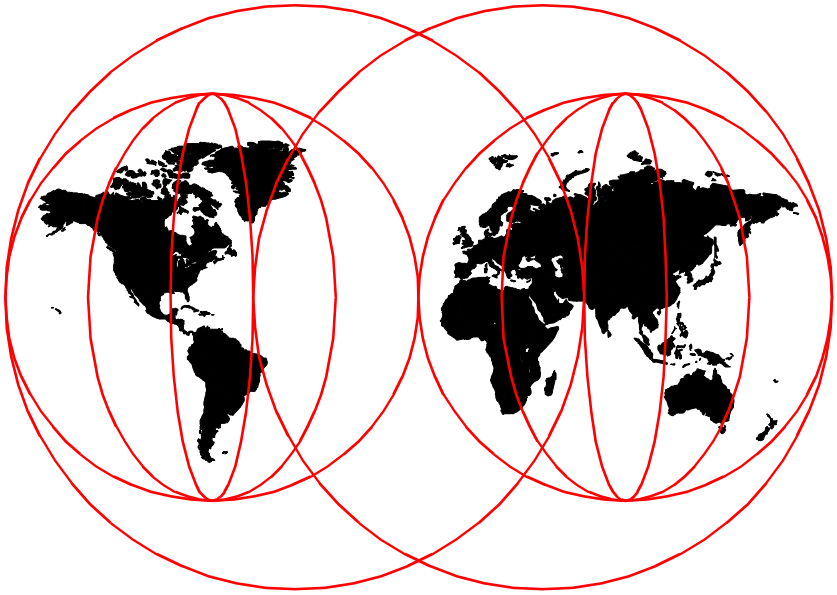
Oracle8*i* Parallel Server on IBM SP Systems: Implementation Guide

Abbas Farazdel, Xianneng Shen, Terry Stevens



International Technical Support Organization

www.redbooks.ibm.com

SG24-5591-00

SG24-5591-00

International Technical Support Organization

Oracle8*i* Parallel Server on IBM SP Systems: Implementation Guide

December 1999

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix I, “Special notices” on page 141.

First Edition (December 1999)

This edition applies to Oracle8*i* Parallel Server, Release 8.1.5 for use with AIX 4.3.2 or later and PSSP 3.1 or later on IBM RS/6000 Scalable POWERparallel (SP) systems.

Note

This book is based on a pre-GA version of a product and may not apply when the product becomes generally available. We recommend that you consult the product documentation or follow-on versions of this redbook for more current information.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. JN9B Mail Station P099
522 South Road
Poughkeepsie, New York 12601-5400

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

Contents

Figures

Tables

Preface

This redbook is designed to help you install, tailor, and configure the new Oracle8*i* Parallel Server, Release 8.1.5 (OPS), on an IBM RS/6000 SP System. It walks you through the process of creating both a small starter multi-instance database and a larger (about 1 TB) example multi-instance database. The instructions are fairly detailed and include the specific commands that are needed to accomplish the tasks at hand.

This redbook is written in cookbook style and pertains only to an IBM SP environment running AIX 4.3.2 or later and PSSP 3.1 or later. It is primarily intended for anyone responsible for:

- Installing and configuring OPS
- Creating a multi-instance OPS database
- Administering OPS

This redbook also discusses the following topics:

- Multi CPU technologies, such as SMP, clusters, and MPP (shared-nothing and shared-disk)
- OPS - IBM RS/6000 SP architecture
- Installing and configuring IBM Virtual Shared Disks (VSD)

The team that wrote this redbook

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

Abbas Farazdel is an SP Technical Consultant, System Strategist, and Senior Project Manager at the International Technical Support Organization, Poughkeepsie Center. Before joining the ITSO in 1998, Dr. Farazdel worked in the Global Business Intelligence Solutions (GBIS) group at IBM Dallas as an Implementation Manager for Data Mining Solutions and in the Scientific and Technical Systems & Solutions (STSS) group at the IBM Thomas J. Watson Research Center as a High Performance Computing Specialist. Dr. Farazdel holds a PhD in Computational Quantum Chemistry and an MS in Computational Physics from the University of Massachusetts.

Xianneng Shen is a programming consultant with the IBM RS/6000 SP Teraplex Integration Center. His current interests include VLDB, scalable

parallel database, data warehouse modeling, design, and implementation on RS/6000 SP systems. Before joining IBM, Xianneng worked as a Senior Advisor at the Cornell Theory Center. He holds a PhD in Electrical Engineering and an M.S. in Computer Engineering from Syracuse University.

Terry Stevens is a Senior Oracle DBA working with Energis Communications in the U.K. He has over 15 years experience in the IT industry including 10 years working with the ORACLE Product Set. His areas of expertise include RS/6000 SP and implementing large Data Warehouse systems.

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

Marcelo Guelfi
IBM Uruguay

Gary McGalliaro, Sana Karam, Mike Li, Peter Lu, Kapil Sharma
Oracle Corporation

Special thanks to the following people whose contributions of manpower and equipment were instrumental to the successful completion of this redbook project:

Joseph M. Catucci, Kurt Sulzbach, Xianneng Shen
RS/6000 SP Teraplex Integration Center, IBM Poughkeepsie

Joseph F. Kelly, James C. Wang, Sharon A. Selzo
RS/6000 Benchmark and Enablement Center, IBM Poughkeepsie

Joseph A. Labriola, David J. Nandor
IBM Poughkeepsie

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 159 to the fax number shown on the form.
- Use the electronic evaluation form found on the redbooks Web sites:

  For Internet users `http://www.redbooks.ibm.com/`
  For IBM intranet users `http://w3.itso.ibm.com`

- Send us a note at the following address:

  redbook@us.ibm.com

Chapter 1. Introduction

The Oracle8*i* Parallel Server (OPS) product and IBM RS/6000 SP (Scalable POWERparallel) systems combine to produce complete database solutions for today's high-end business requirements.

In this chapter, some background material is provided for the reader who is not necessarily versed in the possible architectures for parallel database software and parallel hardware. With this material, you will learn specific details about the RS/6000 SP and the Oracle8*i* Parallel Server and how they complement each other.

The proof is in the testing; so, benchmark data illustrates the performance of OPS on the RS/6000 SP. One sees that adding SP nodes results in an almost linear increase in parallel database processing speed. For example, a 46-node SP produces about 45 times greater performance than a single SP node with the same configuration. Such results show that an SP with Parallel Server is a safe investment for companies facing massive database requirements. As requirements increase, the installation can expand the SP to handle them.

1.1 Background

As the global economy becomes increasingly competitive, companies learn to use information as a strategic asset to gain market advantage. Having collected massive amounts of data over the years, these companies look for tools to help them use this information to gain better and more timely insights into customer requirements and to manage their businesses more effectively.

One way to gain better and more timely insight into customer requirements is to use Decision Support Systems (DSS). Typically, DSS applications query data warehouses for specific customer buying information.

These data warehouses contain a variety of information gathered over the years from the many systems that a company typically depends on to run its everyday business transactions. The amount of information that a DSS must manipulate is massive and tends to grow at a high rate. According to the International Group (Los Altos, California), in 1970, a typical Fortune 500 company possessed eight billion characters of electronic data. By 1990, this amount had increased to 28 trillion characters, and, by the year 2000, the number is expected to increase to 400 trillion characters (see Figure 1 on page 2).

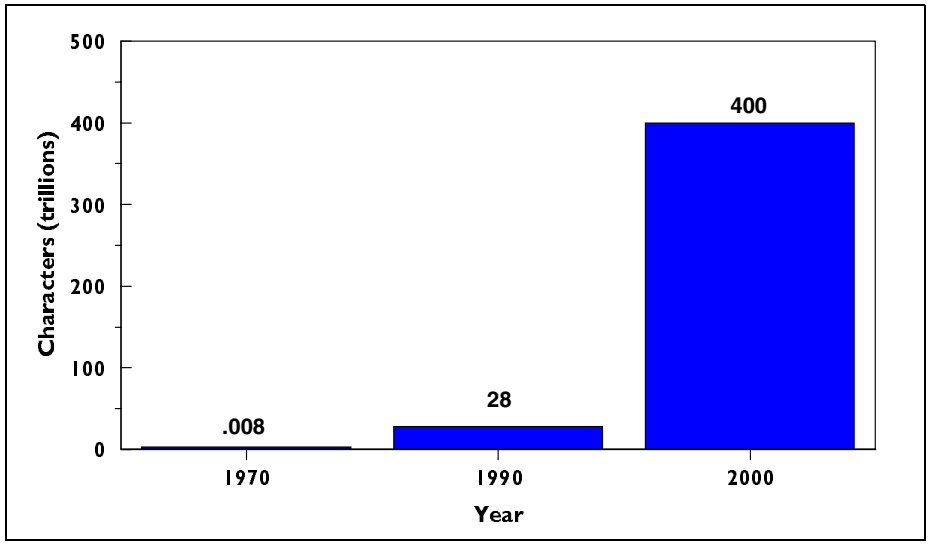


Figure 1. Growth in stored data

To effectively use so much information, companies require technology that efficiently and cost-effectively manages their data. What is more, companies need to make complex queries to the data warehouse, and they need to receive their answers in a timely manner. To do this, companies must invest in dependable hardware and software that provides room for growth.

1.2 Parallel computing is the answer

The enabling technology that allows the efficient and successful implementation of a DSS is found in parallel computing. Traditional single-processor systems simply cannot provide the computing power to manipulate trillions of information bits in an efficient and timely manner. The most powerful and effective solution requires breaking down a workload into separate pieces, each of which is executed by a separate processor. The use of hundreds of processors delivers a performance increase almost unimaginable with a single processor.

The technology to accomplish this feat is known as Massively Parallel Processing (MPP). The IBM RS/6000 SP computer uses MPP technology to achieve its high-end performance.

In the past, companies had to buy proprietary special-purpose machines and operating systems to get SP-like processing power and performance. While these systems could solve large-scale problems, they were expensive to purchase and administer and difficult to scale, eventually running out of capacity.

1.3 Why the RS/6000 SP?

With the introduction of the RS/6000 SP to the marketplace in 1993, MPP technology came of age, and its deployments became viable for businesses that once considered such computing power unattainable. In fact, the RS/6000 SP has become the de facto standard for general-purpose open MPP solutions. It combines the benefits of open systems with the power of large-scale parallel processing and the IBM commitment to business-critical computing.

Since its introduction, the SP supercomputer has built an impressive resume:

- It has been installed in over 70 percent of U.S. Fortune 500 companies as well as scientific and technical institutions worldwide.
- It has been selected by the U.S. Department of Energy for the Accelerated Strategic Computing Initiative (ASCI) project, which will result in the creation of the most powerful computer in human history.
- It has beaten world chess champion and Grand Master, Gary Kasparov.
- It has expanded from zero to approximately 28 percent of the total IBM UNIX-based system revenue.
- It has repeatedly set world records for database and Internet Web serving capability, most recently at the Nagano Winter Olympics.

The RS/6000 SP provides a stable, cost-effective, commercial MPP solution specially suited for scalable high-performance decision support and data warehousing applications as well as a variety of scientific and engineering applications. It represents a new MPP technology generation built from standard off-the-shelf hardware components that run parallel-enabled open systems-based operating systems and standard relational database managers, such as Oracle.

Why has the RS/6000 SP become the de facto standard for such computing? The keys to the acceptance of this new technology include:

- Lower total cost of ownership
- Use of open systems technology

- Easy access to legacy data in the data warehouse
- Cost-effective high performance
- Scalability to ultra-large sizes
- Flexibility to support diverse workloads
- High availability
- Ease of administration

These benefits are only briefly addressed in this redbook, the main subject of which is the Oracle8*i* Parallel Server-IBM RS/6000 SP combination. Further information on the cost of ownership and more is available from IBM and Oracle.

To fully comprehend the advantages of the OPS and IBM RS/6000 SP combination, it is necessary to understand the advent of the MPP architecture. To do this, let us first look at the difference between a symmetric multiprocessing (SMP) computer hardware system, a computer hardware cluster, and a massively parallel processor (MPP) computer hardware system. Let us also examine the differences between a shared-nothing and a shared-disk hardware/database architecture.

1.4 Hardware overview: SMP, clusters, and MPP

Various parallel computing architectures have been in existence for more than 10 years. The goal of parallel computing is to achieve high-performance computing in a cost-effective manner. Instead of using a single very powerful (but also very expensive) processor, the goal is to divide complex tasks among a number of inexpensive processors.

Traditional uniprocessor systems are limited in their processing power. Once the CPU's processing power is saturated, there is no way to increase the system computing power.

Symmetric multiprocessing (SMP) systems have multiple CPUs sharing main memory and disk access. Such shared memory access requires a common memory bus that has a fixed bandwidth and can, therefore, only support a limited number of CPUs. If this number is exceeded, increasing the number of system CPUs does not result in a performance gain since the fixed-bandwidth memory bus is a bottleneck. SMP system scalability is fundamentally limited by the fixed-bandwidth memory bus, no matter how many bus design advances are made or whether a proprietary bus is used.

Clusters provide the next step in complexity and performance after SMP systems. With a cluster architecture, multiple SMP or uniprocessor nodes are coupled using a fast interconnect mechanism (typically Ethernet or FDDI). Usually, the cluster nodes share a common disk pool. This architectural concept eliminates the shared memory access bottleneck. However, the interconnect has a fixed bandwidth; so, clusters introduce this new bottleneck, which limits system scalability.

Note that a cluster interconnect scalability limitation is intuitively obvious when you consider how a so-called fast interconnect can hamper performance when CPUs or different nodes need to communicate and cooperate. At such times, processor-level speeds are required, and fast interconnect is not fast enough to keep pace with processor needs.

The MPP architecture removes the interconnect bottleneck. With an MPP system, the goal is to have a scalable interconnect. With each node added to the system, the bandwidth of the interconnect increases.

A scalable interconnect means that an MPP system offers a solution for applications requiring very large complex queries (for example, in data warehousing environments) and fast OLTP processing and applications requiring huge disk storage. The scalable MPP system is often the only solution, especially if the application growth is very fast and/or unpredictable.

As an additional advantage, note that MPP systems also support scalable parallel I/O subsystems. These subsystems allow the MPP to increase disk throughput as the number of I/O processors increases. With potentially hundreds of I/O channels and network adapters, such well-balanced systems are ideal to support client-server database systems where the required CPU power and disk space grow with the increasing number of user network connections.

Note

The latest trend is a combination of SMP and MPP systems, which is implemented in IBM RS/6000 SP. In this design, SMP systems are integrated as nodes inside an MPP system to combine the advantages of both technologies.

1.5 Software overview: Shared-nothing and shared-disk

Now that we have an understanding of the MPP architecture and its antecedents, let us look at modern database software. In today's parallel

database market, there are two basic database architectural approaches. These are the shared-disk and the shared-nothing designs.

A shared-disk database architecture is characterized by the fact that all hardware nodes share all the database data. Examples of databases using this approach are Oracle's Parallel Server and the IBM mainframe-based HPQS.

In contrast, in a shared-nothing database architecture, the database is divided into partitions based on partition keys (one key per table). Each node has exclusive access to a single partition. Examples of this architecture include Informix XPS, Sybase MPP, IBM UDB EEE, Tandem Nonstop SQL, and NCR Teradata.

Both database architectures have strengths and weaknesses that are discussed later. However, it is important to remember that the issue of an architecture's strengths and weaknesses is secondary to the most important issue, which is: How well does the overall database product serve the requirements of a given application or business need? To help make this decision, it is useful to understand the basics of shared-disk and shared-nothing environments.

1.6 Compare and contrast

A shared-disk environment is inherently more flexible. Logically and physically, there is only one database. This uniqueness allows the addition of nodes and disks to increase system performance without database reconfiguration. Since every node has access to the entire database, adding a node provides an immediate increase in system throughput. On the other hand, additional communication overhead is incurred since data consistency must be maintained across nodes.

If the data can be partitioned well, a shared-nothing environment might be more scalable because there is no shared resource. But, there is reduced flexibility. For example, if the query pattern changes and does not match the current database partitioning scheme well, performance might be very bad, and the database might have to be repartitioned. It might also be difficult to accommodate a varying workload with a shared-nothing environment's fixed partitions.

In a shared-nothing environment, it is imperative that hardware capabilities be matched with equally-impressive system and performance administration tools. Only the RS/6000 SP has all of the hardware and software to meet such needs. Specifically, IBM system software provides for the administration

of the entire SP complex as a single system. Added to the mix are equally-easy IBM AIX management and administration tools.

1.7 OLTP environment

In an OLTP environment, a shared-disk environment allows flexible load balancing. Since all the nodes have access to all the data, incoming connections and tasks can be evenly spread across nodes. On the other hand, global coherency control is required to maintain data consistency.

In a shared-nothing environment, data coherency control is purely local, since each data partition can only be accessed by one node. Load balancing is somewhat inflexible because the data can only be accessed by the node directly attached to it. If all the users access the same portion of the data, one node might be overloaded while the others remain idle. Also, if a transaction updates more than one operating system instance, a resource-intensive distributed commit protocol is necessary to ensure consistency across nodes.

A shared-nothing environment is typically not considered a viable vehicle for OLTP applications because of the need for atomic transactions, whose components all go to completion as a unit (commitment) or are completely canceled (rollback). However, when the high availability features of the RS/6000 SP are coupled with the enhanced flexibility offered through IBM Parallel System Support Programs (PSSP), a shared-nothing environment becomes, in many cases, an ideal OLTP platform.

With PSSP and proper database planning and deployment, the SP can be configured as a large OLTP database server. Additionally, the high availability features of the SP make it virtually impervious to unplanned outages, and the PSSP can minimize or eliminate downtime due to planned or routine maintenance.

1.8 Query environment

In a query environment, a shared-disk database makes it possible to dynamically change the degree of CPU parallelism. This environment also allows the advantage of flexible load balancing. However, the data shipping used in this approach may require more interconnect bandwidth.

In a shared-nothing environment, functions are shipped to nodes for queries and processed locally, and only the results are shipped back to the coordinator node. Therefore, this approach may require less interconnect bandwidth.

On the other hand, queries on non-partitioned keys involve all the system nodes. Such queries are also much more sensitive to data skew. If one partition is significantly larger than the others, it can still only be worked on by one node. Therefore, the response time for the query is limited by the processing time for the largest partition, while the other nodes cannot be used to speed up the process.

The best database architecture to suit your needs will depend on what applications you run. If the query type is hard to predict, a shared-disk architecture is certainly more flexible. If a query type to be run is fairly predictable, or if much is known in advance about what queries will be run, a shared-nothing environment can be more scalable.

1.9 The hardware: RS/6000 SP architecture

IBM RS/6000 SP combines advanced hardware usually built on RS/6000 POWER technology and advanced system software PSSP using the IBM AIX operating system. There are four basic physical components of an SP (See Figure 2 on page 9):

frame This is a containment unit consisting of a rack to hold computers together with supporting hardware, including power supplies, cooling equipment, and communication media, such as the system Ethernet.

node This is a complete RS/6000 server system comprised of processors (POWER3 or PowerPC), memory, internal disk drives, expansion slots, and its own copy of the AIX operating system. A node has no display head or keyboard; so, human user interaction must be done remotely.

switch This is the medium that allows high-speed communication between nodes. See Section 1.10, “The SP Switch” on page 10.

CWS The Control Workstation (CWS) is a stand-alone AIX workstation with a display and keyboard and possessing the hardware required to monitor and control the frame(s) and nodes of the system.

An SP node can be classified as an Internal node or an External node. The former is housed in an SP frame, and the latter sits outside any frame (see Figure 2 on page 9). There are two sizes for SP frames: The Tall frame, which has 16 slots and is 75.8 inches high, and the Short frame, which has eight slots and is 49 inches high.

Internal nodes are available in three form factors: Thin, Wide and High. A Thin node occupies one slot of an SP frame; a Wide node occupies one full drawer (two slots), and a High node occupies two full drawers (four slots).

SP systems with a range of 1 to 512 nodes are generally available, although larger systems have been delivered and are successfully being used today. The SP nodes are packaged in from one to nine logical frames. Depending on the types of nodes, the number of physical frames can be greater.

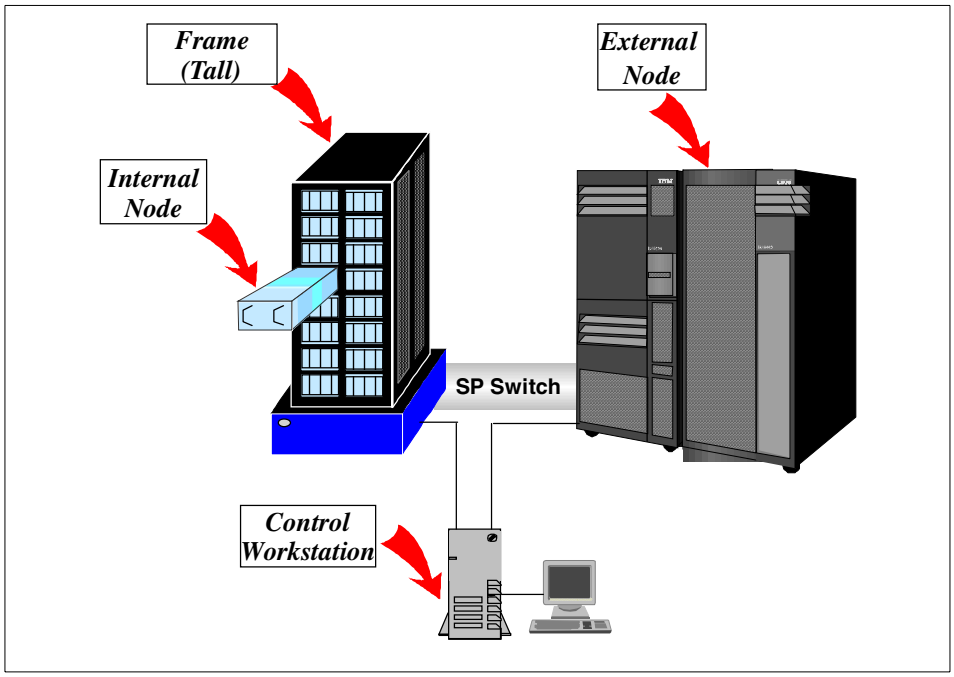


Figure 2. Typical one-frame RS/6000 with an external node

Table 1 shows different current node types offered by IBM as of September 13, 1999. The first five entries in Table 1 are internal nodes, and the last three entries are external nodes. For up-to-date information, go to the following Web site: `http://www.rs6000.ibm.com/resource/technology`

Table 1. Current SP node types

Node (MHz)	Processor	n-Way SMP	PCI Slots	Memory
Thin Node (332)	PowerPC 604+LX	2, 4	2	256 MB - 3 GB
Wide Node (332)	PowerPC 604+LX	2, 4	10	256 MB - 3 GB

Node (MHz)	Processor	n-Way SMP	PCI Slots	Memory
Thin Node (200)	POWER3 RS64	1, 2	2	256 MB - 4 GB
Wide Node (200)	POWER3 RS64	1, 2	10	256 MB - 4 GB
High Node (222)	POWER3 RS64	2, 4, 8	5-53	1 GB - 16 GB
S70 Node (125)	POWER3 RS64	4, 8, 12	14-56	512 MB - 32 GB
S7A Node (262)	PowerPC RS64-II	4, 8, 12	14-56	1 GB - 32 GB
S80 Node (450)	PowerPC RS64-III	6, 12, 18, 24	14-56	2 GB - 64 GB

1.10 The SP Switch

The current day SP uses a shared-nothing model in which each node has its own memory, disk, CPU, and so on. In order for a group of nodes to work on a problem concurrently, they must share data and status as needed via inter-node messages. These messages are sent by the components of a running application and are delivered via packets sent over the communication network chosen for the application.

To enhance the performance of these communication tasks, IBM provides a special hardware medium - the SP Switch. This switch supports a high-speed communication network (presently at 300 MB/sec peak bidirectional) that provides applications with a low-latency high-bandwidth path for message passing. Since September, 1996, the SP has been available with its second generation of switch technology - the SP Switch. The switch is considered the heart of the SP.

Each SP frame may contain an SP Switch board. The nodes of the frames are connected to their respective switch boards via a special SP Switch adapter and corresponding switch cables, and the switch boards of the system are connected via the same type of cables. Thus, a high-speed communication network that allows the nodes to communicate with each other to share data and status is formed. The primary purpose of this high-speed network is the support of solving problems in parallel.

Communication over the switch is supported by the IBM CSS (Communication SubSystem) software, which is shipped with the SP. This software is responsible for sending and receiving packets on behalf of applications to and from other nodes. It also sends packets to switch hardware components as part of its monitoring and controlling functions.

If a component of the switch network (switch board, adapter, cable, node, or software) is not functioning correctly, the CSS software is responsible for recognizing this and reporting this problem via the AIX error log. The software will also take recovery actions automatically as deemed most appropriate for the health of the system, which may mean removing the offending component from the switch network.

1.11 The database: Oracle Parallel Server

Oracle's Parallel Server is the Oracle foundation for the IBM RS/6000 SP. This software allows multiple Oracle instances to simultaneously run on different SP nodes while maintaining the single image look to the end user.

