches:
  voluntary    19
  involuntary  596
```

Figure 20. SAS job using more memory than physical available

The goal is to increase the throughput of the system and to get the CPU time spent in the program as low as possible. We find that the real time it takes to execute a SAS program as a function of the memory size is, generally, as shown in Figure 21 on page 63. If you allocate too small an amount of memory the program will be slow because it lacks sufficient internal buffer space; as the SAS program is allowed to increase the memory usage, the runtime is constant or the increase insignificant until we hit the physical memory limit. If we further increase the memory size allocated by SAS, the execution time will increase dramatically because of paging activity and a lack of space for I/O buffers.

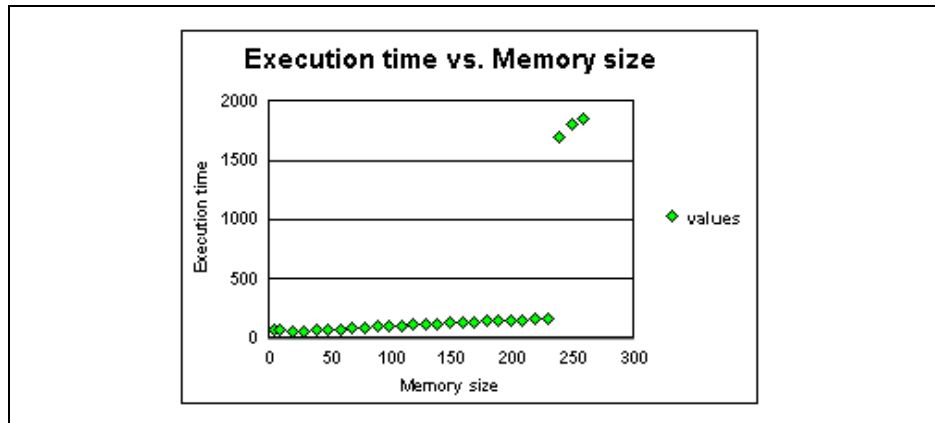


Figure 21. Time to run a SAS program as function of memory

In the tested program, there is no significant performance gain when the SAS parameters, MEMSIZE and SORTSIZE, are increased from 5 MB to 50 MB with input data set size at 500 MB. The recommendation is, therefore, to run all SAS programs with a small amount of memory; we find that a value of 20 MB is good for most programs. The question from the system administrator is then: How do we set restrictions on the memory usage for SAS users?

There are other ways to limit the memory usage; you can:

- Limit each user in the config.sas file in the user's home directory
- Limit all SAS users in the config.sas file in the SAS root directory
- Limit the amount of memory for each process submitted by a SAS user
- Limit the number of concurrent processes for each user

A way to configure the system could be to configure system-wide default values for memory size in the sas.config file in the SAS root directory. The user can then overrule the defaults for each user by adding parameters to command line SAS invocations or by adding parameter values to a local config.sas file. The system administrator can put a system wide limit on the amount of memory used by each SAS user process. This is done by editing the file /etc/security/limits; the format is explained in the file, and the parameters to change are rmss and data. The system administrator can also set a maximum number of processes allowed per user preventing any single user from stealing all the resources. To limit the number of processes per user as the AIX system administrator, type

```
chdev -l sys0 -a maxuproc='<max proc>'
```

The default value is 20, and be aware that this is a system-wide setting and limits all users.

6.3 Tuning disk I/O bound load

In Figure 22, we have captured a vmstat for an I/O bound load. The load is two SAS programs running on a system with 3 GB of memory. Figure 22 shows samples during the execution time, and the blank lines indicate a leap in time. The system is clearly I/O bound because we are waiting for I/O, and there is no paging activity. This system is reading and writing data to disk. From the vmstat table, we can see that SAS is a very well-behaved program. The column fre is the number of free 4K memory pages. The number is dropping while data is read and sorted, but never below the minimum on the free list. The memory pages are used for data buffers and I/O buffers. Figure 23 on page 65 shows the mix at one point in time during execution. If needed, AIX will allow data storage to squeeze the I/O buffer before paging. In Figure 22, SAS finishes the calculations and releases some memory, and the number of free pages increases. Now, when there are free pages, AIX uses them for I/O buffers, and the number of free pages drops again.

kthr		memory				page				faults				cpu			
r	b	avm	fre	re	pi	po	fr	sr	cy	in	sy	cs	us	sy	id	wa	
0	0	112941	625418	0	1	1	18	665	0	109	175	29	2	1	93	4	
1	2	140169	595275	0	0	0	0	0	0	536	725	102	15	24	10	51	
0	2	374913	110728	0	0	0	0	0	0	339	244	38	1	5	0	94	
0	3	374913	99525	0	0	0	0	0	0	331	305	38	3	8	0	88	
0	2	374913	87224	0	0	0	0	0	0	333	314	38	4	11	0	85	
0	3	374913	75947	0	0	0	0	0	0	341	319	40	4	8	0	88	
0	3	374913	75947	0	0	0	0	0	0	336	179	37	0	1	0	99	
0	2	374913	64258	0	0	0	0	0	0	321	332	40	3	9	0	88	
0	3	374913	64258	0	0	0	0	0	0	324	249	49	0	1	0	99	
0	3	374913	53707	0	0	0	0	0	0	312	372	49	4	7	0	89	
0	3	374913	53707	0	0	0	0	0	0	320	190	36	0	1	0	99	
1	1	62871	355525	0	0	0	0	0	0	359	694	32	6	62	0	32	
0	2	62871	340542	0	0	0	0	0	0	591	857	34	6	17	0	77	
0	3	62871	239290	0	0	0	0	0	0	513	1003	38	7	18	0	75	
0	3	62871	223766	0	0	0	0	0	0	546	889	38	6	16	0	78	
0	2	62871	165734	0	0	0	0	0	0	511	1310	42	7	24	0	69	
0	3	62871	153782	0	0	0	0	0	0	525	761	41	4	14	0	83	
1	2	62679	258039	0	0	0	0	0	0	486	973	36	6	33	0	61	
0	2	62679	379662	0	0	0	0	0	0	410	311	40	0	19	0	80	
1	1	61699	608991	0	0	0	0	0	0	277	262	55	0	27	28	45	
0	1	61699	608991	0	0	0	0	0	0	207	346	42	0	0	99	0	

Figure 22. Free pages during program execution

Figure 23 shows output from the `svmon` command during program execution.

```

memory                i n u s e
size  inuse  free   pin   work  pers  clnt
786432  571301  215131  15805  322510  238791  0
.

```

Figure 23. Memory usage during program run

From Figure 22, we can conclude that SAS is a well-behaved program that uses the system efficiently and releases resources as soon as they are not needed anymore; this can be seen by the fact that the memory on the free list is increasing before the program actually ends. This means that SAS is giving up memory as soon as it is no longer needed and before the SAS program is terminated.

The next question is: How do we optimize SAS usage of the system? Table 19 shows average execution times for different values of MEMSIZE and SORTSIZE = MEMSIZE * 0.5, all other parameters being constant. It is clear that larger memory buffers do not improve performance. The Real time and system time increase while the user time is constant. The explanation is that the user program is fixed and has a given execution time, but, because AIX has to handle I/O in smaller buffers, the system spends more time waiting and moving data around.

Table 19. CPU time as function of MEMSIZE parameter

Memory size	Real time	User time	Sys time
32 MB	145	21	24
256 MB	210	21	35
600 MB	333	21	40

As we see from Table 19, we do not get performance increments with larger buffers. Therefore, we recommend that you keep default values for buffer sizes around 32 MB. In the next chapter, we will examine RS/6000 and AIX to investigate how the file system can be improved.

Chapter 7. AIX file system, SAS system, and performance

In this chapter, we will look at performance for various disk and file system configurations. To test the performance, we are using our SAS sort program; we are running the same SAS program with the same parameter settings in all tests. The SAS program and command line parameters we are using are:

```
sas -memsize 32m -sortsize 16m -stimefmt z -sysparm 2 -log <LOGFILE>
sort.sas
```

`sort.sas` is a SAS sort program, which takes an input dataset, sorts it, and writes the output to a SAS output data set. Before we look at file system configuration, we want to find out which ones are busy and which are not. The `iostat` command gives information about disk usage during program execution, and, with the information, we can determine how much each fileset is used. The disk partitions and their usage are described in Figures 4 through 9 on pages 24 through 26.

Table 20 lists the number of busy intervals during program execution; we see that the SAS workspace is the most used file system, and the SAS output file system is the next most used. The disk system is described in detail in Chapter 3, “Application test environment” on page 23.

We will, therefore, use these two most used file systems to test and tune file system performance, since changes to these file systems will have the most impact on execution time. The test system is a two-CPU RS/6000, and SAS is a single threaded program; this means that one SAS program will run and use only one CPU, and, to put a load on both CPUs, we must execute two concurrent SAS instances during each test.

Table 20. .Table 20File system usage

File system and disk	Average usage number of busy five-second intervals
Page space disk 0 and disk 1	7
Input dataset disk4 - disk6	16
Output dataset disk7 - disk 9	17
SAS workspace disk10 - disk13	29
Sas programs disk17	3
Free disks disk14 disk15 disk16	0

7.1 Testing performance of SAS workspace

Figure 24 on page 69 lists the execution time for five various disk configurations for the SAS workspace file system. For each configuration, we ran two tests with two instances at a time in all four program runs with each disk configuration. As one might expect, the execution time is decreasing with more disk drives. The most significant drop in execution time is from four to five disks in the logical volume; more disks barely improve the performance. This is because the time spent waiting for I/O to the SAS workspace is decreasing with more disk drives, and so is the performance gain. The test with seven disks is misleading; the performance improves with more disks even though the improvement from six to seven disks is so small that the uncertainty in the time measurement make it look different. The example shows a slight decline in performance.

In conclusion, when configuring an AIX system for SAS, a large number of disk drives improves the performance. A small RS/6000 two-CPU system, such as the one tested, can utilize five to seven disks in each file system. The most important ones are the SAS workspace, the SAS output file system, and the SAS input file system.

Figure 24 on page 69 shows execution times for various SAS workspace configurations.

```

- 3 disks in logical volume SAS workspace
real 215.76 user cpu 21.37 system cpu 25.77
real 221.45 user cpu 20.86 system cpu 26.11
real 225.01 user cpu 21.52 system cpu 26.26
real 225.04 user cpu 21.27 system cpu 25.42
AVG. 221.82          21.26          25.89

- 4 disks in logical volume SAS workspace
real 221.30 user cpu 20.63 system cpu 26.61
real 221.45 user cpu 20.76 system cpu 26.57
real 221.68 user cpu 21.42 system cpu 26.34
real 222.11 user cpu 21.34 system cpu 26.52
AVG. 221.64          21.04          26.51

- 5 disks in logical volume SAS workspace
real 131.24 user cpu 20.90 system cpu 24.25
real 131.27 user cpu 20.75 system cpu 23.67
real 121.39 user cpu 20.66 system cpu 24.95
real 121.42 user cpu 20.62 system cpu 23.73
AVG. 126.33          20.73          24.15

- 6 disks in logical volume SAS workspace
real 122.67 user cpu 21.56 system cpu 25.02
real 122.63 user cpu 21.30 system cpu 23.71
real 123.63 user cpu 21.16 system cpu 25.34
real 111.40 user cpu 21.64 system cpu 23.39
AVG. 120.08          21.41          24.37

- 7 disks in logical volume SAS workspace
real 125.73 user cpu 20.39 system cpu 24.66
real 125.73 user cpu 20.39 system cpu 24.66
real 117.59 user cpu 20.45 system cpu 24.71
real 117.53 user cpu 20.99 system cpu 25.10
AVG. 121.64          20.55          24.78

```

Figure 24. Execution times for various SAS workspace configurations

In Figure 25 on page 70, we see the impact of adding the spare disks to both the SAS workspace and the SAS output file system. We are testing two configurations: Four + five disks and five + five disks in an output file system and SAS work space. We also see that the SAS output file system benefits from having more disk drives because the average execution time decreases from 133.25 Sec. to 103.10 Sec.

In conclusion, all the SAS file systems (the file system where the input dataset is placed, the fileset where the output dataset is placed, and the fileset where the SAS workspace is placed) benefit from having more disk drives. Therefore, distribute all drives among the file systems with a focus on the SAS workspace.

```

Output filesystem = 4 disks SAS work = 5 disks
real 141.64 user cpu 20.67 system cpu 24.50
real 140.77 user cpu 20.54 system cpu 23.92
real 111.98 user cpu 20.23 system cpu 23.30
real 111.11 user cpu 20.59 system cpu 24.47
real 147.01 user cpu 20.45 system cpu 24.39
real 147.00 user cpu 20.49 system cpu 24.06
AVG. 133.25 20.50 24.11

Output filesystem = 5 disks SAS work = 5 disks
real 148.04 user cpu 20.49 system cpu 23.92
real 148.88 user cpu 20.75 system cpu 24.36
real 63.92 user cpu 20.11 system cpu 24.20
real 76.79 user cpu 21.09 system cpu 23.27
real 85.98 user cpu 20.93 system cpu 22.93
real 95.01 user cpu 20.32 system cpu 23.98
AVG. 103.10 20.62 23.78

```

Figure 25. Non striped disk configuration

7.2 Testing disk striping

The next test, shown in Figure 26 on page 71, is the same SAS test program and two concurrent instances, but, now, the SAS workspace is configured as a disk stripe with a disk stripe size of 8 KB. The rule for ideal disk stripe size is: the number of disks in a stripe * stripe size, and it should be equal to or a multiple of BUFSIZE * BUFNO. With a stripe size of 8KB, this rule is observed, and the load will be evenly distributed among the disk drives.

Surprisingly, the performance does not improve with disk striping; however, it is better the more disks we have in the stripe. The best runtime achieved was 154 seconds, and it was more stable around 190 seconds; however, we had times of around 120-130 seconds with non-striped disks. The reason for this is that the disk access is a high number of read and writes of small randomly-placed disk blocks rather than large sequential disk access.

In conclusion, the SAS workspace should not be configured as a striped file system. This may not be the case for the input and output datasets; these file systems are accessed more sequentially and may benefit from striped file systems.

```

Volumegroup with 5,6 or 7 disks in 8k disk stripe

-5 disk in disk stripe
real 218.49 user cpu 21.31 system cpu 23.45
real 218.44 user cpu 21.62 system cpu 24.56
real 216.88 user cpu 21.00 system cpu 23.85
real 217.73 user cpu 21.68 system cpu 24.15
real 223.12 user cpu 21.97 system cpu 23.34
real 221.58 user cpu 22.07 system cpu 22.53
AVG. 219.37 21.61 23.65

-6 disk in disk stripe
real 154.23 user cpu 21.19 system cpu 23.52
real 177.75 user cpu 22.01 system cpu 23.17
real 200.90 user cpu 21.39 system cpu 24.19
real 200.03 user cpu 21.71 system cpu 23.54
real 192.22 user cpu 21.86 system cpu 24.45
real 192.18 user cpu 21.40 system cpu 25.00
AVG. 186.22 21.59 23.98

-7 disk in disk stripe
real 182.41 user cpu 20.99 system cpu 24.46
real 183.32 user cpu 20.79 system cpu 24.70
real 181.01 user cpu 21.04 system cpu 24.99
real 181.02 user cpu 20.59 system cpu 24.48
real 182.62 user cpu 20.89 system cpu 25.93
real 182.76 user cpu 20.59 system cpu 24.50
AVG. 182.19 20.82 24.84

```

Figure 26. Striped disk configuration

In the next test cases, we will examine the effect of enabling the hardware write cache on the disk adapter and, at the same time, increase the queue depth from 3 to 5. The queue depth is the number of disk I/O requests the operating system can write ahead to the disk controller. In Figure 28 on page 74, we have the average execution time with the write cache enabled and queue depth equal to 5, and, in Figure 27 on page 73, we have the numbers with write cache disabled and queue depth equal to 3. A set of three tests are performed with a different number of disks in the file system containing the output dataset each time. The three tests in the test set are with three, four, and five disks in the file system hosting the SAS work space. Finally, three tests are performed with five disks in the file system holding the SAS output dataset and the SAS workspace placed on a striped file system, also with five disks. These 12 tests are made with the write cache enabled and disabled. All tests are repeated 10 times, and a complete list of test results is presented in Appendix A, “The SAS System on an IBM RS/6000 SP” on page 85. This chapter only lists the average values.

The test set had the following configurations:

- Output data set= hdisk6 hdisk7 hdisk8
 1. SAS work= hdisk9 hdisk10 hdisk11
 2. SAS work= hdisk9 hdisk10 hdisk11 hdisk12
 3. SAS work= hdisk9 hdisk10 hdisk11 hdisk12 hdisk13
- Output data set= hdisk6 hdisk7 hdisk8 hdisk14
 4. SAS work= hdisk9 hdisk10 hdisk11
 5. SAS work= hdisk9 hdisk10 hdisk11 hdisk12
 6. SAS work= hdisk9 hdisk10 hdisk11 hdisk12 hdisk13
- Output data set= hdisk6 hdisk7 hdisk8 hdisk14 hdisk15
 7. SAS work= hdisk9 hdisk10 hdisk11
 8. SAS work= hdisk9 hdisk10 hdisk11 hdisk12
 9. SAS work= hdisk9 hdisk10 hdisk11 hdisk12 hdisk13
- Output data set= hdisk6 hdisk7 hdisk8 hdisk14 hdisk15
 10. SAS work on striped disk hdisk9 hdisk10 hdisk11
 11. SAS work on striped disk= hdisk9 hdisk10 hdisk11 hdisk12
 12. SAS work on striped disk= hdisk9 hdisk10 hdisk11 hdisk12 hdisk13

```

Without Cache 1
AVG. real 154.43 user cpu 20.72 system cpu 24.48

Without Cache 2
AVG. real 143.83 user cpu 20.67 system cpu 24.35

Without Cache 3
AVG. real 125.47 user cpu 20.72 system cpu 23.68

Without Cache 4
AVG. real 145.55 user cpu 20.71 system cpu 23.91

Without Cache 5
AVG. real 131.06 user cpu 20.73 system cpu 24.07

Without Cache 6
AVG. real 142.42 user cpu 20.65 system cpu 23.92

Without Cache 7
AVG. real 154.29 user cpu 20.73 system cpu 23.93

Without Cache 8
AVG. real 137.27 user cpu 20.55 system cpu 24.00

Without Cache 9
AVG. real 114.18 user cpu 20.76 system cpu 23.58

Without Cache 10
AVG. real 250.72 user cpu 20.82 system cpu 23.34

Without Cache 11
AVG. real 210.52 user cpu 20.67 system cpu 23.78

Without Cache 12
AVG. real 197.05 user cpu 20.70 system cpu 23.60

```

Figure 27. Average execution time without write cache

