



Analysis of Flexible Database Clusters with IBM eServer BladeCenter and Oracle9i Real Application Clusters (RAC)

Comprehensive Analysis

Phil Horwitz and Martha Centeno

IBM eServer xSeries Performance Development & Analysis
Research Triangle Park, NC USA

Kevin Closson

PolyServe, Inc.
Beaverton, OR USA

Abstract

Modern clusters may rival the established UNIX® based server, but are they manageable? What are the ramifications of performance and availability at the application level? What is the impact on total cost of ownership (TCO)? Answering these questions is the focus of this white paper.

Oracle9i RAC is designed to help build flexible, high-performance, highly available, clustered database solutions on Linux®. Connecting such clusters to a fault-resilient Fibre Channel storage area network (SAN) lays the foundation for a computing infrastructure known as a Flexible Database Cluster (FDC).

The synergy of IBM® eServer® BladeCenter®, PolyServe Matrix Server, and Oracle9i RAC makes the Flexible Database Cluster (FDC) a powerful platform for supporting multiple applications. This white paper describes a proof-of-concept that validates the architecture and technology of the Flexible Database Cluster and confirms that:

- PolyServe Matrix Server and Oracle9i RAC perform extremely well on the BladeCenter.
- The BladeCenter architecture and technology help provide an unparalleled high-availability platform for implementing Flexible Database Clusters.
- IBM and PolyServe are leading the development of technology for scale-out computing.
- The architecture and technology of the Flexible Database Cluster help enable on-demand computing. Cluster nodes provide a pool of flexible resources for use among applications. The availability of Oracle9i RAC is enhanced because nodes can be dynamically reprovisioned using Matrix Server to cover the loss of another node.
- The Flexible Database Cluster provides strong management tools such as Matrix Server for performance and availability. A single large cluster is now easier to manage than many small clusters.
- A general-purpose cluster filesystem such as the one included with Matrix Server provides a single-system feel and greatly enhances manageability. A shared Oracle home used by all nodes also simplifies management. Support is available for all database operations that require a filesystem.

- Improved manageability, scalability, expandability, availability and asset utilization in an FDC configuration also can help dramatically lower total cost of ownership (TCO) relative to a UNIX-based IT environment.

The audience for this paper is clients who are interested in implementing an FDC infrastructure with the IBM eServer BladeCenter and Oracle9i Real Application Cluster (RAC), PolyServe Matrix Server and the Linux operating system.

Clustering and Oracle9i RAC

Since the release of Oracle9i Real Application Clusters (RAC) during Oracle Open World Europe in June 2001, RAC has had the capability to scale a single application horizontally to clusters as large as 8 nodes or greater—without application modification. This scalability has allowed RAC to gain mindshare within the clustered database world.

Another trend that has been gaining momentum is Server Consolidation with large, flexible Intel® processor-based clusters running Linux under Oracle9i RAC. Emerging clustered architectures such as the IBM BladeCenter lend themselves to configuring large clusters in reasonable physical space at low cost. Connecting such clusters to a fault-resilient Fibre Channel SAN lays the basic foundation for the computing infrastructure known as a Flexible Database Cluster (FDC).

Flexible Database Clusters are the perfect combination of server consolidation and Oracle9i RAC. Compared to several small and separate clusters, each running RAC, a large FDC offers management, availability, and performance characteristics that are difficult to ignore.

Significant testing and proof of concept were required to advance the FDC concept. To that end, IBM and PolyServe joined forces to build a 14-node BladeCenter running SUSE Linux Enterprise Server and then attached it to a formidable SAN configured with over 100 physical disk drives. This large cluster was the target of a series of tests that took an in-depth look at running and managing not just a single application, but several applications simultaneously.

Unlike most RAC studies to date, this testing was much more than the typical exercise of measuring throughput for a single application as nodes are added to the cluster. While those studies are valuable proof points, they lack information about the management aspects of large clusters, what happens when things go “wrong,” and how applications can benefit from dynamically shifting servers from under-utilized applications to applications where they are more useful. To that end, the tests included powering off servers while running large numbers of connected Oracle users and dynamically shifting servers from one application to another. Critical measurements such as recovery times and throughput impact were also analyzed. The tests and measurements are described in detail later in this paper.

The paper also includes performance information, but in a slightly different light than the norm. The FDC serves as a platform on which nodes can be shifted between applications without halting service to client systems. Therefore, tests were conducted to measure speedup to applications that were granted additional nodes dynamically.

Flexible Database Clusters

The concept of a Flexible Database Cluster is based on building a cluster that is large enough to support several databases. Instead of using one cluster for an online transaction database and two other clusters for decision support and development databases, a Flexible Database Cluster can support all three of these databases.

For many years large SMP systems have supported more than one Oracle database in what is routinely referred to as *Server Consolidation*—concentrating the workloads from several small SMPs into one large SMP. Server Consolidation, a trend that started in the late 1990s, was seen as a way to increase efficiency in the datacenter by reducing administrative overhead.

If consolidation adds value in an SMP environment, why wouldn't it add value in a cluster environment? The answer is not simply that it does add value, but, in fact, consolidation delivers more value in clustered environments than in SMP environments, particularly when supporting Oracle9i RAC.

A report provided by Meta Group¹ in 2002 makes the point that the "Hard Partitions" technology brought to market by Sun and Sequent has been recognized as valuable in consolidation. Hard Partitions offer fixed system provisioning to given applications within a single SMP system. With these systems, administrators can dedicate, for example, 16 processors to application A and another 16 processors to application B. The report also focuses on the value of dynamic partitioning (e.g., LPAR, VPAR) and suggests that it is critical for supporting consolidation; that is, flexibility is key.

These SMP-oriented Server Consolidation systems support the ability to dedicate CPUs to important workloads. Likewise, less important workloads (e.g., untested applications still under development) are cordoned off to other CPUs to limit any disturbance they might cause to other applications. Arguably, clusters do this even better than SMPs with domains or partitions. After all, with SMP systems, there are usually many system components, such as memory controllers and I/O adapters that are shared among partitions or domains. With clusters, however, an application running on nodes 1 and 2 does not share server-level resources with applications on nodes 3 and 4. The SAN is shared, but partitionable.

Server Consolidation is a good thing and, because of architectural differences, clusters perform even better in some ways than single SMP systems. Why then are so many IT organizations supporting a small dedicated cluster for each application? If several different clusters all support Oracle9i RAC based applications, why not consolidate?

The Meta Group report of 2002, along with many others, further credits Server Consolidation as enabling IT savings through reductions in the cost of support and skills, capital equipment, and system management. Fewer systems require less management.

If consolidating simple, single servers into one large server yields IT savings, how much more so does consolidating complex clusters into one large cluster?

The Flexible Database Cluster concept lowers administrative overhead and offers higher availability and on-demand scalability beyond that of several small clusters. It is an architecture well worth considering. A Flexible Database Cluster is more than just a big cluster. The prime ingredients are systems software and deployment methodology.

Large Database Clusters with Oracle9i RAC—At a Glance

The thought of assembling a 14-node cluster, much less managing one, conjures up visions of cabling and OS configuration nightmares. This mentality is likely rooted in the UNIX-based clustered systems of the 1990s. Clusters of that era were configured with very few nodes primarily due to limitations in Oracle Parallel Server.

Oracle9i RAC has changed all of that. The scalability and availability characteristics of RAC are compelling reasons to build large clusters. The economic benefit of combining powerful Intel processor-based servers running Linux in a large cluster makes the idea even more palatable. The question is why make a large database cluster, and what are the management considerations?

The answer hinges on the technology that is being assembled into the cluster.

¹ See the Meta Group report on ZDNet at: <http://techupdate.zdnet.com/techupdate/stories/main/0,14179,2878371-1,00.html>

The base requirement for an Oracle9i RAC cluster is a set of servers with shared disk (e.g., JBOD) access and LAN/interconnect connectivity. Strictly speaking, nothing more is required. For reasons explored throughout this paper, it is unlikely that such a stripped down cluster will be as manageable at a large node count. Nor would such a cluster be configured to offer the management and flexibility attributes of the Flexible Database Cluster (FDC) model studied during this project.

Concerns over building and maintaining large clusters for Oracle9i RAC generally fall into six categories, although not necessarily in this order:

- Storage configuration and management
- Storage connectivity and configuration
- OS configuration and management
- Oracle product installation, configuration and maintenance
- Database file locations and space management
- Operational methodology

Storage Configuration and Management

Today, building large clusters such as the Flexible Database Cluster in this proof of concept is actually quite simple, due in part to Fibre Channel SAN technology. Even simpler are large clusters with advanced Intel processor-based clusters such as the IBM BladeCenter.

Storage management is also much simpler and more powerful with today's technology. A great example is the IBM TotalStorage® DS family.

Storage Connectivity and Configuration

Modern technology is also making it easier to connect and configure SAN switches. In the FDC proof-of-concept system, the switch was easily configured via the IBM TotalStorage SAN Fibre Channel Switch Specialist management tool. The switch features an embedded Web browser interface for configuration, management, and troubleshooting. The BladeCenter integrates Fibre Channel switch hardware packaged to nearly eliminate cabling.

OS Configuration and Management

Configuring Linux to support a large cluster is much simpler than legacy UNIX. Enhanced configuration tool kits are available with Linux distributions such as SUSE Linux Enterprise Server (SLES8), which provides the KDE Konsole² utility. This utility feeds key input to all or some nodes of a cluster, which is useful for redundant administrative tasks. This is but one example of the added value that SUSE SLES 8 offers.

SUSE SLES8 also provides a helpful package in RPM form called **orarun**. This package assists administrators with node-level RAC requirements such as environment variables, kernel parameters, and automated startup/shutdown of key Oracle components such as OCMS, SQL*Net listeners and GSD (global services daemon).

Every small improvement that makes a cluster feel more like a single system is critically important, even more so in the case of the Flexible Database Cluster.