Oracle Parallel Server consists of the normal Oracle executables, a software component called the Distributed Lock Manager (DLM), and special I/O routines that use IBM virtual shared-disk (VSD) software (see Figure 3).

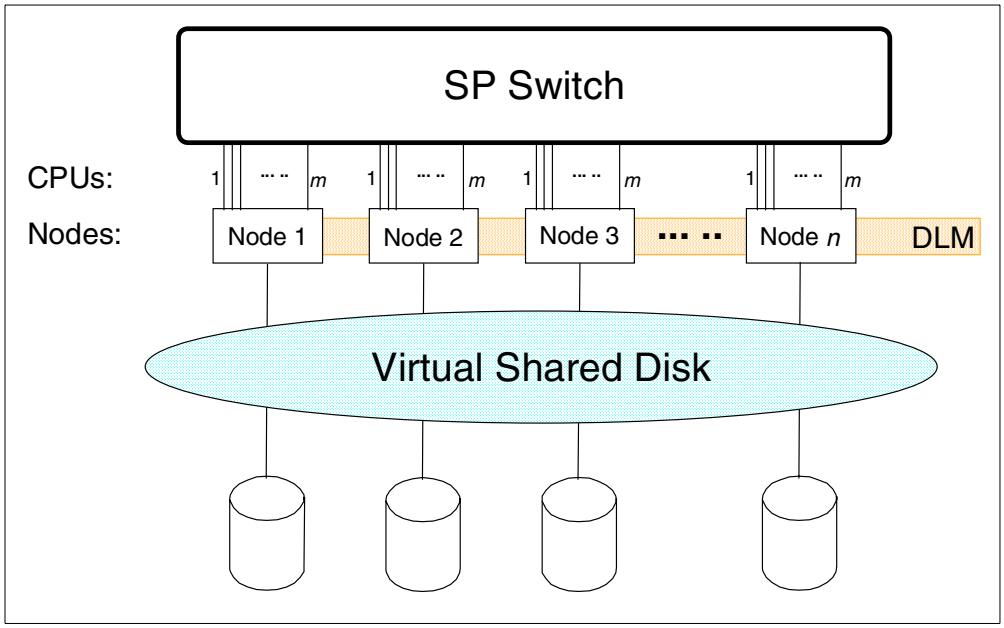


Figure 3. Oracle Parallel Server on RS/6000 SP architecture

The distributed lock manager coordinates the resources between Oracle instances on RS/6000 SP nodes. Since one Oracle instance runs on each RS/6000 SP node, Oracle uses IBM VSD software to allow instances to access disks that are physically attached to other nodes. Data to be exchanged between nodes is passed through the IBM SP Switch as shown in Figure 3.

The latest version of OPS is Oracle8*i* Parallel Server, Release 8.1.5, which is described fully in the following Oracle publications:

- *Getting to Know Oracle8i*, A68020-01
- *Oracle8i Concepts*, A67781-01
- *Oracle8i Parallel Server Concepts and Administration*, A67778-01

If you want to learn more about the IBM SP system, refer to the IBM redbook *The RS/6000 SP Inside Out*, SG24-5374. See “How to get IBM Redbooks” on page 147.

Chapter 2. Installation and configuration

This chapter describes step-by-step how to install and configure the Oracle8*i* Parallel Server (OPS) on the nodes of an IBM RS/6000 SP system. It is intended for anyone responsible for creating an OPS installation on an SP system. It assumes this person has a working knowledge of AIX and SP and has experience with Oracle.

Note

If you plan to use Oracle8*i* with an existing database from a prior release of Oracle, you must upgrade or migrate that database before mounting it using Oracle8*i*. For migration steps, refer to *Oracle8i Migration Guide*, A67774-01.

2.1 Finding installation and migration information

For more information, refer to the following manuals:

- *Oracle8i Installation Guide for AIX-Based Systems*, A67728-01
- *Oracle8i for AIX-Based Systems, Release Note*, A67730-01
- *Oracle8i Migration Guide*, A67774-01
- *Oracle8i Parallel Server, Setup and Configuration Guide*, A67439-01
- *IBM PSSP: Installation and Migration Guide*, GA22-7347
- *IBM PSSP: Administration Guide*, SA22-7348

2.2 The SP system we used

The system used for this redbook project was an RS/6000 SP system as shown in Figure 4 on page 14.

2.2.1 Software configuration

The SP system consisted of the following software components:

- AIX 4.3.2
- PSSP 3.1
- Oracle8*i* Parallel Server, Release 8.1.5

2.2.2 Hardware configuration

The SP system consisted of the following hardware components:

- A tall frame (model 550, 75.8" high) with eight drawers
- Eight 332 MHz SMP Wide nodes, that is, one node per drawer.

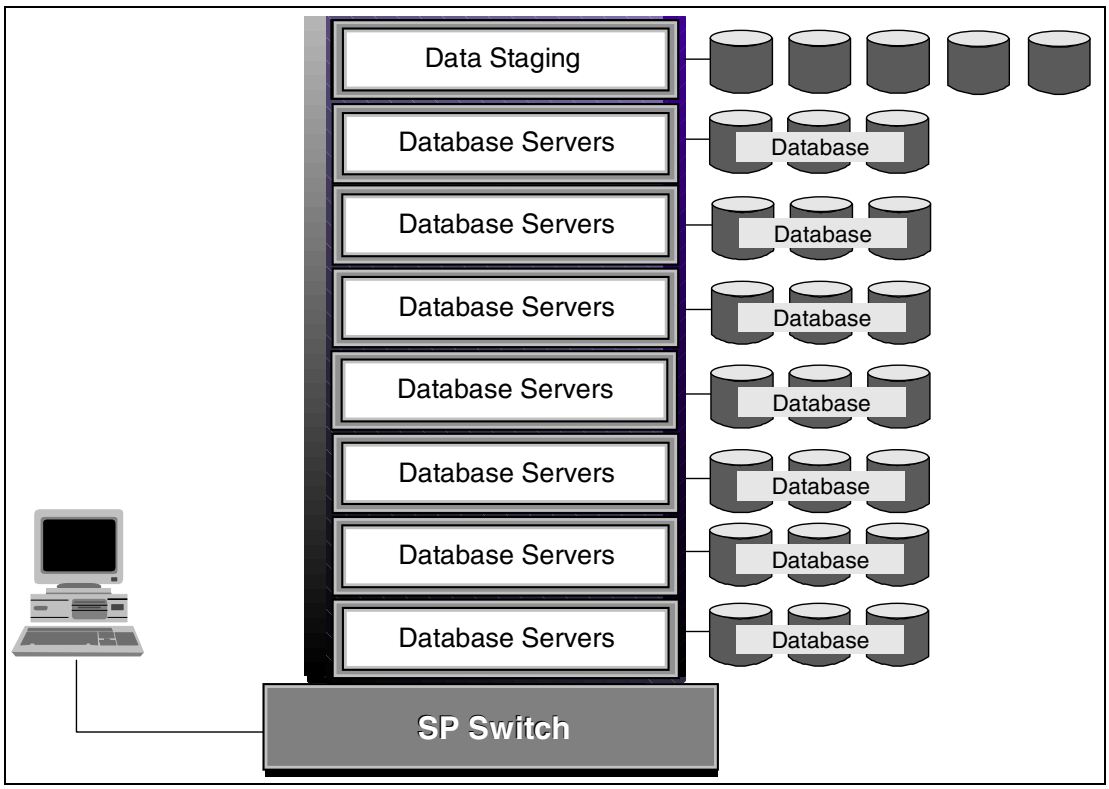


Figure 4. The SP system we used

Each node in our SP had the following:

- Four PowerPC 604e processors running at 332 MHz clock cycle.
- 3 GB of RAM.
- 128 KB of L1 cache (Instruction).
- 128 KB of L1 cache (Data).
- 1 MB of L2 cache.
- 10 PCI I/O expansion slots (three 64-bit, seven 32-bit).

- 18.2 GB of SCSI-2 Fast/Wide internal disks.
- Four SSA adapters (see Section 6.1, “Configuring the SSA disks” on page 85).
- The SP Ethernet network hostnames: spn01, spn03, ..., spn13, spn15.
- The SP Switch network hostnames: spx01, spx03, ..., spx13, spx15.
- A Control Workstation (Model 365) with:
  - 18.2 GB of SCSI internal disks.
  - 80 MB of RAM.
  - Hostname: spcws.

For any parallel processing, such as OPS, you want all your parallel processing nodes to have workloads as equal to each other as possible; otherwise, the node with the highest workload will be the bottleneck. Therefore, we configured seven of the SP nodes (spn01 - spn13) as our OPS database nodes (nodes that run OPS and contain an Oracle database). The eighth node, spn15, was designated as our staging node, that is, the node used for staging raw data - for example, by sorting and getting the data ready for loading onto the OPS nodes.

2.3 Running commands on SP nodes concurrently

You can use the `dsh` command from the SP control workstation to execute commands remotely on some or all nodes in your SP system. Although the `dsh` command uses rsh to issue remote commands, the `dsh` command offers better performance than an rsh loop because the remote commands run concurrently by default. The `dsh` command provides several ways for you to specify on what set of nodes you want your commands to run. For full details and additional parallel management commands, refer to Chapter 3, “Parallel Management Commands”, of the *IBM PSSP: Administration Guide*, SA22-7348.

In this chapter, we use a method based on the concept of a *working collective*, that is, the set of nodes to which dsh sends the commands. It can be specified by the WCOLL environment variable. If you want commands to run only on, say, nodes spn01, spn03, and so on through spn11 and spn13, you need to create a file (named, for example, /.opsnodes) on the control workstation, spcws, that contains the hostnames of these nodes (one hostname per line). Then, set the value of WCOLL to the full pathname of this file as shown in the following screen:

```
spcws# cat /.opsnodes
spn01
spn03
spn05
spn07
spn09
spn11
spn13
spcws# dsh date
dsh:  5025-507 Working collective environment variable not set
spcws# export WCOLL=/.opsnodes
spcws# dsh date
spn01: Fri Apr 16 18:58:39 EDT 1999
spn03: Fri Apr 16 18:58:40 EDT 1999
spn05: Fri Apr 16 18:58:40 EDT 1999
spn07: Fri Apr 16 18:58:40 EDT 1999
spn09: Fri Apr 16 18:58:40 EDT 1999
spn11: Fri Apr 16 18:58:40 EDT 1999
spn13: Fri Apr 16 18:58:40 EDT 1999
```

The following are useful points:

- To run a command on a specific node or nodes, use the `-w` flag, and, to run a command on all the SP nodes, use the `-a` flag for `dsh`.

```
spcws# dsh -w spn01,spn03 date
spn01: Fri Apr 16 18:58:39 EDT 1999
spn03: Fri Apr 16 18:58:40 EDT 1999
spcws# dsh -a date
spn01: Fri Apr 16 18:58:39 EDT 1999
spn03: Fri Apr 16 18:58:40 EDT 1999
spn05: Fri Apr 16 18:58:40 EDT 1999
spn07: Fri Apr 16 18:58:40 EDT 1999
spn09: Fri Apr 16 18:58:40 EDT 1999
spn11: Fri Apr 16 18:58:40 EDT 1999
spn13: Fri Apr 16 18:58:40 EDT 1999
spn15: Fri Apr 16 18:58:39 EDT 1999
```

- The path used when resolving the `dsh` command on the target nodes is the path set by the user with the DSHPATH environment variable. If DSHPATH is not set, the path used is the rsh default path /usr/ucb:/bin:/usr/bin. The DSHPATH environment variable only works when the user's remote login shell is the Bourne or Korn shell.

2.4 Task A. Satisfy prerequisites

This section describes the steps you take to make sure the system meets the hardware, software, memory, and disk space requirements for the products you intend to install.

2.4.1 Step 1: Verify hardware requirements

Ensure that each OPS node of your SP system satisfies each and every one of the following requirements:

1. A minimum of 128 MB of memory (RAM) is required for a node to install the Oracle8*i* products. However, a production RDBMS environment supporting many users requires significantly more memory. To list the memory devices on a node, use the following command (sample output on a node is shown):

```
$ lsdev -Cc memory
mem0 Available 00-0B 128 MB Memory Card
mem1 Available 00-0F 128 MB Memory Card
```

   To display the amount of real memory in kilobytes on a node, use the following command as root. Sample output on a node is shown:

```
# bootinfo -r
262144
```

   Note that 262144 bytes is indeed equal to 2x128 = 256 MB.

2. Paging space on the node is recommended to be at least twice the amount of RAM. To list the characteristics of all paging spaces on a node, enter the following command. Sample output on a node is shown:

```
$ lsps -a
Page Space  Physical Volume   Volume Group    Size   %Used  Active  Auto  Type
hd6         hdisk0           rootvg         512MB       9     yes   yes    lv
```

To add an additional paging space to a system, use the `mkps` command, and, to change the size or other characteristics of an existing paging space, use

the chps command. For example, to change the size of the myvg paging space, use the following command:

```
# chps -s4 myvg
```

This adds four logical partitions to the myvg paging space.

3. A CD-ROM device is needed on a host (for example, the control workstation) that can be NFS mounted on, at least, one of the OPS nodes. To display CD-ROM devices, use the following command. Sample output on a node is shown:

```
$ lsdev -Cc cdrom
cd0 Available 00-01-00-1,0 CD-ROM Drive
```

2.4.2 Step 2: Verify disk space requirements

When installing Oracle8*i*, the Oracle Universal Installer (OUI) gives you a choice of three installation types: Minimal, Typical, and Custom. For OPS, you must select the Custom option, which requires at least 700 MB of disk space.

This approximate space requirement does not take into account the size of your database. Again, in a production RDBMS supporting many users, significantly more disk space is required.

In addition, OUI needs at least 50 MB of free space in the /tmp directory of the SP node to run properly. In order to increase the size of /tmp by at least, say, 5 MB (that is, 10240 512-byte blocks), use the following command as root:

```
# chfs -a size=+10240 /tmp
```

If the specified size increase (here, it is 10240) is not evenly divisible by the physical partition size, it is rounded up to the closest number that is evenly divisible.

2.4.3 Step 3: Verify AIX software requirements

Perform the following steps to verify the AIX software requirements:

1. Oracle8*i* requires AIX Version 4.2.1 or later. Note that AIX 4.1 and 4.2.0 are no longer supported by Oracle8*i*. To determine the AIX maintenance level of your node, enter the following command. Sample output on a node is shown:

```
$ oslevel
4.3.2.0
```

   - If the AIX level is later than 4.3.0, for example, 4.3.1 or 4.3.2, all the required fixes for AIX 4.3.0 are implicitly installed.
   - If the AIX level is 4.3.0, ensure that fixes with the following Authorized Program Analysis Report (APAR) numbers are installed:
     - IX71948
     - IX72696
     - IX79679
     - IX81863
     - IX89087
     - IX89382

     To find out whether fixes, such as IX71948 and IX79679, are installed, use the following command. Sample output on a node is shown:

```
$ instfix -ik "IX71948 IX79679"
    All filesets for IX71948 were found.
    All filesets for IX79679 were found.
```

     To display information about each fileset associated with each APAR, use the `-ivk` flags with the aforementioned command.
   - If the AIX level is later than 4.2.1 but earlier than 4.3.0, all the required fixes for AIX 4.2.1 are implicitly installed.
   - If the AIX level is 4.2.1, ensure that fixes with the following APAR numbers are installed:
     - IX62429
     - IX67174
     - IX67978

- IX68932
- IX70521
- IX70737
- IX71865
- IX78933
- IX81957
- IX86229
- IX88178

In addition, with AIX 4.2.1, the version of the libc.a library needs to be 4.2.1.10 or higher. To display the libc.a version, enter the following command. Sample output on a node is shown:

```
$ lslpp -l bos.rte.libc
  Fileset                      Level  State      Description
  ----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  bos.rte.libc               4.3.2.0  COMMITTED  libc Library
```

In case you need it, a higher version of libc.a can be downloaded from the following IBM Website:

`http://service.software.ibm.com/support/rs6000`

2. The OUI graphical interface requires an AIX-supported window manager that supports Motif, such as dtwm, twm, or olwm.
3. You need a Web browser (preferably Version 4.0 or higher of Netscape Navigator or Microsoft Internet Explorer) to view online documentation.

2.4.4 Step 4: Verify software required for OPS

Perform the following steps to verify software required for OPS.

1. If your node is part of an HACMP/ES cluster, HACMP/ES Version 4.3 with AIX 4.3.2 is required.
2. If your node is an RS/6000 SP, refer to Table 2 and Table 3 for requirements.

Table 2. PSSP, VSD, and RVSD software requirements

Software	Requirement
PSSP	Version 2.4 with AIX 4.2.1 or 4.3.1 Version 3.1 or later with AIX 4.3.2 or later

Software	Requirement
RVSD	Version 2.1.1 with PSSP 2.4 Version 3.1 or later with PSSP 3.1 or later
VSD	Version 2.1.1 with PSSP 2.4 Version 3.1 or later with PSSP 3.1 or later

Table 3. PSSP, VSD, and RVSD software required fixes

Software	APAR Numbers
PSSP 2.4	IX80287, IX80366, IX80541, IX80611, IX80811, IX80880, IX82444, IX83367, IX84276, IX84277, IX84278, IX84280, IX84281, IX84282, IX84283
PSSP 3.1	IX83670
RVSD 2.1.1	IX86937

For more information about SP-related APARs, refer to the Service status hyperlink at the following IBM Website:

`http://www.rs6000.ibm.com/support/sp/`

Note the following information:

- OPS requires a shared disk subsystem using shared raw character devices to hold database data, log, and control files. These raw devices must be Virtual Shared Disks (VSDs) or Hashed Shared Disks (HSDs).

  At this stage, you only need to create shared disks if you choose the OUI option of automatically launching the Oracle Database Configuration Assistant (DBCA) at the end of installation in order to create a demo database. We suggest that you *not* choose the option. After Oracle installation and configuration is completed, refer to Chapter 5, “Creating a small starter database” on page 69.

  For more information, refer to *IBM PSSP: Managing Shared Disks*, SA22-7349 and *Oracle8i Parallel Server, Setup and Configuration Guide*, A67439-01.

- For additional OPS considerations, see pages 1-6 of the *Oracle8i, Migration*, A67774-01.

- If you have HACMP and PSSP concurrently on your OPS nodes, see the “HACMP and PSSP on the Same Machine” section on page 1-7 in the *Oracle8i Installation Guide for AIX-Based Systems*, A67728-01.
- For product-specific information in addition to the hardware and software requirements provided so far, see “Additional Product-Specific Installation Requirements” starting on page 1-8 in the *Oracle8i Installation Guide for AIX-Based Systems*, A67728-01.

2.5 Task B. Set the AIX environment

This section describes the steps needed to prepare your environment for installing Oracle8*i* Parallel Server.

2.5.1 Step 5: Create Oracle software filesystem (as root)

Oracle8*i* requires a filesystem for the software. We recommend having this filesystem defined locally on each and every OPS node as opposed to using NFS. The mount point of this filesystem can have any name. You can choose, for example, /u01, which is also an Optimal Flexible Architecture (OFA)-compliant name. OFA is described in detail in Appendix A, "Optimal Flexible Architecture" of the *Oracle8i Administrator’s Reference for AIX-Based Systems*, A67729-01.

To define space (with size according to Section 2.4.2, “Step 2: Verify disk space requirements” on page 18) for the /u01 filesystem, you can perform one of the following procedures depending upon the configuration of the node. Perform either Procedure I, II, or III. Do not perform more than one procedure.

Procedure I:

This procedure is the simplest and can be used if an existing volume group has enough free space. To display the names of all volume groups within a node, use the `lsvg` command on the node. To display information about a particular volume group, such as rootvg, use the following command. Sample output on a node is shown:

```
$ lsvg rootvg
VOLUME GROUP:   rootvg                  VG IDENTIFIER:  0000161728e009b4
VG STATE:       active                  PP SIZE:        16 megabyte(s)
VG PERMISSION:  read/write              TOTAL PPs:      1084 (17344 megabytes)
MAX LVs:        256                     FREE PPs:       814 (13024 megabytes)
LVs:            10                      USED PPs:       270 (4320 megabytes)
OPEN LVs:       9                       QUORUM:         2
TOTAL PVs:      2                       VG DESCRIPTORS: 3
STALE PVs:      0                       STALE PPs:      0
ACTIVE PVs:     2                       AUTO ON:        yes
MAX PPs per PV: 1016                    MAX PVs:        32 This is screen.
```

As you can see, the characteristics and status of both the logical and physical partitions of the volume group, rootvg, are displayed. In the above example, the volume group, rootvg, has more than 13 GB of free space.

When there is a volume group with enough space, use that volume group (for example, rootvg), and, within it, define a new filesystem for the Oracle software with mount point /u01.

Using SMIT:

TYPE `smit crfs`

SELECT Add a Journaled File System

SELECT Add a Standard Journaled File System
Move the cursor to your volume group (for example, rootvg), and press Enter.

TYPE Size of file system (for example, 1600000) and /u01 as mount point

SELECT **yes** for Mount AUTOMATICALLY at system restart?

PRESS **Do** to create the filesystem.

Using AIX commands:

The following command example creates the `/u01` filesystem using volume group rootvg, which is 800 MB in size, and mount point `/u01`. It also adds the `/u01` into the `/etc/filesystems` file to be automatically mounted at system restart.

```
# crfs -v jfs -g rootvg -a size=1600000 -m /u01 -A yes -a frag=4096 -a nbpi=4096 -a ag=
```

Procedure II:

This is similar to Procedure I except that you first create a logical volume before adding a new filesystem on that logical volume. This procedure allows you to optionally customize the disk space allocated to your logical volume and, in turn, to your filesystem using about 20 different logical volume attributes (see the man pages of mklv or the Add a Logical Volume dialog screen of SMIT).

Using SMIT:

TYPE `smit storage`

SELECT Logical Volume Manager

SELECT Logical Volumes

SELECT Add a Logical Volume

TYPE The volume group name (for example, rootvg), or press F4, and select rootvg.

TYPE The logical volume name (for example, ora_lv).

TYPE The number of logical partitions (for example, 200).

SPECIFY Any other attribute that you desire.

PRESS **Do** to create the logical volume.

PRESS **Cancel** four times.

SELECT File Systems

SELECT Add / Change / Show / Delete File Systems

SELECT Journaled File Systems

SELECT Add a Journaled File System on a Previously Defined Logical Volume

SELECT Add a Standard Journaled File System

SELECT Your logical volume (for example, ora_lv) after pressing F4.

TYPE /u01 as the mount point

SELECT **yes** for Mount Automatically at system restart?

PRESS **Do** to create the filesystem.

Using AIX commands:

The following `mklv` command example creates the logical volume ora_lv using volume group rootvg of 800 MB size (assuming logical partition of 4 MB). The `crfs` command creates the /u01 filesystem using logical volume ora_lv. The

latter command also adds the /u01 into the /etc/filesystems file to be automatically mounted at system restart.

```
# mklv -y ora_lv rootvg 200
# crfs -v jfs -d ora_lv -m /u01 -A yes -p rw -t no
```

Procedure III:

This is similar to Procedure II except that you allow your filesystem to have its own volume group rather than using an existing volume group.

You cannot create a separate volume group if all disks are in use or if the SP node has only one disk. Note that when working with disks that are larger than 4 GB, you must specify the physical partition size to be 8 MB or more for your volume group.

Using SMIT:

TYPE	`smit storage`
SELECT	Logical Volume Manager
SELECT	Volume Groups
SELECT	Add a Volume Group
TYPE	The volume group name (for example, ora_vg)
TYPE	Physical volume names or press F4 and select
PRESS	**Do** to create the volume group
PRESS	**Cancel** three times.
SELECT	Logical Volumes.
SELECT	Add a Logical Volume.
TYPE	The volume group name (for example, ora_vg) or press F4 and select ora_vg.
TYPE	The logical volume name (for example, ora_lv).
TYPE	The number of logical partitions (for example, 200).
SPECIFY	Any other attribute that you desire.
PRESS	**Do** to create the logical volume.
PRESS	**Cancel** four times.

SELECT File Systems

SELECT Add / Change / Show / Delete File Systems

SELECT Journaled File Systems

SELECT Add a Journaled File System on a Previously Defined Logical Volume

SELECT Add a Standard Journaled File System.

SELECT Your logical volume (for example, ora_lv) after pressing F4.

TYPE /u01 as mount point

SELECT **yes** for Mount Automatically at system restart?

PRESS **Do** to create the filesystem.

Using AIX commands:

The following commands will create a new volume group called ora_vg, using hdisk1 as the physical volume. The volume group is varied on. The `mklv` command creates the logical volume ora_lv using volume group ora_vg (size is 800 MB assuming a logical partition of 4 MB). The `crfs` command creates the /u01 filesystem using logical volume ora_lv. The latter command also adds the /u01 into the /etc/filesystems file to be automatically mounted at system restart.

```
# mkvg -f -y ora_vg hdisk1
# varyonvg ora_vg
# mklv -y ora_lv ora_vg 200
# crfs -v jfs -d ora_lv -m /u01 -A yes -p rw -t no
```

2.5.2 Step 6: Create Oracle database filesystems (as root)

Oracle8*i* requires at least one database filesystem, such as /u02, on each and every SP node that runs OPS. However, it requires at least three filesystems, such as /u02, /u03, and /u04, when creating an OFA-compliant installation.