```

With writecache1
AVG. real 166.45 user cpu 20.63 system cpu 23.81

With writecache2
AVG. real 160.68 user cpu 20.56 system cpu 23.61

With writecache3
AVG. real 153.93 user cpu 20.58 system cpu 23.97

With writecache4
AVG. real 164.26 user cpu 20.75 system cpu 23.70

With writecache5
AVG. real 154.93 user cpu 20.57 system cpu 23.81

With writecache6
AVG. real 138.25 user cpu 20.62 system cpu 23.67

With writecache7
AVG. real 165.20 user cpu 20.72 system cpu 23.78

With writecache8
AVG. real 173.36 user cpu 20.60 system cpu 24.55

With writecache9
AVG. real 140.47 user cpu 20.71 system cpu 23.66

With writecache10
AVG. real 211.78 user cpu 20.91 system cpu 23.64

With writecache11
AVG. real 190.97 user cpu 20.80 system cpu 23.73

With writecache12
AVG. real 188.29 user cpu 20.99 system cpu 23.55

```

Figure 28. Average execution time with write cache enabled

Figures 27 and 28 list the execution times for the twelve tests with the cache disabled and the twelve tests with the cache enabled. The tables hold the values for real time, user time, and system time. System time is the amount of time the system spent in OS service routines for a process. User time is the amount of time the system spends in the user program for a process. And the real time is the time it takes to complete the process. All tests are the same SAS job with the same SAS parameters. We expect the program steps required to be the same for all test runs, and, accordingly, the user and system time are constant for all test runs; only the real time differs when the disk cache configuration is changed. The values are consolidated in the following two figures.

Write cache		DISABLE	ENABLE	Relative perf.
Test 1	AVG. real	154.43	166.45	93
Test 2	AVG. real	143.83	160.68	90
Test 3	AVG. real	125.47	153.93	82
Test 4	AVG. real	145.55	164.26	89
Test 5	AVG. real	131.06	154.93	85
Test 6	AVG. real	142.42	138.25	103
Test 7	AVG. real	154.29	165.20	93
Test 8	AVG. real	137.27	173.36	79
Test 9	AVG. real	114.18	140.47	81

Figure 29. Non-striped test runs with relative performance change

In Figure 29, we have the real time values for the non-striped configuration and the relative improvement of enabling the write cache over disabled write cache. In all test cases (except test 6) the performance is decreased when write cache is enabled; the value in test 6 is due to two fast test runs with the cache enabled. There is some uncertainty in the numbers because no two test runs are exactly the same. Data may be placed differently on disks from run to run; interrupts may occur, and other parameters may influence the real time. It seems as if the write cache decreases the performance for non-striped disks. The reason for this is the way AIX and the JFS file system work. When a program writes to disk, it can be done in two ways: Either forced to disk immediately or write to I/O buffer. In a multiuser environment, it is good behavior to write to the I/O buffer rather than the hard drive, and SAS is a very well-behaved program.

Write cache		DISABLE	ENABLE	Relative perf.
Test 10	AVG. real	250.72	211.78	118
Test 11	AVG. real	210.52	190.97	110
Test 12	AVG. real	197.05	188.29	105

Figure 30. Striped test runs with relative performance change

In Figure 30, we have the real time values for the striped configuration and the relative improvement of enabling the write cache over disabled write cache. In all test cases, the performance is improved when write cache is enabled. The reason the write cache improves performance with striped disks and not with unstriped disks is the different physical disk layout that exists when running striped disk configuration, and the real time values are still higher than the non-striped ones without write cache.

In conclusion, SAS uses the system in an efficient way, utilizes I/O buffers, and avoids forced I/O to disk write. Therefore, it is not recommended that you make use of a disk write cache.

Chapter 8. The user community and performance considerations

This chapter hopes to raise some issues that require attention when you are considering an installation and administration strategy. Before configuring the system and setting default SAS or AIX parameters, it is imperative to have an understanding of the user community, their requirements, and even their motivations for using SAS.

Most systems and individuals will suffer in an unregulated application environment. The SAS user, like any other, benefits from the invisible hand of the system administrator providing some guidance and education. The following sections ought to help outline system administration and user education requirements based on a thorough understanding of the SAS community.

8.1 Community assessment

Preparation for a SAS installation begins with an assessment of the user community and their requirements. For each of the following subsections, you should produce a relevant summary of the anticipated usage by the SAS community. These profiles will be vital in creating user and functional requirements documentation. These, in turn, will drive the technical (hardware, software, and networking) requirements pertinent to a SAS/AIX environment.

8.1.1 General usage

Size the SAS community.

- Simply knowing how many individuals require SAS access goes a long way toward anticipating system demand. There is no need to grant every AIX user SAS privileges. Ultimately, it will prove simplest to create a *sasuser* group. User privileges can be managed with AIX group administration tools.

Compare the audience to the overall AIX community.

- The relative share of SAS users of all AIX accounts should help in guiding an allocation strategy of disk, processor, and memory resources.

8.1.2 User profiles

Demands for system resources and administrator support are linked not only to the SAS workload but also to the skills present in the user community. Continue the community assessment by profiling users in the following areas:

SAS skill set

- SAS has very broad capabilities that interact with the abilities of its users. Classifying users into, say, novice, intermediate, and expert categories based on experience and self-reported ability can help manage expectations.

UNIX skill set

- Users arrive with equally divergent AIX skills. For example, those with little comfort on the platform may maximize time spent on a PC and prefer an interactive session when on AIX. In contrast, those with extensive AIX backgrounds will undoubtedly spend more time on the server and be more likely to submit multiple processes.

Estimate the time to be spent on the SAS application.

- Poll your incoming SAS community on their expected usage needs to supplement historical patterns, where available. Daily, weekly, or intense but sporadic usage are possible groupings.

8.1.3 Expected SAS application

Why use SAS? Understanding the basic motivations behind SAS usage is critical in forecasting and meeting user needs with minimal system impact. Obtain estimates on how the SAS users will distribute their application work among the following three broad functional categories:

Data processing

- The reading, writing, and manipulation of data files should be limited as much as possible. Users should be encouraged to share multipurpose SAS files wherever possible. Even though, in Chapter 5, “AIX performance tools” on page 51, that data processing need not be memory-intensive, the manipulation of very large files will benefit from physical memory.

Modeling/data mining

- Modeling tends to promote interactive and intense SAS use. Given more time or higher processing speed, these users will seek to maximize work. Any system improvements could be partly offset as users increase model complexity and the number of considered solutions.

Reporting

- This can be the least demanding in terms of system resources. However, if some users only use SAS for reporting purposes, their

skills may be limited or rusty which might lead to inefficient coding and unnecessary repetition.

8.1.4 Access and interface of choice

Users will access SAS in one of two ways:

- Indirectly, using PC SAS and SAS/CONNECT, or
- Directly, using the AIX command line

Execution will either be in

- Interactive mode (GUI, PC, or AIX)
- Background or foreground (AIX command line only)

A user's comfort with the operating system, file editors, and preference for a graphical user interface (GUI) will drive both their access point decision and their interface of choice.

Very different usage patterns will become evident among the groups. It is possible, however, for a user to use both channels. The indirect users may rely more on their PC's SAS license (and processor). Server use might then be limited to larger jobs or to accessing server files. At the same time, these users may use other resources in copying and downloading SAS files to their local disk.

The command line users generally have better AIX skills and will likely spend a disproportionate amount of time on the server's processor, whether or not SAS is installed on their PC. With the command line comes increased flexibility in job control. It is easier for these users to submit simultaneous (background) jobs and demand more system resources.

8.1.5 Disk access

Administrators are expected to know which disks are locally or remotely attached. Users are generally not expected to know this, especially when multiple hosts are available. They cannot be assumed to know the costs of processing data remotely or the benefits of processing locally. Everyone will benefit if the users are educated about the differences and encouraged to:

- Act locally

Reading and writing speed will always be best when disks are mounted locally. Speed is further increased if the disk being read from and the one being written to are not the same.

- Limit remote disk I/O

Clearly, larger processes or projects will benefit most from improvements in I/O speed. Depending on the task, benefits might accrue if the user observes an appropriate strategy:

- Use the server local to the disk. In a multiple host or node environment, the user may be able to simply log in to an appropriate host and submit SAS locally from there.
- Read from remote disk once and write locally. Limiting the number of times the a remote file is read will dramatically cut processing time. The problem need not be compounded by writing remotely as well. A user may need to read from a remote disk but should be able to work and write out SAS data files within locally. Use separate libnames.
- Read once, write once. If the end result must be delivered to the remote disk and the project is large, a user should consider bringing (necessary parts of) the file over to the local disk. Once localized, run the data processing, reporting or modeling tasks locally before returning the resulting data files back to the remote disk.

8.1.6 Data sources

The format and channel from which SAS users retrieve their data sources should be considered. In all cases, a user should attempt to limit the number of fields and records to those required. This objective can be aided by executing the selection or data processing on the source system whenever possible.

These could be the top three sources:

- FTPd flat files - It is perhaps simplest, but not necessarily most efficient, to obtain files directly. Users may need some education on FTP. Remember to transfer these files in binary mode.
- Relational databases (local or remote) - Drawing files from an RDBMS is becoming easier with the SAS/CONNECT product. This will, undoubtedly, require cooperation between the database and system administrators. Users may require education on SQL usage. They should also be encouraged to execute all possible selection and data processing commands directly in SQL on the RDBMS and take full advantage of the SAS pass-through SQL facility offered in SAS/CONNECT.
- Larger SAS files on a remote server - All efforts should be made to limit the size of files being transferred. In many cases, a SAS licence will be available on the remote server. Take advantage of this and use SAS remotely to create SAS data file(s) while taking care of some record selection and data processing. On compatible operating systems, SAS

files are directly readable once transferred; otherwise, SAS transport data sets should be used.

In the latter two access methods, users' efficiency should surge with inclusion of the SAS/CONNECT software. See Section 8.2, "Remote access setup" on page 83, for more extensive detail on SAS/CONNECT.

Documentation is available for each of these methods. Furthermore, it is very specific to the SAS procedure, operating system, and ancillary software involved. SAS documentation for the PROC's SQL, CPORT, CIMPORT, DATASET, DOWNLOAD, UPLOAD, and SAS/CONNECT are among the useful and necessary references. Preempt difficulties by providing users with both the educational and reference materials they need.

8.1.7 SAS files on disk

You will want to anticipate file usage on your system. A strategy covering the access permissions and acceptable life and size constraints on permanent SAS files requires adherence. An entirely unregulated approach will prove problematic.

SASWORK should see the most I/O activity and should receive appropriate attention. Refer back to Section 4.2.1, "Work space" on page 34, for a full discussion.

Permanent SAS file disk space must also be planned for. Some issues you need to consider are:

- Space - How much space need be set aside?
- Common versus shared - Will users need individually partitioned disk space or will a common volume group prove sufficient?
- Durability - How long will users be allowed to keep SAS files before deletion or archival?
- Archival - Develop and adhere to an archival policy.
- Large files - Do you want to allow for large files on your file system? See Section 4.2.4, "Enabling very large file (> 2 GB) access" on page 36.

Old SAS files can be a nuisance and a drain on resources. You may also think about implementing either AIX file compression or making the SAS compression option the default.

Monitoring SAS disk space is an ongoing system administration requirement. After reviewing opportunities for data compression or behavioral change, the disk space dedicated to SAS might be expanded or altered as necessary.

8.1.8 Job scheduling

SAS processes will complete faster if a bit of discipline in job scheduling is employed. Our examples in Section 4.3.2, “Sorting processes” on page 40, show just that.

Recall that an individual sort with memory constraints finished in 126 seconds. Four consecutively submitted jobs should finish in roughly 500 seconds. Yet, when four simultaneous jobs were submitted (under identical conditions) on our two processor RS/6000, each process averaged 596 seconds. In the multi-process case, the jobs were competing over both CPU and paging space. In the former, each job had the full attention of one processor to itself, leaving the other available for other tasks and other users.

We submit that an optimal schedule for our four tasks would be to process in pairs, thus, leveraging the power of both processors. (We leave this open for confirmation.)

If it is optimal to have one process attached to a processor at any one time; we must advise users never to submit more than one task. If multiple tasks must be run, submit no more jobs than there are free processors available. CPU, I/O, and paging space contention will likely deny users the resources they expect when submitting many processes at once.

Admittedly, it is difficult to coordinate the actions of several users. Still, some efficiencies are sure to be gained if users can reduce CPU allocation requests on lower priority tasks or schedule the submission of certain tasks for submission during periods of low system use. SAS jobs that might be ideal for delayed or batch scheduling include:

- Routine data-intensive reports
- Model updates, model scoring, etc.
- Data file creation and regular downloads
- Any large tasks whose results are not immediately required

Users should first be educated in the use of the `at` command used to submit commands at a later time (perhaps after everyone goes home). If large jobs are run at regular intervals, distribute material on writing shell scripts that release commands for execution at defined intervals. Introduce users to the `nice` command, a useful AIX command that downgrades CPU allocation requests on low-priority processes. Taken together, the above should combine to produce a solid user introduction to job scheduling.

8.1.9 Other software

SAS is rarely the only application on a system. Be aware of what other applications need to interact with SAS. You might categorize an application among the following groups:

- Relational databases
- Statistical software in addition to SAS
- PC/Client software
- Other AIX applications

For some of these applications, notably RDBMS software, enabling ODBC connectivity will be a primary concern. This is too specific a topic to address here. SAS and application-specific documentation is a better source.

While AIX administrators rarely assume responsibility for PC applications, it is still best to know which of these see the most use interactive use with SAS. They may see unanticipated usage patterns that affect your system.

With a better understanding of available and preferred software and a grasp of users' motivations for running SAS, the administrator can help by making productivity recommendations. In some cases, users could even be encouraged to ease use of the SAS application by shifting their usage to other more appropriate applications.

8.1.10 System objectives

SAS usage needs to be measured and tempered with an understanding of the system's purpose in the first place. If the server is primarily dedicated to the needs of the SAS user group, this may not be a concern. SAS usage defines just one of perhaps several user communities.

The areas of concern that should be addressed in balancing resources among communities are the same that we have been stressing throughout: Memory, CPU, I/O, and file space.

8.2 Remote access setup

For those with users not always resident on the LAN (that is, most systems), a bit of additional thought and configuration is needed. Any end-user with telnet capabilities should be able to access the server. Lead your users to a preferred connectivity channel by distributing connectivity instructions. Obviously, security will be an increased concern. Plan for this, and distribute users with gateway information if a gateway server is used.

A few PC users still prefer the AIX-Windows environment. Those with sufficient bandwidth can install third-party X-Windows emulator software on the client and manage their AIX sessions from within the emulator.

To enable remote users to run the SAS program on the server from SAS on the client, SAS/CONNECT software must be installed. Once installed, users can issue instructions to SAS. For example:

Computing services

- Enable users to submit SAS code on the server using data on the server from the client's SAS session.

Remote library services

- Manage and process remote SAS data libraries and files from the client.

Data transfer services

- Use PROC UPLOAD/DOWNLOAD to submit SAS files between client and server, thus, obviating the need for creating and FTPing SAS XPORT files.
- Use the SAS Remote SQL Pass-through capabilities to submit SQL commands directly to the server's RDBMS, and output SAS data files to local or remote libraries.

There are costs associated with using SAS/CONNECT. The most obvious will be those for the SAS/CONNECT licence and SAS licences for the users' client machines, if not they are already funded. Increases in data traffic and related network administration duties are among the costs of another sort.

Benefits may accrue to the user and the server community. Increased flexibility is certainly one. Users are free to move more processing from the server to the client. For users more comfortable with the SAS GUI but not privy to the bandwidth and software required to pipe the AIX SAS GUI back to the client, SAS/CONNECT can provide a user-friendly environment.

Appendix A. The SAS System on an IBM RS/6000 SP

There are various approaches to implementing the SAS System on an RS/6000 SP in such a way that it will take advantage of the scalable architecture of the SP. This appendix will briefly describe how implementing processes, conventions, and an infrastructure can provide a scalable, robust, and powerful environment for SAS System computing on the SP.

In its simplest form, executing the SAS System on the SP entails installing the SAS System on an SP node and running it only on that node. However, this simple form does not exploit the scalable architecture of the SP.

One way to exploit the scalable architecture of the SP is to define a cluster of SAS System nodes to provide processing power over and above what a single uniprocessor or single SMP node provides. With the SP Switch and built-in management facilities of the Parallel System Support Programs (PSSP), the SP provides a scalable environment that, when properly planned for and implemented, is extremely robust, easily scaled, and reliable for 24X7 applications.

Here are some areas that need to be discussed when setting up a cluster of SAS System nodes:

- SP configuration for cluster operation
- Load balancing
- Data sharing

A.1 SP configuration for cluster operation

A set of SP nodes may be designated for interactive use for multiple SAS users. This pool of nodes is sometimes referred to as a cluster. The SAS System should be installed on every node participating in the SAS System cluster. The nodes may or may not have the same version of the SAS System installed; as long as they are downward/upward compatible, this is not an issue. One reason for having different SAS System versions installed is to test a new version/release of the SAS System against the SAS applications before cluster-wide roll out of this new version/release.

When executing the SAS System, a work directory must be specified for the SAS System to use as a temporary workspace. For the purposes of this discussion, the name of this directory will be /saswork. A file system with a mount point of /saswork must be defined on each node of the cluster.

Common user IDs, passwords, and home directories can be implemented in a variety of ways. PSSP's user management provides a simple way to accomplish this by propagating the required password and group files to all nodes of the SP and utilizing the amd daemon to automatically mount the users' home directories when they log in to any node. Another method of maintaining user IDs and passwords across multiple machines is NIS. Explicitly mounting home directories or using the automount daemon will ensure common home directories. Each customer environment or preference will dictate which of these methods is chosen.

Once these steps are complete, we have a system in which a user may log in to any node in the cluster and execute the SAS System. We will now discuss a way to balance the workload across all nodes in the cluster.

A.2 Load balancing

Load balancing across a cluster of SP nodes is accomplished via a LoadLeveler feature called Interactive Session Support (ISS), which is a licensed program product provided at no extra charge with PSSP. ISS is used to distribute logins and application sessions across a pool of servers in a manner that is completely transparent to users and applications.

ISS interfaces with a TCP/IP nameserver to translate machine names to internet addresses. The function of ISS is to recommend to the nameserver the IP address of the *least loaded* node. Instead of specifying the actual machine name of a particular server, the user specifies the name of a pool that has been set up by the LoadLeveler administrator. The set of nodes to be used for SAS System processing are defined in a pool and given a name. In this example, we will use the name *spsas*. A user connects via telnet or uses SAS/CONNECT software, specifying *spsas* as the hostname. ISS has recommended which actual IP address to route the session to based on the current load of the servers in the pool. The definition of *least loaded* is customizable and can be modified by the LoadLeveler administrator.

At this point, the user is ready to access data. In a load-balanced clustered environment, the user(s) can be sent to any node in the cluster, but not necessarily always to the same node. The user must be able to access the required data no matter which node the data physically resides on. Next, we will address sharing the data across all nodes in the SAS System cluster.

A.3 Data sharing

Sharing of data across the cluster of SP nodes can easily be accomplished using the new general parallel file systems (GPFS). GPFS provides file system services to parallel and serial applications running on the RS/6000 SP. Its main benefit to SAS applications is that it allows users shared access to files that may span multiple disk drives on multiple SP nodes. You can get more details on how GPFS can work with the SAS System via the white paper located on the following external Web site:

<http://www.sas.com/partners/ibm/Gpfsfina.pdf>

With a little more administrative work, a similar approach can be done via NFS mounting of the disks on the various SP nodes that will be used by the SAS System. However, there are two areas of interest that must be addressed if you want to pursue this approach. The first is the performance implications of writing to NFS mounted file systems. The second is the location of data, that is, knowing on which file system the data of interest is located. Each of these can be addressed easily with proper planning and, by setting a few conventions and SAS System operational procedures, can be made transparent to the users. You can get more details on how this can be done via the white paper located on the following external Web site:

http://www.sas.com/partners/ibm/RISC_DB2.pdf

To summarize, the SAS System can exploit the IBM RS/6000 SP architecture very effectively.

Appendix B. Disk space and RAM requirements

The following sections describe the memory and disk requirements for your RS/6000 system when running SAS and SAS applications.

How much disk space and RAM do I need for my SAS application?

One of the most frequently asked questions with regard to the SAS System is how much disk space (for both permanent and temporary SAS data files) and memory does a customer need for their SAS applications. There is no exact equation to determine this because every customer's usage of the SAS System differs. However, there are some good guidelines in place to help your customer determine, based on their SAS usage, how much disk space will be needed.

Disk space for permanent SAS data files

The SAS System stores data in a SAS data file (formerly called a SAS data set) in a tabular fashion. To determine the size of this data file, simply multiply the number of rows in the table by the record length of the longest row. The information stored in the header of the SAS data file is roughly the size of two rows.

So, to determine the amount of disk space needed for the permanent SAS data files, the customer can use the above equation to determine the size of the data warehouse. Don't forget to query all the users with regard to how much data they will be keeping in their personal area.

Memory needs

The amount of memory the SAS System uses depends on what tasks the users are carrying out within the SAS System. Below are several common SAS System tasks with some guidelines regarding how much memory is needed for each task.

Data Warehouse Setup - These jobs are generally long running jobs with many steps that run in batch mode after hours. The steps are more I/O-intensive in nature; so, a lot of memory is not needed. If the steps include manipulating the data through DATA steps, summarizations, and univariate analysis, the jobs will rarely take more than 32 MB of physical RAM.

MDDDB Cube Creation - The creation of these cubes requires enough memory (preferably physical RAM) to build the N-way table for the MDDDB (a formula is available to help you determine how much memory is needed in the *Tips to Using the SAS System* white paper).

FREQ Procedure - For each variable in a table request, PROC FREQ stores all of the levels in memory. If all variables are numeric and not formatted, this requires about 84 bytes for each variable level. When there are character variables or formatted numeric variables, the memory that is required depends on the formatted variable lengths, with longer formatted lengths requiring more memory. The number of levels for each variable is limited only by the largest integer that your operating environment can store.

MEANS and SUMMARY Procedures - These two procedures employ the same memory allocation scheme across all host environments. When class variables are involved, these procedures must keep a copy of each unique value of each class variable in memory. There is a formula in the Version 7 documentation that can help you estimate the memory requirements needed to group the class variable.

If you do not have enough memory allocated to the SAS session, you will get a message that you need to adjust the SUMSIZE parameter associated with the SAS session. There is more information about doing this in the Version 7 documentation for the MEANS procedure.

SORT Procedure - This procedure may run faster if the input data set can be held in physical RAM. If not, it is best to limit the memory used by these procedures by setting the SORTSIZE parameter between 16 MB and 32 MB.

Statistical Analysis - Several of the statistical analysis procedures (PHREG, LOGISTIC, MIXED, and the Enterprise Miner product) create a utility file while doing their calculations. These utility files are held in memory. In order to get the best performance possible, these utility files need to fit in physical memory. Starting with Version 7, we have documented the computer resources needed for all the products.

GUI Applications - There are several SAS System applications that are built using the SAS/AF product. These include SAS/EIS software and the applications build with this tool, Enterprise Miner, CFO Vision software, HR Vision software, the Analyst Application, just to name a few. We generally recommend a minimum of 32 MB of physical RAM per user (unless there are many graphics and images; in that case, we recommend 64 MB per user).

Work Space Needs

The amount of disk space needed in the SASWORK area depends on what tasks within the SAS System the users are doing. Below are several common SAS System tasks with some guidelines regarding how much disk space is needed for each task.

Data Warehouse Setup - As mentioned previously, when building a data warehouse, there are generally several steps to the process. The user can start by extracting data from several sources and then merging the information together. If the steps used in manipulating the original data create temporary SASWORK files, you will need enough work space for all these files in the SASWORK directory.

MEANS and SUMMARY Procedures - If these procedures do not have enough memory to execute, they must write partially complete primary types to disk while it processes input data. When this happens, one or more merge passes may be required to combine type levels in memory with those on disk. In addition, if you use an order other than DATA for any class variable, these procedures group the completed type on disk. For this reason, the peak disk space requirements can be more than twice the memory requirements for a given type.

SORT Procedure - The SORT procedure creates two utility files during the sort process as well as a temporary file containing the records in sorted order. Because of this, it is good to plan on having disk space available for three times the size of the file being sorted in the SASWORK area.

Enterprise Miner - When using the Enterprise Miner product to do data mining, you need to make sure you have enough disk space for the files stored in the project library. The amount of space needed depends on what nodes are used in the project. As a general rule, you should plan on four times the size of the input data file being analyzed, but note that this space may be as much as 10 times the size of the input data file. (A paper detailing the resources needed when using Enterprise Miner is available.)

Statistical Analysis - Several of the statistical analysis procedures (PHREG, LOGISTIC, MIXED, and the Enterprise Miner product) create a utility file while doing their calculations. Starting with Version 7, we have documented the computer resources needed for all the products.

We realize that the preceding information is vague, but we hope it can help you give your customers some pointers as to what resources (both memory and disk space) are needed when running their SAS applications.

Appendix C. Optimizing systems performance

This Appendix is copied from the SAS Institute Homepage, which you can visit at: <http://www.sas.com/partners/ibm/optimize.html>

This paper is designed to help you maximize the performance of the SAS System. The following sections suggest ways to increase the efficiency of your SAS job in terms of the three critical resources: I/O, memory, and CPU time. While you may not be able to take advantage of every technique for every job, you can choose the ones that are best suited for your particular job.

Note

For a comprehensive list of efficient programming tips, see *SAS Programming Tips: A Guide to Efficient SAS Processing*, MS56150, and the SAS Companions for the various operating systems.

C.1 Techniques for optimizing I/O

I/O is one of the most important factors for optimizing performance. Most SAS jobs consist of repeated cycles of reading a particular set of data in order to perform various data analyses and data manipulation tasks. To improve the performance of a SAS job, you need to reduce the number of times the SAS system accesses disk or tape devices.

You can do this in two ways:

- Modify your SAS programs to reduce the number of times you have to process the data. There are ways to reduce the number of times you process data including using WHERE processing, using indexes, using both engines and data sets efficiently, and accessing data through views.
- Reduce the number of data accesses by processing more data each time the device is accessed. See details on the BUFNO=, BUFSIZE=, CATCACHE=, and COMPRESS= options in C.4, "SAS system options" on page 95.

Another way of improving efficiency is to use the DROP, KEEP, and LENGTH statements to reduce the size of any given observation and to use the OBS= and FIRSTOBS= options to reduce the number of observations processed.

When you create a temporary data set and include only the needed variables and observations, you can reduce the number of I/Os required to process the

data. See Chapter 4 "Starting with SAS Data Sets," in the book, *SAS Language and Procedures: Usage*, Version 6, First Edition, ISBN 1-55544-371-0, for more information on DROP, KEEP, LENGTH, OBS, and FIRSTOBS.

One final way to improve efficiency is to balance the I/O processing by keeping the SAS WORK directory and system swap files on separate disk drives. That way, you get the most for your money.

C.2 Techniques for optimizing memory usage

If memory is your critical resource, several techniques can reduce the dependence on increased memory. However, most of them will also increase I/O processing or CPU usage.

By increasing the value of the MEMSIZE system option, you can decrease the processing time because the amount of time spent on paging is reduced.

You can make trade-offs between memory and other resources. To make the most of the I/O subsystem, you need to use more and larger buffers. These buffers must share space with the other memory demands of your SAS session.

C.3 Techniques for optimizing CPU performance

Executing a single stream of code takes approximately the same amount of CPU each time that code is executed. Optimizing CPU performance in these instances is usually a trade-off, and the cost is using more memory (see the MEMSIZE System Option).

Because the CPU performs all the processing needed to perform an I/O operation, options or techniques that reduce the number of I/O operations often have a positive effect on CPU usage:

- Storing Compiled Code for Computation-Intensive DATA Steps - Another technique that can improve CPU performance is to store DATA step code that is executed repeatedly as compiled code rather than as SAS source code. This is especially true for large DATA step jobs that are not I/O-intensive. For more information on the Stored Program Facility, see Appendix 3 of the *SAS Language*, Version 6, Cary, NC: SAS Institute Inc., 1990, ISBN 1-55544-381-8.
- Reducing Search Time for SAS Executable Files - Your default configuration file specifies a certain order for the directories containing

SAS executable files. You can rearrange the directory specifications in the PATH option so that the most commonly accessed directories are listed first. Place the least commonly accessed directories last.

- Specifying Variable Lengths - When the SAS System processes the data vector, it typically moves the data in one large operation rather than individual variables. When data are properly aligned (in 8-byte boundaries), data movement can occur in as little as two clock cycles (a single load followed by a single store). Unaligned data are moved by more complex means (at worst, a single byte at a time). This would be at least eight times slower for an 8-byte variable. Many high-performance RISC processors pay a very large penalty for movement of unaligned data. When possible, follow these suggestions for keeping data aligned:
 - Leave numeric data at full width (8-bytes) (Note that the SAS System must widen short numeric data for any arithmetic operation. On the other hand, short numeric data can save both memory and I/O).
 - Keep character data in multiples of 8 bytes in length. This obviously wastes memory, but it does keep data aligned.

These suggestions are especially important when processing a data set by selecting only specific variables and when clause processing. It is important that the selected variables be properly aligned.

C.4 SAS system options

The following are SAS system options for controlling resource allocation by SAS programs.

- BUFNO= Option

The BUFNO= system option specifies the number of buffers to be allocated for processing a SAS data set. The number of buffers is not a permanent attribute of the data set, and it is valid only for the current SAS session or job. The BUFNO= option applies to SAS data sets opened for input, output, or update.

Note

Using the BUFNO= system option can speed up execution time by limiting the number of input/output operations required for a particular SAS data set. The improvement in execution time, however, comes at the expense of increased memory consumption.

PC platform default - 3

UNIX platform default - 1

- **BUFSIZE= Option**

The BUFSIZE= option specifies the size of input/output buffers for SAS data sets. The size of the input/output buffers is permanently associated with the SAS data set. If the number of bytes is greater than 0 when a SAS data set is created, that number is used as the default value for the BUFSIZE= data set option. If the BUFSIZE= data set option is not used and the number of bytes for the BUFSIZE= system option is 0, the SAS System chooses a host system default value that is optimal for the SAS data set.

Note

Using the BUFSIZE= system option can speed up execution time by limiting the number of input/output operations required for a particular SAS data set. The improvement in execution time, however, comes at the expense of increased memory consumption.

PC platform default - 0

UNIX platform default - 0

- **CATCACHE= Option**

CATCACHE=n

The CATCACHE= system option specifies the number of SAS catalogs to keep open. If n is greater than 0, the SAS System places up to that number of open-file descriptors in cache memory instead of closing the catalogs. If n is 0, no open-file descriptors are kept in cache memory.

You can use the CATCACHE= system option to tune an application by avoiding the overhead of repeatedly opening and closing the same SAS catalogs. Increasing the value of the CATCACHE option can potentially improve performance by keeping the catalogs needed for a SAS application in memory during the entire SAS Session.

Note

The increased performance in catalog I/O can also use considerable memory resources; use this technique only if memory issues are not a concern.

PC platform default - 0

UNIX platform default - 0

- COMPRESS= Option

COMPRESS= YES|NO

The COMPRESS= system option specifies whether observations in a newly-created SAS output data set are compressed (variable-length records) or uncompressed (fixed-length records). The record type is a permanent attribute of the SAS data set.

Compressing a data set reduces the size of the data set by reducing repeated consecutive c characters to two- or three-byte representations. To uncompress observations, you must use a DATA step to copy the data set and specify COMPRESS=NO for the new data set.

The advantages gained by using the COMPRESS= data set option include the following:

- Reduced storage requirements for the data set
- Fewer input and output operations necessary to read from or write to the data set during processing.

Note

Using the COMPRESS= system option prevents access to a SAS data set by observation number. Also, using this option increases the CPU time for reading a data set because of the overhead of compressing and uncompressing the records.

- IMPLMAC Option

The IMPLMAC system option controls whether macros defined as statement-style macros can be invoked with statement-style macro calls or if the call must be a name-style macro call.

Note

When you use the IMPLMAC system option, processing time is increased because the SAS System checks every SAS statement to determine whether the beginning word is a macro call. When you use the IMPLMAC system option in conjunction with the MAUTOSOURCE system option, the MRECALL system, or both, processing time can be increased further.

- MEMSIZE= Invocation Option

-MEMSIZE= n | nK | nM | nG | MAX

The MEMSIZE option specifies a limit on the total amount of memory the SAS System uses at any one time. The operating system may use

additional amounts of memory. A MEMSIZE option value of 0 tells the SAS System to use all available memory, up to the system limit. Too low a value will result in out-of-memory conditions. When you increase the value of SORTSIZE, you will need to increase the value of MEMSIZE. This option can take the following values:

- n - Specifies the amount of memory in bytes
- nK - Specifies the amount of memory in kilobytes
- nM - Specifies the amount of memory in megabytes
- nG - Specifies the amount of memory in gigabytes

Note

The MEMSIZE option must be set during the invocation of the SAS System by modifying the CONFIG.SAS file or passing it as parameter to the SAS command.

PC platform default - 0

UNIX platform default - 32 MB

- MSYMTABMAX= Option

MSYMTABMAX= n | nK | nM | nG | MAX

The MSYMTABMAX= system option specifies the maximum amount of memory available to the macro variable symbol table(s). Once this value is reached, additional macro variables are written out to disk. The value you specify with the MSYMTABMAX= system option can range from 0 to the largest non-negative integer representable on your host. The values can be expressed as follows:

- n - Specifies the amount of memory in bytes
- nK - Specifies the amount of memory in kilobytes
- nM - Specifies the amount of memory in megabytes
- nG - Specifies the amount of memory in gigabytes
- MAX - Specifies the maximum amount of memory available

PC platform default - 4M

UNIX platform default - 8K

- MVARSIZE= Option

MVARSIZE= n | nK | nM | nG | MAX

The MVARSIZE= system option specifies the maximum size for in-memory macro variables. If the size is larger than this value, variables are written out to disk. The value you specify with the MVARSIZE= system option can range from 0 to the largest non-negative integer representable on your host.

The value can be expressed as follows:

- n - Specifies the amount of memory in bytes
- nK - Specifies the amount of memory in kilobytes
- nM - Specifies the amount of memory in megabytes
- nG - Specifies the amount of memory in gigabytes
- MAX - Specifies the maximum amount of memory available

PC platform default - 4K

UNIX platform default - 512K

- RESIDENT=num

RESIDENT= specifies whether an SCL entry is saved in resident memory the first time that it is executed instead of being re-read from the catalog on subsequent calls. Ranges for <num> are:

- < 0 to save in memory only SCL entries containing a METHOD statement with the /RESIDENT option.
- = 0 to save no SCL entries in memory.
- > 0 to save <num> entries in memory. By default, the number of SCL entries saved in memory is 64.

When an SCL entry executes, SCL searches resident memory for the entry. If the search is successful, the entry moves to the top of the search list. An SCL entry that is called frequently remains at or near the top of the list and, thus, is found more quickly. When an SCL entry is not found on the search list, the last entry on the search list (the least- recently used) is removed, and the new entry is inserted at the top of the list.

- SORTSIZE = memory-specification

The SORTSIZE= system option specifies the maximum amount of memory available to the SORT procedure (or the sort utility specified with the SORTPGM= system option). The 'memory-specification' can be one of the following:

- MAX - Specifies that all available memory can be used
- n - Specifies the amount in bytes

- nK - Specifies the amount in kilobytes
- nM - Specifies the amount of memory in megabytes

Specifying the SORTSIZE= option in the PROC SORT statement temporarily overrides the setting for the SORTSIZE= system option. The value of the SORTSIZE= system option is the default. When you increase the value of SORTSIZE, please make sure you increase the value of MEMSIZE as well.

Note

Using this option can help improve sort performance by restricting the virtual memory paging controlled by the host operating system. If the SORT procedure needs more memory than you specify, it uses a temporary utility file. As a general rule, the value you use for SORTSIZE= should be set to less than the physical memory available to your process.

PC platform default - 2M

UNIX platform default - 16M

C.5 SAS procedures that use extra resources

The following section describes options that will cause the SAS system to take up more resources.

- CONTENTS with FMTLEN Option

The FMTLEN option prints the default length of formats and informats if they do not have a specified length. If you omit the FMTLEN option, the CONTENTS statement still prints the informat or format, but it does not include the length unless the informat or format has a specified length.

Note

When you use the FMTLEN option, the SAS system uses additional CPU time, I/O time, and memory to load the format and determine its length.

Default length of format: length of the longest formatted value

Default length of informat: longest informatted value

- FREQ with TABLES Statement - EXACT Option
TABLES requests / EXACT;

The EXACT option requests Fisher's exact test for tables that are larger than 2X2. The computational algorithm is the network algorithm given by Mehta and Patel (1983).

Note

- This option is not turned on when the ALL option is specified.
- This test is very intensive in the use of memory and cpu time. It is NOT recommended when $n / ((r-1)(c-1)) > 5$ or when $\text{MIN}(r, c) > 5$. (n is the sample size. r is the number of rows. c is the number of columns.)

- LOGISTIC

For each BY group, define the following:

- K = number of response levels
- $C = 1 +$ number of explanatory variables
- $m1 = K + C$
- $m2 = 1 + m1$
- $m3 = 16m1(m1 + 5)$
- $m4 = 8C(C + 4) + 4m2(m2 + 3)$

The minimum working space needed to process the BY group is m3 bytes. For models with more than two response levels, a test of the parallel lines assumption requires an additional workspace of m4 bytes. However, if this additional memory is no available, the procedure skips the test and finishes the other computations.

If sufficient space is available, the relevant variables and observations from the input data set are also kept in memory; otherwise, the input data set is reread for each evaluation of the likelihood function and its derivatives, with the resulting execution time of the procedure substantially increased.

- MDDB

The minimum working space and virtual memory needs for creating an MDDB are as follows:

- For every MDDB, there are 900 bytes of overhead .
- For every analysis variable, there are 676 bytes of overhead.
- For every class variable, there are 340 bytes of overhead + (the maximum formatted length of the variable * the number of values)+ (unformatted length of variable * the number of variables).

- For each hierarchy, there are 296 bytes of overhead (always at least one - NWAY).
- For each hierarchy: (the number of dimensions * 4 + the number of analysis vars * the number of stats * 8) * the number of crossings in the hierarchy.
- MEANS with CLASS Statement
CLASS variable-list

The CLASS statement assigns the variables used to form subgroups. The CLASS statement has basically the same effect on the statistics computed as that of the BY statement. The differences are in the format of the printed output and in the sorting requirements of the BY statement.

Note

Theoretically, the maximum number of combinations of CLASS levels is 200 million. Realistically, it becomes a machine-dependent estimate, limited solely by the amount of computer memory available. The maximum number of CLASS variables is 30.

- MULTTEST
PROC MULTTEST keeps all of the data in memory to expedite resampling. A large portion of the memory requirement is thus $8 \cdot \text{NOBS} \cdot \text{NVAR}$ bytes, where NOBS is the number of observations in the data set, and NVAR is the number of variables analyzed, including CLASS, FREQ, and STRATA variables.
If you specify PERMUTATION=number (for exact permutation distributions), then PROC MULTTEST requires additional memory. This requirement is approximately $4 \cdot \text{NTEST} \cdot \text{NSTRATA} \cdot \text{CMAX} \cdot \text{number} \cdot (\text{number} + 1)$ bytes, where NTEST is the number of contrasts, NSTRATA is the number of STRATA levels, and CMAX is the maximum contrast coefficient.
The execution time is linear in the number of resamples.
- NPAR1WAY
Although the computational algorithm is fast, the computational time can still be prohibitive depending on the number of groups, the number of distinct response variables, the total sample size, and the speed and memory available on your computer. You can terminate exact computations and exit to the NPAR1WAY procedure at any time by pressing the system interrupt key (refer to the SAS Companion for your system) and choosing to stop computations.

- PHREG

The PHREG procedure performs regression analysis of survival data based on the Cox proportional hazards model. This procedure is very compute intensive and will perform faster if given more memory. A simple algorithm to determine the minimum working space (in bytes) needed to process the BY group is $\max\{12n, 24pp + 160p\}$ where n is the number of observations in a BY group, p is the number of explanatory variables, and pp is the square of p .

If sufficient space is available, the input data set is also kept in memory. Otherwise, the input data is reread from the utility file for each evaluation of the likelihood function and its derivatives, with the resulting execution time substantially increased.

- REG

The REG procedure is efficient for ordinary regression; however, requests for optional features can greatly increase the amount of time required.

The major computational expense in the regression analysis is the collection of the cross-products matrix. For p variables and n observations, the time required is proportional to np^2 . For each model run, REG needs time roughly proportional to k^3 , where k is the number of regressors in the model. Add an additional nk^2 for one of the R, CLM, or CLI options and another nk^2 for the INFLUENCE option.

Most of the memory REG needs to solve large problems is used for crossproducts matrices. PROC REG requires $4p^2$ bytes for the main crossproducts matrix plus $4k^2$ bytes for the largest model. If several output data sets are requested, memory is also needed for buffers.

10) SORT

The SORT procedure sorts observations (arranging them in order by values of one or more variables) and is one of the most common operations performed with the SAS System. You can now specify the SORTSIZE= option when you invoke this procedure. Specifying the SORTSIZE= option in the PROC SORT statement temporarily overrides the setting of the SORTSIZE= system option. The value of the SORTSIZE= systems option is the default. The SORTSIZE= system option was discussed earlier in this appendix.

Note

When you invoke the SORT procedure, the computer system uses either a sorting module provided by SAS Institute, a sorting utility provided with the operating system, or a sorting utility provided by an independent vendor.

It is important to remember that to sort a SAS data set, you need enough disk space to hold the original file and at least two more files of the same size as the original one. This will depend on the number of BY variables.

- SUMMARY with CLASS Statement

CLASS variable-list

The CLASS statement assigns the variables used to form subgroups. The CLASS variable may be either numeric or character, but normally each variable has a small number of discrete values or unique levels. The CLASS statement has an effect on the statistics computed similar to that of the BY statement.

Note

Theoretically, the maximum number of combinations of CLASS levels is 200 million. Realistically, it becomes a machine-dependent estimate, limited solely by the amount of available computer memory. The maximum number of CLASS variables is 30.

C.6 Performance considerations of DATA step views

Using DATA step views can improve the efficiency of programming and applications development. However, the requirements placed on machine resources can increase or decrease depending on the methods of data processing that you replace by using DATA step views. The impact on machine resources is determined by the access pattern of the consuming task (DATA step or PROC step). The consuming task can request the retrieval of data in two ways: A single pass or multiple passes.

When one pass is requested, no data set is created. Compared to traditional methods of processing, the one-pass access pattern increases performance by decreasing the number of input/output operations and elapsed time.

When multiple passes are requested, the view must build a spill file that contains all generated observations so that subsequent passes can read the same data read by previous passes. Whenever the consuming task needs to access only the data across BY groups, the SAS System optimizes multiple passes by reusing space within the spill file whenever the BY groups change. With this optimization, the amount of disk space required is the cumulative size of the largest BY group generated rather than the cumulative size of all observations generated by the view.

Both the single-pass access pattern and the multiple-pass access pattern incur a certain overhead in CPU time and memory requirements. As a general rule, CPU time increases by approximately 10 percent. This increase is due to an internal host supervisor requirement and will be addressed in a future release of the SAS System.

Concerning memory utilization, when a DATA step references a DATA step view, the overhead incurred is associated with additional storage required to execute the DATA step view. When a PROC step references a DATA step view, the additional memory incurred is associated with the DATA step that executes the view.

Appendix D. Disk controller cache test results

In Chapter 7, “AIX file system, SAS system, and performance” on page 67, we tested the impact of having writecache on the disk adapter. In this appendix, we have the complete list of execution time results, whereas we had only average values in Chapter 7.

Twelve tests were made with write cache, and twelve tests were made without write cache. We tested various configurations of SAS output and work datasets. The reason for choosing these datasets was that this is the most used partition, and a change in performance would then have the most impact here. The tests were done with three, four, and five disk drives in a non-striped configuration holding the output dataset and a five disk striped configuration with an 8 KB stripe size holding the output dataset. For each of these four output dataset configurations, we had three configurations for the SAS work space: Three, four, and five disks non-striped in all 12 tests. These 12 tests were done with and without the disk adapter write cache enabled.

Tests 1 -3 SAS output dataset on hdisk6, hdisk7, and hdisk8

- Test 1 SAS work dataset on hdisk9, hdisk10, and hdisk11
- Test 2 SAS work dataset on hdisk9, hdisk10, hdisk11, and hdisk12
- Test 3 SAS work dataset on hdisk9, hdisk10, hdisk11, hdisk12, and hdisk13

Tests 4-6 SAS output dataset on hdisk6, hdisk7, hdisk8, and hdisk14

- Test 4 SAS work dataset on hdisk9, hdisk10, and hdisk11
- Test 5 SAS work dataset on hdisk9, hdisk10, hdisk11, and hdisk12
- Test 6 SAS work dataset on hdisk9, hdisk10, hdisk11, hdisk12, and hdisk13

Tests 7-9 SAS output dataset on hdisk6, hdisk7, hdisk8, hdisk14, and hdisk15

- Test 7 SAS work dataset on hdisk9, hdisk10, and hdisk11
- Test 8 SAS work dataset on hdisk9, hdisk10, hdisk11, and hdisk12
- Test 9 SAS work dataset on hdisk9, hdisk10, hdisk11, hdisk12, and hdisk13

Tests 10-12 SAS output dataset on hdisk6, hdisk7, hdisk8, hdisk14, and hdisk15

- Test 10 SAS work dataset on hdisk9, hdisk10, and hdisk11

- Test 11 SAS work dataset on hdisk9, hdisk10, hdisk11, and hdisk12
- Test 12 SAS work dataset on hdisk9, hdisk10, hdisk11, hdisk12, and hdisk13

D.1 EXECUTION time with write cache is disabled

In this section, we list the date from the test with write cache disabled and disk write Queue depth = 3

Without writecache - Test 1

real 189.62 user cpu 20.02 system cpu 25.03

real 189.59 user cpu 20.66 system cpu 24.92

real 162.15 user cpu 20.90 system cpu 24.65

real 162.08 user cpu 20.90 system cpu 24.99

real 156.32 user cpu 20.94 system cpu 24.33

real 156.44 user cpu 20.86 system cpu 23.80

real 159.02 user cpu 20.95 system cpu 23.52

real 158.95 user cpu 20.54 system cpu 25.20

real 105.06 user cpu 20.90 system cpu 24.01

real 105.08 user cpu 20.51 system cpu 24.35

AVG. 154.43 20.72 24.48

Without writecache - Test 2

real 149.02 user cpu 20.25 system cpu 24.79

real 148.85 user cpu 20.52 system cpu 24.12

real 135.53 user cpu 21.07 system cpu 23.39

real 137.04 user cpu 20.79 system cpu 22.69

real 177.37 user cpu 20.38 system cpu 25.41

real 180.73 user cpu 20.94 system cpu 26.47

real 108.24 user cpu 20.94 system cpu 23.14

real 106.15 user cpu 20.59 system cpu 23.13
real 147.66 user cpu 20.66 system cpu 25.86
real 147.67 user cpu 20.58 system cpu 24.45
AVG. 143.83 20.67 24.35

Without writecache - Test 3

real 191.51 user cpu 20.30 system cpu 24.99
real 191.49 user cpu 20.67 system cpu 23.83
real 109.54 user cpu 21.27 system cpu 22.88
real 111.81 user cpu 21.15 system cpu 22.24
real 95.48 user cpu 20.32 system cpu 25.10
real 97.82 user cpu 21.07 system cpu 22.92
real 104.56 user cpu 20.74 system cpu 24.38
real 104.52 user cpu 20.42 system cpu 23.81
real 124.97 user cpu 20.56 system cpu 23.44
real 123.00 user cpu 20.67 system cpu 23.17
AVG. 125.47 20.72 23.68

Without writecache - Test 4

real 212.52 user cpu 20.67 system cpu 25.19
real 212.56 user cpu 20.51 system cpu 25.25
real 129.32 user cpu 20.95 system cpu 22.93
real 129.17 user cpu 21.22 system cpu 23.02
real 137.17 user cpu 20.43 system cpu 23.90
real 136.29 user cpu 20.73 system cpu 23.65

real 153.82 user cpu 20.50 system cpu 24.27
real 152.96 user cpu 20.40 system cpu 23.76
real 95.40 user cpu 20.52 system cpu 23.96
real 96.31 user cpu 21.17 system cpu 23.12
AVG. 145.55 20.71 23.91

Without writecache - Test 5

real 167.85 user cpu 20.80 system cpu 23.41
real 167.98 user cpu 20.91 system cpu 23.71
real 129.28 user cpu 20.59 system cpu 24.74
real 130.85 user cpu 20.60 system cpu 23.21
real 121.93 user cpu 20.42 system cpu 24.30
real 121.04 user cpu 20.28 system cpu 24.08
real 157.52 user cpu 21.20 system cpu 24.14
real 157.55 user cpu 21.28 system cpu 24.64
real 75.52 user cpu 20.66 system cpu 23.90
real 81.04 user cpu 20.52 system cpu 24.54
AVG. 131.06 20.73 24.07

Without writecache - Test 6

real 176.59 user cpu 20.59 system cpu 24.72
real 176.59 user cpu 20.31 system cpu 24.25
real 152.87 user cpu 20.48 system cpu 25.24
real 152.80 user cpu 20.42 system cpu 24.26
real 137.55 user cpu 20.22 system cpu 23.44

real 138.38 user cpu 21.01 system cpu 23.20
real 121.74 user cpu 20.31 system cpu 24.14
real 121.57 user cpu 20.71 system cpu 23.81
real 123.49 user cpu 20.96 system cpu 23.02
real 122.60 user cpu 21.50 system cpu 23.12
AVG. 142.42 20.65 23.92

Without writecache - Test 7

real 230.43 user cpu 20.61 system cpu 24.88
real 230.41 user cpu 21.06 system cpu 24.01
real 139.26 user cpu 20.55 system cpu 24.46
real 139.21 user cpu 20.85 system cpu 24.20
real 130.68 user cpu 21.43 system cpu 23.33
real 129.74 user cpu 20.05 system cpu 23.81
real 126.56 user cpu 20.60 system cpu 22.97
real 127.41 user cpu 20.12 system cpu 24.69
real 144.17 user cpu 21.11 system cpu 23.16
real 145.04 user cpu 20.87 system cpu 23.80
AVG. 154.29 20.73 23.93

Without writecache - Test 8

real 178.01 user cpu 20.59 system cpu 24.49
real 178.04 user cpu 20.51 system cpu 24.74
real 125.89 user cpu 21.08 system cpu 23.08
real 126.07 user cpu 20.51 system cpu 23.50

real 117.18 user cpu 20.80 system cpu 23.22
real 117.38 user cpu 20.48 system cpu 23.56
real 143.11 user cpu 20.22 system cpu 25.22
real 143.17 user cpu 20.79 system cpu 24.07
real 121.49 user cpu 19.88 system cpu 23.96
real 122.39 user cpu 20.59 system cpu 24.14
AVG. 137.27 20.55 24.00

Without writecache - Test 9

real 131.71 user cpu 20.89 system cpu 23.10
real 135.96 user cpu 20.39 system cpu 23.18
real 106.46 user cpu 20.59 system cpu 24.05
real 106.29 user cpu 21.24 system cpu 23.33
real 112.22 user cpu 20.58 system cpu 24.24
real 112.37 user cpu 20.16 system cpu 23.74
real 102.86 user cpu 20.99 system cpu 23.87
real 94.21 user cpu 20.71 system cpu 22.71
real 119.77 user cpu 20.84 system cpu 23.10
real 119.95 user cpu 21.16 system cpu 24.50
AVG. 114.18 20.76 23.58

Without writecache - Test 10

real 294.42 user cpu 21.04 system cpu 23.64
real 293.57 user cpu 20.88 system cpu 23.30
real 270.04 user cpu 21.01 system cpu 23.60

real 269.19 user cpu 20.75 system cpu 23.80
real 249.52 user cpu 21.09 system cpu 23.21
real 248.67 user cpu 20.38 system cpu 23.31
real 239.78 user cpu 20.56 system cpu 23.03
real 248.98 user cpu 20.83 system cpu 23.63
real 182.90 user cpu 20.78 system cpu 23.51
real 210.09 user cpu 20.89 system cpu 22.37
AVG. 250.72 20.82 23.34

Without writecache - Test 11

real 193.31 user cpu 20.70 system cpu 23.37
real 192.41 user cpu 20.48 system cpu 23.98
real 193.64 user cpu 20.98 system cpu 24.15
real 185.37 user cpu 20.08 system cpu 25.13
real 227.06 user cpu 20.87 system cpu 23.31
real 226.21 user cpu 20.90 system cpu 22.86
real 226.96 user cpu 20.73 system cpu 23.57
real 227.01 user cpu 20.54 system cpu 24.82
real 214.51 user cpu 20.52 system cpu 23.12
real 218.71 user cpu 20.92 system cpu 23.53
AVG. 210.52 20.67 23.78

Without writecache - Test 12

real 218.34 user cpu 20.73 system cpu 24.41
real 217.48 user cpu 20.63 system cpu 23.90