² For information about using KDE Konsole when configuring large clusters for Oracle9i RAC, see this paper on the SUSE website: http://www.SUSE.com/en/business/certifications/certified_software/oracle/docs/9iR2_RAC_sles8_polyserve.pdf

Oracle Product Installation, Configuration and Maintenance

On standard clusters, the Oracle9i Enterprise Edition software must be installed on each node where instances of RAC will be executed. During installation, the Oracle Universal Installer prompts for a list of nodes on which to install the product. It then copies the files for Oracle Home, all 100,000+ of them, to each node in the list.

Installing Oracle is not difficult. However, difficulty arises when Oracle is installed on, for example, 14 nodes³. Configuration must occur in 14 different locations, one for each node. For instance, configuring *init.ora* requires logging in to 14 different systems. Applying patches is difficult as well. For example, the 9.2.0.3 upgrade patch for Linux is more than 280MB in size. This is a lot of software to apply to a lot of Oracle Home locations.

With a general-purpose cluster filesystem⁴, it is quite simple to set up a “Shared Oracle Home.” First, the Oracle Universal Installer is instructed to install on only one node (a single-node cluster install). Then, after Oracle is fully installed on the cluster filesystem as a single-node install, it is very simple to “convert” that Oracle Home to be shared by any number of other nodes.

Converting to a shared Oracle Home is merely the concept of providing for like-name directories and files with node-specific directories and files. For example, it may be advantageous to have the directory *\$ORACLE_HOME/network/admin* set up as a different physical directory when you log into different nodes, yet this indirection needs to be “automagic.” The PolyServe Matrix Server cluster filesystem provides Context Dependent Symbolic Links (CDSL) to do this. Only a few objects require a CDSL, so this is really quite simple.

With a shared Oracle Home, all nodes in the cluster can use the same executables. Also, configuration files are located in the cluster filesystem and can be edited from any node in the cluster. In fact, all configuration files for all nodes can be edited from any node in the cluster. *\$ORACLE_HOME* points to the same directory on all nodes, and that directory is in the cluster filesystem.

Database File Locations and Space Management

Managing a large number of Oracle datafiles can become difficult when several databases are housed in one large cluster, such as those in the FDC proof of concept. Perhaps the most significant technology to assist in this area is a new Oracle9i feature known as Oracle Managed Files (OMF). Using OMF with Oracle9i RAC requires a cluster filesystem such as the PolyServe Matrix Server cluster filesystem.

Oracle Managed Files are preferred in a large FDC because they simplify administration. OMF⁵ provides the following benefits:

- Simplifies deployment of databases. OMF minimizes the time spent making decisions regarding file structure and naming and reduces file management tasks overall.
- Reduces corruption that can be caused by administrators specifying the wrong file. All OMF files are provided a unique name by Oracle.
- Reduces wasted disk space consumed by obsolete files. OMF removes unused files automatically.

Forcing a Database Administrator (DBA) to think about tablespaces as a collection of files takes time away from actual database administration. The “contents” of the files are the actual

³ The Oracle Universal Installer provides for a list of eight nodes for installing Oracle, while the maximum supported node count on Linux is 32. Installing Oracle on a 14-node cluster requires manual file propagation if a shared Oracle Home is not available.

⁴ Such as the PolyServe Matrix Server CFS used in this test. For more information, visit www.polyserve.com.

⁵ For in-depth information about Oracle Managed Files, see the Oracle documentation.

database. Databases consist of rows in blocks grouped into extents and tracked as segments. Those objects reside in datafiles. Deploying with OMF reduces the need for physical administration and allows concentration on logical administration—exactly where the biggest return on DBA effort lies. For instance, why be concerned about what sectors of disk are used by a filesystem? That particular detail is the responsibility of the filesystem. Likewise, DBAs can leave datafile management to the OMF. More time can be dedicated to query optimization and other such tasks that may actually improve performance.

Examples showing how the FDC case study used OMF are provided later in this paper.

Operational Methodology

Monitoring is a major concern in a large cluster environment. If administrators must execute numerous commands to get the status of many nodes, the value proposition quickly dwindles.

Monitoring cluster operations with the type of technology used in the Flexible Database Cluster proof of concept is vastly superior to the technology of the recent past. Comprehensive I/O monitoring at the storage level is possible through modern storage management software. Oracle9i RAC and Enterprise Manager offer a great deal of instance and global database monitoring. PolyServe Matrix Server offers monitoring of the entire cluster from a central console that can be executed either remotely or on any system in the datacenter, and finally, PolyServe's implementation of the Oracle Disk Manager offers unprecedented Oracle-centric, cluster-aware I/O monitoring.

Beyond monitoring, other issues have traditionally affected cluster operations. For instance, administrators have had to connect to specific nodes in the cluster to perform operations such as configuring files in Oracle Home. Also, requiring administrators to remember what node they used to perform an export or what node has a SQL*Plus report on it can be bothersome in today's hectic IT environment.

Concerns such as these have served as roadblocks to deploying the sort of large cluster that can be used as a Flexible Database Cluster.

Infrastructure for a Flexible Database Cluster

This section describes how the Flexible Database Cluster was set up for testing. The core technology for this aspect of the proof of concept was the PolyServe Matrix Server cluster filesystem. Two key Matrix Server product features were germane to this testing: the cluster filesystem and the MxODM I/O monitoring package.

Matrix Server Cluster Filesystem

The Matrix Server cluster filesystem is both general-purpose and optimized for Oracle. These attributes proved quite valuable in the following ways.

General-Purpose

- A single Shared Oracle Home was configured for all 14 nodes.
- Archived redo logging was performed in a cluster filesystem location and compressed.
- Some of the data was loaded with External Tables (an Oracle9i feature supported only on CFS with RAC).

Optimized for Oracle

- All datafiles, control files, online logs, and so on were located in filesystems mounted with the Matrix Server “DBOPTIMIZED” mount option, which implements Direct I/O.
- Oracle Disk Manager⁶ (ODM) was used for asynchronous I/O and improved clusterwide I/O monitoring.

Shared Oracle Home

As described earlier in this paper, a general-purpose cluster filesystem such as the Matrix Server CFS supports setting up a single directory for Oracle Home. This functionality is key to the Flexible Database Cluster architecture.

Context Dependent Symbolic Links

Context Dependent Symbolic Links (CDSL) are a feature of the Matrix Server CFS. They are links to files or directories that are resolved based on the hostname of the node on which the current process is executing. CDSL is an essential feature for the enabling of a shared Oracle Home. It also offers ease of management for several areas such as setting up SQL*Net, archived redo logging, instance logging (e.g., alert logs and trace files) and so on.

It is critical to note that unless a Cluster Filesystem supports Context Dependent Symbolic links, it is impossible to create a fully functional shared Oracle Home. Oracle Enterprise Manager creates many node-specific files located in the following directories, which must be CDSLs:

- `$ORACLE_HOME/network` (the *agent* subdirectory is critical; it contains *.q* files and *services.ora*)
- `$ORACLE_HOME/sysman`
- `$ORACLE_HOME/oem_webstage`

PolyServe provides user documentation for configuring a shared Oracle Home.

CDSL—Oracle Cluster Management Services Setup

Using a shared Oracle Home and CDSL made it simple to set up Oracle Cluster Management Services for 14 nodes on the FDC.

In Figure 1, the Oracle Cluster Management Services subdirectory (*oracm*) is actually a CDSL linking the directory `.oracm.${HOSTNAME}` to *oracm*. As an example, logging in to blade6 will automatically link *oracm* to *oracm.blade6*. It is easy, however, to modify the contents of any of the *oracm* CDSL directories because they are just simple directories.

⁶ For in-depth information regarding Oracle Disk Manager, see the white paper on the Oracle Technology Network: http://otn.oracle.com/deploy/availability/pdf/odm_wp.pdf.

```

Linux
blade1 oracle $ cd $ORACLE_HOME/
blade1 oracle $ ls -ld oracm*
lrwxrwxrwx 1 oracle dba 16 2004-02-19 19:37 oracm -> oracm.{HOSTNAME}
drwxr-xr-x 5 oracle dba 120 2004-02-19 19:36 oracm.blade1
drwxr-xr-x 5 oracle dba 120 2004-02-24 14:32 oracm.blade10
drwxr-xr-x 5 oracle dba 120 2004-02-24 14:32 oracm.blade11
drwxr-xr-x 5 oracle dba 120 2004-02-24 14:32 oracm.blade12
drwxr-xr-x 5 oracle dba 120 2004-02-24 14:32 oracm.blade13
drwxr-xr-x 5 oracle dba 120 2004-02-24 14:32 oracm.blade14
drwxr-xr-x 5 oracle dba 120 2004-02-18 18:30 oracm.blade2
drwxr-xr-x 5 oracle dba 120 2004-02-18 18:30 oracm.blade3
drwxr-xr-x 5 oracle dba 120 2004-02-18 18:30 oracm.blade4
drwxr-xr-x 5 oracle dba 120 2004-02-24 14:32 oracm.blade5
drwxr-xr-x 5 oracle dba 120 2004-02-24 14:32 oracm.blade6
drwxr-xr-x 5 oracle dba 120 2004-02-24 14:32 oracm.blade7
drwxr-xr-x 5 oracle dba 120 2004-02-24 14:32 oracm.blade8
drwxr-xr-x 5 oracle dba 120 2004-02-24 14:32 oracm.blade9
blade1 oracle $ █