Use one of the procedures discussed in Section 2.5.1, “Step 5: Create Oracle software filesystem (as root)” on page 22, to accomplish this.

2.5.3 Step 7: Create the OSDBA group (as root)

Create an AIX group named `dba` whose members will have Oracle DBA privileges. Oracle8*i* documentation refers to this group as the OSDBA group. You may name this group something other than `dba` if you plan to give the OSOPER group a name that is different from the OSDBA group name (see Section 2.5.4, “Step 8: Create the OSOPER group (as root)” on page 27).

The OSDBA group names and group IDs must be identical for all the OPS nodes accessing a single Oracle database. An easy way to accomplish this is to create a new group account called, for example, `dba` on the control workstation using the following command:

```
# mkgroup -A dba
```

Then, propagate this information to all OPS nodes using the SP file collection facility, the `supper` command:

```
# dsh "/var/sysman/supper update user.admin"
```

The -A flag in `mkgroup` sets yourself (that is, root) as the administrator of the group.

2.5.4 Step 8: Create the OSOPER group (as root)

This step is optional; skip it if the OSDBA group (see Section 2.5.3, “Step 7: Create the OSDBA group (as root)” on page 27) is named `dba`. In this case, the OUI assigns both Oracle DBA and Operator privileges to the AIX group, `dba`, by default.

Otherwise, create an AIX group, whose members will have Oracle Operator privileges. You need to name this group something other than `dba`. Use the procedure outlined in Section 2.5.3, “Step 7: Create the OSDBA group (as root)” on page 27.

2.5.5 Step 9: Create the oinstall group (as root)

Create an AIX group named `oinstall`. This group will own the OUI oraInventory directory. The user account that runs the installer must be a

member of this group. Use the procedure outlined in Section 2.5.3, “Step 7: Create the OSDBA group (as root)” on page 27, to create this group.

2.5.6 Step 10: Create the hagsuser group (as root)

Create an AIX group named `hagsuser`. Members of this group are allowed to initialize themselves with the IBM Group Services API. For more information, refer to *RSCT: Group Services Programming Guide and Reference*, SA22-7355.

The user account that owns the Oracle software needs to be a member of the `hagsuser` group. Use the procedure outlined in Section 2.5.3, “Step 7: Create the OSDBA group (as root)” on page 27, to create this group if it does not already exist.

2.5.7 Step 11: Create an AIX Oracle software owner account (as root)

This AIX user account (for example, `oracle`) owns the Oracle8*i* software after installation. You must run the installer OUI from this account.The `oracle` user account characteristics (username, user ID, primary group, group set, home directory, password, and so on) must be identical for all the OPS nodes accessing a single Oracle database.

You can accomplish this by following either procedure A or procedure B depending upon your preferences and the way your SP system is configured. Do not perform more than one procedure.

You can always use procedure A; however, procedure B may be used only if the attributes amd_config and usermgmt_config in your SP system are set to true (see below).

Briefly, in procedure A, you create an AIX user account on the control workstation and then replicate it on any set of SP nodes. You also cross-mount the home directory of the user on these nodes.

In procedure B, you create an SP user (an AIX user that is automatically replicated - except for the password - on all the SP nodes). The home directory of the user is mounted automatically via the AIX automount daemon.

Procedure A:

This procedure may always be used.

1. On the control workstation, create an AIX user with the following attributes:

- User Name: oracle
- Primary Group: dba
- Group Set: dba, oinstall, hagsuser
- HOME Directory: /home/spcws/oracle
- Password: for example, oracle

2. Next, propagate the information to the nodes using the `supper` command. You can use the SMIT interface or the following commands:

```
spcws# mkuser pgrp=oinstall groups=dba,oinstall,hagsuser \
home=/home/spcws/oracle oracle
spcws# lsuser -f oracle
.
.           Ensure attributes are acceptable. Otherwise change them using chuser.
.
spcws# passwd oracle
Changing password for "oracle"
oracle's New password:
Re-enter oracle's new password:
spcws# dsh "/var/sysman/supper update user.admin"
```

3. Although any host that is accessible through the network can be selected to physically contain the home directory, here, we chose the control workstation `spcws`. In addition, you need to create an empty directory (for example, `/home/spcws/oracle`) on each OPS node for mounting the home directory of the `oracle` account (some sample output is shown in the following screen):

```
spcws# /usr/sbin/mknfsexp -d '/home/spcws' -t 'rw' '-B'
spcws# dsh "mkdir /home/spcws"
spcws# dsh "mount spcws:/home/spcws /home/spcws"
spcws# dsh df | grep spcws
spn01: spcws:/home/spcws 98304     10096   90%     6626    27% /home/spcws
spn03: spcws:/home/spcws 98304     10096   90%     6626    27% /home/spcws
spn05: spcws:/home/spcws 98304     10096   90%     6626    27% /home/spcws
spn07: spcws:/home/spcws 98304     10096   90%     6626    27% /home/spcws
spn09: spcws:/home/spcws 98304     10096   90%     6626    27% /home/spcws
spn11: spcws:/home/spcws 98304     10096   90%     6626    27% /home/spcws
spn13: spcws:/home/spcws 98304     10096   90%     6626    27% /home/spcws
```

Procedure B:

This procedure may only be used if the attributes amd_config and usermgmt_config in your SP system are set to TRUE.

To check these attributes, display your SP's configuration data from the System Data Repository (SDR) using the following on the control workstation:

Using SMIT:

TYPE `smit list_data`

SELECT List Site Environment Database Information

PRESS **Do** to get the output.

4. Check the amd_config and usermgmt_config attributes.

Using AIX commands:

The following command on the control workstation is equivalent to the aforementined procedure (partial sample output is shown):

```
# splstdata -e
List Site Environment Database Information
attribute               value
-------------------------------------------------------------------------------
control_workstation     spcws
.                       .
.                       .
amd_config              true
.                       .
.                       .
usermgmt_config         true
passwd_file             /etc/passwd
passwd_file_loc         spcws
homedir_server          spcws
homedir_path            /home/spcws
.                       .
.                       .
```

Check amd_config and usermgmt_config attributes. If both attributes are not true, you have to use Procedure A, unless you are willing to change the configuration of your system. However, if both attributes are set to true, you may proceed with the rest of Procedure B outlined here.

1. From the control workstation, create an SP user with the following attributes:
   - User Name: oracle
   - Primary Group: dba
   - Group Set: dba, oinstall, hagsuser
   - HOME Directory: /home/spcws/oracle

- Password: oracle

You can use the SMIT interface or the following commands:

```
spcws# /usr/lpp/ssp/bin/spmkuser pgrp=oinstall groups=dba,oinstall,hagsuser \
home=spcws:/home/spcws/oracle oracle
spcws# passwd oracle
Changing password for "oracle"
oracle's New password:
Re-enter oracle's new password:
spcws# dsh "/var/sysman/supper update user.admin"
```

At this stage, you should be able to log in to any OPS node as `oracle` and have access to oracle's home directory.

2.5.8 Step 12: Set AIX file creation permissions (as oracle)

Perform the following steps to set AIX file creation permissions:

1. Ensure that the umask is set to 022 for the `oracle` account. Enter the `umask` command in order to check the current setting.

```
$ umask
022
```

2. If the `umask` command does not return 022, set it in the `oracle` user profile by inserting the following line in the `.profile` file of the `oracle` account:

```
$ umask 022
```

2.5.9 Step 13: Set environment variables (as oracle)

Perform the following steps to set environment variables:

Before starting the OUI graphical interface, set the following variables by adding appropriate lines to the .profile of oracle and executing the following command:

```
$ . $HOME/.profile
```

ORACLE_BASE This is required if your system is OFA-compliant. This variable specifies the directory at the top of the Oracle software and administrative file structure. The OFA-

recommended value is software_mount_point/app/oracle. For example /u01/app/oracle.

ORACLE_HOME This specifies the directory containing the Oracle software for a particular release. The OFA-recommended value is ORACLE_BASE/product/release. For example, /u01/app/oracle/product/8.1.5.

NLS_LANG This is required if your database uses a character set other than the default US7ASCII (US 7-bit ASCII). A complete list of character sets is given in Appendix A of *Oracle8i Installation Guide for AIX-Based Systems*, A67728-01.

ORA_NLS33 This is required if you are creating a database with a storage character set other than the default US7ASCII. Set ORA_NLS33 to $ORACLE_HOME/ocommon/nls/admin/data before you start OUI or create the database.

2.6 Task C. Oracle server installation

This section describes what you need to start the Oracle Universal Installer (OUI) and install the Oracle8*i* Parallel Server (OPS) software on your OPS nodes.

2.6.1 Step 14: Mount Oracle8*i* CD-ROM (as root)

The Oracle Product CD-ROM is in RockRidge format, which is supported by AIX. Perform the following steps to mount an Oracle8i CD-ROM:

1. Make sure that the CD-ROM drive on the control workstation is *available,* that is, that the drive is defined, configured, and ready for use (sample output is shown in the following screen):

```
spcws# lsdev -Cc cdrom
cd0 Available 00-00-0S-3,0 Multimedia CD-ROM Drive
```

2. Place the CD-ROM in the CD-ROM drive of the control workstation. If you need to remove any CD-ROM, use the `umount` command before pressing the eject button on the CD-ROM drive.
3. Create a CD-ROM mount point directory (for example, `/cdrom`) on the control workstation. The `-p` flag is not needed for our example. It is only

needed when the directory you choose (instead of /cdrom) contains missing intermediate parent directories.

```
spcws# mkdir -p /cdrom
```

4. Mount the CD-ROM on the mount point directory (for example, /cdrom). Go to this directory using the cd command, and make sure you have access to the Oracle Product files and directories.

```
spcws# mount -r -v cdrfs /dev/cd0 /cdrom
spcws# cd /cdrom
spcws# ls
PW-README       index.htm       ora_kstat       .sh      stage
Translations    install         pw-syscall32    rr_moved
doc             loadext         pw-syscall3264  runInstaller
```

2.6.2 Step 15: Make CD-ROM accessible to OPS nodes (as root)

Perform the following steps to make CD-ROM accessible to OPS nodes via NFS:

1. Export the /cdrom directory with read-only permission using the command shown in the following screen:

```
spcws# mknfsexp -d /cdrom -t ro -N
```

With the -N flag, this command does not add an entry to the /etc/exports file. The exportfs command, however, is run with the correct parameters so that the directory is exported.

2. Mount the /cdrom directory of the control workstation on an empty directory (for example, /cdrom) on each and every OPS node.

```
spcws# dsh "mkdir -p /cdrom"
spcws# dsh "mount spcws:/cdrom /cdrom"
spcws# dsh "df|grep cdrom"
spn01: spcws:/cdrom    1113408         0  100%   278353   100% /cdrom
spn03: spcws:/cdrom    1113408         0  100%   278353   100% /cdrom
spn05: spcws:/cdrom    1113408         0  100%   278353   100% /cdrom
spn07: spcws:/cdrom    1113408         0  100%   278353   100% /cdrom
spn09: spcws:/cdrom    1113408         0  100%   278353   100% /cdrom
spn11: spcws:/cdrom    1113408         0  100%   278353   100% /cdrom
spn13: spcws:/cdrom    1113408         0  100%   278353   100% /cdrom
```

2.6.3 Step 16: Run the rootpre.sh script (as root)

Perform the following step to run the rootpre.sh script:

1. As oracle, shut down all running databases (if there are any).
2. As root, run the `rootpre.sh` script on all the OPS nodes.

```
spcws# dsh "cd /cdrom; ./rootpre.sh"
```

The `rootpre.sh` script basically does the following:

- It installs kernel extensions required by Oracle Server relinking.
- It verifies and sets up Asynch I/O.

2.6.4 Step 17: Set up an X-server display environment (as oracle)

You need to run the OUI graphical interface on the node, for example, spn01, where Oracle8*i* will be installed but display it on the control workstation, spcws. To accomplish this, perform the following:

From spcws, open an X-Windows session to the spn01 node, and execute the following commands:

```
spn01$ DISPLAY=spcws:0.0
spn01$ export DISPLAY
```

The DISPLAY variable is used by the X-Windows system to identify the X-Server that will be used for input and output by X applications, such as OUI.

Note that you can also use the IP address of the control workstation instead of its hostname, spcws.

To add the hostname spcws on the list of machines from which the X-Server accepts connections, run the following command from an spcws X-Windows session:

```
spcws$ xhost +spn01
```

2.6.5 Step 18: Start the Oracle Universal Installer (as oracle)

To start the OUI, perform the following steps from an spn01 X-Windows session on the control workstation spcws:

1. Go to the CD-ROM mount point `/cdrom`

```
spn01$ cd /cdrom
```

2. Start the OUI by executing the following command:

```
spn01$ ./runInstaller
```

3. Enter `y` at the *Has 'rootpre.sh' been run by root? [y/n] (n)* prompt. After some time, you should see the Welcome dialog box shown in Figure 5 on page 36. Click **Next** to begin your installation.

Figure 5. Welcome dialog box

Note

You can install Oracle8*i* by using only the Oracle Universal Installer (OUI). Installation can no longer be performed using character mode. You may, however, perform a non-interactive (or silent) install by supplying the OUI with a *response file*, that is, a text file that contains values and variables that are used by the OUI graphical interface during the installation process.

4. In the File Locations dialog box shown in Figure 6 on page 37, enter, for example, `/u01/app/oracle/product/8.1.5` for the value of your

ORACLE_HOME variable in the Destination... field, and click **Next**. You will be prompted to run /tmp/orainstRoot.sh as root before proceeding.

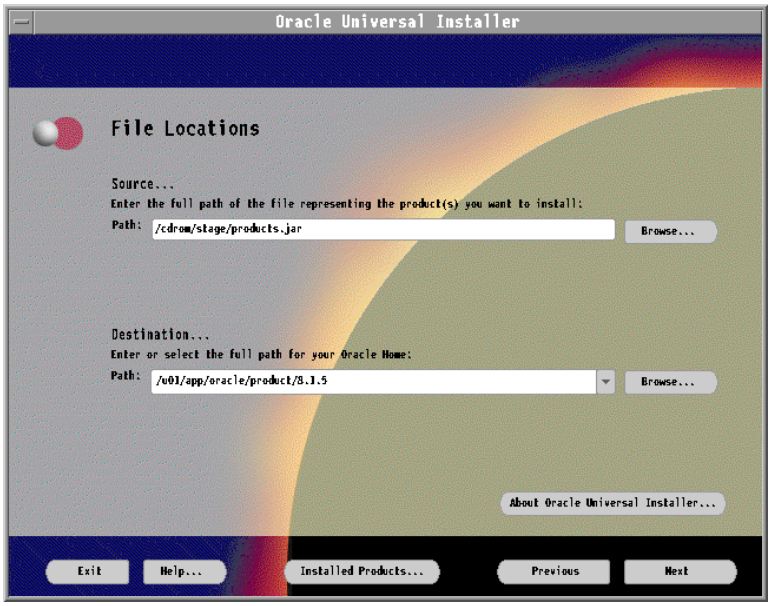


Figure 6. File locations dialog box

Figure 7 on page 38 displays the dialog box prompting you to execute orainstRoot.sh.

Figure 7. Dialog box asking you to execute orainstRoot.sh

5. In a spn01 X-Windows session, perform the following command. Sample output on a node is shown. This creates the pointer file `/etc/oraInst.loc`, which points to the location of the `oraInventory` directory as shown in the following screen:

```
spn01$ su root
root's password:
spn01$ cd /tmp
spn01$ ./orainstRoot.sh
Creating Oracle Inventory pointer file (/etc/oraInst.loc)
Changing groupname of /u01/app/oracle/product/oraInventory to dba.
spn01$ cat /etc/oraInst.loc
inventory_loc=/u01/app/oracle/product/oraInventory
```

6. In the Available Products dialog box shown in Figure 8, select **Oracle8*i* Enterprise Edition 8.1.5.0.0** for the product installation category.

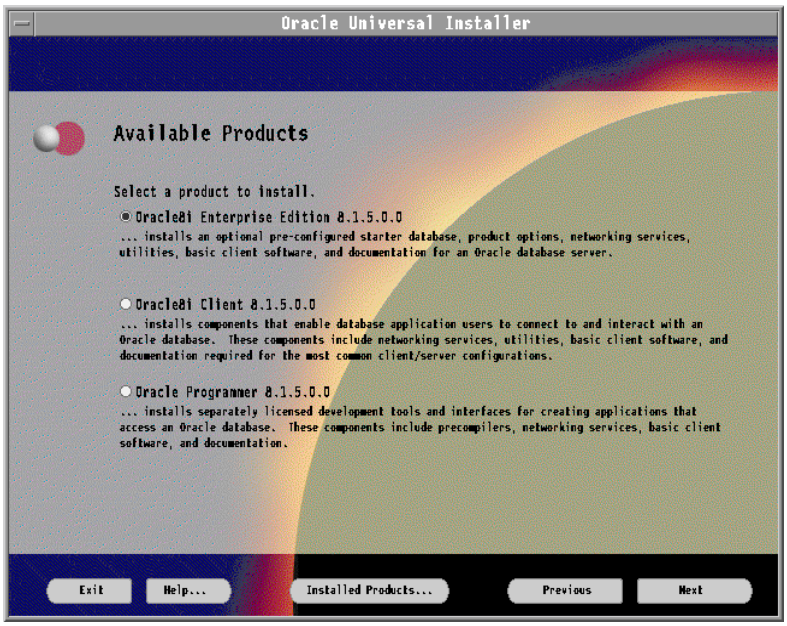


Figure 8. Available products dialog box

7. In the Installation Types dialog box shown in Figure 9 on page 40, select **Custom** and click **Next**.

.

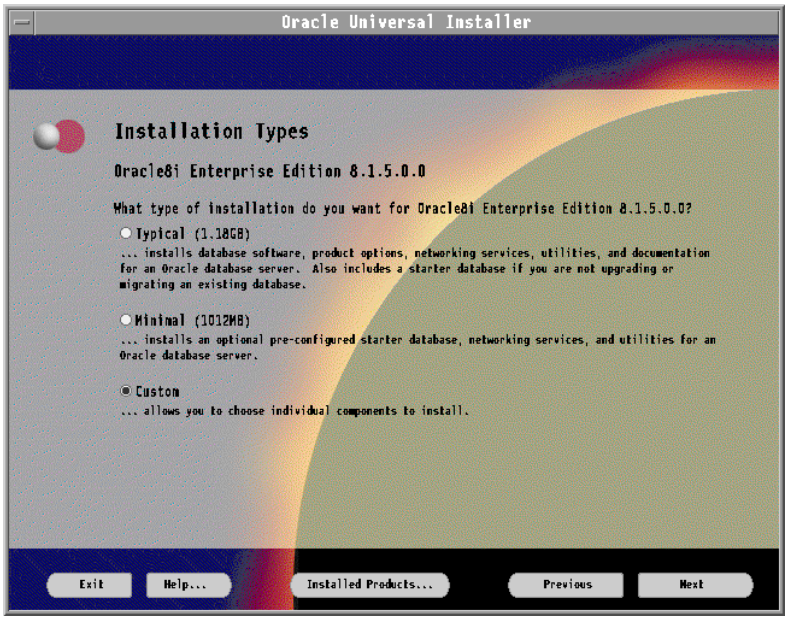


Figure 9. Installation types dialog box

For more information, see "Product Installation Categories and Installation Types" on page 3-1 of the *Oracle8i Installation Guide for AIX-Based Systems*, A67728-01.

8. In the Available Product Components dialog box shown in Figure 10, select the components that you would like to install (including Oracle Parallel Server 8.1.5), and press **Next**.

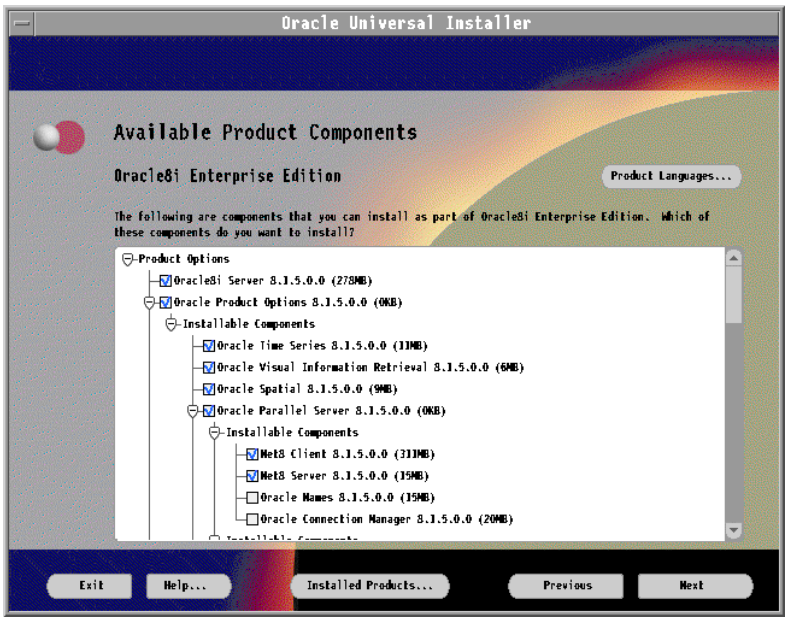


Figure 10. Available product components dialog box

9. Proceed through the remaining dialog boxes until the OUI has finished. If you need assistance at any stage of the installation, access the online help by clicking **Help**.

Note

The OUI graphical interface allows you to install on multiple nodes without needing to telnet on those nodes one by one and run the OUI installer. During the installation on one of the SP nodes, for example, spn01, you will be presented with the *Cluster Node Selection Screen* in which you can select the hostnames of other nodes on which you want Oracle installed. When you are presented with the *Authentication Methods*, make sure you select *Kerberos* if your SP is configured that way.

2.6.6 Step 19: Run the root.sh script (as root)

The OUI creates the root.sh script in the ORACLE_HOME directory. Log in as `oracle`, then cd to ORACLE_HOME and, as `root`, run the root.sh script. This needs to be done on each and every OPS node.

The root.sh script sets the necessary file permissions for Oracle8*i* products and performs other root-related configuration activities.

2.7 Task D. Post-installation tasks

This task concerns what you need to create database objects, establish the user environment, and configure the installed Oracle8*i* products. For complete details, see chapter 4, “Configuring Oracle8*i*” of the *Oracle8i Installation Guide for AIX-Based Systems*, A67728-01.

2.8 Task E. Oracle client installations

This task is necessary if you want to install Oracle client tools, applications, and client interfaces. Check the requirements and instructions in the manuals of those products.

Chapter 3. The database model used

This chapter describes the basic database model used for implementing our production database. The database we are building is a large Data Warehouse of about 1000 GB storing information for a Wholesale Supply company. Later, in Chapter 5, “Creating a small starter database” on page 69, we will describe how to create a small starter database of around 200 MB, and, in Chapter 6, “Creating a larger example database” on page 85, we will describe how to create a larger example database of about 1000 GB. Both of these databases use Oracle8*i* Parallel Server on IBM RS/600 SP.

3.1 Function of the database

The purpose of the database is a primary criteria in determining how to plan and structure it. The database we are creating combines various functions, such as decision support and transaction processing. It describes the activities of a wholesale supplier.

The database has the following characteristics:

- Large volumes of data (about 1000 GB)
- Regular volume data loads from flatfiles
- Complex queries
- Queries are of an ad hoc nature
- Long-running queries
- Low number of database updates
- It needs to be available seven days a week, 24 hours a day

The information that the wholesale supplier requires from this database includes the details of supply, demand, profit, and revenue.

3.2 Database table structure

To make full use of Oracle8*i* features, you need to understand the characteristics and structure of the database objects. For this exercise, the

table structure was already defined. Table 4 describes the tables used in our exercise.

Table 4. Table descriptions

Table Name	Primary Key	Table operations	Function	Table Type
SUPP	suppkey	insert/update/ select	Supplier details	Dimension
PARTS	partkey	insert/update/ select	Part details	Dimension
PARTSP	partkey,suppkey	insert/update/ select	Supply details for parts	Dimension
CUST	custkey	insert/update/ select	Customer details	Dimension
ORDER	orderkey	insert/update/ select	Customer order details	Fact
LINE	orderkey,lineumber	insert/update/ select	Details of items ordered	Fact
NATION	nationkey	insert/update/ select	Customer nation	Dimension
REGION	regionkey	insert/update/ select	Region within nation	Dimension

This was our basic database structure. Details of table sizes can be found in Appendix A, “Database table and index sizes” on page 119.

3.3 Data loading

Data was loaded from flat files stored on a filesystem. There would be an initial data-load of about 100 GB of raw data followed by scheduled daily loads of about 10 GB. Table 5 shows the filename formats for each loadable file. Each file was unsorted.