```
real 220.47 user cpu 20.48 system cpu 23.74
real 207.84 user cpu 20.68 system cpu 22.95
real 169.03 user cpu 20.85 system cpu 24.02
real 181.67 user cpu 20.83 system cpu 23.21
real 199.24 user cpu 20.89 system cpu 23.86
real 199.05 user cpu 21.12 system cpu 23.16
real 158.44 user cpu 20.11 system cpu 23.35
real 198.98 user cpu 20.66 system cpu 23.40
AVG. 197.05    20.70    23.60
```

D.2 EXECUTION time when disk adapter write cache is enabled

In this section, we list the data from the test with write cache enabled and disk write Queue depth = 5; these settings are selected to achieve the highest possible performance.

With writecache - Test 1

```
real 202.76 user cpu 20.86 system cpu 24.36
real 202.77 user cpu 21.30 system cpu 23.85
real 162.73 user cpu 20.62 system cpu 23.44
real 161.64 user cpu 20.47 system cpu 23.44
real 157.48 user cpu 20.69 system cpu 23.50
real 157.50 user cpu 20.50 system cpu 23.92
real 165.44 user cpu 20.35 system cpu 24.07
real 164.92 user cpu 21.10 system cpu 23.08
real 144.62 user cpu 20.30 system cpu 23.61
real 144.68 user cpu 20.14 system cpu 24.81
AVG. 166.45    20.63    23.81
```


With writecache - Test 2

real 181.57 user cpu 20.79 system cpu 23.48
real 182.40 user cpu 20.54 system cpu 24.06
real 144.32 user cpu 20.22 system cpu 23.31
real 143.40 user cpu 20.40 system cpu 23.01
real 191.57 user cpu 20.62 system cpu 24.42
real 191.61 user cpu 20.46 system cpu 24.29
real 124.82 user cpu 20.66 system cpu 23.28
real 125.76 user cpu 20.79 system cpu 23.04
AVG. 160.68 20.56 23.61

With writecache - Test 3

real 172.15 user cpu 20.52 system cpu 24.32
real 172.16 user cpu 20.95 system cpu 24.44
real 145.48 user cpu 20.66 system cpu 23.04
real 144.43 user cpu 21.02 system cpu 22.29
real 161.03 user cpu 20.49 system cpu 24.48
real 160.90 user cpu 19.95 system cpu 24.03
real 120.16 user cpu 20.84 system cpu 24.05
real 120.14 user cpu 20.23 system cpu 24.00
real 171.41 user cpu 20.73 system cpu 24.28
real 171.41 user cpu 20.45 system cpu 24.74
AVG. 153.93 20.58 23.97

With writecache - Test 4

real 200.07 user cpu 20.25 system cpu 24.26
real 199.21 user cpu 20.81 system cpu 24.00
real 179.14 user cpu 21.14 system cpu 23.49
real 178.94 user cpu 20.98 system cpu 23.08
real 178.43 user cpu 20.38 system cpu 24.63
real 178.42 user cpu 20.26 system cpu 24.57
real 90.18 user cpu 20.70 system cpu 23.06
real 93.87 user cpu 21.08 system cpu 23.37
real 171.75 user cpu 20.70 system cpu 23.64
real 172.57 user cpu 21.19 system cpu 22.87
AVG. 164.26 20.75 23.70

With writecache - Test 5

real 173.27 user cpu 20.41 system cpu 23.99
real 173.15 user cpu 20.68 system cpu 23.70
real 179.49 user cpu 20.49 system cpu 24.16
real 179.36 user cpu 20.57 system cpu 24.23
real 152.35 user cpu 20.60 system cpu 24.43
real 152.37 user cpu 20.18 system cpu 23.87
real 148.51 user cpu 20.83 system cpu 24.13
real 148.56 user cpu 20.44 system cpu 23.68
real 123.96 user cpu 20.72 system cpu 23.47
real 118.32 user cpu 20.80 system cpu 22.48
AVG. 154.93 20.57 23.81

With writecache - Test 6

real 171.09 user cpu 20.04 system cpu 24.50
real 170.23 user cpu 20.50 system cpu 23.90
real 130.57 user cpu 20.55 system cpu 23.60
real 130.42 user cpu 20.98 system cpu 23.12
real 133.44 user cpu 20.60 system cpu 22.16
real 119.58 user cpu 20.58 system cpu 23.40
real 159.78 user cpu 21.18 system cpu 24.11
real 159.62 user cpu 20.82 system cpu 24.00
real 103.43 user cpu 20.29 system cpu 24.49
real 104.31 user cpu 20.61 system cpu 23.46

AVG. 138.25 20.62 23.67

With writecache - Test 7

real 156.24 user cpu 20.32 system cpu 24.22
real 162.26 user cpu 20.78 system cpu 23.28
real 181.49 user cpu 20.03 system cpu 24.21
real 181.31 user cpu 21.07 system cpu 23.53
real 138.40 user cpu 21.11 system cpu 23.29
real 138.28 user cpu 20.63 system cpu 23.21
real 175.73 user cpu 20.97 system cpu 24.16
real 175.60 user cpu 20.75 system cpu 23.47
real 171.34 user cpu 20.91 system cpu 23.80
real 171.31 user cpu 20.63 system cpu 24.60

AVG. 165.20 20.72 23.78

With writecache - Test 8

real 177.28 user cpu 20.30 system cpu 24.65

real 177.13 user cpu 20.73 system cpu 24.05

real 194.82 user cpu 20.64 system cpu 25.55

real 194.76 user cpu 20.82 system cpu 24.79

real 148.07 user cpu 20.73 system cpu 23.78

real 148.07 user cpu 20.36 system cpu 24.48

AVG. 173.36 20.60 24.55

With writecache - Test 9

real 184.41 user cpu 20.73 system cpu 23.87

real 183.55 user cpu 20.95 system cpu 23.84

real 139.76 user cpu 20.40 system cpu 22.96

real 139.93 user cpu 20.46 system cpu 23.70

real 109.36 user cpu 20.80 system cpu 22.62

real 132.23 user cpu 20.84 system cpu 23.78

real 155.91 user cpu 20.62 system cpu 23.79

real 156.06 user cpu 20.68 system cpu 24.81

real 101.68 user cpu 20.62 system cpu 23.35

real 101.83 user cpu 20.98 system cpu 23.84

AVG. 140.47 20.71 23.66

With writecache - Test 10

real 233.14 user cpu 21.12 system cpu 23.85
real 230.14 user cpu 21.07 system cpu 23.79
real 195.15 user cpu 20.56 system cpu 22.85
real 196.03 user cpu 20.69 system cpu 24.66
real 208.06 user cpu 21.44 system cpu 23.20
real 208.18 user cpu 20.60 system cpu 23.51
AVG. 211.78 20.91 23.64

With writecache - Test 11

real 179.30 user cpu 20.60 system cpu 24.36
real 178.47 user cpu 20.66 system cpu 23.85
real 222.55 user cpu 20.63 system cpu 23.44
real 223.38 user cpu 20.54 system cpu 23.44
real 160.62 user cpu 20.62 system cpu 23.50
real 161.57 user cpu 21.35 system cpu 23.92
real 158.12 user cpu 20.76 system cpu 24.07
real 178.39 user cpu 21.12 system cpu 23.08
real 223.19 user cpu 20.56 system cpu 23.61
real 224.07 user cpu 20.82 system cpu 24.81
AVG. 190.97 20.80 23.73

With writecache - Test 12

real 183.44 user cpu 20.81 system cpu 24.00
real 183.59 user cpu 20.70 system cpu 24.69