```

Figure 1. Oracle Cluster Management Services subdirectory

Figure 2 shows a shell process on blade1 that lists the contents of the *cmcfg.ora* file in its CDSL-resolved *\$ORACLE_HOME/admin/cmcfg.ora* file. It also shows that, while executing on blade1, the *cmcfg.ora* file is visible via the physical directory path of *oracm.blade6/admin/cmcfg.ora*. Finally, using the *rsh(1)* command, the *cmcfg.ora* file for blade6 is listed as a simple path because the CDSL resolves it properly once the shell is generated on blade6.


```

Linux
blade1 oracle $ cat oracm/admin/cmcfgr.ora
HeartBeat=10000
ClusterName=TEST
PollInterval=1000
MissCount=5
ServicePort=9998
KernelModuleName=hangcheck-timer
PrivateNodeNames= int-blade1 int-blade2 int-blade3 int-blade4 int-blade5 int-blade6 int-blade7 int-blade8 int-blade9 int-blade10 int-blade11 int-blade12 int-blade13 int-blade14
PublicNodeNames= blade1 blade2 blade3 blade4 blade5 blade6 blade7 blade8 blade9 blade10 blade11 blade12 blade13 blade14
HostName=int-blade1
CmDiskFile=/mnt/ps/oradata1/quorum
blade1 oracle $ cat oracm.blade6/admin/cmcfgr.ora
HeartBeat=10000
ClusterName=TEST
PollInterval=1000
MissCount=5
ServicePort=9998
KernelModuleName=hangcheck-timer
PrivateNodeNames= int-blade1 int-blade2 int-blade3 int-blade4 int-blade5 int-blade6 int-blade7 int-blade8 int-blade9 int-blade10 int-blade11 int-blade12 int-blade13 int-blade14
PublicNodeNames= blade1 blade2 blade3 blade4 blade5 blade6 blade7 blade8 blade9 blade10 blade11 blade12 blade13 blade14
HostName=int-blade6
CmDiskFile=/mnt/ps/oradata1/quorum
blade1 oracle $ rsh blade6 cat oracm/admin/cmcfgr.ora
HeartBeat=10000
ClusterName=TEST
PollInterval=1000
MissCount=5
ServicePort=9998
KernelModuleName=hangcheck-timer
PrivateNodeNames= int-blade1 int-blade2 int-blade3 int-blade4 int-blade5 int-blade6 int-blade7 int-blade8 int-blade9 int-blade10 int-blade11 int-blade12 int-blade13 int-blade14
PublicNodeNames= blade1 blade2 blade3 blade4 blade5 blade6 blade7 blade8 blade9 blade10 blade11 blade12 blade13 blade14
HostName=int-blade6
CmDiskFile=/mnt/ps/oradata1/quorum
blade1 oracle $
blade1 oracle $

```

Figure 2. Shell process listing the contents of *cmcfgr.ora*

Using a shared Oracle Home and CSDLs reduces administrative overhead. The administrator can view and edit all of the configuration files for all nodes while logged in on any node in the cluster. Without a shared Oracle Home, the administrator would need to log in to each of the 14 nodes and edit the *cmcfgr.ora* file to set up OCMS.

It is important to note that if a given node name does not appear in the list assigned to the *PrivateNodeNames* and *PublicNodeNames* parameters in the *cmcfgr.ora* file, it cannot join the Oracle cluster. That is, it cannot join until all instances of OCMS have been shut down—a task that requires all database instances clusterwide to be shut down. For example, if there is another node called blade15 in the cluster, it cannot dynamically join the Oracle RAC cluster (OCMS) until all database instances are shut down, all copies of the *cmcfgr.ora* file are edited to add blade15 in the *PrivateNodeNames/PublicNodeNames* parameter assignments, and OCMS has been rebooted on each node.

This fact impacts sites that rely on a cold-standby server in the event of a node failure. In a four-node cluster scenario, losing node 4 and replacing it with the cold-standby node means that *cmcfgr.ora* needs to be configured in advance to accommodate its node name in the active cluster.

The FDC architecture is much more flexible. All nodes in the cluster are included in the OCMS member list. They all run Oracle9i RAC; the only thing that varies is which instances run on the various nodes.

Matrix Server Oracle Disk Manager

PolyServe Matrix Server provides an ODM implementation called MxODM to support the Oracle Disk Manager interface. Although MxODM offers improved datafile integrity through clusterwide file keys for access, its main benefit in the FDC architecture is monitoring. MxODM also enables Oracle9i with asynchronous I/O on the direct I/O-mounted filesystems where it stores datafiles and other database files such as redo logs.

The MxODM I/O statistics package provides the following basic I/O performance information. These reported items are referred to as the Core Reporting Elements.

- Number of file Read and Write operations
- Read and Write throughput per second in Kilobytes
- Count of synchronous and asynchronous I/O operations
- I/O service times
- Percentages

The Core Reporting Elements can be provided at the following levels:

- Clusterwide Level. Provides aggregate information for all database instances on all nodes.
- Database Global Level. Limits information to a named database (e.g., PROD, DEV, FIN, DSS).
- Instance Level. Limits information to a named instance (e.g., PROD1, PROD8, DEV1, FIN4, DSS_6).
- Node Level. Limits information to a named node (e.g., rhas1.acme.com, rhas6.acme.com). This information is the aggregate of all instance activity on the named node. If a node hosts instances accessing different databases (e.g., \$ORACLE_SID=PROD1, \$ORACLE_SID=DEV1), the Core Reporting Elements will reflect the combined information for all instances on the named node.

Because MxODM understands Oracle file, process, and I/O types, the **mxodmstat(8)** command offers very specialized reporting capabilities. On complex clustered systems, it is nearly impossible to take a quick look at the clusterwide or per-instance activity for a given subsystem of the Oracle Server.

For instance, on an 8-node cluster with six PROD instances, two DEV instances, and Parallel Query Slaves active only on nodes 1 through 4 on the PROD database, a DBA will find it extremely difficult to associate clusterwide impact to the PQO activity. Likewise, quickly determining the DBWR activity for only the PROD instances on nodes 1 through 6 is nearly impossible—without MxODM.

MxODM offers “canned” reporting that focuses on the following key Oracle “subsystems.”

Parallel Query Option (PQO). This query returns the Core Reporting Elements for only the Parallel Query Option slaves (e.g., ora_p000_PROD1, ora_p001_PROD3, etc.). This is an extremely beneficial set of information because it allows DBAs to get a top-level view of the impact PQO is having on the cluster, either as a whole or at the node level.

Log Writer. This query focuses on only the **lgwr** processes and their activity at the cluster level, database level, or node level. Because all of the Core Reporting Elements can be returned in this query, it is beneficial for DBAs to maintain streaming output of this query showing **lgwr** activity at either the cluster level or broken down by database, instance, or node.

Database Writer. This query is of the utmost value. It too can return all Core Reporting Elements at all Reporting Levels; however, it can also limit reporting to only **dbwr** process activity. DBAs

can glance at **mxodmstat(8)** output and easily determine the average **dbwr** I/O service times for all databases clusterwide, or can focus on specific databases, nodes, or instances.

Although there is too much functionality in the **mxodmstat** package to describe in this paper⁷, the following examples show some of the helpful monitoring used during the FDC project.

Monitoring several databases that have instances spread across a 14-node BladeCenter is a daunting task. A bird's-eye view can be obtained with **mxodmstat**, as was done on the FDC test system. The first command in Figure 3 shows top-level I/O activity for all databases in aggregate broken out into reads and writes. The second command shows top-level I/O by database for all three databases. This is the aggregated total I/O with breakout for the count of synchronous and asynchronous I/O as well as I/O service times. Note that without the **-a op** argument/option pair, the I/O is not broken out by reads and writes.

```

Linux
$ mxodmstat -i5 -a op
          Read
Sync  Async  KB/s  Ave ms  Sync  Async  KB/s  Ave ms
5018  6087  798799  14     16   2251  14933  15
2448  2743  361485  17     7    981   6298   14
1997  2114  279438  13     5    775   5067   12
2048  2123  281085  12     5    782   5046   16
1408  1786  233820  15     4    709   4802   17

$ mxodmstat -i5 -D
          dss
Sync  Async  KB/s  Ave ms  Sync  Async  KB/s  Ave ms  Sync  Async  KB/s  Ave ms
9     5939  740196  10     10   341   5267   51   3957  1410  36687  15
3     2160  269139  10     7    192  2635   60   1701  553   15682  17
3     2151  268134  10     6    168  2222   44   1966  552   17877  13
3     2093  260989  10     3    321  3495   65   1932  528   17481  14
4     2081  259287  10     3    182  2468   46   1890  544   17133  12
3     2154  268570  10     4    93   1737   43   1923  535   17441  16
3     2116  263581  10     4    186  2663   47   1958  626   18427  13

$

```

Figure 3. Monitoring databases with **mxodmstat**

The **mxodmstat** output in Figure 4 shows a flurry of DSS database activity.

```

Linux
$ mxodmstat -i10 -D prod dss dev
          prod
Sync  Async  KB/s  Ave ms  Sync  Async  KB/s  Ave ms  Sync  Async  KB/s  Ave ms
3160  2714  34339  15     4    0.40  66     1     6     38   1624   4
2735  2775  33268  18     3    0.40  53     1     3     30   1364   5
2566  2301  29048  17     3    0.30  51     1     3    228  2917   53
2596  2119  27663  16     3    0.30  51     2     4    110  1808   53
2295  2110  25998  15     3    0.36  47     1     3   398  4200   59
2304  2085  25625  16     3    0.30  51     1     6     69  1469   38
1978  1932  21946  22     3    0.30  51     1     3     28  1212   5
2393  2541  29718  15     3    745  92698  12     4     28  1273   4
2584  2176  28052  16     3   2111 263050  10     4    224  2796   59
2604  2203  28431  16     3   2130 265467  10     3    142  2223   82
2490  2262  28155  16     6   2102 262051  10     4   436  4529   66
2537  2140  27485  16     3   2060 256738  10     6   117  1923   55
2658  2277  29227  16     3   1618 201694  10     3   148  2340   54
2581  2342  29259  15     3    0.40  48     1     4     82  5052   41
2596  2284  28791  15     3    0.40  51     1     4   118  4972   44
2565  2313  28975  16     3    0.20  54     1     3   404  4338   56

$

```

Figure 4. **mxodmstat** shows database activity

⁷ For complete documentation for the PolyServe MxODM I/O monitoring package, see the *MxODM User Guide*. For general information about MxODM, see www.polyserve.com.