Table 5. Filename formats

Table Name	Filename
SUPP	supp.dat.*nnn*
PARTS	parts.dat.*nnn*
PARTSP	partsp.dat.*nnn*

Table Name	Filename
CUST	cust.da.*nnn*
ORDER	order.dat.*nnn*
LINE	line.dat.*nnn*
NATION	nation.dat.*nnn*
REGION	region.dat.*nnn*

3.4 User queries

The type of information required from this database includes the following:

- Details of revenue generated by a particular customer over a period of time
- Details of placed orders

The queries run against the database had the following characteristics:

- They were ad hoc in nature.
- They were very complex.
- They included a rich breadth of operators and selectivity constraints.
- They generated intense activity on behalf of the database server.

3.5 Why Oracle8*i* Parallel Server on RS/6000 SP?

This type of database is very suitable for OPS on RS/6000 SP. The processing power of SP systems grows almost linearly with the number of processors used; that is, the SP systems are fully scalable.

The Oracle Parallel Server is a resource-sharing system that increases availability and performance by partitioning the workload across multiple nodes.

Oracle8*i* has many new features for traditional OLTP and data warehouse applications including the following:

- Cache fusion greatly improves inter-instance communication.
- System administration is easier because of the enhancement of Oracle Parallel Server Management.
- There is a new Oracle Universal Installer.

- Oracle Database Configuration Assistant enables the creation, modification, or deletion of an Oracle database. A wizard interviews the user for information about the database. At the end of this process, a starter database can be created immediately, or an SQL script, which can be modified and used for database creation later (this is the recommended way), is generated.
- Replication is improved. Data can be replicated to servers that are closer to users and have only the data those users need, thus, providing better performance.
- There is parallel DML. Insert, Update, and Delete operations can now be run in parallel for each transaction.
- Direct Path Load API is a set of OCI interfaces that provide an application access to the direct path load engine in the Oracle server. It is now possible for an application (for example, PRO-C, PRO-COBL, and so on) to use direct and parallel load from within the application.
- Security has been improved. There is now row-level security.
- There are materialized views and summary management, which allows frequently-requested summaries to be stored and maintained. Views are maintained automatically. The Summary Advisor recommends summaries to create or delete.
- There is an improved performance optimizer.
- It has enhanced star-query processing as well as performance improvements.
- There are function-based indexes.
- It has improved partitioning.
- There are transportable tablespaces. You can copy tablespaces from one database to another. This is very useful for exact copies of data as opposed to subsets, but databases must have the following:
  - The same O/S, Oracle version.
  - The same Oracle blocksize.
  - The same Oracle character set.
- There are CUBE and ROLLUP operators.
- The SAMPLE function is useful for data mining in avoiding full table scans.
- The Database Resource Manager controls and limits the amount of processing resources available to a user or set of users.

- Operation monitoring shows, for example, how far a SELECT has progressed.

3.6 Summary

This chapter described the basic database model we used for implementing Oracle8*i* Parallel Server on RS/6000 SP. In addition, we highlighted those areas of Oracle8*i* Parallel Server to be utilized and exploited in our configuration.

In the next chapter, we will discuss the details of planning the database configuration and layout.

Chapter 4. Database planning

The crucial point to remember when creating a large database on RS/6000 SP is to *plan your database*. In this chapter, we will deal with the physical database layout for our VLDB on Oracle8*i* Parallel Server for RS/6000 SP. We will highlight key points and decision alternatives and make recommendations based on our experiences. Appendixes A-G will prove useful for performing this function.

4.1 Overview

The database we are creating is a typical data warehouse database featuring daily data loads and ad hoc queries from users.

Our schema was already defined in Chapter 3, “The database model used” on page 43. For the Database Planning phase, we performed the following steps:

1. Size the database.
2. Decide the physical layout of the database.

   This is very important for Oracle8*i* Parallel Server. The aim is to distribute I/O and CPU usage evenly across all nodes and disks. We have to decide an approach concerning:

   - The degree of parallelism
   - Striping
   - Partitioning
   - Logical volume (LV) and Virtual Shared Disk (VSD) naming schemes
3. Decide on storage and special parameter values for the database objects.
4. Decide on system parameter values.

4.2 Database sizing

When determining the sizing requirements of hardware, you need to size disk space, CPU power, memory, and network capacity.

Configuring a system is more than just working out how many nodes and how much disk space you need:

- You need a balanced system.
- You need adequate backup facilities.

- You need to plan for failures.

For this exercise, the hardware configuration was already defined in Section 2.2, “The SP system we used” on page 13.

Here, we illustrate the Disk Space Sizing performed. This is crucial in deciding:

- The physical layout of files
- The storage parameters for objects
- How to partition objects

4.2.1 Disk space

You need to size for the following:

- Software executables including RDBMS and any applications
- The Physical database including Tablespaces, Redo Log files, Rollback Segments, and Archive Log files
- The Staging Area including space for flat files waiting to be loaded
- The Export Area including space for database EXPorts and other DBA tasks

4.2.1.1 Software executables

We recommend that you:

- Allow for at least two occurrences of RDBMS executables and applications per node. This allows space for upgrades.
- Do not NFS mount the Oracle home, because the NFS filesystem will be a single point of failure.

Table 6 shows the sizes of our software executables.

Table 6. Sizes of the executables

Software	Size (MB)	Number of occurrences	Total size (MB)
Oracle8*i* Parallel Server[RDBMS]	1200	2	2400
Application	100	2	200

4.2.1.2 Physical database

We recommend that you:

- Allow space for future growth. Try to estimate your growth rate.

- Plan for mirroring of redo logs and database files.
- Provide adequate free space for temporary files (for example, log and output files.)
- Allow adequate space for the archive log directory (if archive log mode is enabled. Remember, if archive log directories fill up, all database activity halts!

Table and index sizes
We sized all our tables and indexes for the initial data-load and estimated growth rates for each. Table sizes are in Appendix A, “Database table and index sizes” on page 119, and index sizes are in Appendix A.2, “Index sizes” on page 119.

Materialized views
Materialized views are a new feature of Oracle8*i*. A materialized view provides indirect access to table data by storing the results of a query in a separate schema object. Unlike an ordinary view, which does not take up any storage space or contain any data, a materialized view contains the rows resulting from a query against one or more base tables or views. Materialized Views allow for faster data warehouse query processing.

For sizing, the schema that contains the materialized view must have a sufficient quota in the target tablespace to store the rows returned by the materialized view.

5. Rollback segments

Rollback segments contain information required for read consistency and to undo changes made by transactions that roll back or abort.

We recommend that:

- You create many rollback segments (8 or more per node)
- Each node use different (private) rollback segment

Redo logfiles
The following is required:

- Each node needs its own set of private redo logs.
- They must be raw files.

We used four groups of two logfiles per node and 128 MB per logfile

Archive logging will be enabled. We estimated the size requirement to be 10 GB.

Temporary tablespace
The temporary segment space should be large enough to hold all the data required for sorting. For our Data Warehouse, full table sorts will be required. Our initial estimate was that it must be big enough to hold the largest table (LINE).

Tablespace sizing
Tablespace sizing involves allocating enough space to store tables, indexes, and materialized views. Appendix C, “Tablespace sizes (without partitioning)” on page 123, lists the sizes for each tablespace taken as a whole (that is, without any partitioning).

Remember, when creating a datafile that will use the raw device, make sure that the datafile is smaller than the size of the raw device. The size of an Oracle database created in a raw partition must be at least two Oracle block sizes smaller than the size of the raw partition.

4.2.1.3 Staging area
We recommend that you:

- Allow plenty of space for flatfiles waiting to be loaded by Sql*Loader
- Allow plenty of space for any presorting of data

We allocated node 15 as the staging area. This will be used for receiving and sorting dataload files.

4.2.1.4 Export area
We recommend that you:

- Allow plenty of free space to allow for database reorganizations and other DBA tasks (especially EXPorts)

4.2.2 Extent sizing
An *extent* is a specific number of contiguous data blocks allocated for storage. We followed these recommendations:

- Use a uniform extent size.
- For tablespaces with one large table in them, ensure that the default extents are large enough so that there are only a few of them yet small enough to actually fit in the files that make up the tablespace without wastage.
- For tablespaces that are to have many small/medium size tables, you need to study the likely sizes of each table and set the defaults

appropriately so that it does not take more room than needed and does not consume large numbers of extents (more extents slow a database).

- When PARALLEL=TRUE is used for parallel loader, the INITIAL extent is not used. In this case, we recommend setting a small INITIAL extent (for example, 32 KB).

There are two space management parameters, called PCTFREE and PCTUSED, which need to be sized for each database object. See Appendix B, “Table physical properties” on page 121, for details about table extent sizing.

PCTFREE specifies the percentage of a data block to be reserved as free space for possible updates to existing rows in the block. PCTUSED sets the minimum percentage of a block that can be used for row data plus overhead before new rows will be added to the block.

All of our tables were used for INSERT only with minimum UPDATEs. We set the following values for PCTFREE and PCTUSED:

PCTUSED=99 and PCTFREE=0

4.3 Deciding on the physical database layout

The proper layout of datafiles is critical to database operations. This will ensure that no node or disk becomes a bottleneck.

We followed these recommendations when planning the physical layout of our database:

- Do not NFS mount the ORACLE_HOME directory.
- Have a separate ORACLE_HOME on each node. All Oracle code resides on a filesystem on each node.
- Have separate INIT.ORAs on each node.
- Use raw-devices for the creation of all database files, including all tablespaces, log files, and control files. This is a prerequisite for Oracle8*i* Parallel Server.
- Use private rollback segments (unique to a particular instance).
- Place the set of rollback segments for each instance on a different device.
- Each instance should have its own set of redo logs.
- Put redo logs on their own separate disk.
- Put archived redo logs on their own separate disk.

- Set MAXINSTANCES to a value greater than the maximum number of concurrent instances.
- Set MAXLOGFILES to the maximum number of threads (instances) possible multiplied by the maximum anticipated number of groups per thread).
- Set MAXLOGHISTORY to a large number, such as 1000.
- There will be a large number of raw devices. Datafiles of a parallel server database use raw devices; so, set MAXDATAFILES large enough to allow for the addition of files to tablespaces (it can be set to UNLIMITED in Oracle8*i*.)

4.3.1 Determining the degree of parallelism

The number of parallel execution servers associated with a single operation is known as the *degree of parallelism*. For example, If the degree of parallelism is six, six query sessions will be used automatically.

The degree of parallelism is specified at the statement level (with HINTS or the PARALLEL clause) and at the table or index level (with CREATE or ALTER statements).

The following factors affect the choice of value for DEGREE:

- The number of CPUs available
- The number of disks storing the table

For our largest tables (LINE and ORDER) we chose the number of table partitions as our degree of parallelism.

4.3.2 Striping

We examined three methods of striping:

- **Operating System Striping** - The IBM LVM (Logical Volume Manager) in AIX provides the capability to stripe data across multiple disk spindles to reduce disk contention. OS striping is usually flexible and easy to manage. Stripe size can be from 4 KB to 128 KB.
- **Hashed Shared Disks (HSD)** - The IBM HSD software is additional software that can be installed over VSD (Virtual Shared Disk) in order to provide the ability to stripe over existing VSD devices. It is useful in helping distribute I/O over several nodes. Stripe size can be from 48 KB to 1 GB.

- **Manual Striping** - This requires more DBA planning and effort to set up. For manual striping, add multiple files (each on a separate disk) to each tablespace.

Here are some key points:

- OS striping may not be suitable for a DSS database that performs many sequential operations on each disk.
- For parallel scan operations, such as full table scan, OS striping increases the number of disk seeks.
- OS striping across nodes disables disk affinity (see Section 4.3.3, “Disk affinity” on page 55).
- OS striping will impact availability unless used with hardware redundancy (for example, RAID5).
- Stripe size should be at least as large as the I/O size (DB_BLOCK_SIZE * DB_MULTIBLOCK_READ_COUNT).
- Stripe over at least as many devices as CPUs.
- Use a large stripe size (a least 64 KB).
- The only way to fully examine the performance issues with striping is to try it out with different configurations.

For our DSS database, we adopted manual striping.

4.3.3 Disk affinity

The disk affinity feature is an important shared-nothing optimization for the Oracle Parallel Server. When a query is processed in parallel with disk affinity, Oracle knows what data is on the local disks of a given node. The nodes that are local to the requested data are primarily used for query processing to increase performance.

When a Parallel Data Manipulation Language (PDML) statement is executed across an Oracle Parallel configuration in an MPP environment, the parallel slaves are allocated to the instances based on the locality of the table datafiles. Designing a database in an MPP environment that needs to support PDML must take the distribution of parallel slaves into account.

For our configuration, we localized 12 partitions per node for our largest tables (LINE and ORDER). The datafiles that make up each partition should be local to that node.

4.3.4 Database partitioning

Put simply, the objective of database partitioning is to make users on different instances access different parts of the database. This is especially important for updates. Oracle8 introduced the ability to declare a key-partitioned table (*Range Partitioning*). A key-partioned table is one where the logical table is defined as a series of columns and constraints, and then a series of physical partitions are designated to store the rows, with each partition being assigned to a particular range of values for the partition key. Oracle8*i* introduces Hash Partitioning, in which rows are placed in partitions based on hashes of partition keys, and Composite Partitioning, which involves partitioning by range *and* hash.

Partitioning addresses the key problem of supporting very large tables and indexes by allowing users to break the tables into smaller more manageable pieces called partitions.

All partitions of a table or an index have the same logical attributes, although their physical attributes can be different. For example, all partitions in a table share the same column and constraint definition, and all partitions in an index share the same index columns. However, storage specifications and other physical attributes, such as PCTFREE, PCTUSED, INITRANS, and MAXTRANS can vary for different partitions of the same table or index.

Some key points follow:

- **Range partitioning** is well suited for historical databases with time-dependent keys. Examples include the financial year for tables accessed by financial applications or the hire date for an employee table.
- **Hash partitioning** provides a very simple way of breaking data into evenly-sized containers to be spread across multiple disks or multiple nodes in an SP system. Query performance is improved by spreading I/O across multiple disks, and the performance of parallel DML may also be improved. This partitioning method is also used when you do not know how much data will map into a given range beforehand and/or the size of range partitions differs substantially. Hash partitioning eliminates reorganization due to skew in partition size.
- **Composite (range/hash) partitioning** uses hash subpartitions under range partitions to provide the ease-of-management advantages of range partitioning coupled with the data placement advantages of hash partitioning.

The user specifies ranges of values for the primary partitions of the table or index; then, it specifies a number of hash subpartitions. Data skew is unlikely

because the user can always add or drop subpartitions within a partition to maintain even distribution of each container.

Perform the following steps to implement partitioning:

1. Identify the tables that are suited for partitioning based on scalability, availability, maintenance, or performance reasons.

   Tables matching any of the following criteria may be candidates for partitioning:

   - Very large tables
   - Tables with bulk delete operations
   - Tables with time-based keys
   - Tables with frequent maintenance operations
   - Tables that are accessed for large subsets of information (all California customers, for example)

2. Define the partition method and partition keys. Have your business users identify the range of values on which the partitions should be based.
3. Define an indexing strategy. Identify indexes that need to be global, such as those that are on different columns than the main columns on which the table is partitioned; an example would be the index of Social Security numbers of an employee table partitioned on the employee ID number. You will need to decide whether to partition the index in the same way that the table is partitioned, by a different method, or not at all.

   Select the index type, based on performance and maintenance requirements, from the following options:

   - Local prefixed partitioned index
   - Local nonprefixed partitioned index
   - Global prefixed partitioned index
   - Regular B-tree index
   - Regular bitmapped index

4. Define the new physical layout. The database layout needs to include tablespace design, number, size, the placement of datafiles, and the physical characteristics of each partition. Keep the following factors in mind during this process:

   - Physically separate partitions for parallel scans and maintenance operations.

- Put partitions in different tablespaces than their writable counterparts in order to make them read-only.
- Design to allow for changes over time.
- Because partitions are different objects, they can have different physical characteristics (size, number of extents, percentage of free and used space, and so on).

5. Define a partition-maintenance strategy. Determine your system's partition-rotation requirements. Most partitioning schemes are based on time-dependent keys, and you will have to add and drop partitions routinely. The idea is to determine the most efficient way of doing rotation. First, determine which tables require the partition-maintenance routines. Then, define the partition-rotation strategy for the tables that require partitions to be added and dropped on a routine basis. You need to answer the following questions:
   - Should you split a partition or do an add and drop? Adding is preferable but can only be done at the top of the partition list. You would need to split a partition to add a partition in the middle, to divide a partition that is too large, or when there is no upper bound on the table (use MAXVALUE for the partition less-than clause). Use drop to eliminate partitions you no longer need. Dropping or splitting partitions invalidates global indexes, but adding a partition does not.
   - How can you control tablespace design and physical placement when splitting/adding partitions?

     You can give each partition its own tablespace. Although this may lead to fragmentation of free space across many datafiles, it simplifies making the tablespace read-only when the partition is read-only, and it allows non-fragmented reuse of the space when the partition is dropped.
   - How should you deal with global indexes? Anytime you drop, exchange, split, or perform another similar operation on a partition that massively changes a group of RowIDs, it will leave global indexes invalid. When a global index is invalid, full table scans will occur instead. These operations should be timed to reduce the impact of having to do full table scans while index rebuilds take place.
   - Where and how should you use the parallel operations? Parallel DML can be useful when rebuilding global indexes, merging partitions, or doing deletes without invalidating global indexes.
   - Can the tablespaces where the partitions reside be made read-only? Keeping each partition on a separate tablespace allows you to make

the tablespace read-only when no more changes will occur to that partition.

Partitioning can potentially provide many advantages:

- **Increased parallelism** - This allows you to do parallel DML operations with one process per partition.
- **Reduced backup times** - You can also separate older data from more recent data and put it into read-only tablespaces. You then need to back up this data only once after partitioning.
- **Increased availability** - If a tablespace with one partition becomes unavailable, you can, nevertheless, access the remainder of the table.
- **Increased manageability** - You can manage the table in its entirety or as individual partitions, which means that you can independently analyze, rebuild, or even back up every partition.
- **Reduced data set for queries** - Irrelevant partitions will automatically be eliminated, thus, only allowing queries access to the necessary partitions.
- **Faster elimination of old data** - If partitioning by time, you can simply truncate older data instead of going through costly deletes.

We recommend the following:

- Each partition should be spread over a number of devices on each node (see Section 4.3.3, “Disk affinity” on page 55). This avoids I/O bottlenecks when not all partitions are being scanned (for example, when some have been eliminated.)
- You should use partitioned tables when:
  - Very large tables are frequently scanned by a range predicate on a column that would make a good partitioning column.
  - New data is loaded and old data is purged periodically, and this can be utilized by an add/drop partition system.
  - Tables in which users create new data on a regular basis but rarely modify the data once it has been entered (for example, invoices, customer orders, and purchase orders).
  - Summary tables used for analysis purposes in which data is stored for a specified time period and then deleted and replaced with new summary information, such as sales figures that are stored for two years and then automatically replaced with new data.
- Maximize the use of local indexes (local means per partition) for maintainability and global indexes (global means whole table) for

performance. If global indexes are needed on a table, beware of the following:

- Global indexes will need to be set as UNUSABLE prior to a parallel insert and rebuilt following it. A parallel insert cannot be performed into a global index.
- Global unique indexes will need to be set as UNUSABLE prior to a parallel update operation and rebuilt following it. A parallel update cannot be performed on a global unique index.
- If a table partition needs to be recovered and a global index exists on the table, the global index will need to be rebuilt after the recovery.

We adopted the following for partitioning:

- Nodes 1,3,5,7,9,11, and 13 will run the database (there is one instance per node).
- Node 15 will be used as the staging area for incoming dataloads.
- Nodes 1,3,5,7,9,11, and 13 will run the application for users.
- Table LINE will be partitioned by range (using the SHIPDATE key). There will be 84 monthly logical partitions as follows (see Appendix B.3, “Table partitioning” on page 122):
- Table PARTS will be partitioned by range (using the SIZE key). There will be 50 logical partitions.
- Tables ORDER and CUST will be partitioned by hash into 84 partitions.
- To facilitate backup and recovery, each partition is stored in its own tablespace.
- To better balance I/O, each matching partition for the ORDER and LINE tables has its first datafile on a different disk. For example, LINE_P1 (1st datafile) is on a different disk than ORDER_P1.

Our goal is *no bottlenecks*!

4.3.5 Volume groups

We recommend the following:

- Put all database disks into a volume group.
- Use a naming convention (the same logical group name on each node plus the node number).
- Use Korn shell scripts for creating VGs. Write scripts for each node.

We used the following naming convention for VGs as shown in Table 7.

VG name = oradatavg<node_number>

Table 7. Volume group names

Node number	Volume Group	PP size
1	oradatavg01	8 MB
3	oradatavg03	8 MB
5	oradatavg05	8 MB
7	oradatavg07	8 MB
9	oradatavg09	8 MB
11	oradatavg11	8 MB
13	oradatavg13	8 MB
15	oradatavg15	8 MB

4.3.6 Logical volumes

We recommend the following:

- Keep tablespace names short.
- Use a naming convention (remember the Logical Volume name must have a maximum length of 15 characters).
- Use Korn shell scripts for creating LVs, and write scripts for each node.
- Make it simple (same sizes on each node).
- Create a disk map for physical volume and database map (see Appendix E, “File layout for system tablespaces” on page 127, and Appendix F, “File layout for user tablespaces” on page 129).

Here are some suggested naming conventions:

- LV = <tsname><df#>N<node_number>
- LV = <dbname><tsname><df#>N<node_number>
- LV = <tsname><df#>N<node_number>D<diskno>

where:

- dbname=database name (uppercase)
- tsname=tablespace name (uppercase)
- df#=file number (01 or 001 for 1st file)

- node_number = node number of SP node,
- diskno = disk number of the hard dsk where datafile resides.

We used the following naming convention for LVs:

- LV = <tsname><df#>N<node_number>

For example, for the first file of tablespace TEMP on node 13, LV = TEMP01N13.

4.3.7 Virtual Shared Disks (VSDs)

Virtual Shared Disks provide access to non-local disks. They behave like raw logical volumes. They can be configured using SMIT or commands. VSDs are accessed through the SP Switch; so, the network overhead is very low.

We recommend the following:

- Use a naming convention; this is the same as the Logical Volume convention, except the prefix is *vsd* (remember the VSD name has a maximum length of 31 characters).
- Use Korn shell scripts for creating VSDs, and write scripts for each node.

We used the following naming convention for VSDs:

- VSD= vsd.<tsname><df#>n<node_number>

where

- tsname=tablespace name (lowercase)
- df#=file number (01 or 001 for 1st file)

For example, for the first file of tablespace TEMP on node 13, VSD = vsd.temp01n13.

4.3.8 Finalizing the database file layout

Create a disk map for physical volume and a database map (see Appendix E, “File layout for system tablespaces” on page 127, and Appendix F, “File layout for user tablespaces” on page 129).

Appendix C, “Tablespace sizes (without partitioning)” on page 123, gives the tablespace details (before partitioning) and, most importantly, allocated disks and nodes. It provides a good overall look at how the database is spread across nodes and disks.

Appendix D, “Tablespace file layout - a summary” on page 125, gives the tablespace details (before partitioning) and, most importantly, allocated disks and nodes. It provides a good overall look at how the database is spread across nodes and disks.

Appendix E, “File layout for system tablespaces” on page 127, gives the file layout that we used for the SYSTEM database (SYSTEM tablespace, control files, redo logfiles, rollback segments, and USERS and TOOLS tablespace).

Appendix F, “File layout for user tablespaces” on page 129, gives the physical file layout for the database user tablespaces. It gives the location, LV name, and VSD name for each database file.

4.4 Special considerations for Parallel DML

Parallel DML (parallel insert, update, and delete) automatically uses parallel execution mechanisms to speed up or scale up large DML operations against large database tables and indexes. Parallel DML is useful in a decision support system (DSS) environment where performance and scalability of accessing large objects are important. Parallel DML complements parallel query by providing you with both querying and updating capabilities for your DSS databases.

In order to maximize the increase in speed gained from using the Parallel DML, you need to consider the following issues.

4.4.1 Free lists and free list groups

For each database object, such as a table, partition, or index, Oracle keeps track of blocks with space available for inserts, or for updates that may cause rows to exceed the space available in their original block. A user process that needs free space can look in the master free list of blocks that contain free space. If the master free list does not contain a block with enough space to accommodate the user process, Oracle automatically allocates a new extent.

A free list group, however, is a set of free lists that you can specify for use by one or more particular instances. Each free list group provides free data blocks to accommodate inserts or updates on tables and partitions and is associated with instances at startup.

A parallel server has multiple instances, and process free lists alone cannot solve the problem of contention. However, free list groups effectively reduce pinging between instances.

When enabled, free list groups divide the set of free lists into subsets. Descriptions of process free lists are stored in separate blocks for the different free list groups. Each free list group block points to the same free lists, except that every instance gets its own. If there are more instances than free list groups, multiple instances hash into the same free list group. This ensures that the instances do not compete for the same blocks. In OPS, always use free list groups along with process free lists.

The optimal value of FREELISTS depends on the expected number of concurrent inserts for this table (default=1).