```
real 186.64 user cpu 21.13 system cpu 24.33
real 186.78 user cpu 20.56 system cpu 23.17
real 190.49 user cpu 20.95 system cpu 24.02
real 190.35 user cpu 20.71 system cpu 23.95
real 178.38 user cpu 21.42 system cpu 24.58
real 177.49 user cpu 20.82 system cpu 22.43
real 202.96 user cpu 21.01 system cpu 23.05
real 202.81 user cpu 21.35 system cpu 22.89
AVG. 188.29      20.99      23.55
```

Appendix E. The SAS System and DB2 partitioned databases

This appendix is a white paper written by P.W. Leathem of the IBM Corporation.

We will describe various approaches to implementing SAS on an RS/6000 SP and integrating SAS and DB2 partitioned database processing. The paper will describe how implementing processes, conventions, and an infrastructure can provide a scaleable, robust, and powerful environment for SAS computing on the SP. These approaches, processes, examples, and sample scripts are the result of work conducted at multiple customer locations. We will focus on the following topics:

- SAS on a cluster of SP nodes
 - SP configuration for cluster operations
 - Load balancing
 - Data sharing
- Practical flexibility
 - Scenarios
- SAS and DB2 partitioned databases on the RS/6000 SP
- SAS cleansing/transforming of data
- Extracting large amounts of data from db2 partitioned databases via SAS
- DB2 partitioned database parallel extract
 - CAE for AIX configuration
 - SQL modifications and table view definitions
 - Query restrictions
 - Parallel extract processing flow
 - Parallel extract SAS node alternatives
 - Single node parallel extract processing
 - Multiple node parallel extract processing
 - Multiple uniprocessor/multiple SMP SAS nodes parallel extract
 - Implementation issues
- Hybrid parallel extract
- Multiple DB2 logical nodes implementation
 - Hybrid parallel extract

- True parallel extract
- Summary

E.1 SAS on a cluster of SP nodes

In its simplest form, executing SAS on the SP entails installing SAS on an SP node and running it. Nodes on an SP currently range from a 66 Mhz to a 160 Mhz uniprocessor up to a 12-way SMP node. This paper will describe the implementation of a cluster of SAS nodes to provide processing power over and above what a single uniprocessor or SMP node provides. With the SP Switch and the built-in management facilities of PSSP, the SP provides a scaleable environment that, when properly planned for and implemented, is extremely robust, easily scaled, and reliable for 24X7 applications.

The following implementation steps will be discussed.

- SP configuration for cluster operation
- Load balancing
- Data sharing

E.1.1 SP configuration for cluster operation

The SP configuration that must be performed for the operation of a cluster of SAS nodes consists of the following steps:

1. Install SAS on each node.
2. Create saswork file system on each node.
3. Ensure common user ID and password.
4. Ensure common home directory.

SAS should be installed on every node participating in the SAS cluster. The nodes may or may not have the same version of SAS installed, as long as they are downward/upward compatible. This allows a convenient method of new version/release testing before cluster-wide roll out.

When executing SAS, a work directory must be specified for SAS to use as a temporary workspace. For the purposes of this discussion, the name of this directory will be */saswork*. A file system with a mount point of */saswork* must be defined on each node of the cluster.

Common user IDs, passwords, and home directories can be implemented in a variety of ways. PSSP user management provides a simple way to accomplish this by propagating the required passwd and group files to all

nodes of the SP and utilizing the *amd* daemon to automatically mount the user's home directory when they log in to any node. Another method of maintaining user IDs and passwords across multiple machines is NIS. Explicitly mounting home directories or using the SUN automount daemon will ensure common home directories. Each customer environment or preference will dictate which of these methods is chosen.

Once these steps are complete, we have a system by which a user may log in to any node in the cluster and execute SAS. We will now discuss a way of balancing the workload across all nodes in the cluster.

E.1.2 Load balancing

Load balancing across a cluster of SP nodes is accomplished via a LoadLeveler feature called Interactive Session Support (ISS), which is provided free with PSSP. ISS is used to distribute logins and application sessions across a pool of servers in a manner that is completely transparent to users and applications. ISS interfaces with a TCP/IP nameserver to translate machine names to Internet addresses. The function of ISS is to recommend to the nameserver the IP address to translate a machine name to a name based on the load of each machine. Instead of specifying the actual machine name of a particular server, the user specifies the name of a pool that has been set up by the LoadLeveler administrator. The set of nodes to be used for SAS processing is defined in a pool and given a name. In this example, we will use the name *spsas*. A user telnets, or uses SAS/Connect, specifying *spsas* as the hostname. ISS has recommended which actual IP address to route the session to based on the current load of the servers in the pool.

At this point, the user is ready to access data. In a load-balanced clustered environment, the user(s) can be sent to any node in the cluster, but not necessarily always to the same node. The user must be able to access the required data no matter which node the data physically resides on. Next, we will address sharing the data across all nodes in the SAS cluster.

E.1.3 Data sharing

Sharing of data across the cluster of SP nodes is accomplished via NFS the same way it would be for any cluster of machines. The difference is that the SP Switch makes NFS sharing of data practical due to the high bandwidth available between nodes. The file systems on which users store permanent SAS datasets that they wish to maintain will be termed data storage file systems. The steps for configuring the data storage file systems for the cluster include:

- Define data file system(s) on each node
- NFS export the file system to the other nodes
- NFS mount the other nodes' file systems on each node

To provide a manageable environment for all of your users, it is strongly recommended that you implement file and directory naming conventions for your data storage. A suggested naming convention for file system names is to include the node number (either SAS node number or SP node number) in the file system name. It is also recommended that users utilize subdirectories (using their user ID as the directory name) to prevent file-naming conflicts. For example, it is very common for two different users to both specify an output dataset named *junk* or *test*. If they write output to a common directory rather than to their own subdirectory, conflicts will arise.

- For example, let us assume, for the purpose of this discussion, that we have four nodes in our SAS cluster and that the data file systems have the following names: /nxx/data where xx is the node number (01, 02, 03, or 04)

Each user will have a subdirectory under this file system based on their user ID; so, the user, jqpublic, on node 02 would then use the following: /n02/data/jqpublic for storing SAS data files.

Once each node has the data file system defined, NFS can export that file system to the other nodes. Now, each node can NFS mount the other node file systems over the SP switch.

We now have a cluster of SP nodes that is referenced by the name *spsas*. The nodes all have SAS installed, and each has a SAS work file system (*/saswork*). A data storage file system is defined locally on each node and each node has NFS mounted the other nodes' data storage file system(s).

Figure 31 on page 125 represents the configuration of a cluster of four nodes providing SAS processing power.

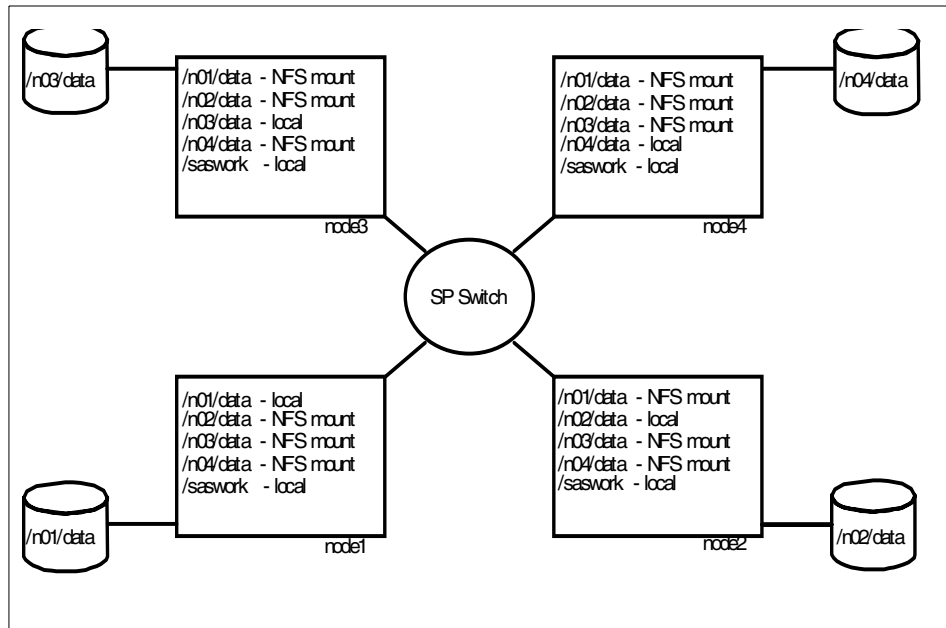


Figure 31. SAS cluster configuration on SP, LoadLeveler ISS Pool -spsas

There are two areas of interest in our configuration that must be addressed. The first is the performance implications of writing to NFS mounted file systems. The second is the location of data, that is, knowing on which file system the data of interest is located. Each of these can be addressed easily with proper planning and, by setting a few conventions and SAS operational procedures, can be made transparent to the users. While reading files from an NFS file system provides acceptable performance, writing large files to an NFS mounted file system incurs severe performance penalties. When using ISS to route users to the least utilized node, it is important that this NFS write problem be resolved. If the user is sent to node 4, they need to specify /n04/data/userid as the output library; if sent to node 2, the user needs to specify /n02/data/userid as the output library. However, you cannot make the assumption that the user will always know to which node he or she was assigned; so, to make this transparent to the user, a simple addition/modification to the autoexec.sas file can be made. Simply set a macro variable `&local` equal to the appropriate file system for that user based on the hostname that has been assigned to their login or connect session. The user now simply specifies the libname `&local` for all large output datasets, and their files will be saved in the appropriate directory structure.

Now that we have a convention and method for ensuring that all writes of large output datasets are to a local file system, we need to implement a way to locate a dataset that we wish to work with. The following example will help to illustrate this requirement.

- For example, a user telnets to *spsas* on Friday and is routed to node 3. The *autoexec.sas* file has set the variable *&local* to be */n03/data/userid*. The user creates the SAS dataset *myout.ssd01* specifying *libname=&local*. The user now logs off and goes home for the weekend. The same user telnets to *spsas* on Monday morning and is routed to node 1 this time. The user wants to perform some additional processing on the dataset *myout.ssd01* that was created on Friday. The dataset *myout.ssd01* is in the file system */n03/data/userid*, but this user is now assigned to */n01/data/userid*. We must provide a way for the user to know this. The simplest way to accomplish this is to provide an AIX shell script that when invoked with a dataset name, searches the data file systems on every node in the SAS cluster and returns the directory that the dataset resides on. A script developed at one customer site is shown below. This script checks the four nodes of the SAS cluster for a specified dataset.

```
#!/bin/ksh
# script name: /usr/local/bin/locate
clear
FILENAME=${1}""
EXISTS=0
echo \n"Date Created   Owner   Group  File Name"\n
FILES=`find /n${DIR}/data/ -name "${FILENAME}" -print `
if [[ ${FILES} != '' ]]
then
    EXISTS=1  ls -l ${FILES} |awk ` { print $6,$7,$8," ",$3,"
    ",$4," ",$9 } `
fi
if [[ ${EXISTS} = '0' ]]
then
    echo " No files matching criteria "${FILENAME}\n
fi
```

With this script, a user can locate a dataset and then specify the appropriate *libname* to access this dataset. With a small amount of additional SAS initialization processing, we can take this one step further and make locating datasets completely transparent to the user. This involves implementing a

metadata concept, and your SAS representative should be contacted for alternatives applicable to your specific environment.

E.1.4 Practical flexibility

Different workloads may run best on different machines. Long running statistical models may run better on a uniprocessor as opposed to sharing the resources with a high number of ad hoc users on an SMP node. In this environment, both types of nodes can be incorporated, and workloads can be segregated to provide optimum response to all user types. A discussion of some of the scenarios that may be encountered will show the power and flexibility of running SAS in this environment.

Scenarios:

- A customer has a group of users that are primarily ad hoc reporting users. There are two or three users that run complex statistical models that produce data that the reporting users then process. An SMP node is configured as the only node in the ISS pool *sasrpt*. Two uniprocessors are in a *saspower* pool. The data file systems are shared among all three machines. The power users utilize the *saspower* pool to run their jobs and ad hoc users use the *sasrpt* pool. If the ad hoc workload increases, another SMP node can be added to the *sasrpt* pool and the data redistributed without any inconvenience to users. Users do not even realize that an additional machine was added. Likewise, if six months from now the power users need additional horsepower, another uniprocessor node can be added to the *saspower* pool. This approach will prevent power users from adversely impacting the ad hoc users, but the data generated by the power users is available to all.
- A cluster is configured with five nodes. One of the nodes takes a hit and is down. If the IBM HACMP for machine takeover has been implemented, the data file system from the problem node is taken over by another node; the NFS mounts are adjusted, and processing continues. Once the problem node is repaired, normal processing can be resumed.
- A cluster is configured with four nodes. A new SAS version is installed on one node and complete regression testing can be performed against real data. Once verified, the new version can be propagated to the remaining nodes.
- A cluster is configured with six nodes. Four of these nodes are utilized for parallel extracts from a parallel database. Since these extracts consume large amounts of resources on these nodes, additional users are routed via ISS to the other two nodes while these extracts are running. Once the

extracts are complete, the datasets are available to all users in the SAS cluster.

Parallel extract will be discussed later in this document.

E.1.5 SAS and DB2 partitioned databases on the RS/6000 SP

There are two topics that will be discussed in regard to using SAS and DB2 partitioned databases on an SP. The first is the use of SAS as a data transformer/cleanser to prepare data to be loaded into a partitioned database. The second is the extracting of large amounts of data from a DB2 partitioned database for processing by SAS.

E.1.6 SAS cleansing/transforming of data

Data to be loaded into a parallel database frequently requires some type of transformation, cleanup, or reduction. Any of these operations will be generically termed *cleansing* for this discussion. This cleansing can take place where the data currently exists (MVS for example) or on the SP since this is the final destination. For this discussion, it is assumed the data is currently on MVS and is to be cleansed on the SP.

There are three steps involved in moving the data from its current location to its final destination in the parallel database. These steps are the following:

- Transfer the data to the SP
- *Cleanse* the data
- Load the data

Transferring the data to the SP

Transferring the data is typically done with standard FTP or a product, such as Client Input Output/sockets (CLIO/S). CLIO/S is an IBM product that provides for high speed data transfer from an MVS system to an RS/6000 or SP node. The destination on the target SP node can be either a disk file or an AIX named pipe. An AIX named pipe is a special file that can be written to by one process and read by another via a memory buffer. A disk file must be used if your SAS cleansing procedure must read through the data multiple times. If the SP is not at AIX 4.2 or higher, splitting datasets larger than 2 GB into smaller chunks (< 2 GB) will be required. AIX named pipes can be used if your SAS cleansing procedure only needs to read the data once. AIX named pipes reduce the disk space required on the SP node and eliminate the 2GB file size limitation on pre-AIX 4.2 systems. To create a named pipe, use the *mknod* command specifying the name of the pipe. When the FTP or CLIO/S FTP is performed, specify the name of the pipe as the target file.

Cleansing the Data

Once the data has been transferred to the SP node (either on disk or in a pipe waiting to be read), the cleansing can take place. The final output of the SAS proc(s) must be either an ASCII delimited or non-delimited file, which may include packed decimal and binary numeric fields. The restrictions on having packed or binary numeric data in the dataset to be loaded are documented in the DB2 UDB Command Reference. Again, the output of the SAS proc(s) can be a disk file or an AIX named pipe. Since the DB2 split and load processes are single pass, there is no instance that would require writing to a disk file. The use of a named pipe will again save disk space and eliminate file size limitations.

Loading the Data

Now that the cleansed data exists in a disk file or in a pipe waiting to be read, the data can be split and loaded. Note that the use of AIX named pipes requires both the pipe writer and pipe reader be executing concurrently. If you submit the FTP job or command, the cleansing program must be running to read data from the FTP pipe. To pipe the entire process, all pieces (FTP, SAS cleanser, and autoloader) must be started together. Figure 32 represents the processing flow when using SAS as a data *cleanser*.

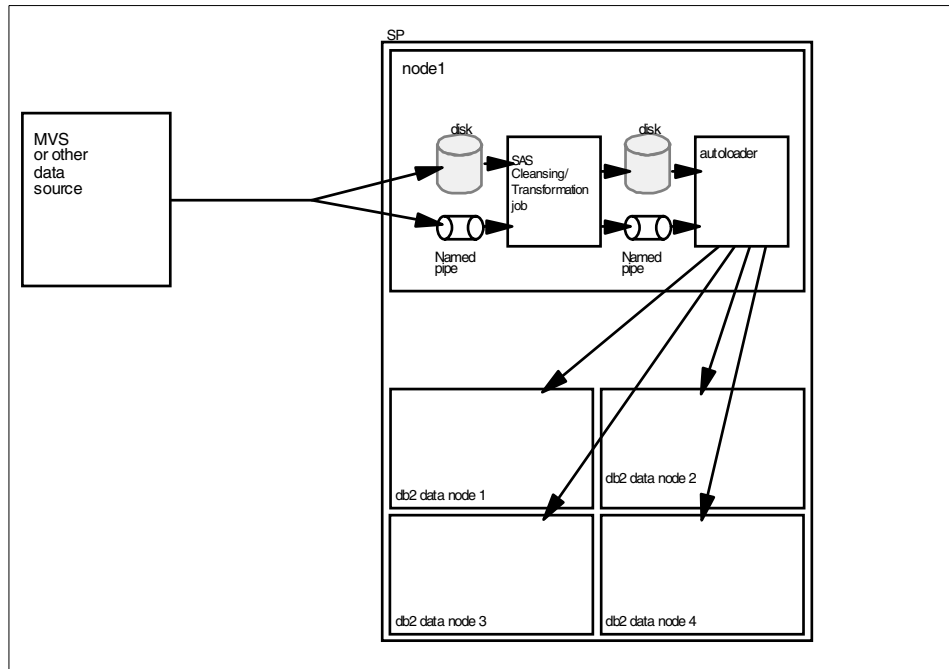


Figure 32. Data cleansing/transforming with SAS on SP

E.2 Extracting large amounts of data from DB2 partitioned databases

The conventional method of accessing DB2 partitioned databases is to configure DB2 Client Application Enabler (CAE) on a non-database node to point to a node in the database complex, which then becomes the coordinator node. Programs then utilize CAE to connect to the database via a TCP/IP socket and issue SQL. The results are returned via the TCP/IP socket to the application. This paper will assume that SAS is running on a non-database node(s).

With no custom implementation steps, the default access of SAS to a DB2 partitioned database is as follows:

1. The SAS program connects to a database via the coordinator node.
2. The query is sent to the coordinator node, and the result set is gathered and then sent back to the SAS program.
3. The coordinator node serializes the return of the result set and adversely impacts the response time of *large* extracts of data.
4. Performance degradation is encountered because the coordinator must accept the result set from each of the data nodes, combine them, and then forward the result set to the SAS program. This access is depicted in Figure 33 on page 131.

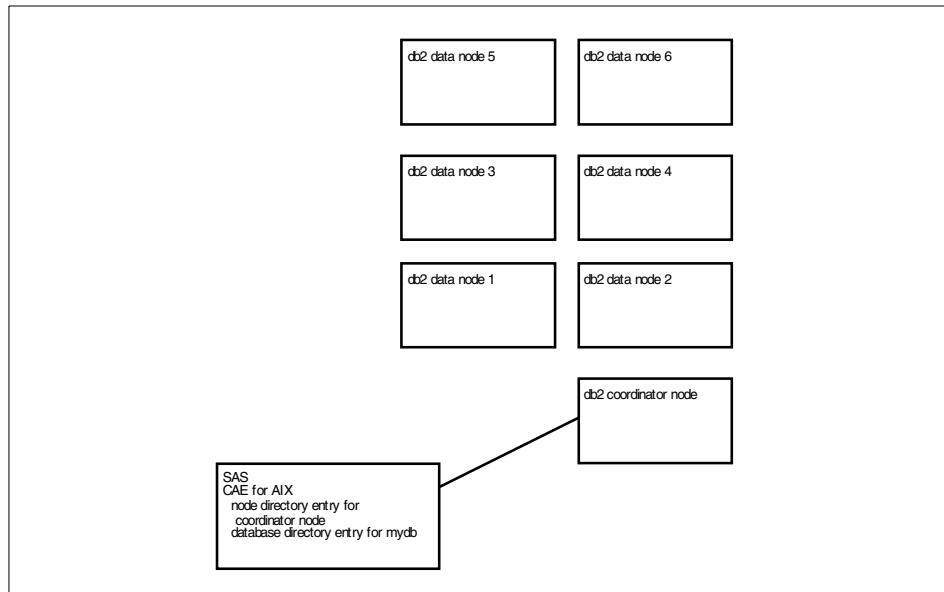


Figure 33. SAS to DB2 partitioned database - Default access

E.3 SAS to DB2 partitioned database parallel extract

A DB2 partitioned database is optimized to analyze extremely large amounts of data and return *the answer*, where *the answer* implies a small result set. Business requirements often necessitate the extraction of *large* amounts of data from a partitioned database for processing by SAS. But just how much is a *large* amount of data? This amount will depend on the response time that can be tolerated, but it is typically over 500 MB. While results sets this size are not large in relation to the amount of data in a warehouse or data mart, they are large when compared to what most ad hoc or reporting users process.

To greatly improve the extraction large amounts of data from a DB2 partitioned database, a method called parallel extract can be used. The high-level concept of parallel extract is to operate on each node in the partitioned database environment independently and then merge the results received from each of the nodes. This is accomplished by connecting directly to each of the data nodes and extracting only the data held on that particular node. This eliminates the true *coordinator node* function. Figure 34 on page 132 represents this parallel extract access.

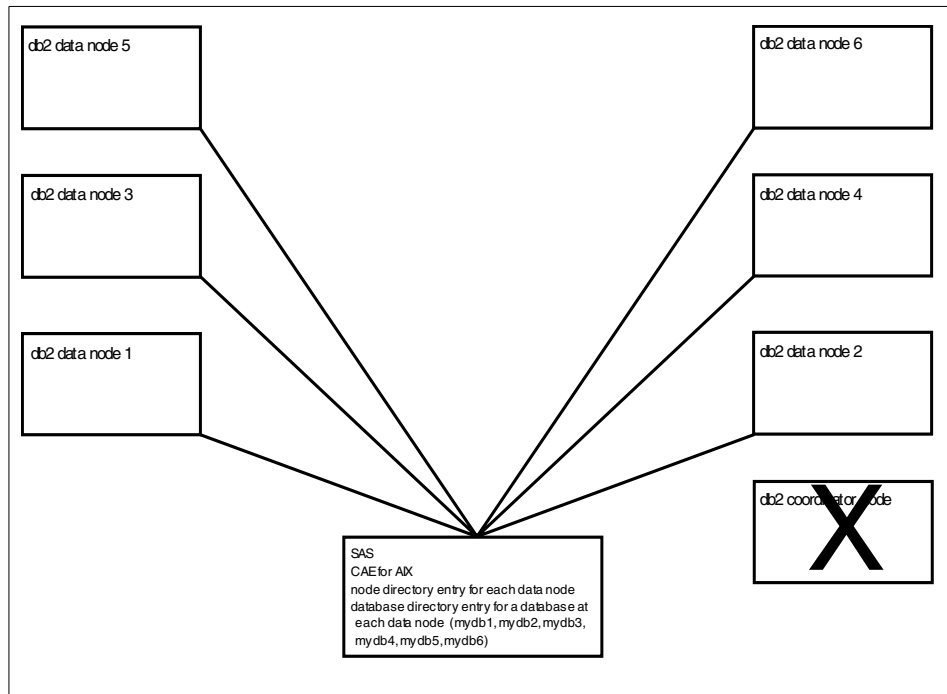


Figure 34. SAS to DB2 partitioned database parallel extract access

The following are required to support this type of access

- Infrastructure (CAE configuration and View Definition)
- SQL modifications
- SQL restrictions

The infrastructure that must be in place is implemented by configuring CAE for AIX correctly and, optionally, defining the required views to the tables. These views are only required if users will be accessing views or if the DBAs require them.

E.3.1 CAE for AIX configuration

The connection to the database is provided by CAE for AIX. This is done to reduce the resources of the SAS node(s) required to connect to the database. This method also isolates the database from the SAS node(s). If the SAS node is part of the database instance, a large amount of memory will be taken up by DB2PE or UDB leaving less for SAS processing. To document the

configuration of CAE to support parallel extracts, the following assumptions must be made.

- There are six data nodes
- The database name is mydb
- The switch interface names are sphps01, sphps02, ..., sphps06

The first step is to catalog the required TCP/IP nodes on the SAS node with CAE. Examples of the node name and remote host parameters of the catalog tcpip node command are the following:

Table 21. Node name and remote host

Node Name	Remote host
dbdatan1	1sphps0
dbdatan	2sphps02
dbdatan3	3sphps0
dbdatan	4sphps04
dbdatan5	sphps05
dbdatan6	sphps06

Now the following databases can be catalogued.

Table 22. Catalogued databases

Alias	Database name	Node
mydb1	mydb	dbdatan1
mydb2	mydb	dbdatan2
mydb3	mydb	dbdatan3
mydb4	mydb	dbdatan4
mydb5	mydb	dbdatan5

E.3.2 SQL modifications and table view definitions

To support parallel extraction of data from a DB2 partitioned database environment, slight modifications are required to either the SQL statements used or the table view definitions. DB2PE and UDB provide a “*where nodenumber(colname) = clause*”, which is an extension to standard SQL. The argument to the where clause can be either the keyword *current node* or an integer denoting the DB2 node number to be operated on. If *current node*

is specified, the node executing the query is the only node from which data is considered. If users will be accessing the base tables directly (not via views), the *where nodenumber(colname)= current node* can be specified directly in the SAS Proc SQL.

If views are being used, the clause cannot be specified in the SAS proc as DB2PE, and UDB only supports this clause when applied to base tables. If views are desired or required, the *where nodenumber(colname) = clause* can be specified in the view definition itself. The use of the current node keyword requires that the SAS node(s) has a database cataloged in CAE for every data node.

E.3.3 Query restrictions

If certain restrictions are not applied to all parallel extracts, the benefits of this method may not be realized and/or the results may be unpredictable. These restrictions include the following:

- Joins of two or more tables must result in a collocated join by DB2PE or UDB.
- Functions, such as count distinct, sum, average, etc., must be done after the extract via SAS processing.
- Order by and group by will not be correct unless further processing is performed in SAS

E.3.4 Parallel extract processing flow

Now that CAE is configured and the required views are defined, we have the ability to extract data from each of the data nodes. A SAS job that issues SQL against a database and writes the data to a SAS dataset is the starting point of a parallel extract. A process must be implemented that can take this SAS job and generate multiple SAS jobs to extract data from each node. Once these SAS jobs have been executed, the output from each must be merged together for the complete answer. The goal is to automate this process via a program or UNIX script. The processing flow of a parallel extract from a DB2 partitioned database environment is represented in Figure 35 on page 135.

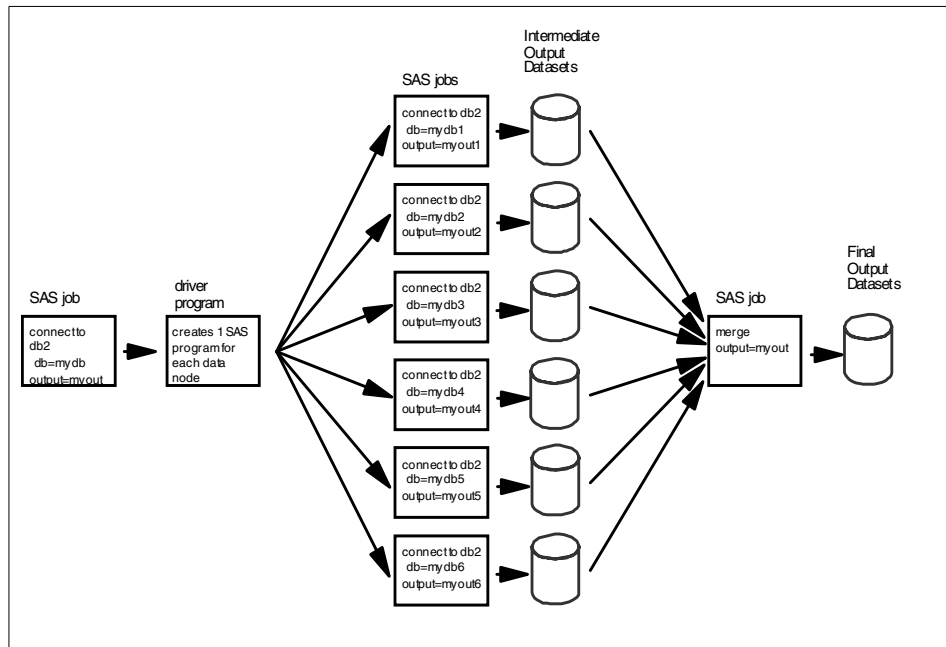


Figure 35. SAS to DB2 partitioned database access

E.3.5 Parallel extract SAS node alternatives

The SAS processing power for a parallel extract can be provided by either an SMP node or a group of uniprocessor or SMP nodes. The number and types of SAS nodes required are dictated by response time requirements and the available resources. As discussed previously in this document, a cluster of SAS nodes can be seamlessly integrated into this processing environment. For parallel extracts, the following performance characteristics were observed at one client location.

If the SAS nodes and database nodes are RS/6000 model 390 class uniprocessors, a 1-SAS node to 4 or 5-database nodes relationship is fairly balanced. This means that if a SAS node was running an extract against four or five database nodes concurrently, all processor resources were consumed on the SAS node. If the SAS node was running against less than four database nodes, the SAS node processor has idle time left. This should only be used as a rule of thumb. A parallel extract was run against 20 database nodes using a single uniprocessor SAS node, and it was successful. This takes longer, but it was still significantly faster than the standard access. If choosing between multiple uniprocessors or a single SMP node for parallel

extracts, the single SMP node is slightly easier to manage. For very large partitioned database environments, multiple SMP nodes may be utilized.

An example of both a single node parallel extract and a multiple node parallel extract will be illustrated next.

E.3.6 Single node parallel extract processing

When using a single uniprocessor or SMP node for the parallel extract, the implementation of the parallel extract processing flow is straightforward. The steps are as follows:

1. Generate the node-specific SAS programs.
2. Generate the merge program.
3. Execute the node-specific programs concurrently.
4. When all have completed, execute the merge program.

The driver program is invoked specifying *myjob.sas* as the program to run. The driver program generates six SAS database access programs and a SAS merge program. The six SAS database access programs are then submitted. Each of these SAS programs writes a corresponding intermediate SAS dataset named *myoutx.ssd01*. Once the driver program detects that all of the database access programs have completed, the SAS merge program is submitted. The SAS merge program then writes the final output dataset.

This is represented pictorially in Figure 36 on page 137.

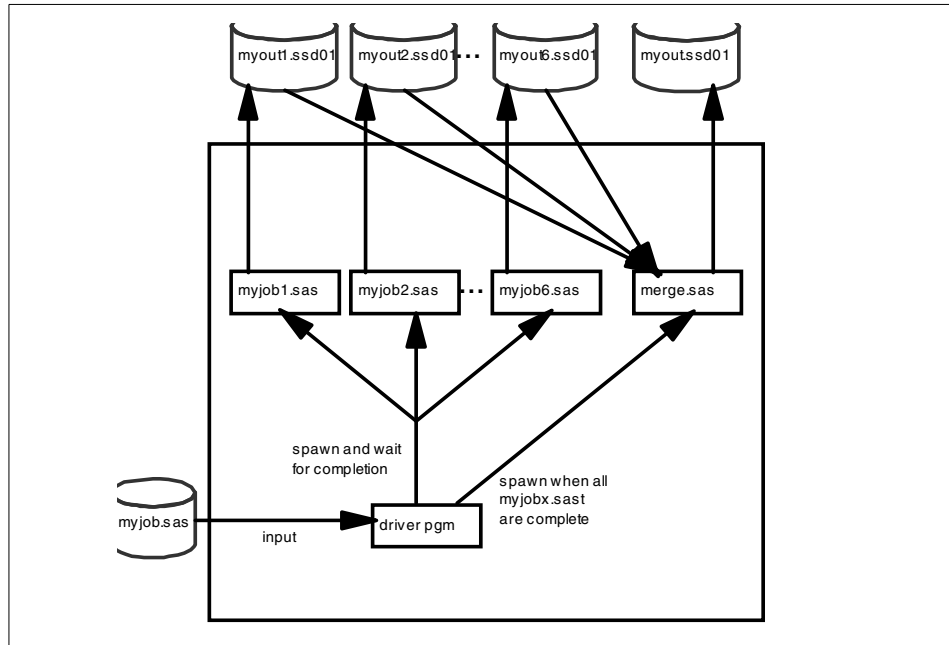


Figure 36. Single SAS node parallel extract implementation

E.3.7 Multiple node parallel extract processing

When using multiple uniprocessor or SMP nodes for the parallel extract, the parallel extract processing flow implementation is slightly modified. The execution of the generated SAS programs is distributed evenly to the participating nodes. As previously discussed, the participating nodes need to be configured as a SAS cluster. The steps are as follows:

1. Generate the node-specific SAS programs.
2. Generate the merge program.
3. Distribute and execute the node-specific programs concurrently.
4. When all have completed, execute the merge program.

If we have two nodes in our SAS cluster, the following will occur: The driver program is invoked specifying *myjob.sas* as the program to run. The driver program generates six SAS database access programs and a SAS merge program. Three of the SAS database access programs are submitted on the local node, and three are submitted via *rsh* to the other SAS node. Each of these SAS programs writes a corresponding intermediate SAS dataset

named *myoutx.ssd01*. Once the driver program detects that all of the database access programs have completed, the SAS merge program is submitted on the local SAS node. The SAS merge program writes the final output dataset. This is represented in Figure 37.

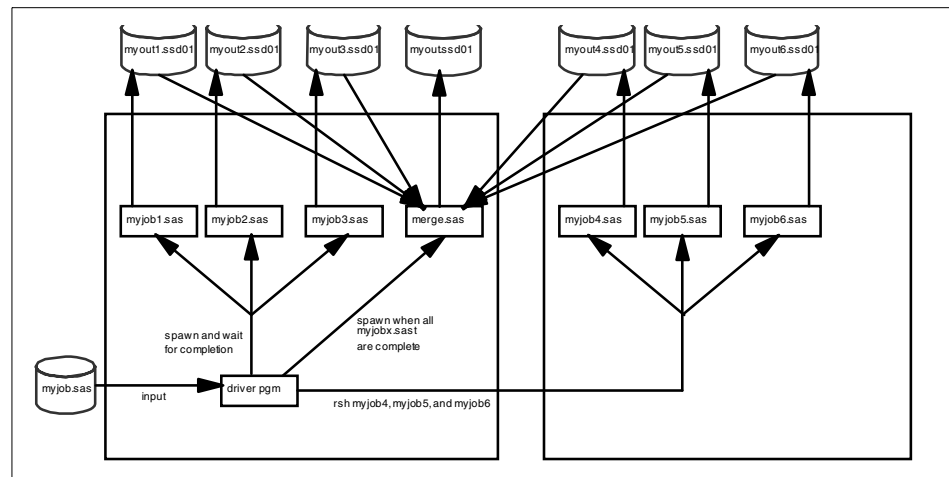


Figure 37. Multiple SAS node parallel extract implementation

E.3.8 Multiple SAS nodes parallel extract implementation

When using multiple uniprocessors or multiple SMP nodes for a parallel extract, the implementation of a cluster of SAS nodes discussed previously is required. This allows multiple nodes to extract data from a set of the data nodes and one of the SAS nodes to merge all of these intermediate datasets together. SAS parallel extract users must also be able to *rsh* commands to the other participating SAS nodes.

There are also a number of issues related to node availability that should be considered. For example, if one of the SAS nodes is not operational, the extract will not be complete, and the resources expended by the other nodes and the database nodes are wasted. The following components on each SAS node should be checked before each submission of a parallel extract:

- Node is up
- Switch is operational
- File systems used (saswork and data storage file systems) are ready
- Database is up

The driver program can check each of these conditions before actually submitting the parallel extract jobs.

E.4 Hybrid parallel extract

For very large numbers of data nodes (>50) or when multiple logical database nodes are running on SMP processors, a hybrid of the standard access and the parallel extract may be desired or required. In a true parallel extract, each database node is returning data for only one node, the node itself. In a hybrid parallel extract, the node accepting the SQL from the SAS program returns data for n number of nodes. Let us now modify our assumptions to illustrate a hybrid parallel extract.

Assumptions:

- Six data nodes
- The odd number nodes, dbdatan1, dbdatan2, and dbdatan3, will function as true coordinator nodes returning data from themselves and one other data nodes.

After cataloging three TCP/IP nodes and three databases with CAE, we need to define a view to select data from two nodes. If there are six data nodes and the odd ones have been cataloged, then $n = 2$. The view that must be defined has the following select syntax:

```
select col1,col2,...,coln from mytable
  where nodenumber(colname) = current node and
         nodenumber(colname) = (current node + 1)
```

The where clause could also be specified directly in the SAS Proc if base tables are being accessed. This would result in three coordinator nodes with each coordinator returning the data for two nodes. This would reduce the number of SAS jobs being executed but increase the time required for the extract because each coordinator is returning data for two data nodes.

The database access for this hybrid parallel extract is shown in Figure 38 on page 140.

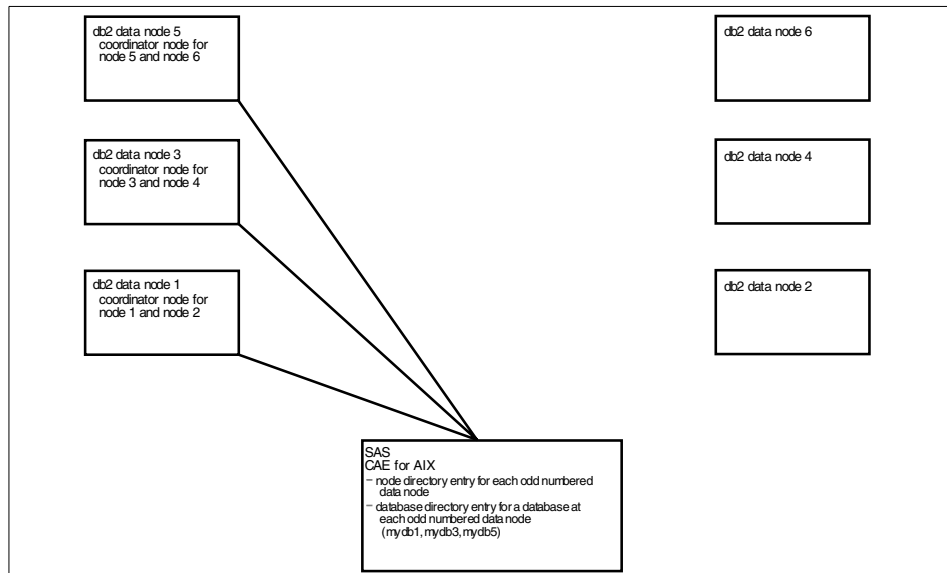


Figure 38. SAS to DB2 partitioned database hybrid parallel extract

E.5 Multiple DB2 logical nodes implementation

The use of SMP nodes running multiple logical DB2 nodes is increasing and will continue to do so. This requires slightly more planning and infrastructure than previously discussed. This is due to the fact that when you catalog a node via CAE, you can only catalog a physical machine (IP address). The first logical node running on that machine will be the one that accepts remote connections. There are two options for proceeding:

- Use a Hybrid parallel extract with one coordinator per physical node
- Implement True parallel extract with SAS on every data node

To describe the first of these scenarios, assume there are two SMP nodes, each running four DB2 logical nodes. Two databases, *mydb1* and *mydb2*, would be cataloged via CAE. This configuration is depicted in Figure 39 on page 141.

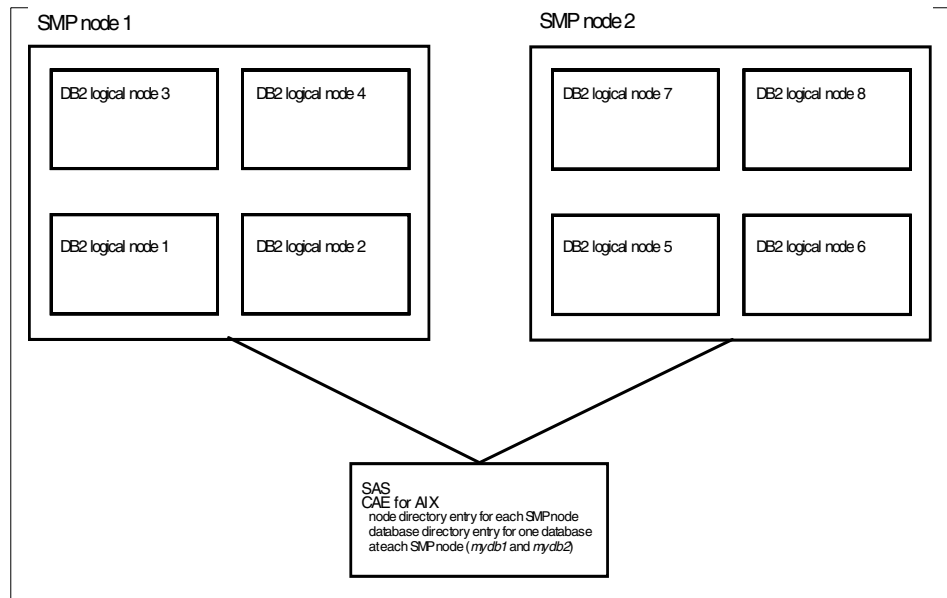


Figure 39. Multiple logical DB2 nodes per SMP node

E.5.1 Hybrid parallel extract

To perform a hybrid parallel extract, a single view is required. The selected syntax is the following:

```
select col1,col2,...,coln from mytable
where nodenumber(colname) = current node and
      nodenumber(colname) = (current node + 1)
      nodenumber(colname) = (current node + 2)
      nodenumber(colname) = (current node + 3)
```

The driver program will generate two SAS jobs: One specifying *mydb1* and the other *mydb2*. When these SAS jobs access the view, the first logical DB2 node on each SMP node would become the coordinator node and would return data for the four DB2 logical nodes on that machine. Again, this reduces the number of SAS jobs required but will take longer to complete than a true parallel extract. This is represented in Figure 40 on page 142.

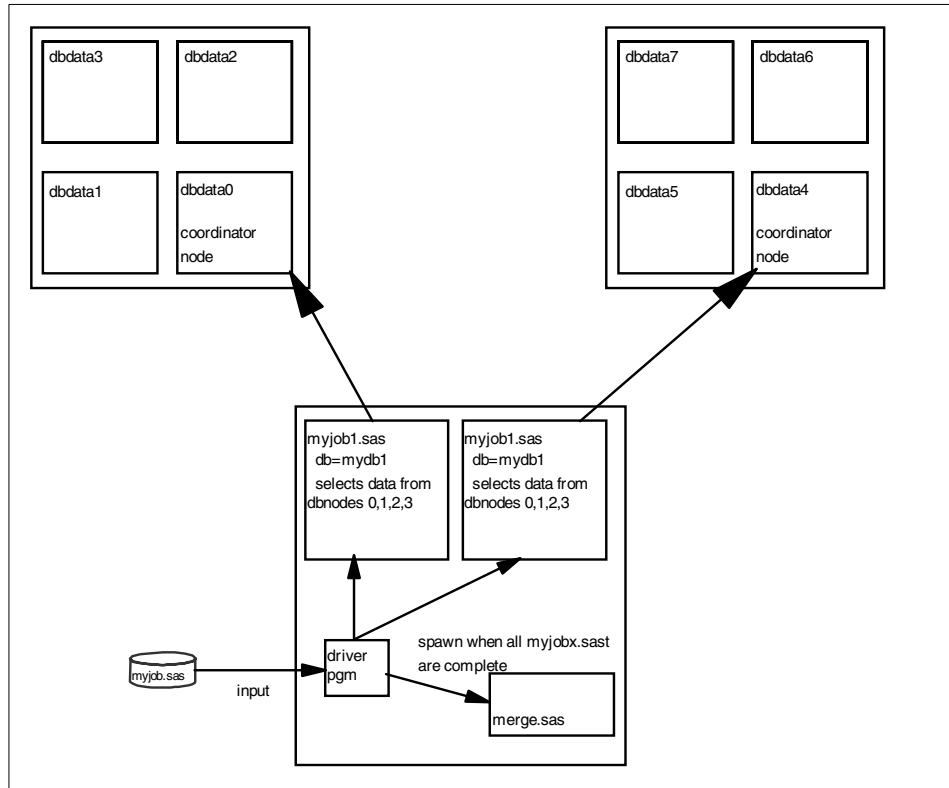


Figure 40. Hybrid parallel extract

E.5.2 True parallel extract

To perform a true parallel extract with multiple logical DB2 nodes the configuration requirements are different, and the submission of the node specific SAS jobs requires an additional parameter:

- Configuration.
- SAS must be installed on the database machines.
- No CAE configuration is required.
- Data Storage file systems must be available for that node's intermediate Datasets.
- Data Storage file systems must be shared as in the cluster configuration.

SAS Job Submission:

When the node specific SAS jobs are distributed via *rsh*, an environment variable must be set before the execution of each of the SAS jobs. The DB2NODE parameter allows the user to specify which logical node to operate on. This implementation is depicted in Figure 41.

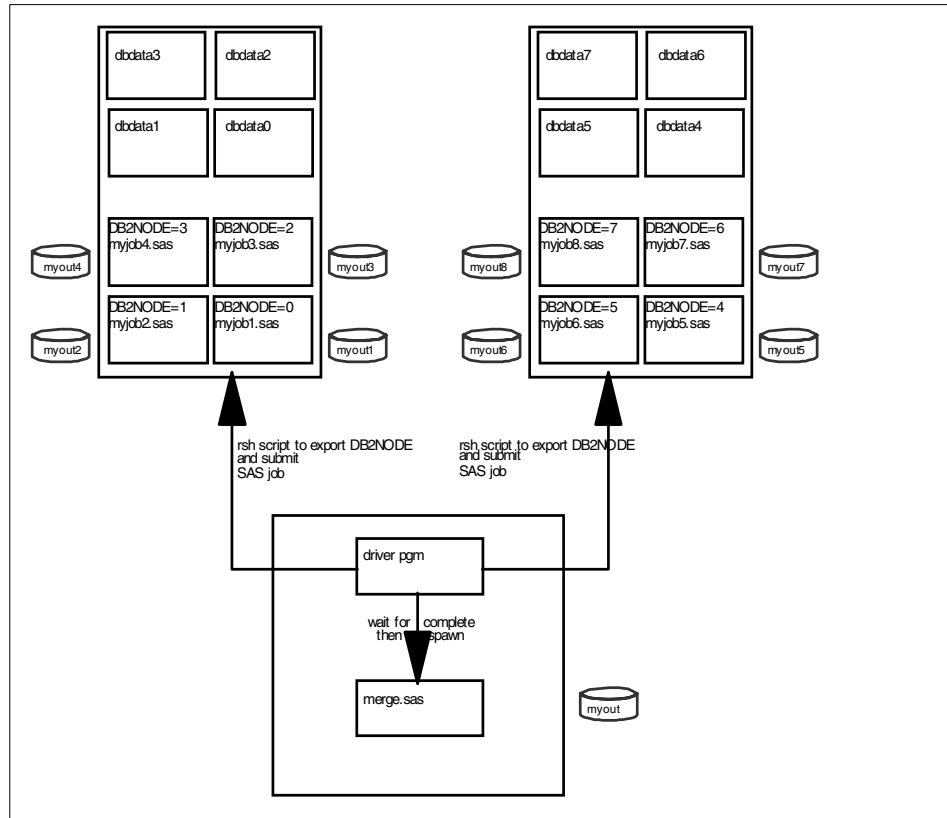


Figure 41. True parallel extract

E.5.3 Summary

The approaches discussed in this paper are not meant to be presented as the only way to implement SAS on an RS/6000 SP or to integrate SAS and DB2 partitioned database processing. They are merely an example of how implementing processes, conventions, and an infrastructure can provide a scalable, robust, and powerful environment for SAS computing on the SP. These approaches and processes were developed in response to delivering solutions at multiple customer locations. They were refined as issues were encountered and new ideas developed. Of course, every site is unique, and

the skills of customer personnel will generate even more new ideas and possibilities. Likewise, should these recommendations be implemented at your site, further refinement, tuning, and improvements will need to be made and tailored to your specific environment, personnel, and requirements.

Appendix F. The SAS system and GPFS

The GPFS file system on RS/6000 SP is a feature of SP systems that delivers very fast data transfer rates. SAS performance will benefit from this, and, therefore, the feature is of interest to SAS users. We include the following article about the GPFS file system here.

Introduction

IBM has introduced a general purpose parallel file system, called the General Parallel File System (GPFS), which runs on the RS/6000 SP. One of the ideas behind developing a parallel file system for the SP is an idea that drives all parallel implementations: Spread workload across many nodes in order to scale up the amount of work you can do while delivering excellent performance.

The RS/6000 and AIX are the platforms of choice for many SAS software users. It was decided to proceed with a project to see whether the SAS System could benefit from the use of GPFS on the RS/6000 SP. The goals of the project were twofold: First, to prove that GPFS and the SAS System were functional, and second, to measure performance characteristics and scalability.

The test project was run at the RS/6000 Teraplex Integration Center in Poughkeepsie, NY. The Teraplex center is used for testing customer's real world environments, with a focus on large-scale Business Intelligence environments. The Teraplex Integration center helps existing IBM customers, potential new customers, business partners, and IBM hardware and software development by integrating, optimizing, and stressing BI systems on gigabyte to terabyte class databases.

The hardware and software configuration, tests, test results, and performance summarization's will be described in this appendix. We will present a GPFS overview, which will describe the capabilities of GPFS, as well as a high-level description of GPFS and how it was configured in these tests. We will also share our experiences setting up and using GPFS in the SAS System environment.

F.1 GPFS overview

The IBM General Parallel File System for AIX (GPFS) provides file system services to parallel and serial applications running on the RS/6000 SP. GPFS allows users shared access to files that may span multiple disk drives on multiple SP nodes.

AIX file system utilities are supported by GPFS. All commands that are currently used by AIX users can continue to be used unchanged for GPFS file system users. The only unique commands are those for administering the GPFS file system, and these commands are available through the automated help panels.

GPFS allows parallel applications simultaneous access to the same files or different files from any node in the configuration while managing a high level of control over all file system operations. It offers extremely high recoverability while maximizing data accessibility.

GPFS improves system performance in a number of ways. It allows multiple processes or applications on all nodes of the SP simultaneous access to the same file. It increases aggregate bandwidth of the file system by spreading reads and writes across multiple disks. It balances the work load evenly across all disks to maximize their combined throughput. It also supports large amounts of data and allows you to have bigger file systems by removing the physical limitation of a single node. It also allows for concurrent reads and concurrent writes to the file system.

GPFS uses a sophisticated token management system to guarantee data consistency while providing multiple independent paths to the same file, by the same name, from anywhere in the SP system.

GPFS is also a logging file system that creates separate logs for each SP node. These logs record the allocation of meta data and aid fast recovery and consistency of data in the event of node failure, even when a node fails while modifying file data.

GPFS allows for adding or deleting disks while the file system is mounted. When the time is right and system demand is low, you can rebalance the file system across all currently-configured disks.

F.1.1 Hardware configuration

The system used at the Teraplex center for the GPFS tests was an eight-node RS/6000 SP. Each of the eight nodes was an identically-configured Symmetric Multiprocessor (called a high node). The eight nodes spanned two SP frames, and all the nodes were connected by the SP Switch, which is a scalable, high-speed, high-availability network.

Each high node had the following configuration:

- Eight - 604 112 MHz PowerPC processors
- 2 GB Memory, two 2.2 GB Internal disks

- Two Serial Storage Architecture (SSA) adapters
- 16 - 4.5GB SSA disks
- Four loops (two loops per adapter), with four disks on each loop

F.1.2 Software configuration

The software configuration was as follows:

AIX Version 4.2.1 NFS Version 2.0 Parallel Systems Support Program (PSSP) 2.4 GPFS Version 1.1 SAS System, Release 6.12.

F.1.3 Physical disk configuration

Each of the nodes had an identical physical disk configuration of two SSA adapters, with two loops on each adapter, and four disks on each loop. See Figure 42 for a diagram of the configuration.

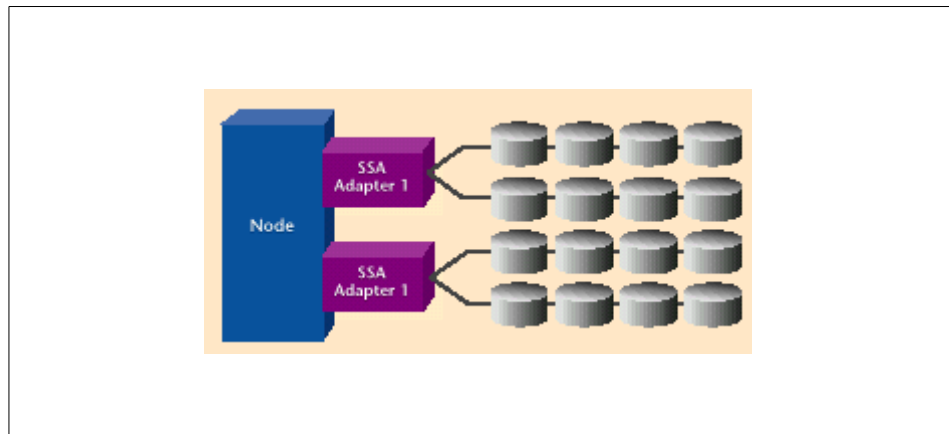


Figure 42. Node configuration

F.1.4 System configuration

The tests were run against three types of file systems: Local, NFS-mounted across the switch, and GPFS. The first set of tests was run against an identical configuration of disks on local, NFS, and GPFS. The second set of tests was run on GPFS where the number of disks and adapters were four times the size of the initial configuration. The disk configurations for each of the different file systems are described in the following sections.

F.1.5 Local disk configuration

Two file systems were defined in each file system type, /DATAFS and /TEMPFS. The /DATAFS file system was used to store the newly-created Household and Person data files. The /TEMPFS file system was used as the \saswork directory.

The disk configuration for the local file system was six 4.5 GB Serial Storage Architecture (SSA) disks for both /DATAFS and /TEMPFS, for a total of 12 disks. Each node of the system configuration had two SSA adapters with 16 disks. Each node of the system configuration had two SSA adapters with 16 disks. Each node of the system configuration had two SSA adapters with 16 disks. Each loop having four disks. The file systems used 12 of those 16 disks.

See Figure 43 for the Local and NFS configurations.

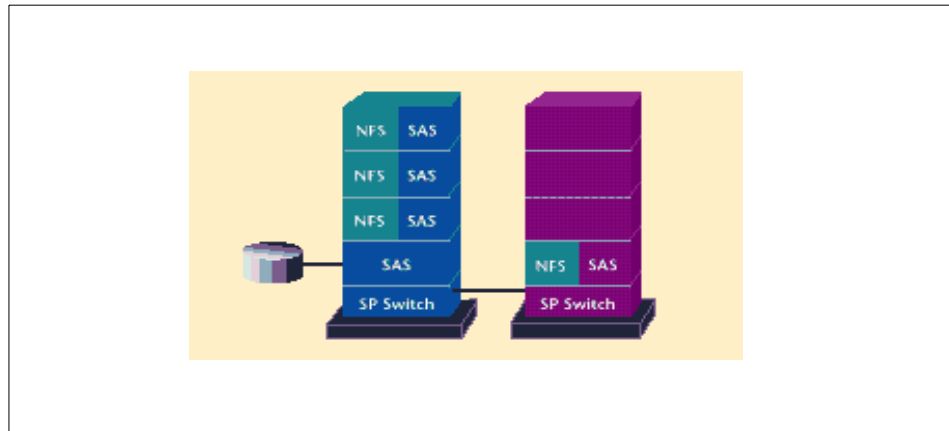


Figure 43. Local and NFS configurations

F.1.6 NFS disk configuration

The local file systems were NFS mounted across the SP Switch for the NFS tests. NFS Version 2.0 is used in AIX 4.2.1. The tests were run on one and four nodes; each of those nodes was different than the node on which the disk was locally attached.

Refer to Figure 43 for the NFS configuration.

F.1.7 GPFS disk configuration

The GPFS disk configuration was identical to the local configuration, in that the /GPFS/DATAFS and /GPFS/TEMPFS file systems each consisted of six 4.5 GB disks on a single node containing two SSA adapters with two loops on each adapter.

Refer to Figure 44 on page 149 for the single node GPFS configuration. The second GPFS disk configuration was four nodes, each containing six 4.5 GB disks for /GPFS/DATAFS and /GPFS/TEMPFS. This second configuration was four times the size of the initial GPFS configuration and four times the size of the local and NFS configurations.

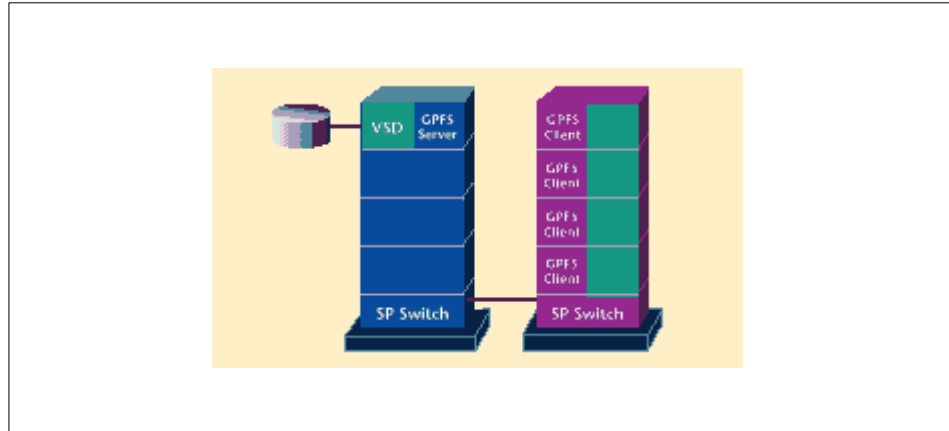


Figure 44. Single node GPFS configuration

See Figure 45 for the four-node GPFS configuration.

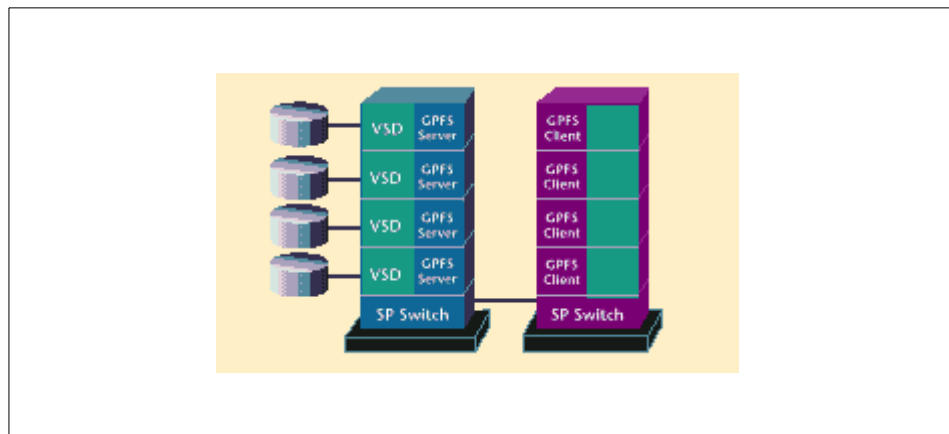


Figure 45. GPFS configuration

F.2 Tests

Four tests cases used in the GPFS testing follow:

- The **Census_Test1.sas** test case reads the raw data supplied from the US Census Bureau files, and creates two SAS data sets: HRECS and PRECS. The HRECS file, generated with data from the state of California, is 120 MB in size, and the PRECS file from the state of California is 340 MB in size. The HRECS SAS data set describes attributes describing households. The PRECS SAS data set describes attributes for individual persons. The three other SAS test cases use the HRECS and PRECS SAS data sets created by the Census_Test1.sas input file.
- The **Census_Test2.sas** test case creates indexes in the HRECS and PRECS SAS data sets to allow the two files to be joined in a later SAS test case. The generation of the indexes is very memory-intensive and allows for more efficient access to data inside a SAS data set.
- The **Census_Test3.sas** test case is a statistical test generating frequency data and summarization data of HRECS and PRECS. Many PROC FREQ steps are executed on many combinations of data in the HRECS data set. Many PROC SUMMARY steps are executed on data in the PRECS data set. This test is very I/O intensive.
- The **Census_Test4.sas** test case collects frequency data on the PRECS data set. This test is very CPU-intensive and consists of two PROC UNIVARIATE steps, a PROC SORT step, and two PROC Data sets steps for cleanup.

F.2.1 Test results

The test results are documented in both table and graph formats. Figure 46 shows three sets of results.

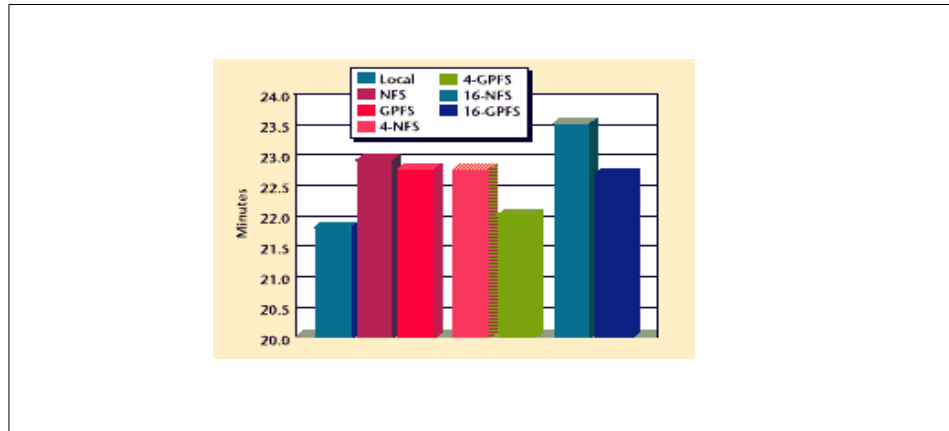


Figure 46. Test 1 results

Each job reads from the same input file and creates a new Household (HREC) and Person (PREC) file. The first set of results, represented by the leftmost three bars, indicates the time to run one job on local, NFS, and one-node GPFS file systems. The results show that while GPFS is slightly better than NFS, the local file system is the better solution for a single job.

The second set of results, indicated by the middle two bars, indicates the time to run four concurrent jobs on NFS and GPFS. The four concurrent jobs use four nodes, each running only one job. In this case, GPFS performs better, on average, than NFS.

The third set of results, represented by the rightmost two bars, indicates the time to run 16 concurrent jobs on NFS and the four node GPFS file system. Four nodes are used, each node running four concurrent jobs. The results for this higher workload indicate that GPFS again performs much better than NFS. Figure 47 shows three sets of results.

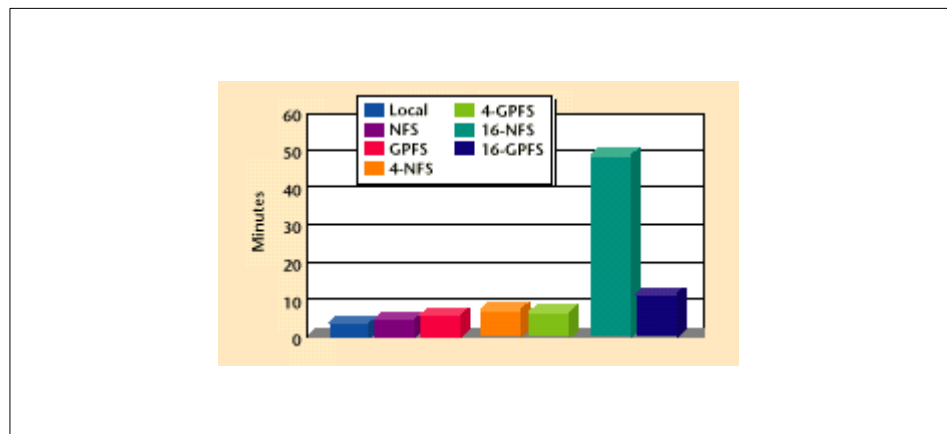


Figure 47. Test 2 results

Each test job sorts both the Household (HREC) and Person (PREC) files and creates indexes to join the files together.

The first set of results, represented by the leftmost three bars, indicates the time to run one job on local, NFS, and one-node GPFS file systems. Again, we see slightly better performance on the local configuration as compared to NFS and GPFS.

The second set of results, indicated by the middle two bars, indicates the time to run four concurrent jobs on NFS and GPFS. The four concurrent jobs use

four nodes, each node running only one job. There is no real difference in performance in this test.

The third set of results, represented by the rightmost two bars, indicates the time to run 16 concurrent jobs on NFS and the four node GPFS file system. Four nodes are used, each node running four concurrent jobs.

As can be seen from the graph, the time to complete the task increases slightly as the workload increases. When the workload is increased to 16 concurrent jobs, the NFS configuration performance is greatly affected. The GPFS configuration sees little or no negative effect from the increased workload and maintains the same relative response times to accomplish the tasks.

Figure 48 shows three sets of results.

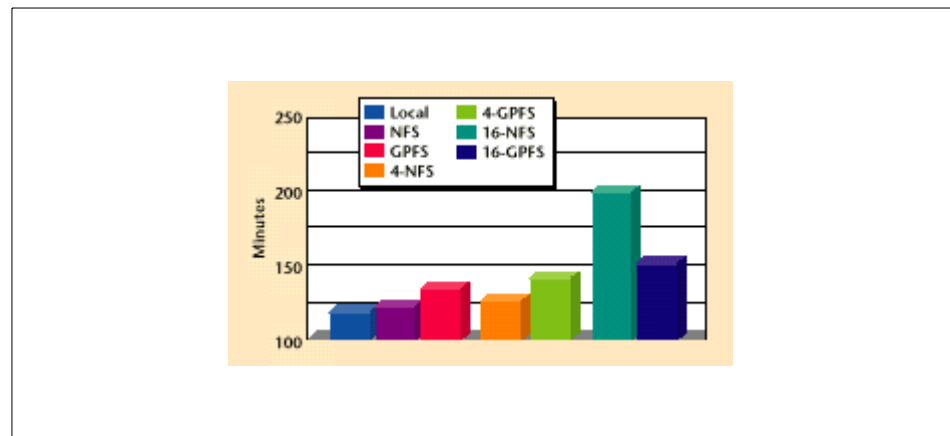


Figure 48. Test 3 results

Each job does sequence calculations and summarizations.

The first set of results, represented by the leftmost three bars, indicates the time to run one job on local, NFS, and one-node GPFS file systems. The aforementioned trend continues where the local configuration performs best.

The second set of results, indicated by the middle two bars, indicates the time to run four concurrent jobs on NFS and GPFS. The four concurrent jobs use four nodes, each node running only one job. In this test, we see that NFS holds a slight advantage over GPFS.

The third set of results, represented by the rightmost two bars, indicates the time to run 16 concurrent jobs on NFS and the four-node GPFS file system. Four nodes are used, each node running four concurrent jobs. Again, we see the GPFS advantage in this test case, where NFS performance is greatly affected in a negative fashion while GPFS yields consistent results despite the increased workload.

Figure 49 shows three sets of results. Each job collects statistics on the files.

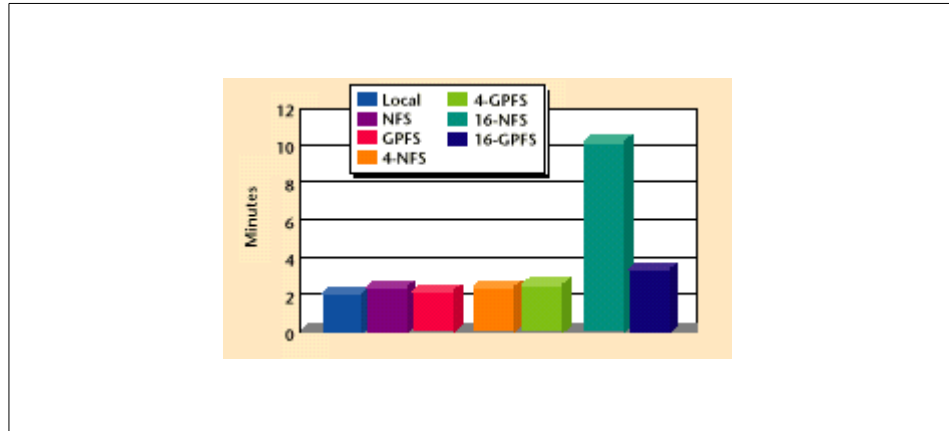


Figure 49. Test 4 results

The first set of results, represented by the leftmost three bars, indicates the time to run one job on local, NFS and one node GPFS file systems.

The second set of results, indicated by the middle two bars, indicates the time to run four concurrent jobs on NFS and GPFS. The four concurrent jobs use four nodes, each node running only one job.

The third set of results, represented by the rightmost two bars, indicates the time to run 16 concurrent jobs on NFS and the four node GPFS file system. Four nodes are used, each node running four concurrent jobs. This set of results demonstrates the severe penalty paid by the NFS file system with a high workload. The results also show how scaling up the number of resources to handle I/O with GPFS can support the larger workload with ease.

F.2.2 Notes on testing

This test project was the first opportunity for us to work with GPFS. We referred to three sources of information: A redbook on GPFS, a whitepaper

that had been written regarding performance with GPFS, and the *General Parallel File System for AIX: Install & Admin Guide*, SA22-7278.

There are a few key parameters that dramatically affect system performance. Since the SP switch is used by GPFS, certain parameters must be set in order to achieve maximum performance. In addition to switch tuning, certain system parameters, such as GPFS block size and VSD *buddy buffers*, must be set accordingly.

One key item to be aware of is that a few of these parameters come from the kernel memory (heap), which is limited in AIX 4.2.1. This limitation is removed in AIX 4.3. We ended up readjusting the number of buddy buffers in order to achieve a proper switch tuning setting. The parameter settings we used were:

Switch:

- thewall=65536
- poolsize=16777216
- spoolsize=16777216

GPFS:

- Block size = 256K

VSD:

The VSDs were defined with the parameters shown in Table 23:

Table 23. VSD parameters

IP packet Size	Initial cache buffers	Max cache buffers	VSD request count	rw request count	Buddy Buffer Min and max size	Size:# size
61440	64	256	256	48	4096 and 262144	33

The GPFS block size was changed to 64 KB, but there was little or no positive effect on performance.

After running the initial set of tests with a one-node GPFS configuration, we used the Systems Management Interface Tool (smit) to change the GPFS configuration. We supplied a file that contained the physical disk descriptions that would be used to define the VSDs. The SMIT panel allows you to specify this disk descriptor file and uses it to automatically build and configure the VSDs.

After the VSDs were configured, the file system was automatically changed to include disks on all four nodes. Any data that was in the file system was also automatically rebalanced across the entire file system using the 64 KB stripe size.

While we did not use this option, GPFS allows for adding disks to the file system while still online.

F.3 Conclusions

The two objectives of our test were achieved. Our first goal, which was to test functionality between the SAS System and GPFS, was demonstrated in that all the tests ran successfully with no changes required to any of the test cases. The output files were compared between the local, NFS, and GPFS tests and were identical. There were no problems with many SAS users on separate nodes accessing the same file in the GPFS file system.

The second goal, which was to measure performance and scalability of the GPFS file system, was also accomplished. The results showed us that scaling up the number of disks, loops, disk adapters, I/O buses, and processors allows for more work to be done with comparable response times. The ability to access a file from any node in an SP system also allows for the scaling up of CPUs and memory to handle SAS workload.

One additional benefit that GPFS can provide SAS users is the ability to define a large \saswork directory that provides a global name space. This simplifies administration of a load-balancing environment. System administrators and users do not have to concern themselves with details as to where data is located and mounted. In defining a \saswork in GPFS, it allows for an accessible file system to any node in an SP complex and allows for simple automated administration of that directory. The system administration of a GPFS file system is exceptionally easy, especially when compared to having to administer a similar NFS configuration. The added benefits of a GPFS configuration are availability and recoverability.

The ability of GPFS to add disks to a file system and rebalance data across those new disks offers balanced scalable growth to a file system. This ability to grow the file system in terms of processors, buses, disks, and adapters is unprecedented. It offers the ability to maintain acceptable performance while increasing workload.

The tests that were run indicate that GPFS is an excellent candidate for solving problems introduced by large numbers of SAS users or jobs where

data needs to be shared or where large amounts of I/O activity can be helped through the addition of disks and adapters.

For more information, see the following:

- *An Introduction to GPFS R1* - White paper available on the Web at:
www.austin.ibm.com/resource/technology
- *GPFS: A Parallel File System*, SG24-5165
- *General Parallel File System for AIX: Install & Admin Guide*, SA22-7278
- RS/6000 Teraplex Integration Center

For more information about the RS/6000 Teraplex Integration Center, contact Frank DeRobertis at DEROB@us.ibm.com.

The information in this appendix comes from an IBM whitepaper, *The SAS System and GPFS - A Scalable Solution*, by Keith F. Olsen and James T. West of the IBM Corporation.

Appendix G. Scalable performance

This appendix deals with achieving scalable performance for large SAS applications and database extracts.

The dramatic drop in the price of computer disk storage from tens of dollars per megabyte to tens of cents per megabyte has greatly increased the collection and storage of massive amounts of data. Many organizations are making significant investments in data warehousing, data marts, and analytical applications to extract knowledge from their operational data. Often, the ability to realize the full value of that data is limited because, even with the fastest computer, the data volumes are so large that they make processing within the requisite batch window impossible.

Not only are Business Intelligence systems likely to be large and grow rapidly in complexity as well as size, but their growth and usage patterns may be difficult to predict. Even more than with typical data processing systems, BI systems need to be built to be scalable. This includes the ability to grow in increments and the ability to harness many resources, all working together to provide computing power for high-performance on large applications when needed. Traditional solutions can't address these dynamic requirements.

Multiprocessor machines provide the ability to economically scale the number of jobs and users as processors are added. These systems can also be used to scale the throughput of single large jobs, such as data extract or analysis. By making use of the many processors in an SMP or MPP system to divide and conquer such large-scale jobs, the use of parallelism can allow users to get maximum utility from their hardware investment.

Among the customers who are already facing the limitations of conventional computing are those who run large SAS applications. The SAS system, developed by the SAS Institute, is the industry leader in statistical data processing. It provides an integrated set of tools for data access, management, analysis, and presentation. SAS customers are looking for ways to more efficiently access and process data, and many are showing interest in parallelism as a solution. The obvious questions are: How much of a benefit can be accrued from using parallelism, how is it implemented, and how difficult is the development involved?

Torrent Systems in Cambridge, MA, provides an off-the-shelf environment for building and deploying parallel applications. Torrent's Orchestrate provides the benefits of parallelism while freeing programmers from having to engage in the complexities of parallel programming. Orchestrate enables any IT

professional or commercial developer to parallelize a new or existing application for improved performance and scalability. To address the above questions, Torrent Systems, along with IBM, developed and ran a set of representative workloads to test the performance impact of applying parallelism to data extraction and SAS analyses, with the goal of decreasing overall processing time and measuring the effect of parallelism on various segments of the applications.

The testing team ran two tests. Both were run on an IBM RS/6000 SP massively parallel system. The first test extracted data from a parallel IBM DB2 UDB database and passed it to a SAS application. The second test used an IBM customer's actual SAS inventory forecasting application, which processed data from flat files. Without special parallel programming, such as that provided by Torrent's Orchestrator for the SAS System, neither could run in parallel, despite the fact that they were running on a parallel hardware platform. The tests were run sequentially and then in parallel, incrementally increasing the number of nodes to test scalability, and the results were compared.

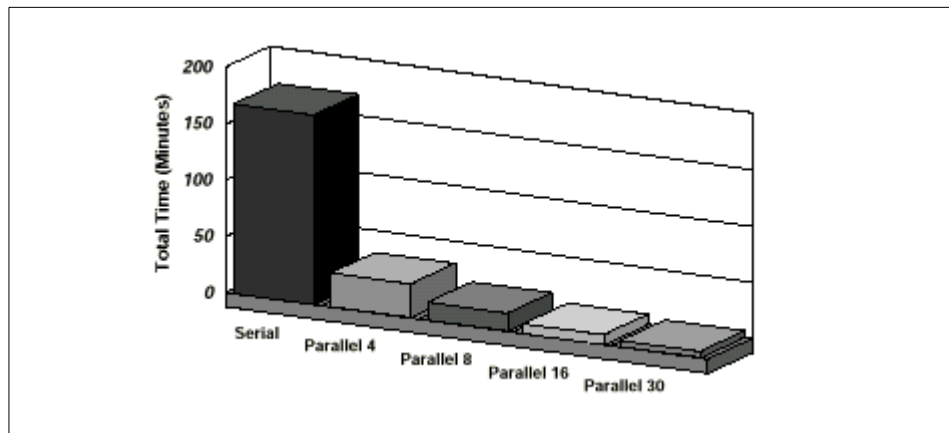


Figure 50. Parallel extraction from DB2UDB

What the tests demonstrate is that the performance of both the database extract and the SAS application are severely limited by the ability to only process on one CPU at a time. This is an artificial limitation if the applications are running on multiprocessor systems. Without adding hardware, in each case, performance increased dramatically when the extracts and processing were run in parallel as opposed to sequential mode on the same machine.

The tests further demonstrate that not only does the use of parallelism provide immediate performance enhancements, but it also provides the ability

of the tested system to scale incrementally. In fact, results showed linear scalability through all 32 processors tested. (An example of linear scalability is where 12 processors provide 12 times the performance of a single processor). Furthermore, because the Torrent software enabled and managed the parallelization, the programmers did not have to write in parallel. They simply executed their applications in the Torrent Orchestrate environment to gain the full benefits of parallel execution. Orchestrate makes scalable software systems commercially feasible.

Problem statement

According to Moore's¹ Law, the number of transistors that can fit on a chip (and, thus, its power) doubles every 18 months. That is not fast enough to keep up with today's Business Intelligence demands. The Tower Group predicts that the size of the average data warehouse will be 10 TB in the year 2000, and 100 TB by 2005. Scalable solutions require both scalable enabling technologies and scalable application design.

Although multiprocessor systems (SMP, MPP, and clusters) are now common in commercial environments, most software is designed to be run serially on a single processor. This severely limits customers' ability to take advantage of these powerful machines to provide the performance and scalability they need to tackle the new class of business intelligence applications. Data extract and refinement tools run slower than they should. Data warehouses are artificially limited in size. The amount of historic data needed for time-series models to be effective is compromised. Analysis applications are run less frequently than desired; use of summarized data is required, and so on.

Commercial processing has begun to make use of parallel technology with parallel RDBMSs. Data in a single table can be partitioned across many CPUs and processed in parallel for greater throughput of queries and other database commands. The IBM DB2 UDB (Universal Database) partitions data across multiple nodes and parallelizes all database functions including insert, update, and delete. Joins, load balancing, table reorganization, data load, index creation, indexed access, backup, and restore are all performed in parallel on multiple nodes. However, the processes that occur outside the database are still run sequentially, creating a bottleneck and limiting the overall performance benefit of using a parallel database:

- SAS applications can very quickly consume large amounts of data.

Thus, it is important to avoid a situation where there is only one connection between the SAS step and the parallel database, in essence, serializing the database. Although the data has been partitioned across multiple nodes, a

¹ Gordon Moore, Chairman of the board of Intel in the 1960s

SAS process can only connect to one DB2 UDB node at a time. Thus, all the data must be passed to that node and then funneled through a single cursor to the SAS application, creating a bottleneck.

- Equally important is how the data tables are partitioned across the processors in the first place.

The database must be partitioned in a way that is consistent with the expected access. If that is not the case, users will find themselves asking for data that is not local to the connected node and having to wait for it to be transmitted, thus, greatly diminishing efficiency.

- SAS applications only run on one processor at a time.

Accessing a database with data on multiple processors is similar to traffic flow on several heavily traveled expressways, with the task of flowing selective vehicles to a large ball park from the expressways. The design requires multiple points of entry to allow vehicles on the multiple expressways to arrive in the ball park's parking lot at about the same time. The expressway vehicles can't all merge into a single traffic lane with one point of entry into the ball park. This would constitute a self-imposed bottleneck.

Moving data from a parallel database to a software application is very similar. The data must arrive at the application via multiple parallel routes, and not back up, waiting to pass through a single entry point. You want to avoid *serializing a parallel database*.

In fact, like databases, many tools and applications can benefit from the application of parallelism. The steps involved in building a data warehouse or data mart, such as data extract, transformation, cleansing, merge, sort, and loading are easily *parallelizable*, and shortening their processing time can provide substantial benefit. Likewise, parallel execution of many data mining and analysis applications could mean the difference between compromise and running the required jobs against the required data volumes in the required frequency.

But, effective parallel processing includes full coordination of the partitioning, processing, error handling, and merging events so that the results are the same as if the application had been run serially. In addition, it means that the processing is done efficiently so as to provide high-performance and scalability as processors are added. Parallel programming has not come without the high cost of development and management. Until now.

The parallel solution

Torrent Systems has developed a product called ORCHESTRATOR for the SAS System (OFS) for enabling SAS applications to execute in parallel and to interface in parallel to the IBM DB2 parallel database system on both SMP and MPP architectures. OFS increases overall system performance by letting you:

- Extract data in parallel from a parallel RDBMS
- Load the results of SAS programs back into the database in parallel
- Process parallel data streams with parallel instances of a SAS DATA or PROC step for much higher throughput rates
- Store large data sets in parallel, thus, providing faster access and eliminating storage restrictions
- Stream data between SAS steps without having to write intermediate results to disk

The bottom line is that you can process much larger volumes of data.

Orchestrator for the SAS System is a commercially available and supported product designed to solve the parallel interface issue with minimal user effort, allowing SAS users to insert simple parallel directives into SAS programs. SAS algorithms that can be partitioned into discrete components can be parallelized using a Multiple Processor Independent Data (MPID) parallel paradigm.

SQL statements that use a GROUP BY clause illustrate a class of problems that could be executed in parallel with each SAS processor working on a separate GROUP or PARTITION of the data. Data extracted from a database in parallel is most efficiently processed in parallel local to the node containing the extracted data. This involves no processor-to-processor communication to move data. Executing SAS applications in parallel and using data extracted in parallel from a DB2 parallel database makes efficient use of the IBM RS/6000 SP system.

IBM and Torrent set out to demonstrate and measure the performance impact of extracting data and running SAS applications in parallel. The results show the dramatic improvements that can be realized by employing Torrent's Orchestrator for the SAS System to fully exploit the power of the IBM RS/6000 SP system.

Testing methodology

There are no standard benchmarks for performance measurement of data access and analytical processing, two time-consuming components of BI

systems. The testing team wanted to ensure that the workloads and configuration were as realistic as possible and that the comparisons between sequential and parallel processing were fair and meaningful. To this end, they used real customer applications. One test used an actual SAS inventory forecasting application running at an IBM customer site. The customer was using an IBM RS/6000 SP system but running SAS applications on a single node and not taking full advantage of the multiple processors.

Segments of the applications were tested independently to demonstrate the impact of parallelism on each type of process, but the overall performance was maximized when the system was run using *end-to-end parallelism* eliminating sequential bottlenecks.

Both uniprocessor and SMP nodes were used in the system configuration that is typical in customer environments and allows measurement of the performance characteristics of each.

In addition, Richard Winter, president of Winter Corporation, and a leading authority on the subject of Very Large Database implementation, conducted an independent analysis of the test procedures, results, and conclusions. His evaluation and comments can be found in “Evaluation and comments” on page 176.

The tests were run at the IBM Teraplex Integration Center in Poughkeepsie, NY. The \$47-million Teraplex is used to integrate and test RS/6000 SP hardware, IBM software, and products developed by IBM Business Partners for very large, end-to-end customer solutions.

Test machine configuration

Readers need to be familiar with the basics of parallel architecture in order to understand the testing procedures. They are explained in “Parallel architecture basics” on page 179.

The IBM RS/6000 SP system hardware can best be described as a cabinet with eight drawers. There are three types of nodes: Thin, high, and wide. Each node type takes up space in these drawers.

The thin node is a uniprocessor node and takes up half a drawer.

The high node, which is an SMP node with between two and eight CPUs, takes up two full drawers. It is twice as high as a normal thin or wide node.

The wide node (not used during this test) takes up one full drawer. It is a uniprocessor node but has more I/O capability than a thin node.

The hardware configuration for the test consisted of high nodes and thin nodes. The thin nodes were hosting the database in different configurations consisting of 4, 8, 16, 32, and, sometimes, 30 nodes. The high nodes were used for running SAS jobs for the single node tests. All processors were connected by a high-speed switching network.

The high nodes were configured as follows:

- Eight 604 PowerPC processors (112MHz)
- 2 GB memory
- Two 2.2GB internal disks
- Four SSA adapters
- 32 4.5GB disks (two loops on each adapter with four disks per loop)

The thin nodes were configured as follows:

- 120MHz Power2SuperChip processor
- 1 GB memory in 28 nodes
- 512MB memory in 4 nodes
- Two 2.2GB internal disks
- One SSA adapter
- Eight 4.5GB disks (two loops with four disks per loop)

The software configuration consisted of currently available products from IBM, the SAS Institute, and Torrent Systems:

- AIX V4.2.1 with Parallel System Support Program (PSSP) V2.3 for all nodes
- DB2 UDB V5.0 parallel database
- SAS V6.12, and SAS/Access
- Orchestrate V3.0
- Orchestrator for the SAS System V3.0

The software used on each node varied depending on the test being run. For example, in some tests, SAS applications only ran on one node, whereas, in other tests, they ran on all nodes. The DB2 UDB database was configured on differing numbers of nodes (4, 8, 16, or 32) depending on the test being run.

Database configuration

The DB2 UDB parallel database was configured using System Managed Storage (SMS). The AIX file system used by UDB was striped across four SSA disks with a single SSA adapter. The stripe size was 64 KB.

Striped SSA file systems under AIX provide exceptional I/O bandwidth when using four or more SSA disks. With a striped file system, all disks are kept busy using striped I/O with optimal data buffering and minimal disk head movement. A disk striped file system provides optimal parallel utilization of the SSA disks.

A study was performed to compare the performance difference between Database Managed Storage (DMS) using user-defined segments on individual disks and a striped file system using SMS. Large data extractions with DB2 UDB measured the SMS striped file system technique as more than 10 percent faster than user-managed segments using DMS.

Census data was used, and the database was partitioned by serial number for 4, 8, 16, and 32 thin nodes.

The tests

Leveraging parallel hardware for data extraction, processing, and load into a parallel warehouse can yield huge gains in the performance of data warehouse applications. To demonstrate the performance gains of both parallel extract and parallel processing, IBM and Torrent devised two tests that extract and then process large amounts of data from a DB2 UDB database in parallel. As the number of CPUs is increased, the entire application execution time falls off dramatically for both the extraction and processing steps. This is referred to as scalable performance. The following sections describe, in detail, the test scenarios and their results.

Test 1: Parallel extraction

Parallel extraction of data from a parallel database is a good example of where SAS systems can benefit from parallelism.

Serial extraction of multiple gigabytes of data from a warehouse is a time-consuming operation. Data distributed across a parallel table of an RDBMS must be streamed from the various table nodes into a coordinator node, that then streams the data out of the database. The flow of data is bottlenecked by the rate at which the coordinator node within the database can read and then write the data.

In the first test, Orchestrator for the SAS System was used to parallelize the data extraction from the database. This was done by enabling SAS/Access to

run on every node, only extracting data local to that node. By executing a SAS/Access process on each node of a parallel table, OFS is able to extract all of the data from the parallel table. OFS removes the sequential bottleneck found at the coordinator node by executing SAS/Access on all of the database nodes.

Figure 51 shows a typical sequential SAS/Access database extraction vs. an OFS-enabled parallel database extraction. Several parallel extractions were performed on a table of data spread out over 4, 8, 16, and then 32 nodes of the database. The volume of data extracted from the table was fixed at a constant volume during all phases of the test. The tests extracted an entire table composed of five million rows (records) of census data. Each record was 304 bytes long and was composed of 126 fields. The total data volume was 1.52 GB.

In the parallel extraction runs, OFS merges the parallel extraction streams into a single SAS data set. This single SAS file acts as a data mart that the SAS expert then uses for further data analysis.

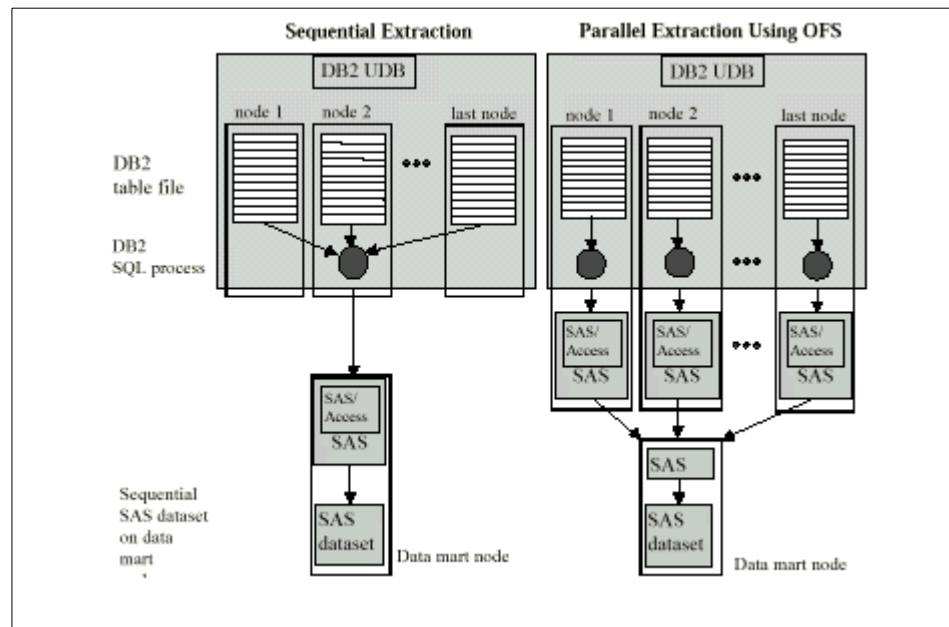


Figure 51. Database extraction into sequential SAS data

In order to evaluate the relative performance of high and thin nodes, tests were run where the sequential SAS data set was collected onto both high and

thin nodes. All processors were connected by a high-speed switched network.

In the sequential extraction, the *sequentialization of the distributed data* occurs when streaming data into the single coordinator node of the database. SAS/Access then uses a single stream to extract the data out of the database. In the parallel extraction test, the need to move data from node to node within the database is eliminated, plus, data is streamed out of the database in parallel. The sequential step that converges the data into a single file occurs outside of the database under the control of OFS. As the tests will show, only parallel extraction provides efficient scalable performance with large volumes of data.

A key feature of the OFS parallel environment is its ability to land data to disks in parallel. Figure 52 shows one of the applications that was run as part of the parallel extraction tests. It is like the parallel extraction test above, but, instead of generating a single sequential SAS data set, OFS generates a parallel SAS data set. The parallel processing test demonstrates the utility and power of parallel SAS data sets.

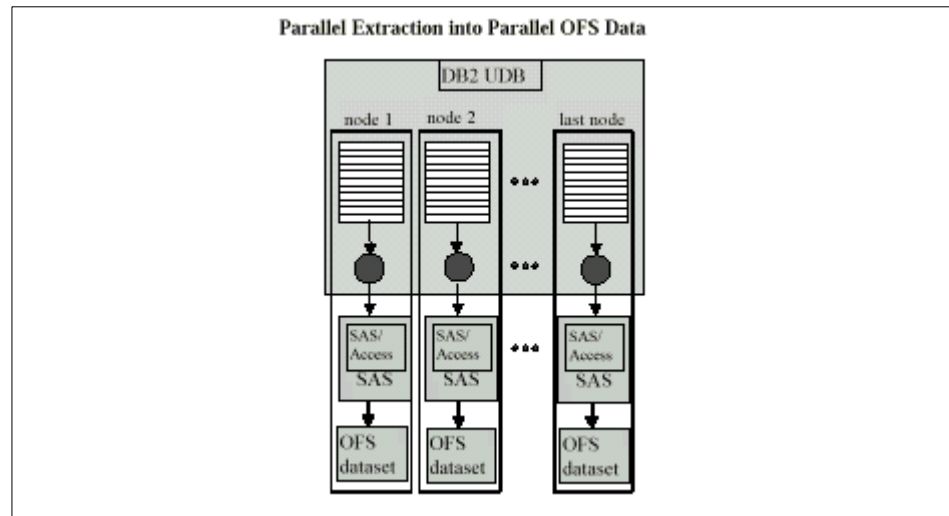


Figure 52. Parallel extraction using OFS

Results of the parallel extraction test

Table 24 shows the performance results of extracting the census data from the parallel database. The subheadings of high and thin designate the data

mart node type. N-to-1 indicates extraction from N nodes into a single data set. N-to-N indicates extraction from N nodes into an N-way parallel data set.

Table 24. Census data

	High node	Thin node
Sequential extraction 32-way to 1	3:16:34	2:48:42
Parallel extraction		
4-way to 1	30:12	30:10
8-way to 1	16:01	16:01
16-way to 1	08:35	08:48
32-way to 1	06:39	05:13
32-way to 32	04:17	
30-way to 30		04:07

Notice the sequential extraction times for the census data. These are the runtimes of the left side of Figure 51 on page 165 for both high and thin node data marts. Compare these to the 32-way-to-1 parallel extraction times (right side panel of Figure 51 where N=32). The difference in these run times (3:16:34 and 2:48:42 vs. 06:39 and 05:13) is the difference between converging a 32-way table to a single file stream inside versus outside of the database.

OFS is extremely efficient at streaming large volumes of data.

Next, notice the drop in times in each of the columns as you progress from the 4-way- to-1 extraction towards the 32-way-to-1 extraction. In this progression, the same volume of data is extracted from the table that is spread out over 4, 8, 16, and then 32 nodes of the database. The run times are shown as a function of the number of nodes in Figure 53. Perfect linear scalability is achieved when doubling the number of processors doubles the increase in speed; four times the number of processors quadruples the increase in speed, and so on.

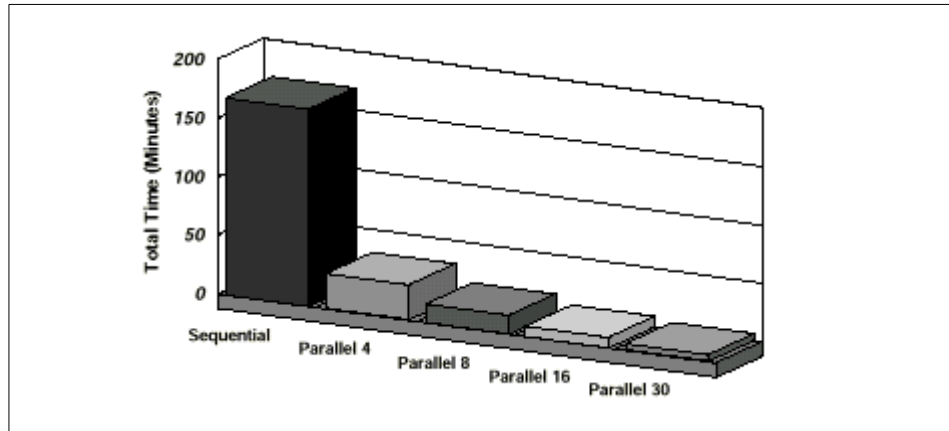


Figure 53. Parallel extraction from DB2 creating a sequential dataset on a thinnode

Parallel extraction from DB2 creating a sequential SAS dataset on a Thin Node OFS provides near linear scalability of the database extraction process. The scalability is shown in Figure 54. The actual data points are plotted in the black curve. The theoretical scalability limit that exists for applications with no sequential bottlenecks is plotted with the heavy gray curve.

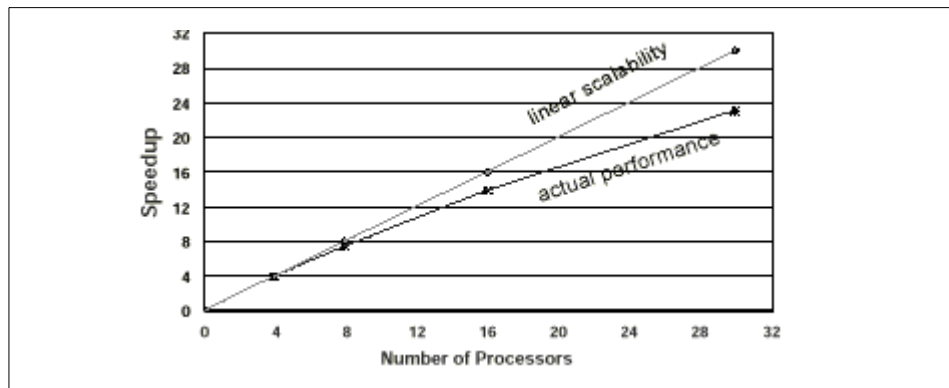


Figure 54. Scalability of parallel extraction from DB2 to equential SAS dataset

As the number of processors used to extract the data increases, the deviation from the 1:1 scalability line increases. This is particularly evident after 16 nodes. This deviation results from the inevitable sequential bottleneck that is created when we merge the extracted data into a sequential SAS file. Parallel extraction is so fast with many parallel clients, that regardless of whether data is streaming from 30, 32, or 128 extraction nodes, the bottleneck becomes

writing data to disk on a single node. The difference between the 32-way-to-1 and the 32-way-to-32-way run times is approximately the time taken to write the data to a single disk (see Table 24 on page 167). In the thin node census data run, the difference is 66 seconds. For the census data set of 1.5 GB, this is consistent with the data mart disk I/O of 23 Mbytes/second and is very close to the hardware specification.

By writing to an OFS parallel data set instead of to the sequential data mart file, OFS allows almost perfect scalability in the extraction. The 32-way-to-32 run time graph lies right on top of the ideal scalability curve in the figure above because its run time is exactly half of the 16-way-to-16 run time (see run times in the table above). This feature means that you can frequently avoid sequential bottlenecks in production mode applications by loading data into parallel data sets. The parallel OFS data sets can be fed directly to the parallel OFS application.

Test 2: Parallel SAS Application

The execution of complex sequential SAS applications against multiple gigabytes of data can be a very time-consuming operation. Every record of the data set passes through the same CPU to undergo processing. The bottleneck of sequential processing can far exceed that observed in sequential database extraction because the SAS application may be CPU-intensive. Thus, you are no longer merely limited by the rate at which you can stream data through a single node; you are further limited by the speed of the processor.

In this second test, we parallelized a large, commercial, SAS inventory forecasting model. This application is a SAS program with 526 lines of code, containing 20 DATA steps and 33 PROC steps including linear regression, freq, sort, sql, summary, transpose, and so forth. This application is one that is currently running in sequential mode at an IBM customer site. The customer stores the data in a flat file.

In the test, the processing was broken into three sections to demonstrate and measure the impact of parallel processing on different steps within the application.

Section 1: Sequential read

Reading the customer's flat file is a sequential bottleneck. In the first section of this test, OFS took the sequential flat file, converted it to a SAS data set, and then created a parallel SAS data set by partitioning it onto multiple nodes of the MPP or SMP measuring the difference in performance between writing the data sets to high vs. thin nodes. This section of the test is very fast and requires a small fraction of the time needed to run the actual SAS application.

The output from this section is used as input to the next step, the SAS application.

Section 2: Parallel execution

In the second section of the test, OFS executes the SAS model on the multiple nodes of the MPP or the multiple CPUs of the SMP. The more nodes executing the SAS model, the shorter the run time. Since each CPU only needs to process that portion of the data residing locally on its node, and the data has been divided up equally among N nodes (where N=1,2,4,8,16, or 32), each CPU must process only 1/Nth of the total data. The whole model can now execute in 1/Nth the time required to run the SAS model sequentially; so, for instance, the SAS model running on four nodes should run in one-fourth the time of the sequential model. After processing, OFS writes the SAS data set out to disk in parallel.

Section 3: Sequential write

In the final section of this test, the parallel data set created by the SAS model is read into SAS in parallel and then merged into a single sequential SAS data set, the output being easily inspected by the data analyst.

The left panel of Figure 55 on page 171 shows the actual test mode used to evaluate the execution time of the three separate sections described above. The intermediate writing to disk was performed only so that a separate runtime for each section could be evaluated. In production mode, you would avoid this intermediate write to disk. The right side panel shows this production mode configuration that avoids writing the data sets to disk.

The parallel model was run on 1, 2, 4, 8, 16, and 32 thin nodes of the RS/6000 SP and on 1, 2, 3, 4, 5, 6, 7, and 8 CPUs of the SMP. Running the parallel model on one node is equivalent to running the sequential model. Regardless of the degree of parallelism of the model, the volume of data processed was fixed at a constant volume during all phases of the test.

Pipelined parallelism

OFS applications containing multiple segments also take advantage of pipelined parallelism. With pipelined parallelism, all OFS segments in an application execute at the same time. A segment may be active, meaning it has input data available to process, or a segment may be blocked as it waits to receive input data from a previous segment.

In contrast, in a normal SAS application, one SAS step executes until completion, then the next SAS step executes, and so on. In this situation, a SAS step must consume all its input data and write all its output data before

the next step can begin. With pipelined parallelism, any OFS segment can execute as long as there is data available for it to use.

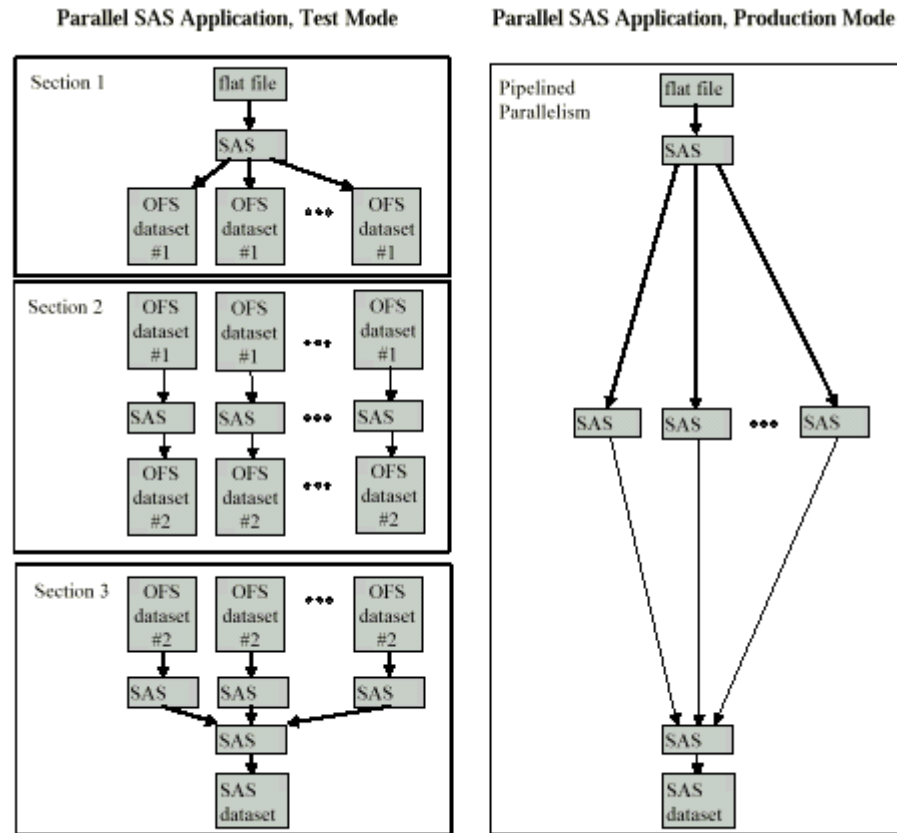


Figure 55. Parallel SAS application

Results of parallel SAS application test

Table 25 on page 172 shows the OFS parallel SAS application test performance results. The second section (the most time-consuming and CPU-intensive) runs the actual SAS application. This section actually scales superlinearly, that is, as the number of processors doubles, the run time is cut by more than a factor of two.

The first and third sections comprise a small fraction of the overall application time. The first and third sections do not scale because they read from and write to a sequential file. It does not matter how many nodes data streams to

or from in these two short sections. The rate-limiting step is the speed of the sequential node.

Table 25. Results of parallel SAS application test

	High node	Thin node
Section 1, Flat file to N-way data set		10:17
1-way	12:02	07:43
2-way	07:34	07:07
4 way	06:41	07:26
8-way	07:15	07:17
16-way		07:00
32-way		
Section 2, Parallel SAS application		1:42:29
1-way	3:19:02	36:36
2-way	1:10:34	15:58
4-way	31:23	07:08
8-way	15:23	04:19
16-way		02:36
32-way		
Section 3, N-way dataset to 1-way dataset		
1-way	01:01	01:01
2-way	01:00	01:02
4 way	01:03	01:04
8-way	01:06	01:08
16-way		01:15
32-way		

Figure 56 on page 173 shows a composite bar chart for the run time of the entire application as a function of the number of nodes executing the application. Each band represents the run time of one of the three sections of the test. The bottom dark-gray band represents the time taken to parallelize the data; the middle light-gray band represents the time taken to run the parallel SAS model code, and the top medium-gray band represents the time taken to stream the reduced model results to a single node. As the number of nodes increases, the fraction of the run time due to the actual SAS inventory forecasting run time (Section 2) decreases. This is a result of the scalability of this section and the lack of scalability of the other two sections. At 32 nodes, section 2 accounts for only 20 percent of the total run time. By this point, the run time is down to 10 minutes.

At 32 nodes, parallelization of the data (Section 1) requires seven minutes, which is 70 percent of the total run time. By leveraging the ability of OFS to create parallel data sets when performing extractions from your database, the sequential nature of this step can be eliminated. In fact, all of Section 1 can be eliminated. OFS provides the tools that will enable you to remove the bottleneck that will invariably exist in your application if you rely on sequential data sets.

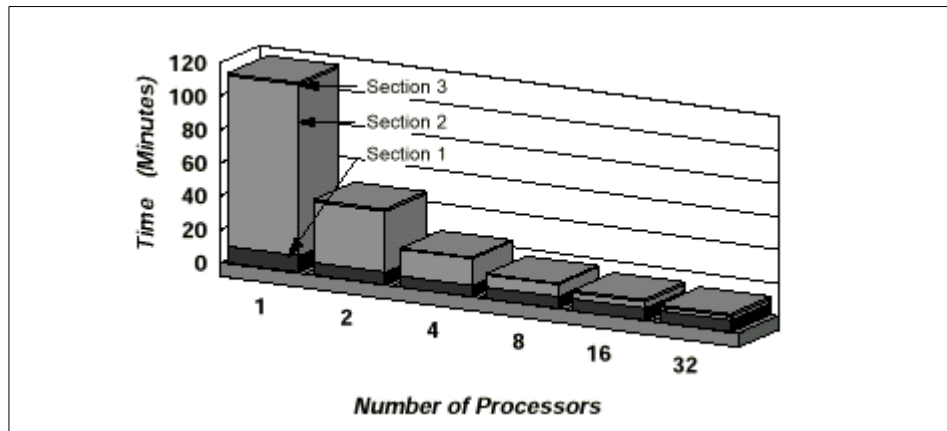


Figure 56. SAS application run time versus number of processors

Figure 56 illustrates that the run times of parallel Section 2 decreased dramatically while the run times of the sequential sections 1 and 3 did not change as processors were added to this application. As discussed previously, there is a fundamental absence of scalability in programs that have sequential bottlenecks like those found in sections 1 and 3. We investigate the scalability of Section 2 in greater detail below.

This chart also illustrates the value of OFS parallel data sets. By storing data in OFS parallel data sets, all of the run time contribution from Section 1 in this bar chart can be eliminated, making the application as a whole scale even better.

Figure 57 on page 174 shows the superlinear increase in speed observed in Section 2. Complex analytic applications running against large volumes of data can be very memory-intensive. When memory limits are exceeded, paging and thrashing can occur. By distributing data in parallel, OFS can provide superlinear speedup because it reduces the memory requirements on any single node below the threshold where these slow processes occur. The paging is eliminated.

This application achieved superlinear speedup using up to eight processors. The slope of the performance curve is about 2.0 in this region. When using more than eight processors, the slope levels off to a value of 1.0 indicating normal linear scaling.

Figure 57 provides a dramatic illustration of how distributing data over many nodes can lighten the load placed on the memory of each individual node and, thereby, reduce the overall work performed by the system. OFS allows an application to run against large volumes of data while still using hardware in the regime for which it has been optimized. In this particular application/hardware configuration, eight nodes were required to bring the system into the optimal performance regime. Beyond eight nodes, the total amount of work performed did not change, and there was normal 1:1 linear scalability.

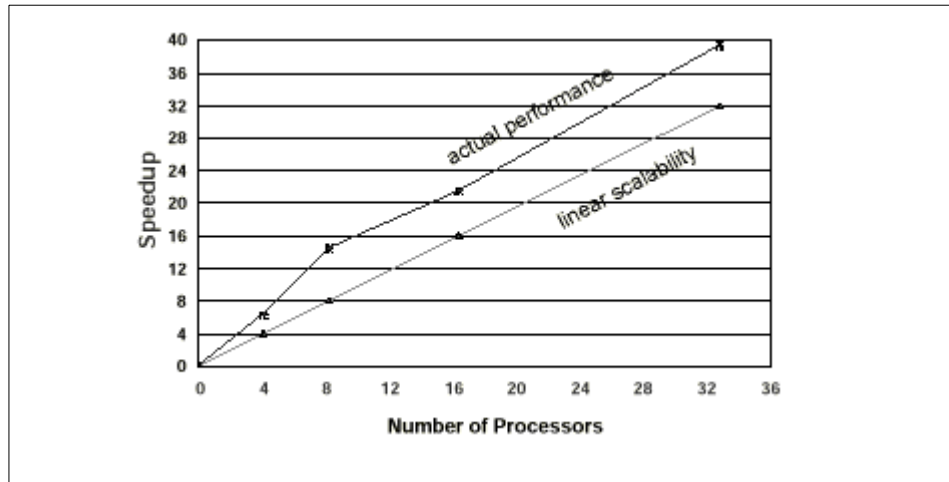


Figure 57. Superlinear speedup of SAS forecasting model

Conclusions

The benchmark performance studies of Torrent System's OFS undertaken at the IBM Teraplex Center have demonstrated the scalable performance that can be achieved when executing database extracts and SAS applications in the OFS parallel environment on IBM SMP and MPP hardware.

Several significant conclusions can be drawn from these results:

- Extracts from a parallel database should be done in parallel.

Using OFS to perform parallel extracts of large data sets from DB2 UDB allows linear scalability because of the extremely efficient mechanism that is

used to stream data. This advantage is achieved even when creating a sequential SAS data set because OFS employs parallel access streams to the database.

- Use of parallel data sets eliminates serious sequential bottlenecks.

Additionally, OFS allows the creation of parallel data sets directly when streaming data out of a parallel RDBMS. OFS parallel data sets eliminate many sequential bottlenecks that arise in the data plumbing between the parallel RDBMS and SAS applications.