Figure 5 shows a drill-down command to see what processes are performing the I/O on the DSS database. The output shows that throughout all of the DSS instances, the I/O is almost entirely performed by Parallel Query Processes.

```

Linux
$ mxodmstat -i10 -D dss -s proc

```

Background				DB Writer				Log Writer				PQO				Foreground			
Sy	As	KB/s	Ave	Sy	As	KB/s	Ave	Sy	As	KB/s	Ave	Sy	As	KB/s	Ave	Sy	As	KB/s	Ave
2	0	38	1	0	0	0	0	0	0	0.20	0	1	0	0	0	0	0	0	0
3	0	51	2	0	0	0	0	0	0	0.30	0	8	0	0	0	0	0	0	0
3	0	51	1	0	0	0	0	0	0	0.30	0	1	0	0	0	0	0	0	0
3	0	50	5	0	0	0	0	0	0	0.40	0	6	0	1754	218363	10	0	0	0
3	0	51	6	0	0	0	0	0	0	0.30	0	4	0	2113	263240	10	0	0	0
3	0	48	6	0	0	0	0	0	0	0.30	0	8	0	2114	263435	10	0	0	0
3	0	48	7	0	0	0	0	0	0	0.40	0	6	0	2132	265597	10	0	0	0
3	0	54	6	0	0	0	0	0	0	0.20	0	5	0	2124	264634	10	0	0	0
3	0	51	8	0	0	0	0	0	0	0.40	0	9	0	2106	262446	10	0	0	0
4	0	61	7	0	0	0	0	0	0	0.50	0	6	0	2118	263824	10	0	0	0
3	0	54	8	0	0	0	0	0	0	0.20	0	5	0	1726	215011	10	0	0	0
3	0	53	1	0	0	0	0	0	0	0.40	0	1	0	0	0	0	0	0	0
3	0	48	1	0	0	0	0	0	0	0.30	0	1	0	0	0	0	0	0	0

Figure 5. mxodmstat drill-down command

Figure 6 shows a focused monitoring of instances DSS1 and DSS2 only, broken out by reads and writes. Note that after some six lines of output (60 seconds), the query plan being serviced by DSS1 and DSS2 switched from 100% asynchronous large reads evenly by both instances to only DSS1 performing small synchronous reads, along with just a few asynchronous writes.

```

Linux
$ mxodmstat -i10 -I dss1 dss2 -a op

```

dss1										dss2									
Read					Write					Read					Write				
Syn	Asy	KB/s	Ave	m	Syn	Asy	KB/s	Ave	m	Syn	Asy	KB/s	Ave	m	Syn	Asy	KB/s	Ave	m
1	1233	153846	10	0	0	0	0	0	0	1	1240	154440	10	0	0	0	0	0	0
1	1046	130483	10	0	0	0	0	0	0	1	1050	130728	10	0	0	0	0	0	0
1	1060	132339	10	0	0	0	0	0	0	1	1058	131645	10	0	0	0	0	0	0
1	1049	130899	10	0.30	0.10	5	5	5	6	1	1052	131013	10	0.30	0.20	5	5	6	6
1	1043	130131	10	0.40	0.30	7	7	7	7	1	1050	130677	10	0.40	0.20	6	6	9	9
8	321	40171	10	0.30	7	60	60	60	2	13	322	40291	9	0.30	0.10	5	5	2	2
529	0	8464	2	0.30	4	324	324	324	9	1	0	21	1	0.30	0.20	5	5	1	1
669	0	10698	1	0.30	2	517	517	517	6	1	0	16	1	0.30	0.20	5	5	3	3
720	0	11522	1	0.30	3	697	697	697	6	1	0	21	1	0.30	0.20	5	5	1	1

Figure 6. Instances DSS1 and DSS2

Cluster Management

Matrix Server is more than a cluster filesystem; it also offers a high-availability framework and SAN management. Furthermore, Matrix Server offers multi-path I/O, which is a pivotal component in reducing and eliminating single points of failure.

Figure 7 shows the Matrix Server console, which provides a bird's-eye view of cluster status. The console can be used with simple point and click to set up filesystems, mount options and advanced Hi-Av process and service monitoring for failover. A command-line interface is also available for cluster configuration.

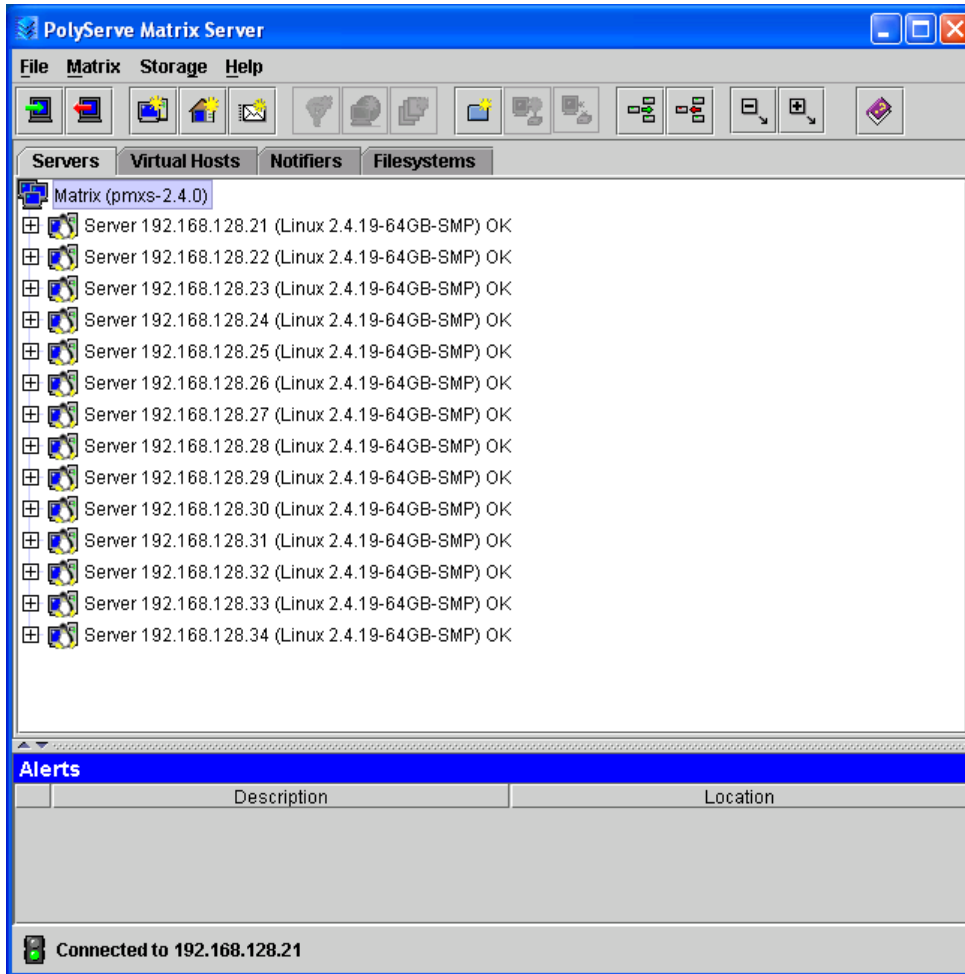


Figure 7. Matrix Server Management Console

Figure 8 presents another view of the Matrix Server console, showing its drill-down capability for obtaining the status of filesystems with a clusterwide view.

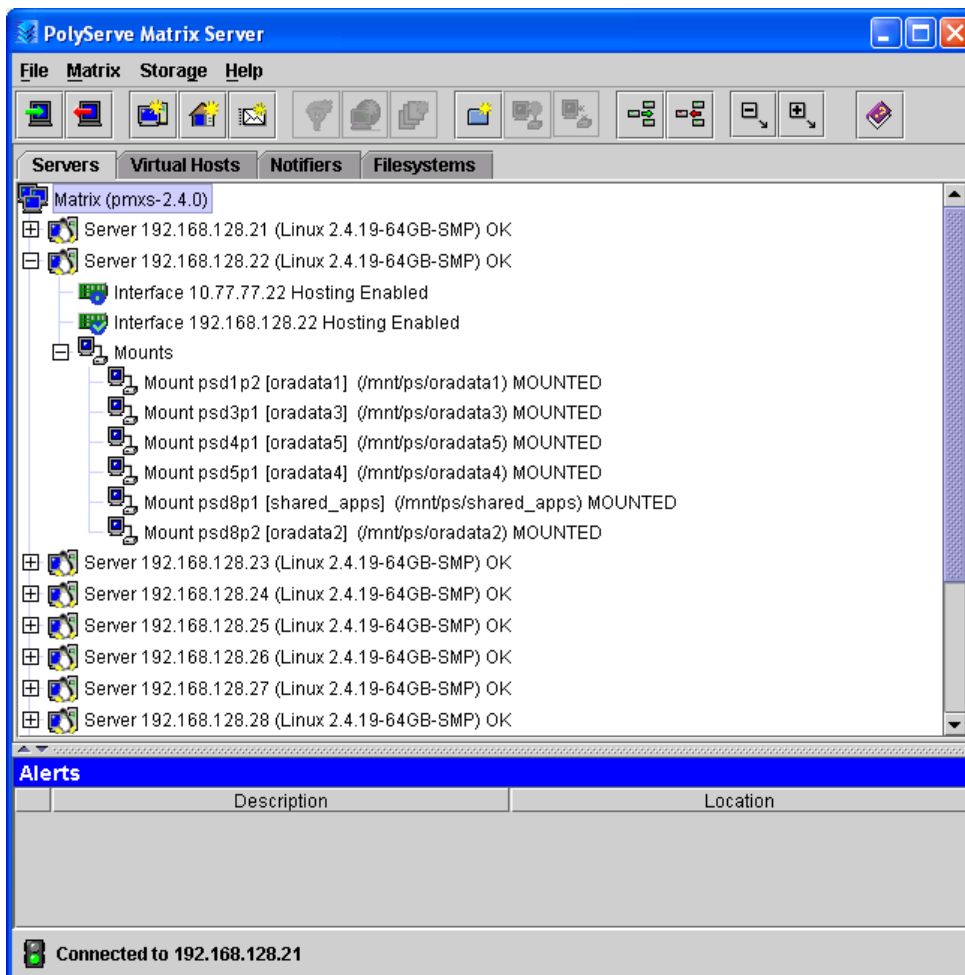


Figure 8. Drill-down on the Matrix Server Management Console

Proof of Concept

State-of-the-art and robust technologies were key to creating a suitable test system to prove the Flexible Database Cluster architecture. Figure 9 shows the cluster system components used for the FDC proof of concept.