Typically, set FREELIST GROUPS to the number of instances on the parallel server. The default is 1. For example,

```
CREATE TABLE dep
(...)
STORAGE (INITIAL 100K NEXT 100K MAXEXTENTS 1024
PCTINCREASE 0
FREELIST GROUPS 7
FREELISTS 32);
```

4.4.2 INITRANS and MAXTRANS

If you have global indexes, a global index segment and global index blocks will be shared by server processes of the parallel DML statement. Even if the operations are not performed against the same row, the server processes may share the same index blocks. Each server transaction needs one transaction entry in the index block header before it can make changes to a block.

INITRANS specifies the initial number of concurrent update transactions allocated within each data block. MAXTRANS specifies the maximum number of concurrent transactions that can update a data block.

For each global index, set INITRANS to a large value, such as the maximum degree of parallelism for this index. Leave MAXTRANS at its default value (255).

4.5 Parallel Cache Management (PCM) locking

In addition to normal transaction locking, another locking mechanism is necessary to prevent multiple instances from accessing the database and interfering with the other instances. The instances in a parallel server must

enable the users to work in a nondestructive manner. This requires cache coherency; that is, data read into the buffer cache must not be an outdated image. Oracle uses the following features to maintain cache coherency:

- **Parallel Cache Management (PCM) locks** - To prevent instances from having inconsistent information in their caches, PCM locks are used to coordinate the use of shared resources, such as tables and indexes. These PCM locks are *not* related to transaction locks. Moreover, PCM locks are owned by instances, and not by transactions. Oracle transaction locking is the same in parallel or exclusive mode.
- **Distributed Lock Manager (DLM)** - The DLM allows applications to synchronize access to resources, so that concurrent requests for the same resource are coordinated between applications using different instances in a parallel server environment.
- **Background Lock Process (LCKn)** - Background lock processes (LCKn) manage the locks used by an instance and coordinate requests for those locks by other instances. They act as the interface between an instance and the DLM.

4.5.1 Overheads for having PCM

On a parallel server, a particular data block can only be modified by a single instance at a time. If one instance modifies a data block that another instance needs, each instance's locks on the data block must be converted accordingly. The first instance must write the block to disk before the other instance can read it. The write to disk and read from disk is called a PING. Coordination is done with messages between the nodes and DLM.

In Oracle8*i* Parallel Server, a method of resolving the problem of pings has been introduced. It is called Cache Fusion. Cache Fusion provides copies of blocks directly from the holding instance's memory cache to the requesting instance's memory cache. There is a new background process called Block Server Process (BSP) to accomplish this.

BSP exists in an OPS environment. It manages outgoing messages to requesting nodes as well as the transmission of consistent read blocks from one node to another.

4.5.2 PCM initialization parameters

To optimize parallel execution on Oracle8*i* Parallel Server, you need to correctly set the GC_FILES_TO_LOCKS parameter. This parameter maps PCM locks to datafiles. This area is important for tuning and is fully explained

in Chapter 15 of *Oracle8i Parallel Server Concepts and Administration*, A67778-01. Some key points related to this parameter follow:

- It defaults to NULL.
- It must be the same on all instances.
- Always specify all datafiles in GC_FILES_TO_LOCKS.
- For datafiles containing read-only data, set the tablespace to read only; then, there will be no PCM locking at all.
- Assign many locks to files that many instances modify concurrently.
- Assign fewer locks to files that do not need to be accessed concurrently by multiple instances.
- Never allocate PCM locks for datafiles of temporary tablespaces.
- Never allocate PCM locks for datafiles that contain only rollback segments. These are protected by the GC_ROLLBACK_LOCKS parameter.
- Allocate specific PCM locks for the SYSTEM tablespace.

Other parameters to be set include:

GC_RELEASEABLE_LOCKS

GC_ROLLBACK_LOCKS

LM_LOCKS

LM_RESS

4.6 System parameters

When an instance starts up, Oracle uses the values found in an initalization parameter file to create the System Global Area (SGA) for that instance. You can use various approaches to define multiple instances:

- Use a common parameter file for multiple instances.
- Use an individual parameter file for multiple instances
- Embed a parameter file using IFILE.

We used individual parameter files for each instance and one common parameter file.

Our naming conventions were as follows:

initxxx.ora = used by all nodes (common to all nodes)

initxxxnn.ora = individual to each instance [xxx=sid, nn=node_number]

For further details of system parameters refer to the *Oracle8i Administrator's Guide*, A67772-01 and to *Oracle8i Parallel Server Concepts and Administration*, A67778-01.

An example common parameter file can be found in Appendix G, "Common init parameter file initdss.ora" on page 131.

An example individual parameter file is shown in Figure on page 96.

4.7 Summary

This chapter described the steps involved in planning the implementation of a database using Oracle8*i* Parallel Server.

In the next chapter, we will show the steps involved in creating a small starter database.

Chapter 5. Creating a small starter database

In this chapter, we describe the steps necessary to create a small starter database using the Oracle Database Configuration Assistant. This assumes that Oracle has already been installed on all OPS nodes.

5.1 Overview

Our task is to create a small starter database using Oracle8*i* Parallel Server on two RS/6000 SP nodes. The node hostnames are *v06n01* and *v06n03*. Oracle8*i* Parallel Server is already installed on both of these nodes. We performed the following steps to create our starter database:

1. Create the volume groups (as root).
2. Create the logical volumes (as root).
3. Create and activate the VSDs (as root).
    1. VSD commands authorization.
    2. Enter VSD information.
    3. Define the VSD global volume groups.
    4. Define the VSDs.
    5. Configure and activate the VSDs.
    6. Check the VSDs.
4. Create a starter database.
5. Configure OPSCTL.
6. Verify instances.

5.2 Create the volume groups (as root)

For our starter database, we created volume groups called oradatavg01 for node 1 and oradatavg03 for node 3. You can use either the SMIT interface or AIX commands on the node:

Using SMIT, perform the following steps:

TYPE `smit storage`

SELECT Logical Volume Manager

SELECT Volume Groups

SELECT Add a Volume Group

TYPE `oradatavg01` in **VOLUME GROUP name**

TYPE `8` in **Physical partition SIZE in megabytes**. Notice that, when you are working with disks that are larger than 4 GB, you must specify the physical partition size to be 8 MB or more for your volume group.

TYPE `hdisk2,hdisk3` in **PHYSICAL VOLUME names** or press F4, and select the physical volumes you want.

PRESS **Do** to create the volume group.

Using AIX commands:

The following commands will create a new volume group called oradatavag01 using hdisk2 and hdisk3 as the physical volumes. The volume group is then varied on (activated).

```
# mkvg -f -y oradatavg01 -s 8 hdisk2,hdisk3
# varyonvg oradatavg01
```

You may repeat the above for node 3 to create oradatavg03.

5.3 Create the logical volumes (as root)

For our starter database, we created the following logical volumes shown in Table 8:

Table 8. Logical volume names and sizes

LV name	Nodes	Size
DR01N1	1	90 MB
SYS101N1	1	180 MB
TMP101N1	1	100 MB
RBS101N1	1	50 MB
INDX101N1	1	50 MB
OEMREP101N1	1	80 MB
USR101N1	1	20 MB
CTL1N1	1	8 MB

LV name	Nodes	Size
CTL2N1	1	8 MB
LOG01N1	1	2 MB
LOG02N1	1	2 MB
LOG01N3	3	2 MB
LOG02N3	3	2 MB

To create the LVs, you can use either the SMIT interface or AIX commands on the node:

Using SMIT:

TYPE `smit storage`

SELECT Logical Volume Manager

SELECT Logical Volumes

SELECT Add a Logical Volume

TYPE `oradatavg01` in **VOLUME GROUP name**, or press F4, and select oradatavg01.

TYPE An LV name from Table 8 (for example, `DR01N1`).

TYPE Size of the LV (in units of PP, which is 8 MB in our example) in the **Number of LOGICAL PARTITIONS**. For example, `12` for DR01N1, which actually makes DR01N1 96 MB in size.

PRESS **Do** to create the logical volume.

Using AIX commands:

The following `mklv` command example creates the logical volume DR01N1 with size 96 MB in oradatavg01 (assuming a logical partition of 8 MB).

```
# mklv -y DR01N1 oradatavg01 12
```

Finally, give ownership of each raw logical volume to the user oracle using the command shown in the next screen. For example, for DR01N1:

```
# chown oracle:dba /dev/rDR01N1
```

You must perform the above procedure for all LVs on both node 1 and node 3.

5.4 Create and activate the VSDs (as root)

If you are new to working with the IBM Virtual Shared Disk facility, you might want to refer to *IBM PSSP: Managing Shared Disks*, SA22-7349 for a more detailed description of VSDs. This manual can also be viewed at or downloaded from the following IBM Web site:

`http://www.rs6000.ibm.com/resource/aix_resource/sp_books/pssp/`

To create and manage IBM VSDs, you can use the VSD Perspective, a graphical user interface that lets you perform all IBM VSD tasks (see *IBM PSSP: Managing Shared Disks*, SA22-7349 or the redbook *SP Perspectives: A New View of Your SP System*, SG24-5180).

Instead of the VSD perspective, you can use VSD commands or the SMIT interface as we do in this section.

5.4.1 VSD commands authorization

Before you can use any VSD commands that operate on multiple nodes, you must have the correct authorization by having Kerberos authority and sysctl authorization. This can be done by adding your principal ID to the /etc/sysctl.acl and /etc/sysctl.vsd.acl files. For user `root`, it would be similar to the following line:

```
_PRINCIPAL root.admin@MSC.ITSO.IBM.COM
```

Where `MSC.ITSO.IBM.COM` should be replaced by the domain name at your site.

You must perform these authorization steps on the control workstation and on nodes that will be IBM VSD clients or servers. In our example, they are node 1 and node 3.

Next, stop and restart the `sysctld` daemon on the control workstation and nodes 1 and 3. Then, check to make sure that VSD commands will function under sysctl control.

```
spcws# sysctl -h <cws> -h <node1> -h <node2> svcrestart
spcws# sysctl -h <cws> -h <node1> -h <node2> sysctl_vsdcheck
```

where <cws>, <node1>, and <node3> are the hostnames of the control workstations node 1 and node 3. You will get no error messages if VSD commands authorization is set up correctly.

5.4.2 Enter VSD information

Recall that, for our starter database, we designated nodes 1 and 3 as the VSD nodes. To enter this information, in addition to VSD tunables, into the SDR, you can use either SMIT or the vsdnode command.

Using SMIT:

TYPE `smit vsd_data`

SELECT VSD Node Information

TYPE `1 3` in **Nodes**, or press **F4** and select.

TYPE `css0` in **Adapter Name for VSD Communications**, or press F4 and select.

TYPE `6307` in **Initial Cache Buffer Count**.

TYPE `12614` in **Maximum Cache Buffer Count**.

TYPE `2000` in **VSD Request Count**.

TYPE `192` in **Read/Write Request Count**.

TYPE `4096` in **VSD Minimum Buddy Buffer Size**.

TYPE `262144` in **VSD Maximum Buddy Buffer Size**.

TYPE `48` in **VSD Number of Max-sized Buddy Buffers**.

TYPE `61440` in **VSD Maximum IP Message Size**.

PRESS **Do**

If you prefer, you can use the following vsdnode command to achieve the same result as the preceding steps.

```
spcws# /usr/lpp/csd/bin/vsdnode 1 3 css0 6307 12614 2000 192 4096 262144 48 61440
```

5.4.3 Define the VSD global volume groups

To define the VSD global volume groups, you can use either SMIT or the vsdvg command.

Using SMIT:

TYPE `smit vsd_data`

SELECT VSD Global Volume Group Information

TYPE `oradatavg01` in **Local Volume Group Name**.

TYPE `1` in **Primary Node on which Volume Group is Resident**.

TYPE `vsd.oradatavg01` in **Global Volume Group Name**.

PRESS **Do**

You must repeat these steps for Node 3.

If you prefer, you can use the following `vsdnode` command to achieve the same result as the preceding steps.

```
spcws# /usr/lpp/csd/bin/vsdvg -g 'vsd.oradatavg01' oradatavg01 1
spcws# /usr/lpp/csd/bin/vsdvg -g 'vsd.oradatavg01' oradatavg01 3
```

5.4.4 Define the VSDs

To designate a node as either having or using a VSD, you may use either SMIT or the `defvsd` command.

For our starter database, we created the VSDs shown in Table 9.

Table 9. VSDs created for starter database

VSD name	LV name
vsd.dr01n1	DR01N1
vsd.sys101n1	SYS101N1
vsd.tmp101n1	TMP101N1
vsd.rbs101n1	RBS101N1
vsd.indx101n1	INDX101N1
vsd.oemrep101n1	OEMREP101N1
vsd.usr101n1	USR101N1
vsd.ctl1n1	CTL1N1
vsd.ctl2n1	CTL2N1
vsd.log01n1	LOG01N1
vsd.log02n1	LOG02N1

VSD name	LV name
vsd.log01n3	LOG01N3
vsd.log02n3	LOG02N3

Using SMIT:

TYPE `smit vsd_data`

SELECT Define a Virtual Shared Disk

TYPE `DR01N1` in **Logical Volume Name**.

TYPE `vsd.oradatavg01` in **Global Volume Group Name**.

TYPE `vsd.dr01n1` in **Virtual Shared Disk Name**.

TYPE `nocache` in **Virtual Shared Disk Option**.

PRESS **Do**

If you prefer, you can use the following `defvsd` command to achieve the same result as the preceding steps.

```
/usr/lpp/csd/bin/defvsd DR01N1 vsd.oradatavg01 vsd.dr01n1 nocache
```

You must repeat the above procedure for all other VSDs shown in Table 9.

5.4.5 Configure and activate the VSDs

The VSDs must be configured, made available, and activated before they can be used. In addition, give ownership of the VSDs to the user oracle.

The following commands must be run on each and every VSD node (nodes 1 and 3 in our example). Note that the `chown` command is run for all the VSDs (Table 9 in our example).

```
# /usr/lpp/csd/bin/cfgvsd -a
# /usr/lpp/csd/bin/startvsd -a
# chown oracle:dba /dev/rvsd.dr01n1
...                          ...
...                          ...
...                          ...
# chown oracle:dba /dev/rvsd.log02n3
```

The -a flag specifies that all VSDs that have been defined are to be configured (cfgvsd command) or started (startvsd command).

You should have installed the IBM Recoverable Virtual Shared Disk (RVSD) software on each VSD node. Run the ha_vsd resett command on each VSD node to start or restart the RVSD subsystem. This is required even if you are not using any twin-tailed disks. Issue the following command from the control workstation:

```
spcws# /usr/lpp/csd/bin/ha_vsd reset
```

5.4.6 Check the VSDs

The following commands are useful for verifying the VSD configuration.

To display configured VSDs and their characteristics, run the following command on the VSD node:

```
# /usr/lpp/csd/bin/lsvsd -l
```

To display the IBM VSD device driver statistics of a node, run the following command on the VSD node:

```
# /usr/lpp/csd/bin/statvsd
```

To display the IBM VSD subsystem definition data from the SDR, issue the following commands from the control workstation:

```
spcws# /usr/lpp/csd/bin/vsdatalst -g
spcws# /usr/lpp/csd/bin/vsdatalst -n
spcws# /usr/lpp/csd/bin/vsdatalst -v
```

The -g flag displays global volume group data, -n displays VSD node data, and -v displays VSD definition data from the SDR.

- To verify an IBM VSD, you can use either the VSD Perspective GUI or the dd command to write to the VSD remotely:

```
# dd if=<remote_vsd_name> of=/dev/null count=1
```

If no error is returned, the node has proper access.

Note

The VSD verification procedure writes data to the VSD. Do not perform this after you have put real data on the VSD.

5.5 Create a starter database

We will use the Oracle Database Configuration Assistant (DBCA) to create our starter database. The database will be across two nodes: 1 and 3.

The Oracle Database Configuration Assistant (DBCA) is provided to simplify the creation of Oracle databases. It can be invoked as a stand-alone Java application. A wizard interviews the user for information relevant to the use and environment of the database. At the end of this process, a pretuned starter database can be created immediately, or an SQL script is generated to create the database at a later time.

We performed the following steps to create our starter database:

1. Log in as `oracle` on node 1.
2. Ensure that the ORACLE_HOME and ORACLE_BASE environment variables are set correctly: For example:

   ```
   ORACLE_HOME=/u01/app/oracle/product/8.1.5

   ORACLE_BASE=/u01/app/oracle
   ```

3. The DBCA needs an input ASCII file containing details of the raw VSDs. This file describes the mapping between the Oracle tablespace name and the raw device. We created the following for our two-node starter database:

dr	/dev/rvsd.dr01n1
sys1	/dev/rvsd.sys101n1
tmp1	/dev/rvsd.tmp101n1
rbs1	/dev/rvsd.rbs101n1
indx1	/dev/rvsd.indx101n1

oemrep1	/dev/rvsd.oemrep101n1
usr1	/dev/rvsd.usr101n1
ctl1	/dev/rvsd.ctl1n1
ctl2	/dev/rvsd.ctl2n1
log1_1	/dev/rvsd.log01n1
log1_2	/dev/rvsd.log02n1
log2_1	/dev/rvsd.log01n3
log2_2	/dev/rvsd.log02n3

The DBCA_RAW_CONFIG environment variable identifies the location of this ASCII file. The DBCA requires this environment variable in order to determine where to create the data files.

We set `DBCA_RAW_CONFIG=/home/users/oracle/create.raw`

4. Check that the DISPLAY environment variable is correctly set. This should be set to the machine name or IP address, X server, and screen being used by your workstation. For example:

   ```
   DISPLAY=9.12.0.5:0.0

   export DISPLAY
   ```

5. Start the DBCA by typing `dbassist`

   The Oracle Database Configuration Assistant welcome page appears as shown in Figure 11 on page 79.

Figure 11. Oracle Database Configuration Assistant welcome page

6. Select **Oracle Parallel Server Configuration**, and click **Next**. The dialog shown in Figure 12 on page 80 appears.

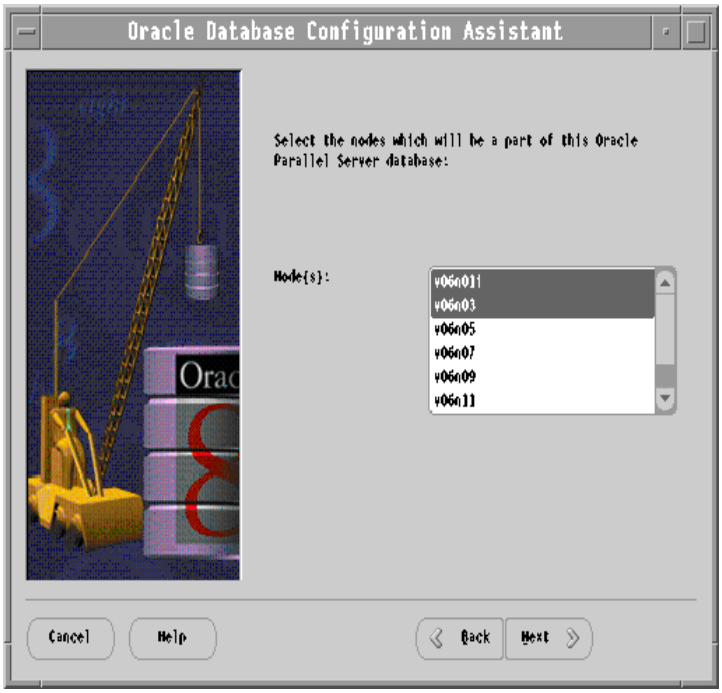


Figure 12. Dialog box asking you to select the OPS nodes

7. Select the nodes where the database is to be installed. You must always select at least two nodes including the local node.
8. Respond to instructions in each Oracle Database Configuration Assistant page. When you get to the page that requires the global database name and SID prefix, do the following:
   1. Enter `dss.ibm` as the global database name.
   2. Enter `dss` as the Oracle System Identifier (SID) prefix.
9. When you get to the last screen, click **Finish** to start the creation of the Oracle Parallel Server database. In case of failures, check the logs in the $ORACLE_HOME/install directory.

   The Database Creation Assistant created the database and the following files:

   - OFA administrative directory structure under $ORACLE_BASE/admin/dss.
   - Parameter files in $ORACLE_BASE/admin/dss/pfile.
10. Create a password file by performing the following steps:

1. Change directory to $ORACLE_HOME/dbs
2. Execute the following command as oracle:

   ```
   orapwd file=orapw<SID> password=??
   ```

   For example: `orapwd file=orapwdss1 password=something`

5.6 Configure OPSCTL

The OPSCTL utility provides a single point of administration of Oracle8*i* Parallel Server. It handles the startup and shutdown of lock manager, instances, listener, and intelligent agents. To configure opsctl, perform the following steps:

1. Verify that the $HOME/.rhosts files have been modified on all nodes, for the oracle user. For example, /home/oracle/.rhosts may contain:
   - v06n01 oracle
   - v06n03 oracle
2. Start the Oracle Parallel Server Communication Daemon (OPSD). The OPSD receives requests from OPSCTL to execute administrative job tasks, such as startup or shutdown. The command is executed locally on each node and the results are sent back locally to OPSCTL:
3. The /etc/services file must be updated on each node to add an entry that will specify the port for the TCP/IP protocol-based service of OPSD. Log in as root and add an entry similar to the following to /etc/services (the port number may vary).

   ```
   opsd 1545/tcp
   ```

   Do the same on each node running oracle. (Use the same port number on all nodes.)
4. Log in as oracle and type the following on all nodes running oracle:

   ```
   opsd log=/oracle/admin/dss/bdump/opsd.log
   ```

   The log directory may be different on your system.
5. To start the OPSD automatically when the system is rebooted (this is recommended), log in as root and add an entry similar to the following to the system startup file:

   ```
   su - oracle -c "opsd log=/oracle/admin/dss/bdump/opsd.log"
   ```
6. Create the following directories:

   $ORACLE_HOME/ops

This is the default directory in which OPSCTL expects to find the configuration files.

7. Set the environment symbol ORACLE_PSRV to the database name. For example:

   ORACLE_PSRV=dss

   OPSCTL will use this to form the name of the config file:

   ${ORACLE_HOME}/ops/${ORACLE_PSRV}.conf

   Ensure that ORACLE_HOME is set.

8. Get the node list. You must provide numbers that correspond to the node names according to what is in the SDR (System Data Repository) of your SP system. To find this information, enter the following command as `root`:

   ```
   /usr/lpp/ssp/bin/SDRGetObjects Node node_number initial_hostname
   ```

   This will produce a reply similar to the following:

   node_number initial_hostname

   1 v06n01

   3 v06n03

9. Create the config file in the $ORACLE_HOME/ops directory.

Here is the config file (called dss.conf) that we created for our 2-node starter database:

```
oracle_home=/u01/app/oracle/product/8.1.5
node_list = "1,3"
oracle_sid_list = "dss1,dss3"
inst_oracle_sid = %p%n
inst_init_ora = /u01/app/oracle/product/8.1.5/dbs/init%p%n.ora
lsnr_listener_name = %m_listener
```

10. Start OPSCTL:

    Type `opsctl start`

    This will start all the listeners (TNS) and instances on all nodes.

5.7 Verify instances

To verify that instances are running, perform the following steps:

1. On any node, enter the following:

   ```
   sql> connect internal/password
   ```

```
sql> select * from v$active_instances;
```

We received the following output:

INST_NUMBER INST_NAME

1 v06n01:dss1

3 v06n03:dss3

2. On all nodes, connect as SCOTT/TIGER, and view the EMP table:

```
sql> connect scott/tiger
sql> select * from EMP;
```

The employee table should be displayed.

5.8 Summary

This chapter described the tasks necessary to create a small starter database using the Oracle Database Configuration Assistant. In addition, we showed how to create Volume Groups (VGs), Logical Volumes (LVs), and Virtual Shared Disks (VSDs).

In the next chapter, we will show how to create a large production database of about 1000 GB.

Chapter 6. Creating a larger example database

In this chapter, we describe the steps necessary to create an example OPS database. Before proceeding, you might want to review Chapter 5, “Creating a small starter database” on page 69, to learn some of the basic steps involved in a less complex example.

The software and hardware configuration of the SP system that we used in the present chapter is described in Section 2.2, “The SP system we used” on page 13.

6.1 Configuring the SSA disks

In our SP system (see Figure 4 on page 14), each OPS node has three drawers of SSA disks with sixteen 4.5 GB disks per drawer. Therefore, there are forty-eight 4.5 GB disks, or 216 GB of external SSA disk space per OPS node, giving a grand total of more than 1.5 TB for our seven OPS nodes. In addition, the staging node (spn15) has five SSA drawers with a total of eighty 4.5 GB SSA disks totaling 360 GB.

We chose to use the mirroring capability of AIX to protect the data from disk failure. Mirroring and RAID disks are commonly used technologies for database systems. In our SP system, all external disks are singly mirrored; that is, there are two copies of each logical volume. Therefore, each OPS node has 24 mirrored pairs of external disks. In order to improve performance and availability, the two disks in each pair are on different PCI buses and different SSA adaptors. Therefore, the database is protected whether there is a disk failure, an SSA adaptor failure, or a PCI bus failure.

6.2 Space layout for the example database

In this redbook, we focus on business intelligence applications; so, the example database is a DSS database. We use a TPC-D-like database of about 100 GB in size.