- Executing SAS applications in parallel yields linear and even superlinear scalability.

Using OFS to execute SAS applications in parallel on the multiple CPUs of an SMP or the multiple nodes of an SP can yield linear and even superlinear scalability. Memory and disk constraints that arise in complex applications running against large volumes of data in sequential applications are overcome by using OFS to distribute the processing to all of the available system hardware. Linear scalability results from distributing the processing load to multiple CPUs. Super-linear scalability can result when distributing memory-intensive applications among multiple nodes, thereby, optimizing the memory processing.

- Repartitioning of a parallel database is necessary in a changing world, but cumbersome and time-consuming without ORCHESTRATOR for the SAS System

Finally, the OFS ability to provide parallel streams of large data sets into and out of a parallel RDBMS allows redistribution of tables within the database in a very simple and efficient manner. The database was re-partitioned numerous times during the testing, from 1 to 4 to 8, etc. nodes in order to measure processing scalability. In any environment, it may be necessary to periodically add nodes or reconfigure a system. With OFS, it is not a daunting task to re-partition the database to take advantage of the additional nodes.

Additionally, as the needs of database users change and the types of queries submitted against various tables of the RDBMS evolve, the OFS ability to dynamically repartition tables *on the fly* according to different keys for the purpose of a particular operation will become more and more valuable.

All parallel all the time

Organizations performing data extraction and refinement, data warehouse and data mart loading, models, analysis, data mining, and other such large-scale batch jobs now have a way to cost-effectively handle growth in volume and complexity. By implementing end-to-end parallelism (parallel

hardware, parallel database, parallel applications) using scalable hardware, such as the IBM SP system and off-the-shelf software from Torrent Systems, they can create an environment that will scale to many times its sequential capacity without having to purchase additional hardware. And, Torrent's software, which enables development and manages deployment of parallel applications, allows users to take full advantage of parallelism without having to invest in parallel programmers.

Being able to incrementally scale processing up to terabytes of data on hundreds of processors provides a great deal of growth capacity for an environment growing in data volume and application complexity.

Evaluation and comments

This section is an independent review of Benchmark of Orchestrator for the SAS System by Richard Winter, President and Founder of Winter Corporation.

Tests performed

Two principal tests were performed. The first illustrated extraction of data from a DB2 UDB database. The second illustrated execution of a SAS application. Without OFS, each of these tasks would ordinarily be performed in a serial manner. This was illustrated in an initial run of each test. Each task was then performed in parallel using OFS.

Results obtained: DB2 extraction test

In the DB2 UDB extraction test, a baseline is established with a run that extracts 1.52 GB of data from DB2 to a single file written by a serial application program. This is a typical first step in setting up SAS processing of data that has been stored in a relational database. The benchmark showed that running the serial output process took approximately three hours on each of two different configurations.

This test points up an issue overlooked by many IT professionals: Even though DB2 UDB is a sophisticated database capable of performing a parallel scan in a highly-optimized manner, its output to any single application program is ordinarily serialized. Thus, the user does not benefit from either the parallel architecture of DB2 or the parallel architecture of the RS/6000 SP in: (a), the feed from the database to the application, or (b), the output from the application to a sequential file.

This same operation was then run under OFS demonstrating both the parallel feed out of the database and parallel I/O from the application to the file system. The resulting runs, operating on up to 32 nodes of the RS/6000 SP, took approximately four minutes.

Linear scalability was demonstrated in this test, as the number of nodes employed was increased from four to 32.

SAS application execution test

This test is designed to illustrate the effect of using OFS to run compute intensive SAS applications in parallel on multiple nodes of the MPP architecture. In this case, the baseline test establishes a serial runtime for the computation-intensive component of the application ranging from about 100 minutes on a thin node to about 200 minutes on a high node.

Running the computation-intensive component in parallel on 32 thin nodes reduced the execution time from 100 minutes to approximately 2.5 minutes. Running the computation-intensive component in parallel on eight high nodes yielded a speedup from approximately 200 minutes to approximately 15 minutes. In this case, the speedup was better than linear due to the elimination of a thrashing effect that existed when the application was executed on a single node.

In this test, there were two components of the SAS application that were not parallelized: Reading from a sequential input file and writing to a sequential output file. These components did not speed up in the test. However, if parallel OFS files were used for these steps, additional parallelism and speedup would be expected to occur.

Review process

In the review process, I examined listings of the tests executed and the results produced by interviewed personnel who participated in the test process; I reviewed a draft of the White Paper presenting the results and conducted my own independent analysis of the test results and conclusions.

I am satisfied that the tests were conducted as described here, that the results are as represented here, and that the conclusions described in the White Paper are appropriate and supported by the test results.

Reviewer's comment

Much of the significance of the capabilities provided by OFS is due to the ease with which preexisting serial applications are transformed to operate in parallel. OFS provides facilities to easily specify how such multistep parallel operations should be set up; it readily feeds the output of one step as the input to another and provides facilities for management of parallel execution.

In principle, this can be done without the use of such a product as OFS: Any user could replicate his or her DB2 extraction application and run multiple copies in parallel to achieve speedup. But, this is not commonly done,

precisely because of the complexity of setting up and managing such processes, particularly when they are simply one step in a larger process. As the number of parallel processes increases and the number of steps increases, it becomes impractical to set up and manage multistream applications manually. The probability of human error becomes unacceptably large, and the difficulties of even determining whether the process completed correctly can become formidable.

Therefore, it is extremely significant that these tests were set up employing previously serial applications and the highly automated facilities of OFS to run them in parallel. It is also important to note that the SAS application did require some straightforward modifications (the effort of one engineer for a few hours) in order to achieve the speedups illustrated.

Some SAS applications could be run in parallel with literally no modification. In my view, the tests are all the more realistic for Torrent's choice to use a real SAS application of some considerable complexity and include in the test protocol the minor modifications required to parallelize it.

Implications of results

In my opinion, the significance of these results is as follows: Large scale analysis, data mining and similar applications are comprised of complex multistep processes. Much of this occurs outside of the relational database, and most of it is, ordinarily, executed serially, even on a parallel hardware architecture.

The steps that are performed outside the database, such as extraction of data to a file or analysis of the data within SAS, must also be performed in parallel to avoid lengthy delays. The two simple steps illustrated in the benchmark (extracting 1.52 GB of data and analyzing it in SAS) would each take hours if performed serially. In these cases, the benefits of parallel operation within the database would be dwarfed by the long run times of the serial steps that followed.

This benchmark shows that OFS can overcome the serial bottlenecks that exist outside the database, dramatically shortening the process as a whole. The results demonstrate linear speedups with up to 32-way parallelism. Steps that would ordinarily take hours in serial execution are shortened to minutes.

The significance, then, is that OFS provides the means to parallelize (and, thus, accelerate) the performance of entire analytic processes involving multiple steps in addition to any database process. In doing so, it not only delivers the performance to realize the speedup itself but also enables the

specification and management of these multi-step processes in a parallel environment.

Conclusion

In my opinion, the OFS benchmark is a fair, appropriate, and accurate demonstration of: (a), the scalability of OFS applications, and (b), OFS as an environment for efficient parallel implementation of previously serial tasks.

About the reviewer

Richard Winter, founder and president of the Winter Corporation, has over 25 years of experience in research, product development, and implementation in the very large database field. He speaks and publishes extensively on scalable databases and is regarded as one of the world's leading authorities on the subject. He can be contacted at:

Winter Corporation

186 Lincoln Street, Suite 611

Boston, MA 02111

(617) 695-1800

Fax: (617) 338-4499

<http://www.wintercorp.com>

Parallel architecture basics

The following sections explain the basics of parallel architecture in order to help the reader understand the testing procedures.

Symmetric Multiprocessing System (SMP)

This is a machine which has a shared memory, and one or more CPUs which use that memory subsystem. It is running one instance of an operating system, which handles the memory sharing, as well as other system components.

Massively Parallel Processing System (MPP)

An MPP system is a collection of processing nodes connected by a high-bandwidth switch. Each node runs its own copy of the operating system and can be either a uniprocessor or an SMP system. The MPP system is often referred to as a *shared nothing* architecture because each node has its own operating system, memory, and disk. You can grow the computer by adding more processors, memory, and disk.

Each architecture is ideally suited for a certain class of applications. An example of a task that requires an SMP system is one where all data needs to be visible to all processors during the analysis. MPP architecture lends itself well to applications in which large amounts of data can be grouped and the groups then processed independently, such as alphabetizing. The IBM RS/6000 Scalable POWERparallel (SP) system is an MPP system. One of the strengths of the SP architecture is that nodes can be uniprocessors or SMP systems combining the benefits of SMP and MPP approaches to optimize processing for a wide range of workloads.

The tests were run on the SP system because of its flexibility and popularity with SAS customers. The distributed memory architecture and shared-nothing design offer an extremely scalable platform that can handle incremental growth without encountering typical performance bottlenecks where too little memory can be allocated to a large processing problem. The configuration that had both SMP and uniprocessors for nodes demonstrated dramatic scalability. This scalability was achieved because the software that was used exploited the capabilities of both the SMP and MPP systems used.

The DB2 UDB parallel database, working in conjunction with the SAS application and Torrent's Orchestrator for the SAS System, made it possible to run processes in parallel in order to achieve speedups and run applications that were previously not possible with serial processing.

Additional information is available on the IBM and Torrent Web sites at

<http://www.ibm.com/> and <http://www.torrent.com/> or from representatives of both companies.

Appendix H. Special notices

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, North Castle Drive, Armonk, NY 10504-1785.

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.

This document contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples contain the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

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:

AIX ®	AIXwindows ®
AS/400 ®	AT
CT	DB2 ®
ESCON ®	IBM ®
LoadLeveler ®	SP
Magstar ®	Micro Channel ®
Netfinity ®	POWERparallel
PowerPC Architecture	PowerPC 604 ®
RS/6000 ®	System/390 ®
XT	

The following terms are trademarks of other companies:

Tivoli, Manage. Anything. Anywhere., The Power To Manage., Anything. Anywhere., TME, NetView, Cross-Site, Tivoli Ready, Tivoli Certified, Planet Tivoli, and Tivoli Enterprise are trademarks or registered trademarks of Tivoli Systems Inc., an IBM company, in the United States, other countries, or both. In Denmark, Tivoli is a trademark licensed from Kjøbenhavns Sommer - Tivoli A/S.

C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other

countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

PC Direct is a trademark of Ziff Communications Company in the United States and/or other countries and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States and other countries. (For a complete list of Intel trademarks see www.intel.com/tradmarx.htm).

UNIX is a registered trademark in the United States and/or other countries licensed exclusively through The Open Group.

SET and the SET logo are trademarks owned by SET Secure Electronic Transaction LLC.

Other company, product, and service names may be trademarks or service marks of others.

Appendix I. Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

I.1 IBM Redbooks publications

For information on ordering these ITSO publications see “How to get IBM Redbooks” on page 187.

- *GPFS: A Parallel File System*, SG24-5165

I.2 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at <http://www.redbooks.ibm.com/> for information about all the CD-ROMs offered, updates, and formats.

CD-ROM Title	Collection Kit Number
System/390 Redbooks Collection	SK2T-2177
Networking and Systems Management Redbooks Collection	SK2T-6022
Transaction Processing and Data Management Redbooks Collection	SK2T-8038
Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
Netfinity Hardware and Software Redbooks Collection	SK2T-8046
AS/400 Redbooks Collection	SK2T-2849
Netfinity Hardware and Software Redbooks Collection	SK2T-8046
RS/6000 Redbooks Collection (BkMgr Format)	SK2T-8040
RS/6000 Redbooks Collection (PDF Format)	SK2T-8043
Application Development Redbooks Collection	SK2T-8037
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

I.3 Other resources

These publications are also relevant as further information sources:

- *SAS Companion for UNIX Environments: Language*, Version 6, Cary, NC: SAS Institute Inc., 1993, ISBN 1-55544-565-9
- *SAS Language*, Version 6, Cary, NC: SAS Institute Inc., 1990, ISBN 1-55544-381-8

- *SAS Language and Procedures: Usage*, Version 6, First Edition, ISBN 1-55544-371-0
- *SAS/STAT User's Guide*, Volumes 1 and 2, Version 6, Fourth Editions, Cary, NC: SAS Institute Inc., 1989, ISBN 1-55544-376-1
- *SAS Procedures Guide*, Version 6, Cary, NC: SAS Institute Inc., 1990, ISBN 1-55544-378-8
- *SAS Programming Tips: A Guide to Efficient SAS Processing*, MS56150
- *AIX Survival Guide*, Andreas Siegart, New York, Addison-Wesley, 1996, ISBN: 0-201-59388-2
- *RS/6000 Facts and Features*, G320-9878
- *General Parallel File System for AIX: Install & Admin Guide*, SA22-7278
- *AIX Version 3.2 & 4 Performance Tuning Guide*, SC23-2365
- *AIX Commands Reference*, Volumes 1 through 6, SBOF-1851
- *An Introduction to GPFS R1* - White paper available on the Web at: www.austin.ibm.com/resource/technology
- *Tips to Using the SAS System*, SAS Institute white paper
- *The SAS System and GPFS - A Scalable Solution*, white paper by Keith F. Olsen and James T. West of the IBM Corporation

I.4 Referenced Web sites

These Web sites are also relevant as further information sources:

- <http://www.sas.com/partners/enterprise/ibm/Gpfsfina.pdf>
- http://www.sas.com/partners/enterprise/ibm/RISC_DB2.pdf
- <http://www.sas.com/partners/ibm/optimize.html>

How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** <http://www.redbooks.ibm.com/>

Search for, view, download, or order hardcopy/CD-ROM Redbooks from the Redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

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.

- **E-mail Orders**

Send orders by e-mail including information from the IBM Redbooks fax order form to:

	e-mail address
In United States	usib6fpl@ibmmail.com
Outside North America	Contact information is in the "How to Order" section at this site: http://www.elink.ibm.com/pbl/pbl

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the "How to Order" section at this site: http://www.elink.ibm.com/pbl/pbl

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the "How to Order" section at this site: http://www.elink.ibm.com/pbl/pbl

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the Redbooks Web site.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and Redbooks by accessing the IBM Intranet Web site at <http://w3.itso.ibm.com/> and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access MyNews at <http://w3.ibm.com/> for redbook, residency, and workshop announcements.

Glossary

CMOS. Complimentary metal-oxide semiconductor.

DASD. Direct Access Storage Device.

IBM. International Business Machines Corporation.

ITSO. International Technical Support Organization.

JFS. Journaled File System. This is the standard file system in AIX.

LP. LogicalPartition. The term is used in AIX for a Logical partition equal to the physical partition unless we are using mirroring where one Logical partition can exist in more PPs, one for each copy in mirror.

LV. LogicalVolume. The term is used in AIX for a Logical volume or disk partition.

OLTP. Online Transaction Processing.

PCI. PCI is a standard low-cost Bus system now used in all RS/6000

PP. Physical Partition. This term is used in AIX for the base physical chunk size in a volume group.

PV. Physical Volume. This term is used in AIX for a physical DASD.

SAS. This is base SAS software from the SAS Institute Inc.

SCSI. Small Computer System Interface. SCSI is a standard for device interface and is typically used for DASD and other devices.

VG. Volume group. A group of physical volumes with the same style.

Index

Symbols

% idle 54
% iowait 54
% sys 53
% tm_act 54
% user 53

Numerics

2493 SCSI-2 Fast/Wide RAID Adapter 14
2494 PCI 3-Channel Ultra SCSI RAID Adapter 15
43P Series 12
6206 PCI Single-Ended Ultra SCSI Adapter 14
6207 PCI Differential Ultra SCSI Adapter 14
6208 SCSI-2 Fast/Wide Adapter 4-A 14
6209 SCSI-2 Differential Fast/Wide Adapter 4-B 14
6215 SSA RAID 5 Adapter with 6222 SSA
Fast-Write Cache Option 15
6222 SSA Fast-Write Cache Option 15
6225 Advanced SerialRAID Adapter 15
64-bit technology 3
7043 Models 12

A

adapter performance 14
adapters, communications
 performance notes 14
adapters, storage
 performance notes 14
 SCSI
 2493 SCSI-2 Fast/Wide RAID Adapter 14
 2494 PCI 3-Channel Ultra SCSI RAID Adapter 15
 6206 PCI Single-Ended Ultra SCSI Adapter 14
 6207 PCI Differential Ultra SCSI Adapter 14
 6208 SCSI-2 Fast/Wide Adapter 4-A 14
 6209 SCSI-2 Differential Fast/Wide Adapter 4-B 14
 SSA
 6215 SSA RAID 5 Adapter with 6222 SSA
 Fast-Write Cache Option 15
 6225 Advanced SerialRAID Adapter 15
AIX 44, 46, 51
AIXwindows 46
algebra 47

aligned 95
aligned data 95
autoexec.sas 125

B

block diagram explanation 2
block size 20
Boardroom data 28
buddy buffers 154
buffer size 38, 40
BUFNO 32, 34, 39
bufno 40
BUFSIZE 32, 33, 34, 36, 39, 96
Business Intelligence 157
BY 36

C

CATCACHE 96
Census Bureau 27
Census Bureau files 150
center 17
CFO Vision 90
change block size 20
chdev 20
CLASS Statement 102
CLASS statement 104
cleanwork 35
CLIO/S 128
Cluster 121
Cluster Operations 121
CMAX 102
commercial computing 13
COMPRESS 97
compression 35
computer system block diagram explanation 2
config.sas612 32
COPIES 18
CPU 93

D

data disks 17
Data Sharing 87, 121
Data sharing 123
DATA STEP 29
data transformer 128
data warehouses 157

- database access programs 137
- DB2 Client Application Enabler 130
- DB2 Partitioned Database 121
- DB2 partitioned database 128
- DB2NODE 143
- design overview, RS/6000 1
- Deskside Server
 - F50 12
- Disk controller 107
- disk layout 17
- disk space 17, 89
- distribution media 19
- DROP 93

E

- elapsed time 44
- Enterprise Miner 90
- EXACT 101
- extranet 13

F

- Facts and Features 4
- FC-AL 3
- feature codes
 - 2493 SCSI-2 Fast/Wide RAID Adapter 14
 - 2494 PCI 3-Channel Ultra SCSI RAID Adapter 15
 - 6206 PCI Single-Ended Ultra SCSI Adapter 14
 - 6207 PCI Differential Ultra SCSI Adapter 14
 - 6208 SCSI-2 Fast/Wide Adapter 4-A 14
 - 6209 SCSI-2 Differential Fast/Wide Adapter 4-B 14
 - 6215 SSA RAID 5 Adapter 15
 - 6222 SSA Fast-Write Cache Option 15
 - 6225 Advanced SerialRAID Adapter 15
- Fibre Channel Arbitrated Loop 3
- file system 18
- file systems 24
- FIRSTOBS 93
- FMTLEN 100
- formatted length 101
- free list 52
- FREQ Procedure 90
- FULLSTIMER 29

G

- GPFS 87, 145

- GPFS block size 154
- GPFS Installation 156

H

- high node 162
- history
 - of RS/6000 product line 1
- household record 28
- HR Vision 90
- HRECS 28

I

- I/O 19, 93
- IBM DB2 parallel database 161
- IMPLMAC 97
- Indexing 36
- INFLUENCE 103
- inner_edge 17
- inner_middle 17
- input 38
- input data 19
- input datasets 17
- installation 17
- Internet 13
- intranet 13
- iostat 51, 53

J

- JFS 18
- journaled file system 18

K

- KEEP 93

L

- large files 36
- large SAS applications 157
- Large Scale Servers 13
- LENGTH 93
- libraries 26
- linear regression 29
- linear speedup 173
- Load Balancing 121
- Load balancing 85, 86
- LoadLeveler 86, 123
- logical partitions 18
- logical volumes 18

LOGISTIC 90, 101

LP 18

lsattr 20

LV 18

M

matrix algebra 47

MAUTOSOURCE 97

maxfree 52

MDDB 101

MDDB Cube Creation 89

MEANS 90

media 20

memory 31, 54, 93

memory requirements 47

MEMSIZE 29, 32, 33, 34, 43, 47, 94

METHOD 99

minfree 52

mirroring 18

MIXED 90

Model 140

facts and features summary 6

Model 150

facts and features summary 6

Model 260

facts and features summary 6

Model F40

facts and features summary 7

Model F50

facts and features summary 7

Model H50

facts and features summary 8

Model H70

facts and features summary 8

Model S70

facts and features summary 9

Model S70 Advanced

facts and features summary 9

Moore's Law 159

mount 21

MRECALL 97

MSYMTABMAX 98

Multiple Node parallel extract 121

multiple node parallel extract 136

Multiple SMP SAS Nodes Parallel Extract 121

multiple uniprocessors for a parallel extract 138

MULTTEST 102

MVARSIZE 98

N

named pipes 128

NPAR1WAY 102

NSTRATA 102

NWAY 102

O

OBS 93

OFS benchmark 179

Optimizing I/O 93

optimizing performance 93

ORCHESTRATOR 175

Orchestrator 158

outer_edge 17

outer_middle 17

output data 19

output datasets 17

P

page faults 52

paging 44

paging space 18

paging spaces 24

parallel applications 146

Parallel Extract 121, 131

Parallel extraction 164

parallel extracts 174

parallel file system 145

parallel processing 13

PATH 95

performance 37

adapters, communications and storage 14

see also adapter performance

PERMUTATION 102

personal records 28

PHREG 29, 49, 90, 103

physical memory 31, 41

physical partitions 17

physical volume 17

pipelined parallelism 170

POWER3 Microprocessor 2

PP size 17

PRECS 28

pre-installation 19

PROC GLM 29

PROC LOGISTIC 29, 49

PROC PHREG 29

PROC REG 29

PROC SORT 28, 40
 Process parallel data streams 161
 processors
 key new technologies 2
 POWER3 2
 RS64/RS64-II 3
 PSSP 85

R

Rack-Mounted Server
 H50 - Entry-Level Enterprise Server 12, 13
 H70 - Entry-Level Enterprise Server 12
 HA50 - Entry-Level Enterprise Server Solution
 12
 HA-H70 - Entry-Level Enterprise Server Solution
 12
 RAM 40, 89
 REG 103
 RESIDENT 99
 rewind 20
 rmss 42, 51
 root volume 17
 rpoolsize 154
 RS/6000 SP 13, 121
 RS64/RS64-II Microprocessor 3
 rsh 138
 run queue 52

S

SAS Cleansing 121
 SAS data 19
 SAS data file 89
 SAS Manager application 19
 SAS workspace 25
 SAS/AF 90
 SAS/Connect 123
 sas612 19
 sasoption 32
 SASROOT 20
 sasvg 26
 SASWORK 27, 34, 43
 saswork 85
 SCALABLE PERFORMANCE 157
 Scalable solutions 159
 SCSI 23
 SCSI adapters
 2493 SCSI-2 Fast/Wide RAID Adapter 14
 2494 PCI 3-Channel Ultra SCSI RAID Adapter
 15
 6206 PCI Single-Ended Ultra SCSI Adapter 14
 6207 PCI Differential Ultra SCSI Adapter 14
 6208 SCSI-2 Fast/Wide Adapter 4-A 14
 6209 SCSI-2 Differential Fast/Wide Adapter 4-B
 14
 serial applications 178
 Serial extraction 164
 SETINIT 21
 Sharing data 87
 Single Node Parallel Extract 121
 single node parallel extract 136
 smit chgtpe 20
 smit makcdr 20
 Sorting 40
 SORTPGM 99
 SORTSIZE 32, 33, 41, 43, 46, 99
 SP 13, 85
 parallel processing 13
 SP 160 MHz Thin node
 facts and features summary 11
 SP 332 MHz SMP Thin node
 facts and features summary 10
 SP 332 MHz SMP Wide node
 facts and features summary 10
 SP nodes 86
 SP POWER3 SMP Thin node
 facts and features summary 11
 SP POWER3 SMP Wide node
 facts and features summary 11
 SP switch 154
 spoolsize 154
 SSA 23, 148
 SSA adapters
 6215 SSA RAID 5 Adapter with 6222 SSA
 Fast-Write Cache Option 15
 6225 Advanced SerialRAID Adapter 15
 SSA Fast-Write Cache Option (# 6222) 15
 statistical analysis 91
 stimefmt 30
 storage
 new technologies 3
 Store large data sets in parallel 161
 STRATA 102
 stripe size 18, 164
 striped configuration 107
 striped disk 24
 SUMMARY 90
 SUMSIZE 90

- svmon 51, 54
- system activity 51
- System Managed Storage 164
- system performance 19
- system planning 19
- system resources 31

T

- tar 21
- tctl 20
- technologies, key new 3
 - 64-bit technology 3
 - processor technologies 2
 - storage technologies 3
- test environment 23
- thewall 154
- thin node 162
- Torrent Systems 158
- True Parallel Extract 122

U

- unaligned data 95
- unformatted length 101
- UNIX 27
- utilities 35

V

- VG 17
- Virtual memory 51
- virtual memory 53
- vmstat 45, 51
- vmtune 51
- volume group 17

W

- wait queue 52
- WHERE 36, 93
- wide node 162
- Winter Corporation 176
- work directory 122
- work space 17, 19
- workgroup server 12
- Workgroup Servers (Entry) 12
- Workstations 12
- write cache disabled 108
- write cache enabled 114

IBM Redbooks evaluation

Implementing SAS on the RS/6000 Family
SG24-5513-00

Your feedback is very important to help us maintain the quality of IBM 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!)

SG24-5513-00
Printed in the U.S.A.

Implementing SAS on the RS/6000 Family

SG24-5513-00