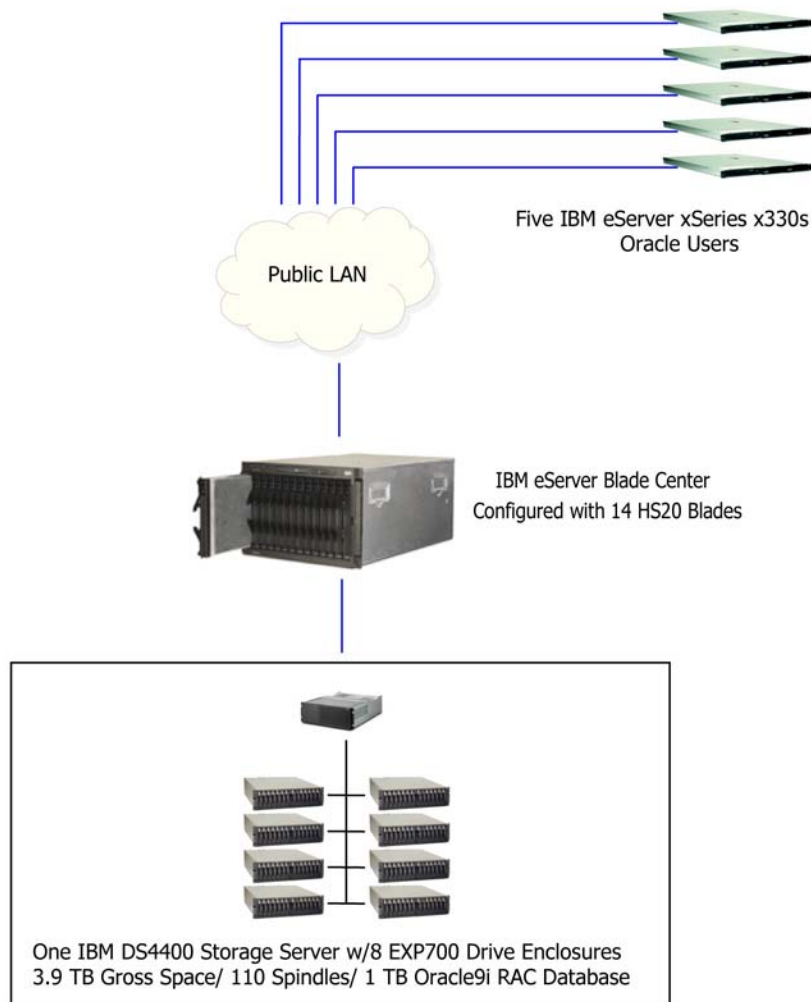


Figure 9. The proof-of-concept Flexible Database Cluster

Overview of the IBM eServer BladeCenter

To support the basic computing infrastructure needed by the Flexible Database Cluster proof of concept, it was important to choose a hardware platform that would showcase flexibility and manageability. The IBM eServer BladeCenter provides both of these attributes.

With emerging blade technologies enabling customers to reverse their “server sprawl” and collapse the complexity of their distributed IT infrastructure, we felt that this proof point would be an ideal opportunity to showcase this technology in a database cluster environment. Blades also deliver better management software, have less cable snarl, more expansion possibilities and smaller footprint requirements.

With those features in mind, it is easy to see why the IBM eServer BladeCenter, with its advantage of being rack-optimized and high-density in an innovative 7U form-factor design, was a perfect choice for deploying the Oracle9i RAC installation. The IBM eServer BladeCenter chassis accommodates up to 14 hot-swap 2-way Intel Xeon™ DP processor-based blade servers and integrates within the chassis such key infrastructure components as Layer 2-7 GB Ethernet Switching, SAN switching and centralized management tools.

With the need to connect large amounts of highly available disk storage to our BladeCenter, the storage subsystem was designed around the IBM TotalStorage DS4400⁸ Storage Server. The DS4400 is a RAID storage subsystem that contains Fibre Channel (FC) interfaces to connect both the host systems and the disk drive enclosures. With its dual 2Gbps controllers and high-availability design, the DS4400 delivers the necessary throughput to support this high-end proof point.

IBM eServer BladeCenter Chassis

The IBM eServer Blade Center chassis can accommodate up to 14 blade servers in its 7U form factor. Resources are shared among all of the blades and include power, switch, management and blower modules. The chassis provides high-speed I/O capabilities for all of the modules, thereby reducing the amount of cabling required in the datacenter. The Management Module, through remote access, allows the control of components in the enclosure.

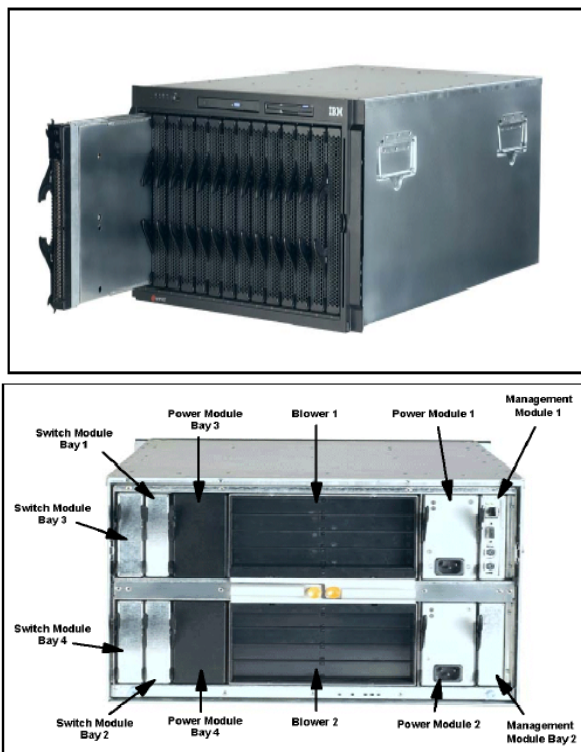


Figure 10. Front and rear views of the BladeCenter chassis

The IBM eServer BladeCenter chassis was configured as follows:

- Standard 48X CD-ROM and 1.44MB floppy accessible from all blades in the Media Tray.

⁸ The IBM TotalStorage FAST products were renamed in September 2004. The FAST700 is now called DS4400; FAST900 is called DS4500. The family is referred to as the DS4000 series.

- Management Module. The center for systems management on the BladeCenter, the Management Module is responsible for monitoring all components in the BladeCenter as well as each individual blade. It has the capability of detecting the condition and state of any of the installed components.
- Two additional 1200-watt hot-swap power modules (two are standard) were required to power blade slots 7-14. Installed as pairs, the power modules provide redundancy and power for robust configurations.
- Two 4-port Ethernet Switch Modules. Although not standard on the BladeCenter unit, the modules were necessary to provide the interconnectivity between the blades and Management Module and the external network. The module is a fully functional Ethernet switch with four external gigabit ports, two internal 10/100 links to the Management Module and 14 internal gigabit links to the blades. Two Ethernet Switch Modules were used in this proof point to support access to the external, public network (eth0) and the internal, private interconnect traffic (eth1).
- One 2-port Fibre Channel Switch Module. With the Fibre Channel Expansion Card in each blade server, the optional 2-port Fibre Channel Switch Module completed the required Fibre Channel connectivity to the SAN. Each port is capable of supporting transmission speeds of up to 2 Gbps after auto-negotiating with the DS4400 Storage Server.
- 14 IBM eServer BladeCenter HS20 blades.

IBM eServer BladeCenter HS20 Blade Server

The BladeCenter HS20 blades are high-throughput, two-way SMP-capable Xeon processor-based blade servers. With support for up to 8GB PC2100 ECC DDR Chipkill™ SDRAM and processor speeds of 2.4GHz to 3.2GHz, these blade servers are highly scalable. An integrated service processor on each blade server enables communication with the BladeCenter Management Module for remote control of server tasks. Also integrated on the HS20 are two Ethernet controllers that can be configured for either fault-tolerance or increased throughput through adapter teaming. With blade server expansion card options such as Myrinet Cluster, Gigabit Ethernet, Fibre Channel and support for both EIDE and SCSI drives, a blade can be tuned to create customized solutions that match application needs.