In the example database, some tables are partitioned and some are not. The partitioned tables are either range-partitioned or hash-partitioned. The latter partitioning scheme is new in Oracle8*i*. For hash partitioned tables, we use one tablespace to hold all the partitions of the table. For range-partitioned tables, however, we use one tablespace for each and every partition of the table.

LINE is a table that contains all the line items. It is the largest table in the example database. It is about 78 GB and is range partitioned into 84 partitions; so, there are 84 partition tablespaces for LINE. These tablespaces are named line1, line2, and so on through line84. Each partition tablespace resides on one datafile; so, there are 84 datafiles.

ORDER is a table that has all the orders and is the second largest table. It is about 17 GB in size and is hash partitioned into 84 partitions. All 84 partitions reside in one tablespace.

PARTS is hash partitioned into 50 partitions, and all partitions reside in one tablespace.

The tables NATION and REGION are very small; so, they reside in the system tablespace. Tables SUPP, CUST, and PARTSP are non-partitioned.

6.3 Tablespaces and datafiles

We created one datafile for each partition tablespace, multiple datafiles for non-partition tablespaces, and one datafile for each hash-partitioned table.

In an SP system, each datafile is a raw logical volume device built on IBM virtual shared disk (VSD) technology. To create VSDs, you need to perform the following steps (also see Chapter 5, “Creating a small starter database” on page 69 for more details):

1. Create VGs on the available disks on each node.
2. Create LVs within the VGs you just created.
3. Enter VSD information for the nodes.
4. Define VSD Global VGs.
5. Define VSD devices.
6. Configure VSD devices.
7. Start VSDs.
8. Change VSD ownership to user oracle, group dba.
9. Verify VSD configurations.
10. Check async I/O settings.
11. Check SP Switch settings.

We wrote script files to perform the above steps. We also had the same configuration and ordering of the SSA disks on each node.

Recall from Section 2.3, “Running commands on SP nodes concurrently” on page 15, that you can use the `dsh` command to remotely execute commands on one or more SP nodes in parallel. You can either issue `dsh` from the control workstation or from an SP node. In our case, we chose the first node, v06n01, to issue the `dsh` command.

Therefore, we logged on to the node v06n01 as root and set our WCOLL environment variable as follows:

```
# export WCOLL=/.opsnodes
```

where `/.opsnodes` is a file that contains the hostnames of our seven OPS nodes.

```
# cat /.opsnodes
v06n01
v06n03
v06n05
v06n07
v06n09
v06n11
v06n13
```

We then carried out the steps listed earlier as described in the following sections.

6.3.1 Create VGs on the available disks on each node

To create VGs, issue the following:

```
dsh /absolute_path/create_vg.sh
```

where `create_vg.sh` is a file containing commands similar to:

```
# /usr/sbin/mkvg -f -y <vg_name> -s <pp_size> <list_of_disks>
# /usr/sbin/varyonvg <vg_name>
```

for each VG that needs to be created and activated.

Note

Consult with the command man pages for full details about all the commands used in this section.

In our SP system, there are 48 disks on each node. They can all be in one volume group or in several volume groups. We put them into two volume groups, generically named vg<node_number> and vg<node_number>a. The former has 32 disks and the latter has 16. Our experience shows that it takes a significantly longer time to create a volume group with more than 32 disks, although AIX allows a maximum of 96 disks per volume group.

For example, for node v06n01, the commands for vg1 (and similarly for vg1a) would be:

```
/usr/sbin/mkvg -f -y 'vg1' -s '8' hidsk4 hdisk41 hdisk40 hdisk5....

/usr/sbin/varyonvg vg1
```

Note that, in our example, the physical partition size is assumed to be 8 MB.

6.3.2 Create LVs within the VGs you just created

To create LVs within the VGs you just created, issue the following:

```
dsh /absolute_path/create_lv.sh
```

where `create_lv.sh` is a file containing commands similar to:

```
# /usr/sbin/mklv -y <lv_name> -a <position> -e <range> -c <copies> \
-w <mirror_write_consistancy> -x <maximum> <vg_name> \
<number_of_logical_partitions> <list_of_disks>
```

for each LV that needs to be created.

For example, for logical volume LOG1N1, the command would be:

```
/usr/sbin/mklv -y 'LOG1N1' -a 'e' -e 'x' -c '2' -w 'n' -x '16' vg1 16 \
hdisk42 hdisk7
```

Here, one copy of the mirrored LV will be on hdisk42, and the other copy will be on hdisk7.

6.3.3 Enter VSD information for the nodes

To enter VSD information for the nodes, issue the following:

```
/absolute_path/vsdnode.sh
```

where vsdnode.sh is a file containing a command similar to the following:

```
# /usr/lpp/csd/bin/vsdnode <list_of_node_numbers> <adapter_name> \
<init_cache_buffer_count> <max_cache_buffer_count> <vsd_request_count> \
<rw_request_count> <min_buddy_buffer_size> <max_buddy_buffer_size> \
<max_buddy_buffers> <vsd_max_ip_msg_size>
```

The preceding command enters VSD information including VSD tunable settings for all the VSD nodes. In general, vsdnode enters IBM VSD information for a node or series of nodes into the SDR.

For example:

```
/usr/lpp/csd/bin/vsdnode 1 3 5 7 9 11 13 css0 6307 12614 2000 192 \

4096 262144 48 61440
```

6.3.4 Define VSD global VGs

To define VSD global VGs, issue the following:

```
/absolute_path/vsdvg.sh
```

where vsdvg.sh is a file containing commands similar to:

```
# /usr/lpp/csd/bin/vsdvg -g <global_group_name> <local_group_name> <node_number>
```

for each VG that you want to be used for VSDs.

For example, the command

```
/user/lpp/csd/bin/vsdvg -g 'vsdvg1' vg1 1
```

adds SDR information indicating that the volume group known as vg1 on node 1 is available for global access and is known to the VSD nodes as vsdvg1. Here, node 1 is the primary and only server.

6.3.5 Define VSD devices

To define VSD devices, issue the following:

```
dsh /absolute_path/defvsd.sh
```

where defvsd.sh is a file containing commands similar to the following:

```
# /usr/lpp/csd/bin/defvsd <logical_volume_name> <global_group_name> \
<vsd_name> [nocache |cache]
```

for each node that you want to designate as either having or using a VSD.

For example, the command

```
/usr/lpp/csd/bin/defvsd LINE11N1 vsdvg1 vsd.line11n1 nocache
```

adds SDR information indicating that, on the globally-accessible volume group vsdvg1, the logical volume known as LINE11N1 is used as a nocached VSD named vsd.line11n1.

Note

IBM recommends that you not use VSD cache buffers (that is, use the nocache option) under normal situations. VSD LRU (least recently used) cache buffers use pinned kernel memory, which can be put to more effective use. Also, when a cache buffer is enabled, every physical read incurs the overhead of searching the cache blocks for overlapping pages and copying data in and out of the cache buffers.
See `http://www.rs6000.ibm.com/support/sp/perf/` and `http://www.rs6000.ibm.com/resource/technology/#sp` for further information about tuning the SP and VSDs.

6.3.6 Configure VSD devices

To configure a VSD device, issue the following:

```
dsh /absolute_path/cfgvsd.sh
```

where `cfgvsd.sh` is a file containing commands similar to:

```
# /usr/lpp/csd/bin/cfgvsd <list_of_vsd_names>
```

to configure the already defined VSDs and bring them to the *stopped* state. See *IBM PSSP: Managing Shared Disks*, SA22-7349 for details.

For example, the command

```
/usr/lpp/csd/bin/cfgvsd vsd.line11n1
```

brings the virtual shared disk vsd.line11n1 from the *defined* state to the stopped state.

6.3.7 Start VSDs

To start a VSD, issue the following:

```
dsh /absolute_path/startvsd.sh
```

where `startvsd.sh` is a file containing a command similar to the following.

```
# /usr/lpp/csd/bin/startvsd <list_of_vsd_names>
```

The preceding command starts the already-configured VSDs and brings them to the *active* state (see *IBM PSSP: Managing Shared Disks*, SA22-7349, for details); that is, it makes them available for use.

For example, the command

```
/usr/lpp/csd/bin/startvsd vsd.line11n1
```

makes the virtual shared disk, `vsd.line11n1`, available and activates it.

6.3.8 Change VSD ownership to user oracle and group dba

To change VSD ownership, issue the following:

```
dsh /absolute path/chown_lv.sh
```

where `chown_lv.sh` is a file containing commands similar to:

```
# /usr/bin/chown <owner>:<group> /dev/r<vsd_name>
```

The preceding script changes the ownership of the VSD raw devices to the Oracle software owner (see Section 2.5.7, “Step 11: Create an AIX Oracle software owner account (as root)” on page 28). For example:

```
/usr/bin/chown oracle:dba /dev/rvsd.line11n1
```

6.3.9 Verify VSD configurations

Follow the procedure discussed in Section 5.4.6, “Check the VSDs” on page 76.

After you complete the previous steps, datafiles are ready for you to use. However, before proceeding to create databases, you need to check the async I/O and SP Switch settings.

6.3.10 Check async I/O settings

OPS on SP uses VSD raw devices plus asynchronous I/O (aio) instead of filesystems to store data. This results in better I/O throughput when dealing with large logical volumes.

A rule of thumb for setting the MAXIMUM number of aio servers (kernel processes dedicated to asynchronous I/O processing) is to use a number that is ten times the number of disks that are to use aio concurrently. Furthermore, set the MINIMUM number of aio servers to half of this maximum number.

In our case, we have 48 disks per node; so, the MAXIMUM is set to 480, and the MINIMUM is set to 240.

After the OPS installation, you can take statistics twice using vmstat -s before any high I/O activity begins and again at the end. Check the field iodones in the output. In this way, you can determine how many physical I/Os are being handled in a given wall clock period. Then, increase the MAXIMUM number of servers, and see if you can get a larger value for iodones in the same time period.

You can change attributes relating to asynchronous I/O using the `chdev` command or SMIT. Likewise, you can use SMIT to configure and remove (unconfigure) asynchronous I/O. Alternatively, you can use the `mkdev` and `rmdev` commands to configure and remove asynchronous I/O. To start SMIT at the main menu for asynchronous I/O, enter `smit aio`.

6.3.11 Check SP Switch settings

For the SP Switch, use the following `lsattr` command to view the current settings for the SP Switch pools, spoolsize, and rpoolsize:

lsattr -El css0

```
# lsattr -El css0
bus_mem_addr   0x04000000 Bus memory address        False
int_level      0xb        Bus interrupt level       False
int_priority   3          Interrupt priority        False
dma_lvl        9          DMA arbitration level     False
spoolsize      524288     Size of IP send buffer    True
rpoolsize      524288     Size of IP receive buffer True
adapter_status css_ready  Configuration status      False
```

For OPS on SP the two SP Switch pool sizes (spoolsize and rpoolsize) are recommended to be about 16 MB each.

Use the `chgcss` command to apply configuration changes to the SP Switch adapter in the ODM database. The command is located in the `/usr/lpp/ssp/css` directory. Configuration changes are later applied to the device when it is configured during the system reboot. You must have root privileges to run this command.

```
# chgcss -l css0 -a rpoolsize=16777216 -a spoolsize=16777216
```

6.4 Preparing to create the example database

At this stage, you have created all the datafiles you need for the example OPS database. Before you create the database, however, you need to carry out the tasks outlined in the following sections.

6.4.1 .profile for AIX user oracle

In Section 2.5.7, “Step 11: Create an AIX Oracle software owner account (as root)” on page 28, we created the AIX user account `oracle`. Log in to the control workstation, and modify the $HOME/.profile file of `oracle`, and set the

necessary Oracle environment variables. In Figure 13, parts of a typical `.profile` for `oracle` are shown:

Figure 13. Part of an example .profile

```
v06n01$ id
uid=221(oracle) gid=202(dba) groups=203(oinstall),204(hagsuser)
v06n01$cd
cat .profile
PATH=/usr/bin:/etc:/usr/sbin:/usr/ucb:$HOME/bin:/usr/bin/X11:/sbin:.
export PATH
...     ...
...     ...
# -- Reminders:
# (1) Update <oracle_version>
# (2) Update <absolute_path_to_lkmgr_conf>
# (3) Update <absolute_path_to_oracle_base>
# (4) Update <absolute_path_to_oracle_paths>: SQL path
nodenum=$(/usr/lpp/ssp/install/bin/node_number)

export ENV=$HOME/.kshrc
export EDITOR=vi

export ORACLE_HOME=/u01/app/oracle/product/8.1.5
export ORACLE_SID = dss$nodenum
export ORACLE_PSRV=dss
export ORATAB=/dev/oratab

ORACLE_TERM=lft
export WCOLL=/.opsnodes

export PATH=$PATH:/usr/bin:/usr/lpp/ssp/rcmd/bin:/usr/etc:/usr/local/bin:\
$ORACLE_HOME/bin:/usr/lpp/csd/bin:/usr/lpp/ssp/bin

alias dbs="cd $ORACLE_HOME/dbs"
alias dump="cd $ORACLE_HOME/rdbms/log"
alias netadm="cd $ORACLE_HOME/network/admin"
alias rdbadm="cd $ORACLE_HOME/rdbms/admin"
alias oh="cd $ORACLE_HOME"
...     ...
...     ...
```

6.4.2 Creating the initialization files

An initialization file is an ASCII text file containing a list of parameters. Each node has its own initialization file (with parameters unique for an instance) named init<sid>.ora (in our example, initdss1.ora, initdss3.ora, and so on through initdss13.ora) in the $ORACLE_HOME/dbs directory. Additionally, there is an initialization file named init<db_name>.ora (in our example, it is initdss.ora) with common parameters shared across the nodes (see Figure 14 on page 95).

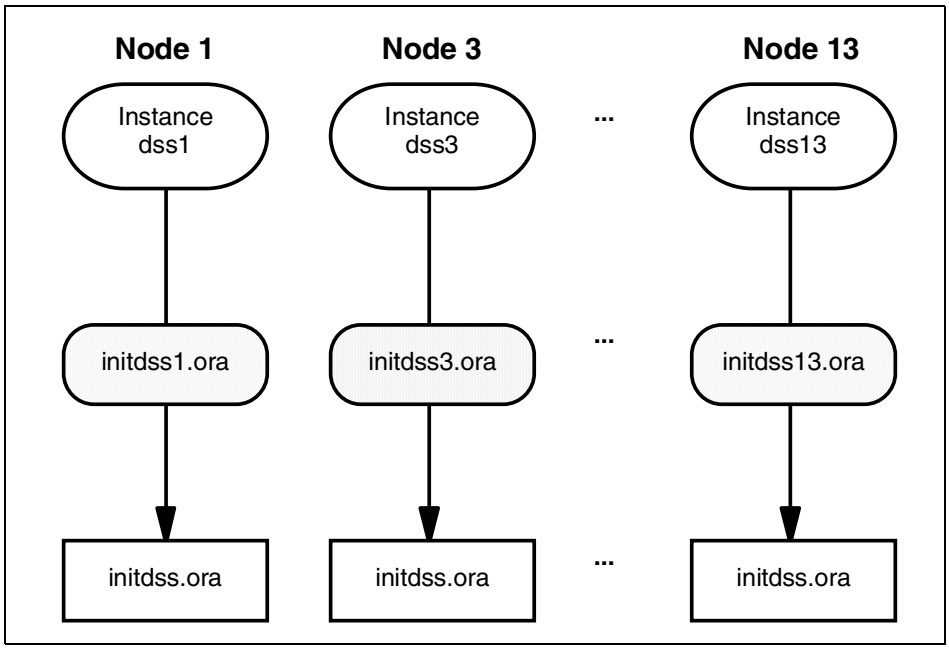


Figure 14. Instance and common initialization files

The init<sid>.ora files point to the init<db_name>.ora file for common parameters and define the following for each instance:

- A unique instance name
- A unique thread number
- Private rollback segments
- Execution of a database as an OPS rather than a single instance

The SID is the value of the DB_NAME parameter in the INIT<DB_NAME>.ORA file and the instance number. For example, if the DB_NAME is dss and the first node has an instance number of 1, its SID is dss1; the third node uses the SID dss3, and so on.

As an example, Figure 15 on page 96 shows the contents of the initialization file initdss1.ora (for node 1 in our example).

```
#***********************************************************************
# instance 1 for dss
#***********************************************************************
#
# -- common init parameter file (initdss.ora).
#


# -- include common database configuration parameters ..

ifile            =  /home/serv6/itso/db/dbs/initdss.ora


# -- configuration parameters for this instance ..


rollback_segments    = (rb1_rbs1,rb1_rbs2,rb1_rbs3,rb1_rbs4
                       )


thread               = 1
instance_number      = 1



# -- instance groups related parameters ..

#instance_groups       = group_a
#parallel_instance_group     = group_a
```

Figure 15. initdss1.ora

The INIT<DB_NAME>.ORA database initialization file is called by the individual instance initialization files through the IFILE parameter.

For OPS, some initialization parameters must have the same value for all instances and some different values. This is accomplished by referencing the common parameter file using IFILE within the individual instance parameter files.

As an example, Appendix G, “Common init parameter file initdss.ora” on page 131, shows the contents of the common initialization file initdss.ora.

For more detailed information and the meaning of initialization parameters, see *Oracle8i Parallel Server, Setup and Configuration Guide*, A67439-01.

6.4.3 Creating an Oracle user to create the database

You need to create an Oracle user account that has all the DBA privileges in order to create, start up, shut down, and connect as internal to the database.

You can use an SQL script similar to the following to create such an Oracle user.

```
rem
rem ************************************************************************
rem
rem Create a DBA with its name the same as the database name.
rem
rem Note:
rem (1) If this script is executed under svrmgrl, delete 'set time on' and 'set time of
rem     These commands work under sqlplus only.
rem (2) sys or system privilege is required.
rem
rem ************************************************************************
rem

set time on
set timing on
spool create_dss_acct.log

CREATE USER dss IDENTIFIED BY dss
   DEFAULT TABLESPACE usr
   TEMPORARY TABLESPACE usr;

GRANT connect,resource,dba TO dss WITH admin OPTION;

spool off
set timing off
set time off

connect system/manager
@?/sqlplus/admin/pupbld;

connect dss/dss
@?/rdbms/admin/catdbsyn;
@?/sqlplus/admin/pupbld;
@?/rdbms/admin/utlxplan;

connect sys/sys
set echo off
```

From the preceding screen, notice that the user `dss` has the following characteristics:

- Default tablespace usr
- Temporary tablespace usr

Also, notice that, after the DBA user `dss` is created, you need to connect to system/manager to run the Oracle script $ORACLE_HOME/sqlplus/admin/pupbld. In addition, you should connect as Oracle user dss to run the Oracle script files $ORACLE_HOME/rdbms/admin/catdbsyn, $ORACLE_HOME/sqlplus/admin/pupbld, and $ORACLE_HOME/rdbms/admin/utlxplan.

6.5 Creating the multi-instance example database

To create an OPS multi-instance database, first, create a standard single-instance database with parallel server disabled, and then extend it to all instances.

6.5.1 Setting initialization parameters

Certain initialization parameters that are critical at database creation or that affect certain database operations must have the same value for every instance in OPS. Be sure that these are identical across all instances before creating a database for your multi-instance environment.

For the single-instance database, you need a generic initialization file, such as initdss0.ora, that does not associate with any instance ID. This file has only one line, which specifies the ifile parameter pointing to the common initialization file, initdss.ora, which we discussed in Section 6.4.2, “Creating the initialization files” on page 94.

6.5.2 Setting CREATE DATABASE options

The following CREATE DATABASE options are specific to OPS, and the values given here are for our specific example database.

- **MAXINSTANCES 14** - This limits the number of instances that can access a database concurrently. Usually, this option is set to a value greater than the maximum number of instances you expect to run concurrently. This way, if instance A fails and is being recovered by instance B, you will be able to start instance C before instance A is fully recovered.
- **MAXLOGFILES 255** - This specifies the maximum number of redo log groups that can be created for the database. Usually, this option is set to the maximum number of threads possible multiplied by the maximum anticipated number of groups per thread.
- **MAXDATAFILES 1022** - This is a generic option. OPS tends to have more data files and log files than standard systems. On your platform, the default value of this option may be too low.

6.5.3 Creating and starting up the database

The CREATE DATABASE statement mounts and opens the newly-created database, leaving the parallel server disabled. You must close and dismount the database, then remount it with parallel server enabled.

Perform the following steps to create and start up the database:

1. Start Server Manager.
2. Connect with SYSDBA privileges.
3. Start up an instance with the NOMOUNT option.
4. Issue the CREATE DATABASE statement.
5. Create all rollback tablespaces and threads on each node.
6. Issue the ALTER DATABASE statement to add private logfiles for each node and enable them.
7. Close and dismount the database by issuing SHUTDOWN.
8. Update the initialization files to be sure they point to the proper rollback segments and threads.
9. Make sure parallel server is enabled.
10. Remount the database by issuing STARTUP

6.6 Starting and stopping Oracle instances in SHARED mode

You can start all Oracle instances in SHARED mode by using the OPS management utility, OPSCTL. Alternatively, you can use rsh for each OPS node to start or stop instances.

On AIX, the OPSCTL utility uses a manually-created file named <db_name>.conf to define Oracle Parallel Server instances and related services.

Note

The <db_name>.conf file must reside in the Oracle Parallel Server's shared or non-shared Oracle home location, even if this file is empty. OPSCTL requires this file to run properly.

Here is an example of a <db_name>.conf configuration file we used in our example:

```
node_list = “1,3,5,7,9,11,13"
inst_oracle_sid="%p%n"
lsnr_listener_name="%p%n"
inst_shutdown_mode="immediate"
```

Here, %p expands to the name of the Oracle Parallel Server, which is set by the value of the ORACLE_PSRV parameter or the environment variable

ORACLE_PSRV. The designation, %n, expands to the node number for the appropriate node.

When you are ready to start up all instances of your database, issue the `opsctl` command from the first OPS node as an Oracle DBA:

```
opsctl -s start inst
```

To stop all the instances, you need to issue the `opsctl` command with the stop option instead of start:

```
opsctl -s stop inst
```

You can also start up all instances of your database sequentially. The following is an example of a Korn shell script that may be used as the user oracle.

```
#!/bin/ksh
#
# -- Reminder:
#
# (1) Provide <absolute_path_to_dbs> directory below
#
#
DirDbs=/u01/app/oracle/product/8.1.5/dbs
#
#
rsh v06n01 -n ". ./.profile; cd $DirDbs; \
/home/serv6/itso/db/admin/oracle/startup_oracle8"&
rsh v06n03 -n ". ./.profile; cd $DirDbs; \
/home/serv6/itso/db/admin/oracle/startup_oracle8"&
rsh v06n05 -n ". ./.profile; cd $DirDbs; \
/home/serv6/itso/db/admin/oracle/startup_oracle8"&
rsh v06n07 -n ". ./.profile; cd $DirDbs; \
/home/serv6/itso/db/admin/oracle/startup_oracle8"&
rsh v06n09 -n ". ./.profile; cd $DirDbs; \
/home/serv6/itso/db/admin/oracle/startup_oracle8"&
rsh v06n11 -n ". ./.profile; cd $DirDbs; \
/home/serv6/itso/db/admin/oracle/startup_oracle8"&
rsh v06n13 -n ". ./.profile; cd $DirDbs; \
/home/serv6/itso/db/admin/oracle/startup_oracle8"&
#
```

Here, the startup script, `startup_oracle8`, contains the following lines:

```
svrmgrl << !
    connect internal
    startup
    exit
!
```

The `rsh` command executes one or more commands at a remote host. It sends standard input from the local command line to the remote command and receives standard output and standard error from the remote command. The -n flag, however, specifies that the `rsh` command should not read from standard input.

To have shell metacharacters interpreted on the remote host, place the metacharacters between " " (double quotes); otherwise, the metacharacters are interpreted by the local shell. Each command is separated from the next command by a ; (semicolon) .

The shutdown script, shutdown_oracle8, is similar to startup_oracle8, except the `startup` command is replaced by the `shutdown` command:

```
svrmgrl << !
    connect internal
    shutdown
    exit
!
```

6.7 Creating the tablespaces

In order for GC_FILES_TO_LOCKS (specified in the common init parameter, file initdss.ora) to work properly, you need to create all the tablespaces for your database. This includes tablespaces for all your tables, indexes, and materialized views, as well as temporary tablespaces. These tablespaces reside in the datafiles you created as described earlier in this chapter.

The following is an example DDL for creating tablespace line1 for the first partition of the table LINE. In this example, there is only one datafile for the tablespace.

```
CREATE TABLESPACE line1
        DATAFILE '/dev/rvsd.line11n1' SIZE 982992K REUSE
        DEFAULT STORAGE (
                                INITIAL     97600K
                                NEXT        97600K
                                PCTINCREASE  0
                                MINEXTENTS   1
                                MAXEXTENTS   unlimited);
```

Here, tablespace line1 is created using the datafile, /dev/rvsd.line11n1, which is 982992 KB in size. The storage clause for the tablespace is with the initial extent of 97600 KB and the next extent of 97600 KB. The minimum number of extents is one, and the maximum number of extents is unlimited. The percentage of increase is zero.