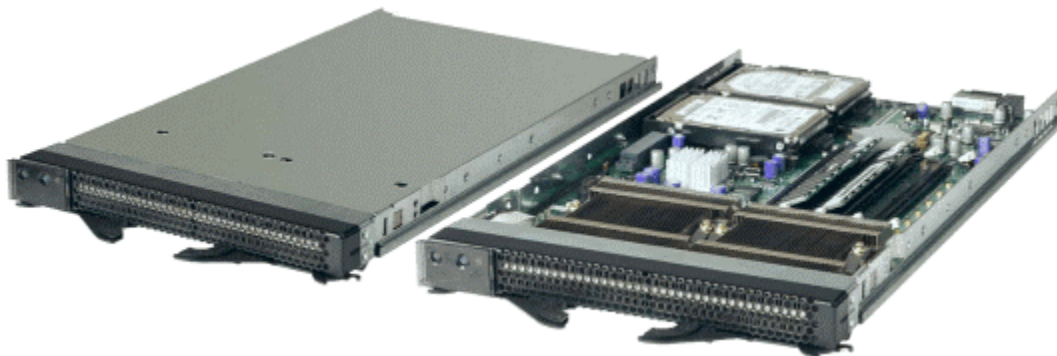


Figure 11. View of an HS20 Blade Server

For the proof point, each blade server was configured with:

- 2 x 2.4GHz Xeon processors
- 2GB PC2100 DDR ECC Chipkill memory
- 40GB IDE drive
- Integrated dual Broadcom Gigabit Ethernet controllers

- Fibre Channel Expansion card
- SUSE Linux Enterprise Server 8 (Service Pack 1)

Managing the BladeCenter

As IT environments become more complex to manage and support, it is important that there be tools and processes that allow the end user to track and maintain the infrastructure. To reduce the sometimes time-consuming task of managing high-density computing environments, the IBM eServer BladeCenter has a built-in, Web-based GUI⁹ that allows remote access to the BladeCenter to remotely power on/off blades and manage I/O modules. Access is gained through a standard Ethernet port and a standard Web browser. The BladeCenter's main menu has four main sections: System Monitors, Blade Tasks, I/O Module, and Management Module Control.

The Monitors menu allows the user to view status, settings and other information for each of the key components configured in the IBM eServer BladeCenter. The displayed information includes:

- System Status of blade servers, I/O modules, management modules and power modules
- Event Log
- Front Panel and Blade Server LEDs
- Vital Product Data for blades, I/O module and Management Module

The screenshot shows the Management Module web interface in a Microsoft Internet Explorer browser window. The address bar shows <http://10.77.77.125/private/main.ssi>. The main content area displays the 'Management Module' for 'Bay 1: WMN315828724'. A left-hand navigation menu includes sections like Monitors, Blade Tasks, I/O Module Tasks, and MM Control. The main table shows the status of 14 blades. Below this table, there is a section for 'I/O Modules' with a table showing details for three modules.

Bay	Status	Name	Pwr	Owner**		Network		WOL*	Local Control			BSE*
				KVM	MT*	Onboard	Card		Pwr	KVM	MT*	
1	●	SN#K10UJ34Y11U	On			Eth	Fib ...	On	X	X	X	
2	●	SN#K10UJ3511CS	On			Eth	Fib ...	On	X	X	X	
3	●	SN#K10UJ35114F	On			Eth	Fib ...	On	X	X	X	
4	●	SN#K10UJ35112J	On			Eth	Fib ...	On	X	X	X	
5	●	SN#K10UJ35318C	On			Eth	Fib ...	On	X	X	X	
6	●	SN#K10UJ35110V	On			Eth	Fib ...	On	X	X	X	
7	●	SN#K10UJ34Y1DH	On			Eth	Fib ...	On	X	X	X	
8	●	SN#K10UJ35110F	On			Eth	Fib ...	On	X	X	X	
9	●	SN#K10UJ33418Y	On			Eth	Fib ...	On	X	X	X	
10	●	BLADE#03	On	X		Eth	Fib ...	On	X	X	X	
11	●	BLADE#02	On		X	Eth	Fib ...	On	X	X	X	
12	●	BLADE#06	On			Eth	Fib ...	On	X	X	X	
13	●	BLADE#07	On			Eth	Fib ...	On	X	X	X	
14	●	BLADE#05	On			Eth	Fib ...	On	X	X	X	

* MT = Media Tray (CD/Floppy/USB) , WOL = Wake on LAN , BSE = Blade Storage Expansion
 ** You can change the KVM and Media Tray ownership on the Remote Control panel (under Blade Tasks).

Bay	Status	Type*	MAC Address	IP Address	Pwr	Details
1	●	Ethernet SM	00:05:5D:9C:95:3C	10.77.77.127	On	POST results available: FF: Module completed POS
2	●	Ethernet SM	00:05:5D:9C:8F:B0	10.77.77.128	On	POST results available: FF: Module completed POS
3	●	Fibre SM	00:C0:DD:01:F2:CF	10.77.77.129	On	POST results available: FF: Module completed POS

Figure 12. Portion of the System Status summary window

⁹ The IBM eServer BladeCenter is also tightly integrated to the IBM Director V4 systems management tool and the Rapid Deployment Manager; however, these system management products are beyond the scope of this paper. For additional information about these products, visit www.ibm.com.

The settings for each blade server can be configured and controlled via the Blade Tasks section of the menu. The following tasks can be performed:

- Power/restart of individual or all blade servers
- Remote control of an individual blade to associate the media tray ownership
- Updates of firmware
- Configuration for each blade for KVM control, media tray control, Wake on LAN and boot sequence

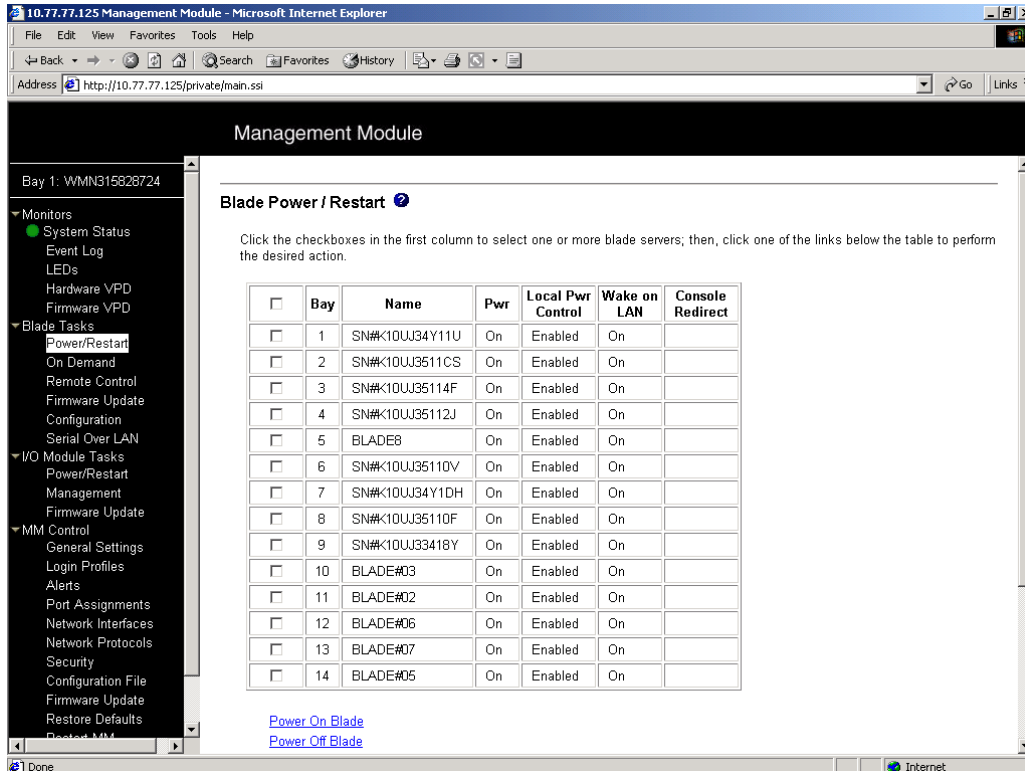


Figure 13. Portion of Blade Task menu selection

The heading of the I/O Module section of the menu depends on what optional switch modules are installed on the BladeCenter. In this particular installation there were two gigabit Ethernet switch modules and two Fibre Channel switch modules. The heading was displayed as "I/O Module Tasks" and the menu allowed the following:

- Power/restart of modules
- Individual switch management setting of IP network addresses and a drill-down for Advanced Management into each module to further configure the switch modules or to generate a telnet session/Web-based GUI to monitor and control the device
- Firmware upgrade capabilities

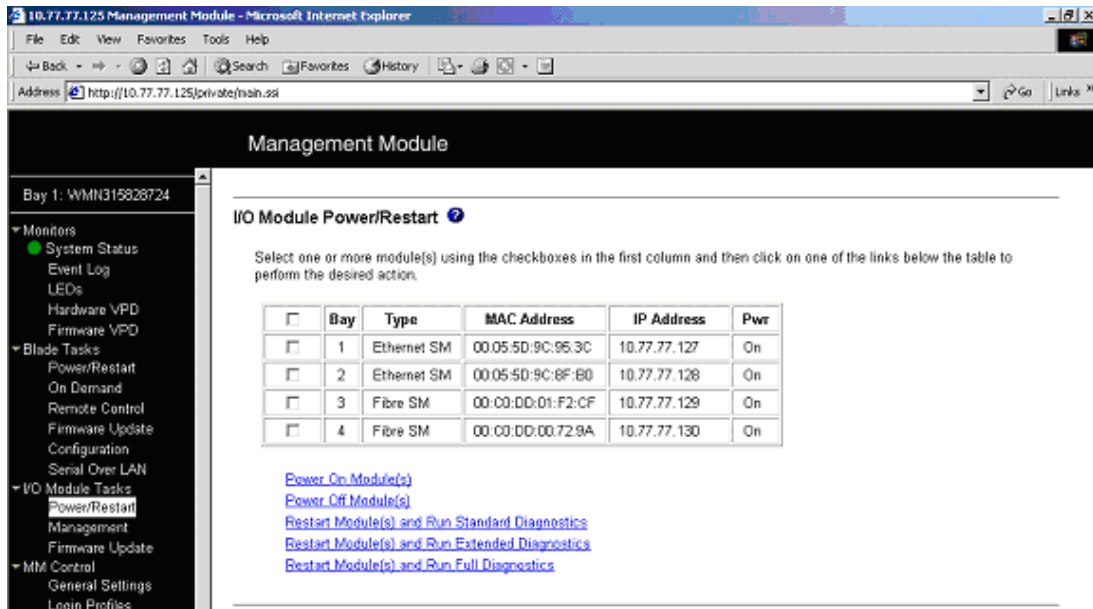


Figure 14. Partial display of I/O Module Tasks

The final module controlled through the Web-based system management GUI is the BladeCenter Management Module. The tasks include:

- General module settings such as name, date and time
- The ability to manage login profiles through ID and password control
- The ability to set alert levels and target users to receive notification
- Port assignments
- Network interfaces
- Network protocols
- Security
- Configuration file management with backup and restore options
- The ability to restore defaults
- The ability to restart the Management Module

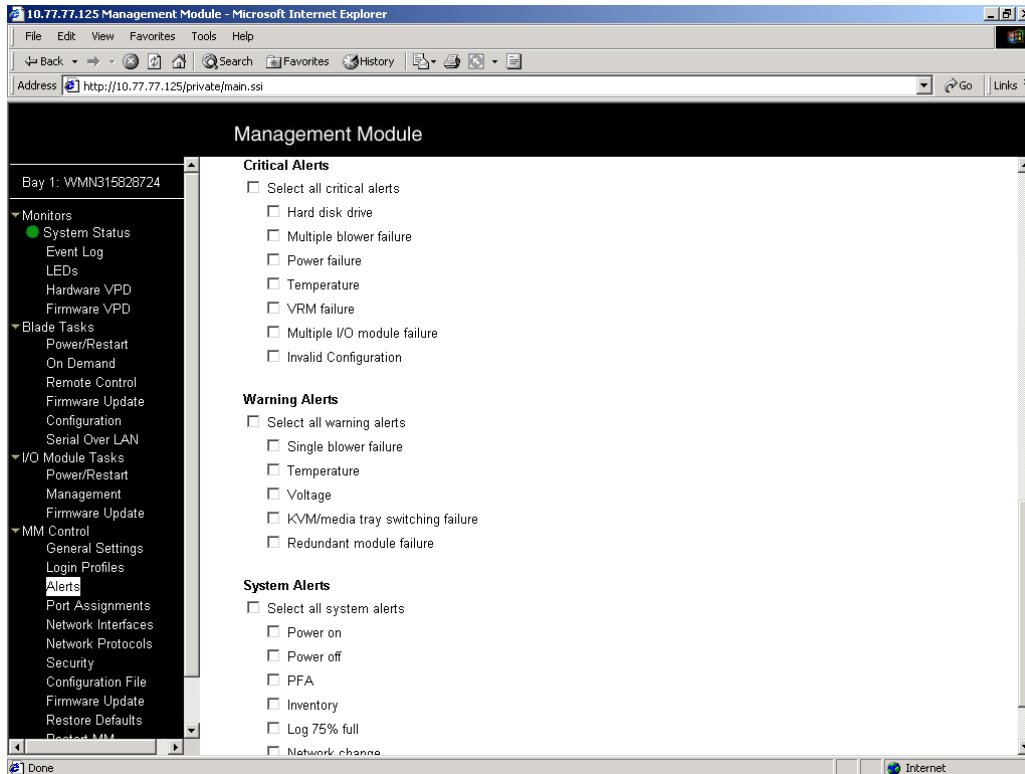


Figure 15. Sampling of alerts that can be monitored

Customizing BladeCenter I/O Modules to Match Application Needs

One of the strengths of a BladeCenter deployment is the flexibility to add modules to enable customized solutions that match application needs. From a physical perspective, adding these modules inside the chassis reduces the complexity of the external infrastructure by reducing cabling, power and space requirements. From a management perspective, complexity is reduced by having a centrally administered installation. Current I/O module options for the IBM eServer BladeCenter include:

- Optical Pass-thru Module allowing unswitched, unblocked network connections to each blade server
- 4-port Gigabit Ethernet Switch Module providing high speed Ethernet connections between each blade server and the outside network environment
- Nortel Networks Layer 2-7 GB Ethernet Switch Module integrating advanced Ethernet functionality into the chassis
- 2-port Fibre Channel Switch Module supporting two FC uplinks at transmission rates up to 2Gbps

The requirements for standard Gigabit Ethernet connectivity and connection to a fault-resilient Fibre Channel SAN for the proof-point were easily met by literally sliding in the 4-port Gigabit Ethernet Switch Module and the 2-port Fibre Channel Switch Module. These modules were immediately recognized by the BladeCenter Management Module and were available for configuration.

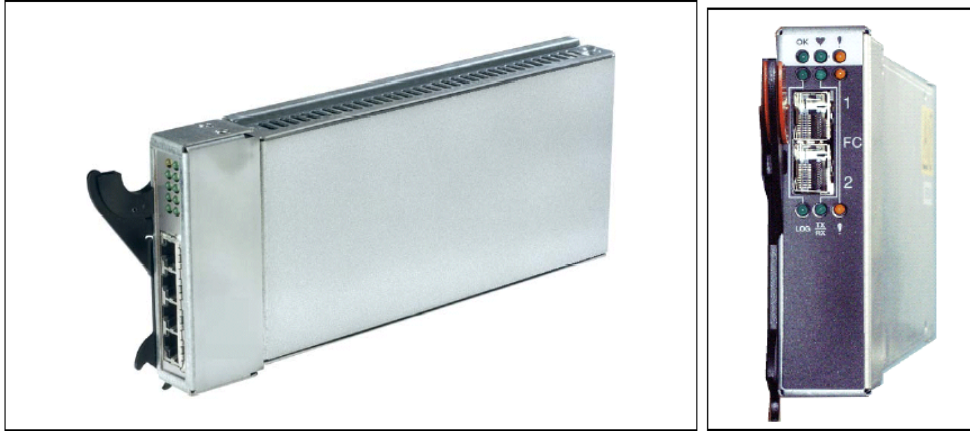


Figure 16. View of 4-port Ethernet Switch Module and 2-port Fibre Channel Switch Module

Managing the BladeCenter 4-port Gigabit Ethernet Switch Module

The Ethernet Switch Module (ESM) can be managed through a telnet session or a Web interface, which is the preferred method. Each module on the BladeCenter is assigned a TCP/IP address that corresponds to the module slot in which the switch is installed. This makes it very easy to start a telnet session directly or to access the Web interface at the assigned address. Through the Web interface, the ability to manage and monitor performance of the Ethernet switch module is only several mouse clicks away.

From the main BladeCenter Management menu, selecting I/O Module Tasks -> Management and then choosing the Advanced Management button for the module of choice provides the end user the capability to drill down into a Web-based, graphical tool for monitoring the 4-port Gigabit switch. For ease of navigation, the menu is broken into four distinct categories:

- Switch configuration for port settings, VLANs, link aggregation, and so on
- Remote management setup
- Network monitoring to display port utilization statistics
- Maintenance tasks such as upgrading firmware, downloading configuration, and restarting modules

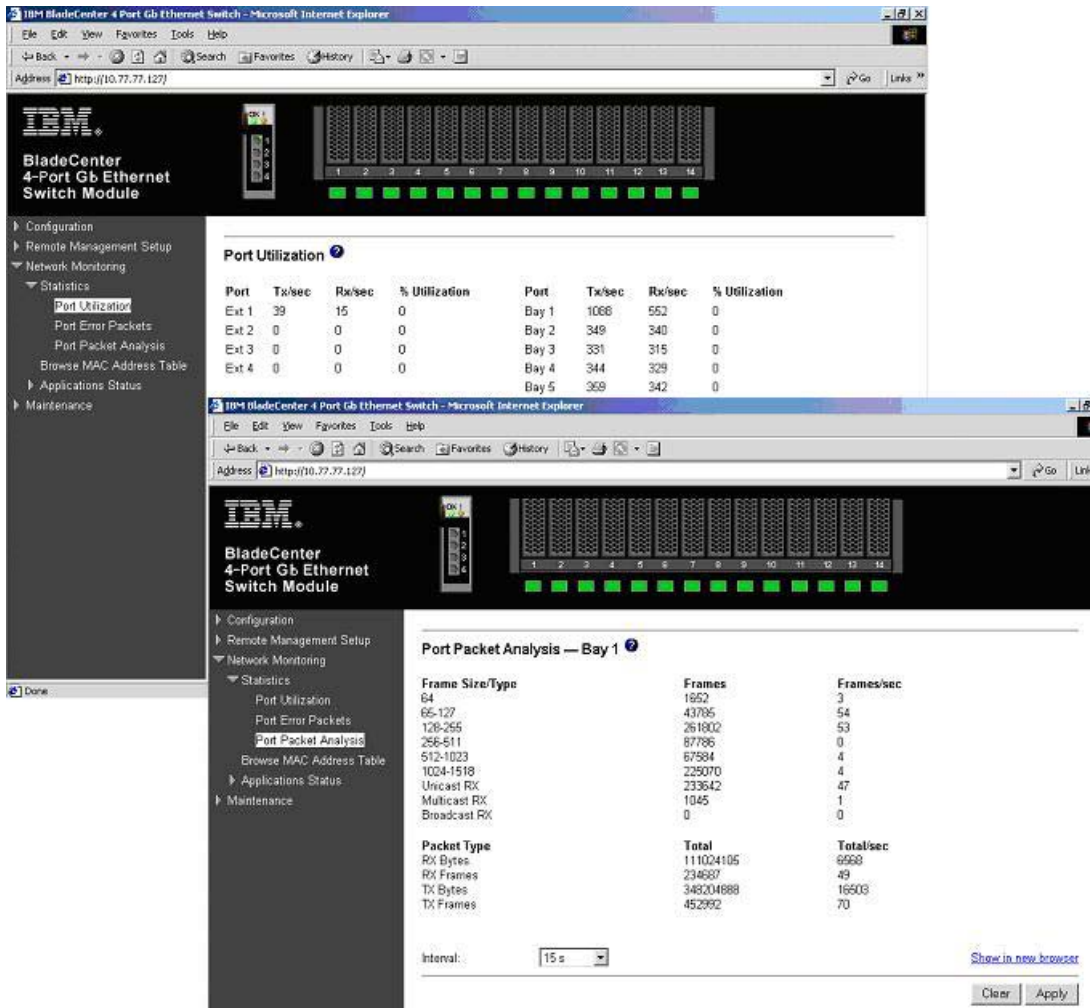


Figure 17. Multiple displays of performance information for the 4-port GB Ethernet Switch Module

Managing the BladeCenter Fibre Channel Switch Module

The configuration and management of the BladeCenter Fibre Channel Switch Module is facilitated with the SAN utility tool. This tool is a Java-based graphical user interface that allows viewing and configuration of ports, zoning and network setup, and troubleshooting through diagnostic functionalities. The SAN utility can be installed and executed from one of the blade servers or from an external monitoring system. It is supported in both Windows and Linux environments. The SAN utility provides a graphical view of the Fibre Channel subsystem and can be used to view, monitor or change network, switch module, and port configuration for one or more fabrics concurrently. There are two basic views available with the SAN utility: Topology Display and Faceplate Display.