The temporary tablespace, tmp, is a large tablespace consisting of many datafiles. It is first created for a datafile; then, it is altered to add the other datafiles one at a time.

```
CREATE TABLESPACE tmp TEMPORARY
        DATAFILE '/dev/rvsd.tmp1n1' SIZE 1245136K REUSE
        DEFAULT STORAGE (
                                INITIAL     30400K
                                NEXT        30400K
                                PCTINCREASE  0
                                MINEXTENTS   1
                                MAXEXTENTS   unlimited);
```

Next, issue:

```
ALTER TABLESPACE tmp ADD DATAFILE
'/dev/rvsd.tmp2n3' SIZE 1245136K REUSE;
```

Datafiles for the tmp tablespace reside on different nodes using a round-robin algorithm. For our example database, the first datafile, rvsd.tmp1n1, is stored on node 1; the second datafile, rvsd.tmp2n3, is stored on node 3; the third datafile, rvsd.tmp3n5, is stored on node 5; the fourth datafile, rvsd.tmp4n7, is stored on node 7; the fifth datafile, rvsd.tmp5n9, is stored on node 9; the sixth datafile, rvsd.tmp6n11, is stored on node 11; the seventh datafile, rvsd.tmp7n13, is stored on node 13; the eighth datafile, rvsd.tmp8n1, is stored on node 1, and so forth.

6.8 Creating the tables

This section discusses the steps necessary to create the needed tables in our example, namely, the tables LINE, ORDER, PARTS, PARTSP, CUST, SUPP, REGION, and NATION.

6.8.1 Creating LINE

LINE is the largest fact table in our example database. It contains roughly 78 percent of the data. LINE is range partitioned by shipdate into 84 monthly partitions. Each node has twelve partitions; each partition resides on one tablespace, and each tablespace uses one datafile. See Appendix H.1, “Example script to create table LINE” on page 137, for an example script to create LINE.

6.8.2 Creating ORDER

ORDER is the second largest table in the example database. It is about 15 percent of the total data in the database. We partition the ORDER table using hash partitioning by o_orderkey. The tablespace for ORDER spans twelve disks on each node. ORDER is partitioned into 84 partitions to match LINE. The following is an example DDL to create ORDER:

```
create table orders(
    o_orderdate         date ,
    o_orderkey          number NOT NULL,
    o_custkey           number NOT NULL,
    o_orderpriority     varchar(15) ,
    o_shippriority      number ,
    o_clerk             varchar(15) ,
    o_orderstatus       char(1) ,
    o_totalprice        number ,
    o_comment           varchar(79))
  pctfree 1
  pctused 99
  tablespace tocil
  storage (initial 32k next 10m maxextents unlimited freelists 32 freelist
  groups 7
  pctincrease 0)
  parallel (degree 8 instances 7)
  nologging
  partition by hash (o_orderkey)
  partitions 84;
```

6.8.3 Creating PARTS

PARTS is a dimension table and is not large. There are queries needing PARTS to join with LINE and ORDER. We range partition PARTS by p_size into 50 partitions. All partitions reside in one tablespace, thus, in the partition clause, you do not need to specify a tablespace for each partition. The following is an example script that can be used to create PARTS:

```
create table parts(
    p_partkey          number NOT NULL,
    p_type             varchar(25),
    p_size             number,
    p_brand            varchar(10),
    p_name             varchar(55),
    p_container        varchar(10),
    p_mfgr             varchar(25),
    p_retailprice      number,
    p_comment          varchar(23))
pctfree 0
pctused 99
tablespace supprt
storage (initial 32k next 7m maxextents unlimited pctincrease 0)
parallel
nologging
partition by range (p_size)
(
partition part1 values less than (2),
partition part2 values less than (3),
...                              ...
...                              ...
partition part48 values less than (49),
partition part49 values less than (50),
partition part50 values less than (MAXVALUE));
```

6.8.4 Creating PARTSP

PARTSP is a dimension table containing supply detail information for PARTS. We do not partition the PARTSP table. It resides on one tablespace, which spans a few disks on all nodes. The following is an example script that can create PARTSP:

```
create table partsupp(
    ps_partkey      number NOT NULL,
    ps_suppkey      number NOT NULL,
    ps_supplycost   number NOT NULL,
    ps_availqty     number,
    ps_comment      varchar(199))
pctfree 0
pctused 99
tablespace prtsup
storage (initial 32K next 10M maxextents unlimited pctincrease 0)
parallel
nologging;
```

6.8.5 Creating CUST

CUST is a dimension table containing detailed customer information. We hash partition CUST into 84 partitions using the customer key. Here is an example script that can create CUST:

```
create table customer(
    c_custkey       number NOT NULL,
    c_mktsegment    varchar(10),
    c_nationkey     number,
    c_name          varchar(25) ,
    c_address       varchar(40) ,
    c_phone         varchar(15),
    c_acctbal       number ,
    c_comment       varchar(117))
pctfree 1
pctused 99
```

```
tablespace tocil
storage (initial 32k next 6m maxextents unlimited freelists 32 freelist
groups 7
pctincrease 0)
parallel (degree 8 instances 7)
nologging
partition by hash (c_custkey)
partitions 84;
```

6.8.6 Creating SUPP, REGION, and NATION

SUPP is a dimension table containing supplier detail information. REGION and NATION are demographic dimension tables. These tables are very small and need not be partitioned. Example scripts that may be used to create these tables are given in Appendix H.2, “Example script to create SUPP” on page 138, Appendix H.3, “Example script to create REGION” on page 139, and Appendix H.4, “Example script to create NATION” on page 139.

6.9 Loading the data

You have several options when it comes to loading data into your OPS database on an SP system.

One option is to sort the data by a partition key, provided the table is range partitioned and each partition has its own tablespace containing one datafile on a particular OPS node. Load the sorted data into partitions in parallel. This is how we loaded the LINE table.

If you cannot presort your data (for example, if you have hash partitioned the table), split the data into a small number (one or two for our example) of files per node. Then, transfer these datafiles to each node in parallel using, for example, FTP over the SP Switch. Alternatively, to load the data, you may NFS mount (over the SP Switch if possible) the file system containing the datafiles from the data staging node (see Figure 4 on page 14) to each OPS node. The latter method should be used as a last resort since it will be a relatively slow process.

Obviously, there are other ways to load data, for example, from tapes, networks, and so on.

In a data warehousing environment, the amount of data to load is usually large. We recommend setting up a data staging node similar to the system we used (see Figure 4 on page 14). As an added advantage, the data staging node can also be used for other things, such as an ADSM server for backup and recovery.

6.9.1 Loading LINE

We sort LINE data in our staging node and split the sorted data into 84 files. We then FTP these files to the OPS nodes using the SP switch. We use SQLLDR to direct the data into partitions in parallel. Each node is responsible for loading twelve partitions. This is a good way of taking advantage of the fact that each partition has its own separate tablespace and datafile.

We use a ksh script similar to the following for loading the data into partitions of LINE:

```
#!/bin/ksh
cd /stage
for file in $(ls line*)
do
   num=${file##line}
   print file=$file
   print num=$num
   /u01/app/oracle/product/8.1.5/bin/sqlldr USERID=dss/dss CONTROL=/
   home/serv6/itso/db/jddl/line/ctl/line$num.ctl DATA=/stage/line.$num
   DIRECT=TRUE PARALLEL=TRUE  2>&1 LOG=/home/serv6/itso/db/
   jddl/line/log/log.line.$num  BAD=/home/serv6/itso/db/jddl/line/bad/
   bad.line.$num &
done
```

Here, the files reside on the filesystem /stage in each node. The control file (see the following paragraph) for loading each partition is in the directory /home/serv6/itso/jddl/line/ctl/line$num.ctl. The logfile will be stored in the directory, /home/serv6/itso/jddl/line/log/, and the bad file is in the directory /home/serv6/itso/jddl/line/bad. The variable $num is the partition number. This script is used to load all 12 partitions on each node at the same time.

As an example, the control file for loading the first partition of LINE is similar to the following:

```
unrecoverable
```

```
load
-- Here is where INFILE should go.
append
into table line
partition (ITEM1)
-- Here is where PARTITION is specified.
append
fields terminated by '|'
(
    l_orderkey          integer external,
    l_partkey           integer external,
    l_suppkey           integer external,
    l_linenumber        integer external,
    l_quantity          integer external,
    l_extendedprice     decimal external,
    l_discount          decimal external,
    l_tax               decimal external,
    l_returnflag        char(1),
    l_linestatus        char(1),
    l_shipdatedate      'yyyy-mm-dd',
    l_commitdate        date 'yyyy-mm-dd',
    l_receiptdate       date 'yyyy-mm-dd',
    l_shipinstruct      char(25),
    l_shipmode          char(10),
    l_comment           char(44))
```

6.9.2 Loading ORDER, CUST, SUPP, PARTS, and PARTSP

We partition ORDER using the hash partitioning function. We cannot presort the data since we do not know the exact hash algorithm used in Oracle8*i* Parallel Server. We split the ORDER data into fourteen pieces, p1 through p14, and transfered two pieces to each of the seven OPS nodes, for example, p1 and p8 to node 1, p2 and p9 to node 2, and so on. The transfer of p1, p2, and so on through p7 is done in parallel, and so is the transfer of p8, p9, and

so on through p14. This is a slower loading process than the one used for LINE (see Section 6.9.1, “Loading LINE” on page 107) because each node loads data into all 84 partitions at the same time.

The loading script is similar to that of LINE:

```
#!/bin/ksh

hostname=$(hostname -s)

/u01/app/oracle/product/8.1.5/bin/sqlldr USERID=dss/dss CONTROL=/
home/serv6/itso/db/jddl/order/order.ctl DATA=/stage/order.1
DIRECT=TRUE PARALLEL=TRUE  2>&1 LOG=/home/serv6/itso/db/jddl/
order/order_${hostname}.log  BAD=/home/serv6/itso/db/jddl/order/
order_${hostname}.bad
```

where the ORDER data is stored in the file order.1 on the /stage directory.

The control file is also similar to that of LINE; however, there is no mention of a particular partition in it since all partitions reside in the same tablespace:

```
LOAD DATA
APPEND
INTO TABLE order
FIELDS TERMINATED BY '|'
(
    o_orderkey          integer external,
    o_custkey           integer external,
    o_orderstatus       char(1),
    o_totalprice        decimal external,
     o_orderdate        date 'yyyy-mm-dd',
    o_orderpriority     char(15),
    o_clerk             char(15),
    o_shippriority      integer external,
    o_comment           char(79))
```

Tables CUST, SUPP, PARTS, and PARTSP are loaded using a method similar to that of ORDER.

6.9.3 Loading NATION and REGION

NATION and REGION are small so that they will not be distributed among the OPS nodes. These two tables physically reside on just one OPS node and are loaded locally. There is only one datafile for NATION and one datafile for REGION.

As an example, here is the ksh script file and the control file for loading NATION:

```
#!/bin/ksh

hostname=$(hostname -s)

/u01/app/oracle/product/8.1.5/bin/sqlldr USERID=dss/dss CONTROL=/
home/serv6/itso/db/jddl/nation/nation.ctl DATA=/stage/nation.dat
DIRECT=TRUE 2>&1 LOG=/home/serv6/itso/db/jddl/nation/
nation_${hostname}.log  BAD=/home/serv6/itso/db/jddl/nation/
nation_${hostname}.bad
```

The control file is:

```
LOAD DATA

APPEND

INTO TABLE nation

FIELDS TERMINATED BY '|'

(

    n_nationkey         integer external,

    n_name              char(25),

    n_regionkey         integer external,

    n_comment           char(152))
```

6.10 Table statistics

You need to run the ANALYZE command against each table or partitions of a table after loading is completed. The ANALYZE statement with the STATISTICS option gathers statistics about the physical storage characteristics of an object (table or partition) and stores the statistics in the data dictionary. Oracle can use these statistics when cost-based optimization is employed to choose the most efficient execution plan for SQL statements accessing analyzed objects. You can also use statistics generated by this command to write efficient SQL statements that access analyzed tables or partitions.

You can estimate statistics using the ANALYZE statement with the ESTIMATE STATISTICS option. When estimating statistics, Oracle gathers representative information from portions of an object. This subset of information provides reasonable estimated statistics about the table. The accuracy of estimated statistics depends on how representative the sampling used by Oracle is. Only parts of an object are scanned to gather information for estimated statistics; so, an object can be analyzed quickly. You can, optionally, specify the number or percentage of rows that Oracle should use in making the estimate.

Here is a portion of the analyze script for LINE:

```
analyze table line  partition (line1) estimate statistics sample 10
PERCENT;

analyze table line  partition (line2) estimate statistics sample 10
PERCENT;

...                 ...

...                 ...

...                 ...

analyze table line  partition (line83) estimate statistics sample 10
PERCENT;

analyze table line  partition (line84) estimate statistics sample 10
PERCENT;
```

You only need to analyze the table if the table is not partitioned or if each partition does not have its own tablespace; so, for ORDER, you can simply use the following:

```
analyze table order estimate statistics sample 10 PERCENT;
```

Now, the tables are ready for query.

6.11 Creating the indexes

Indexes are created in order to improve the performance of SQL execution and to impose constraints. Almost all the techniques used for creating a table are applicable for creating an index. You can create an index on one or more columns of a table or a partitioned table. In our example database, we used different schemes for creating indexes on different tables.

6.11.1 Creating LINE index

The index of the LINE table is made on four columns: l_orderkey, l_returnflag, l_extendedprice, and l_discount. This index is partitioned into 84 partitions to match the 84 table partitions. A script similar to the following is used to create the index:

```
create index l_ored
on line (l_orderkey, l_returnflag, l_extendedprice, l_discount)
pctfree 2
initrans 10
nologging
compute statistics
tablespace tocil
storage (initial 10m next 10m freelists 32 freelist groups 7 maxextents
unlimited pctincrease 0)
parallel (degree 12 instances 7)
global partition by range (l_orderkey) (
partition lored1 values less than (71500001),
partition lored2 values less than (143000001),
...                    ...
...                    ...
...                    ...
partition lored83 values less than (5934500001),
partition lored84 values less than (MAXVALUE));
```

The index, l_ored, is globally partitioned by the range of l_orderkey into 84 partitions. We compute the statistics of the index during index creation.

6.11.2 Creating ORDER indexes

For ORDER, we create two indexes. One index, named o_okey, is on the o_orderkey column and is unique. The other index, named o_clokod, is on the three columns o_clerk, o_orderkey, and o_orderdate. The following script is used to create o_okey:

```
create unique index o_okey
on order (o_orderkey)
```

pctfree 2

initrans 10

nologging

compute statistics

tablespace iorder

storage (initial 10m next 10m freelists 32 freelist groups 7 maxextents unlimited pctincrease 0)

parallel (degree 8 instances 7);

and script:

create index o_clokod

on order (o_clerk, o_orderkey, o_orderdate)

pctfree 2

initrans 10

nologging

compute statistics

tablespace iorder

storage (initial 50m next 50m freelists 32 freelist groups 7 maxextents unlimited pctincrease 0)

parallel (degree 8 instances 7)

local;

to create the index o_clokod.

6.11.3 Creating CUST index

Here is a script used to create the unique index i_c_custkey for CUST:

create unique index i_c_custkey

on cust (c_custkey)

pctfree 0

nologging

compute statistics

tablespace constr

storage (initial 1m next 1m pctincrease 0)

```
parallel;
```

6.11.4 Creating PARTS indexes

We create two indexes. One is the unique index, named i_p_partkey, that is on the p_partkey column and is not partitioned. The other index, named p_cbp, is partitioned and is on the three columns p_container, p_brand, and p_partkey. The index p_cbp is globally partitioned into 200 partitions by range partitioning based on the two columns p_container and p_brand. Here is a script to create i_p_partkey:

```
create unique index i_p_partkey
on parts (p_partkey)
pctfree 0
nologging
compute statistics
tablespace constr
storage (initial 1m next 1m pctincrease 0)
parallel;
```

and here is a script to create p_cbp:

```
create index p_cbp
on parts (p_container,p_brand,p_partkey)
pctfree 1
nologging
compute statistics
tablespace iparts
storage (initial 1m next 5m maxextents unlimited pctincrease 0)
parallel (degree 2 instances 7)
global partition by range (p_container,p_brand) (
partition pcbp1 values less than ('JUMBO BAG','Brand#151'),
partition pcbp2 values less than ('JUMBO BAG','Brand#251'),
...                 ...
...                 ...
...                 ...
```

```
partition pcbp199 values less than ('WRAP PKG','Brand#451'),
partition pcbp200 values less than (MAXVALUE,MAXVALUE));
```

6.11.5 Creating PARTSP, SUPP, REGION, and NATION indexes

The following are the scripts used to create these indexes.

For PARTSP:

```
create unique index i_ps_partkey_suppkey
on partsp (ps_partkey,ps_suppkey)
pctfree 0
nologging
compute statistics
tablespace constr
storage (initial 1m next 1m pctincrease 0)
parallel;
```

For SUPP:

```
create unique index i_s_suppkey
on supp (s_suppkey)
pctfree 0
nologging
compute statistics
tablespace constr
storage (initial 1m next 1m pctincrease 0)
parallel;
```

For REGION:

```
create unique index i_r_regionkey
on region (r_regionkey)
pctfree 0
nologging
compute statistics
tablespace constr
```

```
    storage (initial 32k next 32k pctincrease 0);
```

For NATION:

```
    create unique index i_n_nationkey
    on nation (n_nationkey)
    pctfree 0
    nologging
    compute statistics
    tablespace constr
    storage (initial 32k next 32k pctincrease 0);
```

6.12 Alter tables to add constraint

At this stage of the creation of our example database, you need to designate the unique indexes you have already created as primary key constraints. To do this, you have to alter the tables. You may use alter table statements similar to the following for each table with a unique index:

For CUST:

```
    alter table cust add constraint cpk_c_custkey
        primary key (c_custkey) using index;
```

For NATION:

```
    alter table NATION add constraint npk_n_nationkey
        primary key (n_nationkey) using index;
```

For ORDER:

```
    alter table order add constraint ook_o_orderkey
        primary key (o_orderkey) using index;
```

For PARTS:

```
    alter table parts add constraint ppk_p_partkey
        primary key (p_partkey) using index;
```

For PARTSP:

```
    alter table partsp add constraint pspk_ps_partkey_suppkey
        primary key (ps_partkey,ps_suppkey) using index;
```

For REGION:

```
alter table region add constraint rpk_r_regionkey
    primary key (r_regionkey) using index;
```

For SUPP:

```
alter table supp add constraint spk_s_suppkey
    primary key (s_suppkey) using index;
```

6.13 Materialized views

Materialized views are a new Oracle feature in Oracle8*i*. They are schema objects that can be used to summarize, precompute, replicate, and distribute data. They are suitable in various computing environments, such as data warehousing, decision support, and distributed or mobile computing.

In data warehouses, materialized views are used to precompute and store aggregated data, such as sums and averages. Materialized views in these environments are typically referred to as *summaries* since they store summarized data. They can also be used to precompute joins with or without aggregations.

Oracle's cost-based optimization can make use of materialized views to improve query performance by automatically recognizing when a materialized view can and should be used to satisfy a request. The optimizer transparently rewrites the request to use the materialized view. Queries are then directed to the materialized view and not to the underlying detail tables or views.

Materialized views are similar to indexes in the following ways:

- They consume storage space.
- They must be refreshed when the data in their master tables changes.
- When used for query rewrites, they improve the performance of SQL execution, and their existence is transparent to SQL applications and users.

Unlike indexes, however, materialized views can be accessed directly using a SELECT statement and, depending on the type of refresh required, they can also be accessed directly in an INSERT, UPDATE, or DELETE statement.

A materialized view can be partitioned, and you can define a materialized view on a partitioned table and one or more indexes on the materialized view.

For our example database, we created materialized views and were able to validate that query performance is improved significantly by their use.

Appendix A. Database table and index sizes

This appendix displays size information for the tables and indexes in our example database.

A.1 Table sizes

Table 10. Table sizes

Table Name	Table Operations	Number of rows	Average Row Length (in bytes)	Table Size
SUPP	INSERT/SELECT	3 M	159	477 MB
PARTS	INSERT/SELECT	60 M	155	9300 MB
PARTSP	INSERT/SELECT	240 M	144	34560 MB
CUST	INSERT/SELECT	45 M	179	8055 MB
ORDER	INSERT/SELECT	450 M	104	46800 MB
LINE	INSERT/SELECT	1800 M	112	201600 MB
NATION	INSERT/SELECT	7500	128	0.96 MB
REGION	INSERT/SELECT	1500	124	0.186 MB
			TOTAL:	300794 MB

A.2 Index sizes

Table 11. Index sizes

Index Name	Table Name	Index Size (in K bytes)
SUPP_PK_IDX	SUPP	159 M
PART_PK_IDX	PARTS	3100 M
PSUPP_PK_IDX	PARTSP	11520 M
CUST_PK_IDX	CUST	2685 M
ORDER_PK_IDX	ORDER	15600 M
LINE_PK_IDX	LINE	67200 M

Index Name	Table Name	Index Size (in K bytes)
NATION_PK_IDX	NATION	0.32 M
REGION_PK_IDX	REGION	0.062 M
	TOTAL:	100264 M

Appendix B. Table physical properties

This appendix displays physical properties including storage and partitioning information for the tables in our example database.

B.1 Storage attributes

Table 12. Storage information

Table name	Initial extent	Next extent	Minextents	Maxextents	Pctinrease	Freelists	Freelist groups
SUPP	32 KB	2 MB	Default	unlimited	0	Default	Default
PARTS	32 KB	25 MB	Default	unlimited	0	Default	Default
PARTSP	32 KB	48 MB	Default	unlimited	0	Default	Default
CUST	Default	Default	Default	Default	0	Default	Default
ORDER	32 KB	256 MB	Default	unlimited	0	32	7
LINE	32 KB	512 MB	Default	unlimited	0	32	7
NATION	Default	Default	Default	Default	0	Default	Default
REGION	Default	Default	Default	Default	0	Default	Default

B.2 Physical attributes

Table 13. Physical attributes

Table name	Pctused	Pctfree	Initrans	Maxtrans	Degree of parallelism
SUPP	99	0	Default	Default	Default
PARTS	99	0	Default	Default	Default
PARTSP	99	0	Default	Default	Default
CUST	99	0	Default	Default	Default
ORDER	99	0	10	255	No of partitions
LINE	99	0	10	255	No of partitions
NATION	Default	Default	Default	Default	None

Table name	Pctused	Pctfree	Initrans	Maxtrans	Degree of parallelism
REGION	Default	Default	Default	Default	None

B.3 Table partitioning

Table 14. Table partitioning

Table name	Partitioning key	Key type	Partition type	Partition range
LINE	L_SHIPDATE	DATE	By Range	Monthly
ORDER	O_ORDERDATE	DATE	By Hash	
PARTS	P_SIZE	INTEGER	By Range	1,2,3,4,...maxvalue
CUST	C_CUSTKEY	INTEGER	By Hash	

Appendix C. Tablespace sizes (without partitioning)

This appendix lists the sizes for each tablespace taken as a whole (without any partitioning)

Table 15. Tablespace sizes (without partitioning)

Tablespace Name	Use	Estimated Size	Size Used
SYSTEM	Oracle data dictionary	120 MB	120 MB
RBS1	Rollback segments for node 1	4000 MB	4000 MB
RBS3	Rollback segments for node 3	4000 MB	4000 MB
RBS5	Rollback segments for node 5	4000 MB	4000 MB
RBS7	Rollback segments for node 7	4000 MB	4000 MB
RBS9	Rollback segments for node 9	4000 MB	4000 MB
RBS11	Rollback segments for node 11	4000 MB	4000 MB
RBS13	Rollback segments for node 13	4000 MB	4000 MB
TEMP	Temporary segments	201600 MB	201600 MB
USERS	Users default tablespace	25 MB	25 MB
LINE	Tablespace for LINE table	201600 MB	242688 MB
LINEIDX	Tablespace for LINE index	67200 MB	80896 MB
ORDER	Tablespace for ORDERS table	46800 MB	56320 MB
ORDIDX	Index for ORDERS table	15600 MB	19456 MB
CUST	CUST table	8055 MB	9216 MB
CUSTIDX	Index for CUST table	2685 MB	3222 MB
SUPP	SUPP table	477 MB	512 MB
SUPPIDX	Indexes for SUPP table	159 MB	256 MB
PARTS	PARTS table	9300 MB	10240 MB
PARTIDX	Indexes for PARTS table	3100 MB	4096 MB
PSUPP	PARTSP table	34560 MB	40960 MB
PSUPPIDX	Indexes for PARTSP table	11520 MB	14336 MB
	TOTAL:		711946 MB

Appendix D. Tablespace file layout - a summary

This appendix gives details about the tablespaces before partitioning and, most importantly, about allocated disks and nodes. It gives an overall look at how the database is spread across nodes and disks.

Table 16. Tablespace file layout

Tablespace Name	Use	Size Used	Nodes	Disks Used
SYSTEM	Oracle data dictionary	120 MB	1	4
redo logs	Roll forward	7168 MB	1-13	5,6
RBS1	Rollback segments for node 1	2000 MB	1	7
RBS3	Rollback segments for node 3	2000 MB	3	7
RBS5	Rollback segments for node 5	2000 MB	5	7
RBS7	Rollback segments for node 7	2000 MB	7	7
RBS9	Rollback segments for node 9	2000 MB	9	7
RBS11	Rollback segments for node 11	2000 MB	11	7
RBS13	Rollback segments for node 13	2000 MB	13	7
TEMP	Temporary segments	201600 MB	1-13	8-38
USERS	Users default tablespace	25 MB	1	2
LINE	Tablespace for LINES table	242688 MB	1-13	8-38
LINEIDX	Tablespace for LINES index	80896 MB	1-13	39-49
ORDER	Tablespace for ORDERS table	56320 MB	1-13	8-14
ORDIDX	Index for ORDERS table	19456 MB	1-13	40-44
CUST	CUST table	9216 MB	3	20-29
CUSTIDX	Index for CUST table	3222 MB	3	40-44
SUPP	SUPP table	512 MB	5	30
SUPPIDX	Indexes for SUPP table	256 MB	5	40-44
PARTS	PARTS table	10240 MB	7	21-30
PARTIDX	Indexes for PARTS table	4096 MB	7	40-44
PSUPP	PARTSUP table	40960 MB	9	13-19
PSUPPIDX	Indexes for PARTSP table	14336 MB	9	45-49

Appendix E. File layout for system tablespaces

This appendix describes the physical file layout for the database system tablespaces. It gives the location, LV name, and VSD name for each database file.

Table 17. File layout for system tablespaces

Tablespace Name	Size	Nodes	Disk	LV name	VSD name
SYSTEM	120 MB	1	4	SYSTEM01N01	vsd.system01n01
control file 1	32 MB	1	4	CTRL1N01	vsd.ctrl1n01
control file 2	32 MB	3	4	CTRL2N03	vsd.ctrl2n03
control file 3	32 MB	5	4	CTRL3N05	vsd.ctrl3n05
redo loga (Group 1) [2 mirror copies(a and b) in each group]	128 MB 128 MB 128 MB 128 MB 128 MB 128 MB 128 MB 128 MB 128 MB 128 MB 128 MB 128 MB 128 MB 128 MB	1 1 3 3 5 5 7 7 9 9 11 11 13 13	5 6 5 6 5 6 5 6 5 6 5 6 5 6	LOG01N01a LOG01N01b LOG01N03a LOG01N03b LOG01N05a LOG01N05b LOG01N07a LOG01N07b LOG01N09a LOG01N09b LOG01N11a LOG01N11b LOG01N13a LOG01N13b	vsd.log01n01a vsd.log01n01b vsd.log01n03a vsd.log01n03b vsd.log01n05a vsd.log01n05b vsd.log01n07a vsd.log01n07b vsd.log01n09a vsd.log01n09b vsd.log01n11a vsd.log01n11b vsd.log01n13a vsd.log01n13b
redo loga (Group 2) [2 mirror copies(a and b) in each group]	128 MB 128 MB ... 128 MB 128 MB	1 1 ... 13 13	5 6 ... 5 6	LOG02N01a LOG02N01b ... LOG02N13a LOG02N13b	vsd.log02n01a vsd.log02n01b ... vsd.log02n13a vsd.log02n13b
redo loga (Group 3) [2 mirror copies(a and b) in each group]	128 MB 128 MB ... 128 MB 128 MB	1 1 ... 13 13	5 6 ... 5 6	LOG03N01a LOG03N01b ... LOG03N13a LOG03N13b	vsd.log03n01a vsd.log03n01b ... vsd.log03n13a vsd.log03n13b
redo loga (Group 4) [2 mirror copies(a and b) in each group]	128 MB 128 MB ... 128 MB 128 MB	1 3 ... 13 13	5 5 ... 5 6	LOG04N01a LOG04N01b ... LOG04N13a LOG04N13b	vsd.log04n01a vsd.log04n01b ... vsd.log04n13a vsd.log04n13b

Tablespace Name	Size	Nodes	Disk	LV name	VSD name
RBS1	1024 MB	1	7	RBS301N01	vsd.rbs301n01
	1024 MB	1	7	RBS302N01	vsd.rbs302n01
	1024 MB	1	7	RBS303N01	vsd.rbs303n01
	1024 MB	1	7	RBS304N01	vsd.rbs304n01
RBS3	1024 MB	3	7	RBS501N03	vsd.rbs501n03
	1024 MB	3	7	RBS502N03	vsd.rbs502n03
	1024 MB	3	7	RBS503N03	vsd.rbs503n03
	1024 MB	3	7	RBS504N03	vsd.rbs504n03
RBS5	1024 MB	5	7	RBS701N05	vsd.rbs701n05
	1024 MB	5	7	RBS702N05	vsd.rbs702n05
	1024 MB	5	7	RBS703N05	vsd.rbs703n05
	1024 MB	5	7	RBS704N05	vsd.rbs704n05
RBS7	1024 MB	7	7	RBS901N07	vsd.rbs901n07
	1024 MB	7	7	RBS902N07	vsd.rbs902n07
	1024 MB	7	7	RBS903N07	vsd.rbs903n07
	1024 MB	7	7	RBS904N07	vsd.rbs904n07
RBS9	1024 MB	9	7	RBS1101N9	vsd.rbs1101n9
	1024 MB	9	7	RBS1102N9	vsd.rbs1102n9
	1024 MB	9	7	RBS1103N9	vsd.rbs1103n9
	1024 MB	9	7	RBS1104N9	vsd.rbs1104n9
RBS11	1024 MB	11	7	RBS1301N11	vsd.rbs1301n11
	1024 MB	11	7	RBS1302N11	vsd.rbs1302n11
	1024 MB	11	7	RBS1303N11	vsd.rbs1303n11
	1024 MB	11	7	RBS1304N11	vsd.rbs1304n11
RBS13	1024 MB	13	7	RBS1301N13	vsd.rbs1301n13
	1024 MB	13	7	RBS1302N13	vsd.rbs1302n13
	1024 MB	13	7	RBS1303N13	vsd.rbs1303n13
	1024 MB	13	7	RBS1304N13	vsd.rbs1304n13

Appendix F. File layout for user tablespaces

This appendix describes the physical file layout for the database user tablespaces. It gives the location, LV name, and VSD name for each database file.

Table 18. File layout for user tablespaces

Tablespace Name	Size	Nodes	Disk	LV name	VSD name
TEMP (total 204800M)	1024 MB 1024 MB 1024 MB 1024 MB 1024 MB ...	1 3 5 7 9 ...	8-38	TEMP01N01 TEMP02N03 TEMP03N05 TEMP04N07 TEMP05n09 TEMP[01-05]n[01-09]	vsd.TEMP01N01 vsd.TEMP02N03 vsd.TEMP03N05 vsd.TEMP04N07 vsd.TEMP05n09 ...
USERS	25 MB	1	2	USERS01N01	vsd.users01N01
ORD_P1	1024 MB	1	8	ORD_P101N01	vsd.ord_p101n01
ORD_P2	1024 MB	3	8	ORD_P201N03	vsd.ord_p201n03
ORD_P3	1024MB	5	8	ORD_P301N05	vsd.ord_p301n05
ORD_P4	1024 MB	7	8	ORD_P401N07	vsd.ord_p401n07
...	...	...	...	...	...
ORD_P8	1024 MB	1	9	ORD_P801N01	vsd.ord_p801n01
...	...	...	...	...	...
ORD_P84	1024 MB	13	14	ORD_P8401N13	vsd.ord_p5501n13
LINE_P1	1024 MB 1024 MB 1024 MB 1024 MB 1024 MB 1024 MB 1024 MB	1 1 1 1 1 1 1	8 9 10 11 12 13 14	LINE_P101N01 LINE_P102N01 LINE_P103N01 LINE_P104N01 LINE_P105N01 LINE_P106N01 LINE_P107N01	vsd.line_p101n01 vsd.line_p102n01 vsd.line_p103n01 vsd.line_p104n01 vsd.line_p105n01 vsd.line_p106n01 vsd.line_p107n01

Tablespace Name	Size	Nodes	Disk	LV name	VSD name
LINE_P2	1024 MB 1024 MB 1024 MB 1024 MB 1024 MB 1024 MB 1024 MB	3 3 3 3 3 3 3	9 10 11 12 13 14 15	LINE_P201N03 LINE_P202N03 LINE_P203N03 LINE_P204N03 LINE_P205N03 LINE_P206N03 LINE_P207N03	vsd.line_p201n03 vsd.line_p202n03 vsd.line_p203n03 vsd.line_p204n03 vsd.line_p205n03 vsd.line_p206n03 vsd.line_p207n03
LINE_P3	1024 MB 1024 MB 1024 MB 1024 MB 1024 MB 1024 MB 1024 MB	5 5 5 5 5 5 5	10 11 12 13 14 15 16	LINE_P301N05 LINE_P302N05 LINE_P303N05 LINE_P304N05 LINE_P305N05 LINE_P306N05 LINE_P307N05	vsd.line_p301n05 vsd.line_p302n05 vsd.line_p303n05 vsd.line_p304n05 vsd.line_p305n05 vsd.line_p306n05 vsd.line_p307n05
...	...	...	...	...	...
LINE_P45	1024 MB 1024 MB 1024 MB 1024 MB 1024 MB 1024 MB 1024 MB	5 5 5 5 5 5 5		LINE_P4501N01 LINE_P4502N03 LINE_P4503N05 LINE_P4504N07 LINE_P4505N09 LINE_P4506N11 LINE_P4507N13	vsd.line_p4501n01 vsd.line_p4502n03 vsd.line_p4503n05 vsd.line_p4504n07 vsd.line_p4505n09 vsd.line_p4506n11 vsd.line_p4507n13
...	...	...	...	...and so on	...
LINE_P84	1024 MB 1024 MB 1024 MB	11 11 11	?	LINE_P8401N01 LINE_P8402N03 LINE_P8403N05	vsd.line_p8401n01
PSUPP (40 db files) disks 13-19	1024 MB 1024 MB ... 1024 MB 1024 MB	1 3 ... 1 ...	13 13 ... 14 ...	PSUPP01N01 PSUPP02N03 ... PSUPP08N01 ...	vsd.psupp01n01 vsd.psupp02n03 ... vsd.psupp08n01 ...
PARTS(10 db files) disks 13 - 14	1024 MB ... 1024 MB	1 ... 5	21	PARTS01N01 ... PARTS10N??	vsd.parts01n01 ... vsd.parts10n??
SUPP (1 db file)	512 MB	1	30	SUPP01N01	
SUPPIDX	256 MB	3		SUPPIDX03N03	
CUST (9 db files)	1024 MB	3	20-29	CUST01N01	

Appendix G. Common init parameter file initdss.ora

```
# ***********************************************************************
# initdss.ora - instance configuration parameters, COMMON TO ALL NODES
# ***********************************************************************

db_name                          = dss
compatible                       = 8.1.5            # default: 8.0.0

#***********************************
# -- Control Files
#***********************************

control_file_record_keep_time    = 7                # default : 7 days
control_files                    = (/dev/rvsd.ctl1n1,
                                    /dev/rvsd.ctl2n3,
                                    /dev/rvsd.ctl3n5)

#***********************************
# -- System Global Area
#***********************************

db_block_size                   = 16384             # default: OS dependent
db_block_buffers                = 65536             # default: 50
db_block_max_dirty_target        = 65536            # default: db_block_buffers
log_buffer                       = 1048576          # default: OS dependent
shared_pool_size                 = 52428800         # default: 16MB
shared_pool_reserved_size        = 2621440          # default: 0.05
*shared_pool_size
pre_page_sga                     = false            # default: false
large_pool_size                  = 0                # default: 0
db_block_lru_latches             = 2                # default: cpu_count/2

#***********************************
# -- I/O
#***********************************

db_files                         = 970              # default: OS dependent
db_file_multiblock_read_count    = 4                # default: 8
db_block_checksum                = false            # default: false
db_writer_processes              = 1                # default: 1
disk_asynch_io                   = true             # default: true
tape_asynch_io                   = true             # default: true
use_sigio                        = true


#***********************************
# -- Cursors and Sessions
```

```
#***********************************

open_cursors                      = 50              # default: 50
cursor_space_for_time             = false           # default: false
sessions                          = 225             # default: (1.1*processesr)+5
session_cached_cursors            = 25              # default: 0
session_max_open_files            = 10              # default: 10
object_cache_max_size_percent     = 10              # default: 10
object_cache_optimal_size         = 102400          # default: 102400

#row_cache_cursors                = 100             # obsolete

#***********************************
# -- Transaction
#***********************************

dml_locks                        = 512              # default: 4*transactions
processes                        = 200              # default: 30
transactions                     = 220              # default: 1.1*sessions
enqueue_resources                 = 6000            # defaule: derived
row_locking                       = always          # defaule: always
recovery_parallelism             = 2                # default: OS dependent

#***********************************
# -- Rollback
#***********************************

max_rollback_segments            = 84               # default: 30
#cleanup_rollback_entries        = 20               # obsolete
#discrete_transactions_enabled   = false            # obsolete
transactions_per_rollback_segment= 73               # default: 5

#***********************************
# -- Checkpoint
#***********************************


log_checkpoint_interval           = 9999999         # default: OS dependent
log_checkpoint_timeout            = 0               # default: 0 sec
log_checkpoints_to_alert          = true            # default: false

#***********************************
# -- Parallel Server
#***********************************

gc_rollback_locks                = "0=100:1-28=5EACH"
# gc_files_to_locks               = "1=2000:1EACH9=1536:10-93=1EACH:94-177=1EACH:178-
261=1EACH:318-345=1EACH:346-429=1EACH:430-513=1EACH:514-527=1EACH:528-555=1EACH:556-
562=1EACH:563-646=1EACH:647-660=1EACH:661-695=1EACH:696-702=1EACH:703-737=1EACH:738-
```

```
751=1EACH:752-793=1EACH:794-800=1EACH:801-807=1EACH"
gc_releasable_locks             = 78643            # default: db_block_buffers
gc_defer_time                   = 10               # default: 10
parallel_server                 = true             # default: false
parallel_min_servers           = 4                 # default: 0
parallel_max_servers           = 168               # default: OS specific
parallel_min_percent                 = 50          # default: 0
parallel_broadcast_enabled      = true             # default: false

lm_locks                        = 157286           # 2*gc_releasable_locks
lm_ress                         = 157286           # 2*gc_releasable_locks
lm_procs                        = 400              # 2*processes

#freeze_DB_for_fast_instance_recovery = true       # obsolete
_affinity_on                         = true       # OS dependent

#***********************************
# -- Parallel Query - BM usage only
#***********************************

#_dss_cache_flush              = false              # BM usage only, not supported

#***********************************
# -- Index
#***********************************

#fast_full_scan_enabled        = true               # obsolete

#***********************************
# -- Sort Area
#***********************************

sort_area_size                  = 8388608          # default: OS dependent
sort_area_retained_size         = 8388608          # default: sort_area_size

#***********************************
# -- Dump
#***********************************

background_dump_dest           = ?/rdbms/log       # default: OS dependent
core_dump_dest                 = ?/rdbms/log       # default: $ORACLE_HOME/dbs
user_dump_dest                 = ?/rdbms/log       # default: OS dependent
max_dump_file_size              = 10000000         # default: 10000 blocks

#***********************************
# -- Audit
#***********************************

timed_statistics                = true             # default: false
```

```
audit_trail                     = false             # default: false
audit_file_dest                 = ?/rdbms/audit     # default: ?/rdbms/audit
transaction_auditing            = false             # default: true

#***********************************
# -- Archiver
#***********************************

log_archive_start               = false             # default: false
log_archive_dest                = ?/dbs/arch        # default: OS dependent
log_archive_format              = log_%t_%s.arc

#***********************************
# -- SMP
#***********************************

#_kgl_bucket_count              = 4                 # limit to number of CPUS

#***********************************
# -- Backup and Restore
#***********************************

backup_tape_io_slaves           = true              # default: false

#***********************************
# -- Distributed Database
#***********************************

commit_point_strength           = 1                 # default: 1
db_domain                       = WORLD             # default: WORLD
global_names                    = false             # default: false
distributed_transactions        = 0                 # set tx if DB is distributed
open_links                      = 100               # default: 4
open_links_per_instance         = 4                 # default: 4

#***********************************
# -- Replication
#***********************************

replication_dependency_tracking   = true            # default: true

#***********************************
# -- Profile
#***********************************

resource_limit                    = false           # default: false

#***********************************
# -- Optimizer
```

```
#***********************************

optimizer_features_enable        = 8.1.5            # default: 8.0.0
optimizer_mode                   = choose           # default: choose
optimizer_percent_parallel       = 100              # default: 0
optimizer_search_limit           = 5                # default: 5

#***********************************
# -- sql_trace
#***********************************

sql_trace                        = false            # default: false

#***********************************
# -- Password File
#***********************************

#remote_login_passwordfile        = exclusive        # default: none, passwordfile

#***********************************
# -- Roles
#***********************************

max_enabled_roles                = 20               # default: 20

#***********************************
# -- Query Join
#***********************************

hash_join_enabled                = true             # default: true
hash_multiblock_io_count         = 4                # default: 1
always_anti_join                 = hash             # default: nested_loops
hash_area_size                   = 16777216         # default: 2*sort_area_size

partition_view_enabled           = true             # default: false

#***********************************
# -- Bit Map and Star Schema
#***********************************

star_transformation_enabled      = true             # default: false
create_bitmap_area_size          = 8388608          # default: 8Mb
bitmap_merge_area_size           = 1048576          # default: 1Mb

#***********************************
# -- Oracle 8i
#***********************************

job_queue_processes              = 0                # default: 0
```

```
query_rewrite_enabled           = true
```

Appendix H. Scripts to create the tables in the example database

This section displays some example scripts that may be used to create the tables in our example database.

H.1 Example script to create table LINE

```
create table line(
    l_shipdatedate,
    l_orderkeynumber not null,
    l_discountnumber,
    l_extendedpricenumber,
    l_suppkeynumber not null,
    l_quantitynumber,
    l_returnflagchar(1),
    l_partkeynumber not null,
    l_linestatuschar(1),
    l_taxnumber not null,
    l_commitdatedate,
    l_receiptdatedate,
    l_shipmodevarchar(10),
    l_linenumbernumber not null,
    l_shipinstructvarchar(25),
    l_commentvarchar(44))

pctfree 1

pctused 99

initrans 10

storage (initial 32k next 45m maxextents unlimited freelists 32 freelist groups
7 pctincrease 0)

parallel (degree 84)

nologging
```

```
partition by range (l_shipdate)
(
partition item1 values less than (to_date('1992-02-01','YYYY-MM-DD'))
tablespace line1
,
partition item2 values less than (to_date('1992-03-01','YYYY-MM-DD'))
tablespace line2
,
...                         ...
...                         ...
,
partition item82 values less than (to_date('1998-11-01','YYYY-MM-DD'))
tablespace line82
,
partition item83 values less than (to_date('1998-12-01','YYYY-MM-DD'))
tablespace line83
,
partition item84 values less than (MAXVALUE)
tablespace line84
);
```

H.2 Example script to create SUPP

```
create table supp(
    s_suppkeynumber NOT NULL,
    s_nationkeynumber,
```

```
    s_commentvarchar(101),
    s_namevarchar(25),
    s_addressvarchar(40),
    s_phonevarchar(15),
    s_acctbalnumber)
pctfree 0
pctused 99
tablespace supprt
storage (initial 32k next 512k maxextents unlimited pctincrease 0)
parallel
nologging;
```

H.3 Example script to create REGION

```
create table region(
    r_regionkeynumber,
    r_namevarchar(25),
    r_commentvarchar(125))
tablespace system;
```

H.4 Example script to create NATION

```
create table nation(
    n_nationkeynumber NOT NULL,
    n_namevarchar(25),
    n_regionkeynumber ,
    n_commentvarchar(152))
tablespace system;
```

Appendix I. Special notices

This publication is intended to help both an RS/6000 SP Specialist and/or an Oracle DBA who wants to implement the Oracle8*i* Parallel Server on an IBM RS/6000 SP system. The information in this publication is not intended to be the specification of any programming interfaces that are provided by IBM AIX and IBM Parallel System Support Programs (PSSP). See the PUBLICATIONS section of the Programming Announcement for the products mentioned in this redbook for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information 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.

The following 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	AS/400
AT	CT
IBM ®	Netfinity
NetView	POWERparallel
PowerPC 604	RS/6000
SP	System/390
XT	400

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/or 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 X/Open Company Limited.

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 J. Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

J.1 IBM Redbooks publications

For information on ordering these publications see “How to get IBM Redbooks” on page 147.

- *The RS/6000 SP Inside Out*, SG24-5374
- *Understanding and Using the SP Switch*, SG24-5161
- *RS/6000 SP System Performance Tuning*, SG24-5340
- *SP Perspectives: A New View of Your SP System*, SG24-5180

J.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
AS/400 Redbooks Collection	SK2T-2849
Netfinity Hardware and Software Redbooks Collection	SK2T-8046
RS/6000 Redbooks Collection (BkMgr)	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

J.3 Other resources

These publications are also relevant as further information sources:

- *Oracle8i Server Documentation, Release 8.1.5*
- *Getting to Know Oracle8i*, A68020-01

- *Oracle8i Concepts*, A67781-01
- *Oracle8i Parallel Server Concepts and Administration*, A67778-01
- *Oracle8i Installation Guide for AIX-Based Systems*, A67728-01
- *Oracle8i for AIX-Based Systems, Release Note*, A67730-01
- *Oracle8i Migration Guide*, A67774-01
- *Oracle8i Parallel Server, Setup and Configuration Guide*, A67439-01
- *Oracle8i Administratorís Reference for AIX-Based Systems*, A67729-01
- *Oracle8i Administratorís Guide*, A67772-01
- *AIX-Based Systems, Release 8.1.5: Installation Guide*
- *AIX-Based Systems, Release 8.1.5: Administrator's Reference*

  all of which are available at the Oracle Technology Network site (requires free registration):

  `http://technet.oracle.com/docs/products/oracle8i/doc_index.htm`
- *RSCT: Group Services Programming Guide and Reference*, SA22-7355
- IBM Redpiece: *PSSP 3.1 Survival Guide*, SG24-5344

and

- *PSSP: Installation and Migration Guide*, GA22-7347
- *PSSP: Administration Guide*, SA22-7348
- *PSSP: Managing Shared Disks*, SA22-7349
- *PSSP: Diagnosis Guide*, GA22-7350

  all of which are available at the IBM RS/6000 SP Library site:

  `http://www.rs6000.ibm.com/resource/aix_resource/sp_books/pssp/`

J.4 Referenced Web sites

The following Web sites are also relevant as further information sources:

- `http://www.rs6000.ibm.com/resource/technology`
- `http://www.rs6000.ibm.com/resource/aix_resource/sp_books/pssp/`
- `http://www.rs6000.ibm.com/support/sp/perf/`
- `http://www.rs6000.ibm.com/resource/technology/#sp`

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.ibmlink.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.ibmlink.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.ibmlink.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.

IBM Redbooks fax order form

Please send me the following:

Title	Order Number	Quantity

First name	Last name	
Company		
Address		
City	Postal code	Country
Telephone number	Telefax number	VAT number

☐ Invoice to customer number _____

☐ Credit card number _____

Credit card expiration date	Card issued to	Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Glossary

APAR Authorized Program Analysis Report

ASCI Accelerated Strategic Computing Initiative

AIX Advanced Interactive Executive

API Application Programming Interface

CP Crown Prince

CPU central processing unit

CSS communication subsystem

CWS control workstation

DSS Decision Support System

DB database

DDL Data Definition Language

GB gigabytes

GVG Global Volume Group

HPQS Highly Parallel Query System

HSD Hashed Shared Disk

IBM International Business Machines Corporation

ITSO International Technical Support Organization

LAN Local Area Network

LP logical partition

LRC least recently used

LV logical volume

LVM Logical Volume Manager

MB megabytes

MPP Massively Parallel Processing

NFS Network File System

OFA Oracle Optimal Flexible Architecture

OLTP On-Line Transaction Processing

PCI Peripheral Component Interconnect (Intel bus standard)

PP physical partition

PSSP Parallel System Support Programs

PV physical volume

RAID redundant array of independent disks

RAM random access memory

RSCT RS/6000 Cluster Technology

RVSD Recoverable Virtual Shared Disk

SCSI Small Computer System Interface

SDR System Data Repository

SMIT System Management Interface Tool

SMP Symmetric Multiprocessing

SSA Serial Storage Architecture

VG volume group

VLDB Very Large Database

VSD Virtual Shared Disk

Index

Symbols

A

B

C

D

E

F

G

H

I

L

M

Q

R

S

T

U

V

W

X

IBM Redbooks evaluation

Oracle8*i* Parallel Server on IBM SP Systems: Implementation Guide
SG24-5591-00

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!)

_____

_____

_____

_____

SG24-5591-00
Printed in the U.S.A.

Oracle8*i* Parallel Server on IBM SP Systems: Implementation Guide

SG24-5591-00