The Topology Display shows all switches that are able to communicate and all connections between switches. (A *fabric* is defined as one or more connected switches.) The graphic window of the topology display provides status information for switches, interswitch links, and the Ethernet connection.

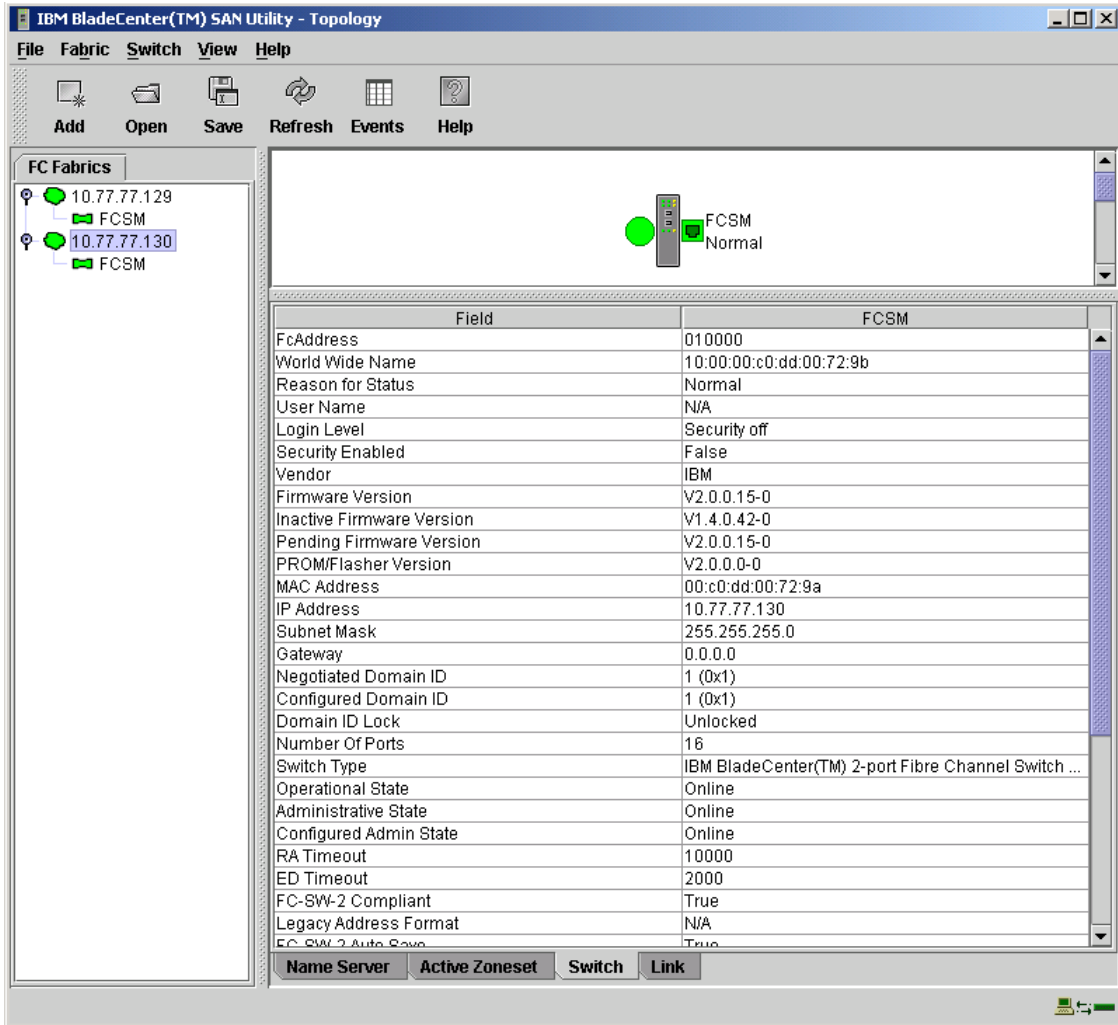


Figure 18. SAN Utility Topology Display

The BladeCenter Fibre Channel Switch Module information can also be viewed through the Faceplate Display, which shows the front of the switch and its related information: switch name, switch operational state, and port status.

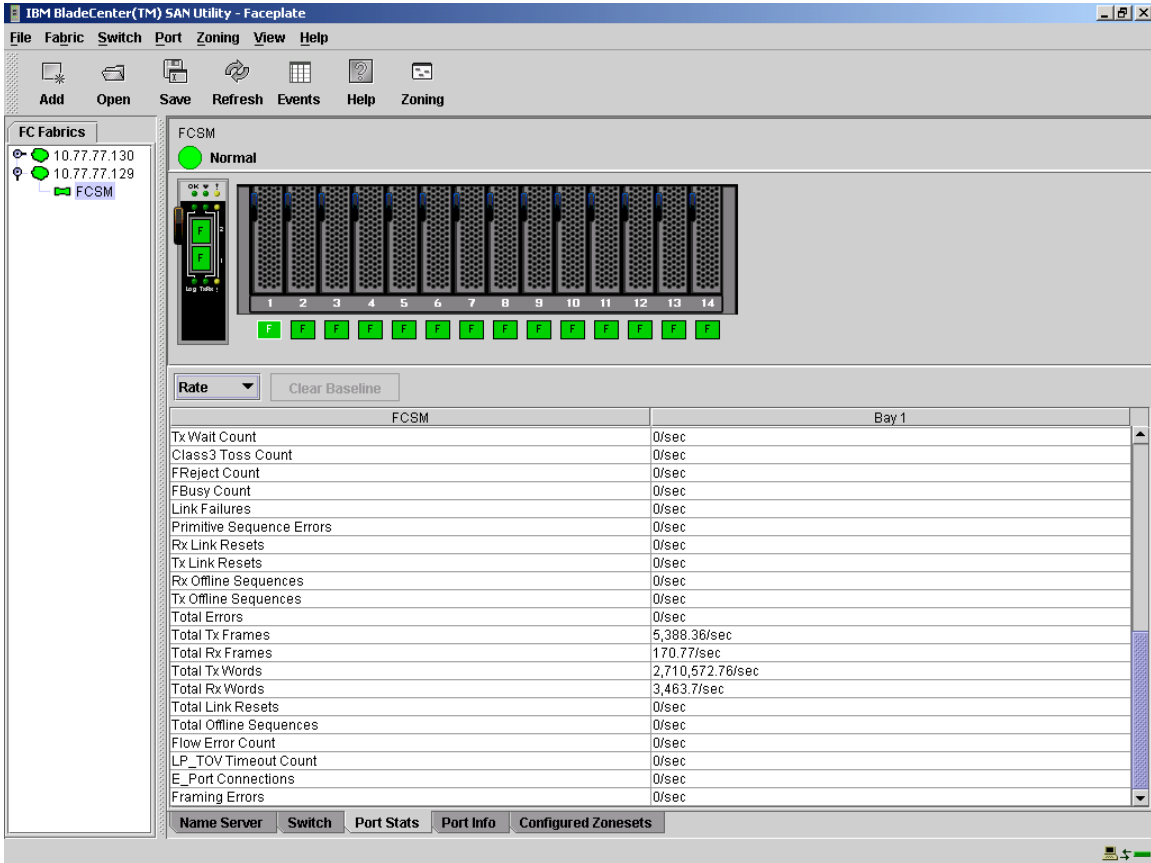


Figure 19. Faceplate display from the SAN Utility

The IBM eServer BladeCenter also includes a management tool called Fabric View. This application provides a method to visually monitor realtime traffic performance for each port on a switch. The graphs can be set up to display either Kbytes/sec or number of frames/sec.



Figure 20. Fabric View

Storage Subsystem

The IBM TotalStorage DS 4000 series is designed to support the large and growing data storage requirements of business-critical applications. The DS 4000 series are RAID controller devices that contain Fibre Channel (FC) interfaces to connect the host systems and the disk drive enclosures. The DS4400 used in this proof point has controllers that use the 2Gbps Fibre Channel standard on both the host side and the drive side. The DS4400 can support up to 224 FC disks. To avoid single points of failure, it also supports high-availability features such as hot-swap RAID controllers, two dual redundant FC disk loops, write cache mirroring, redundant hot-swap power supplies, fans and dual AC line cords.

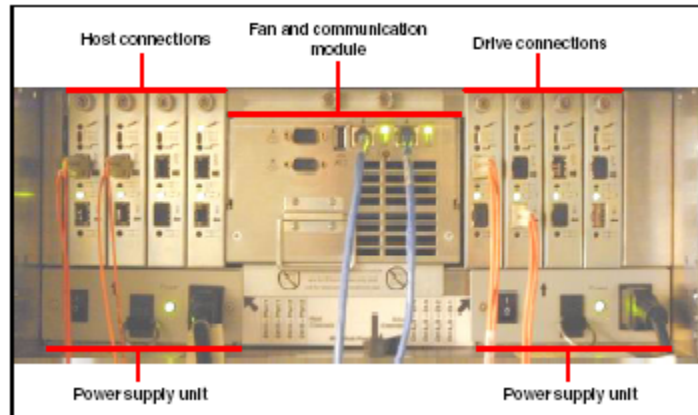
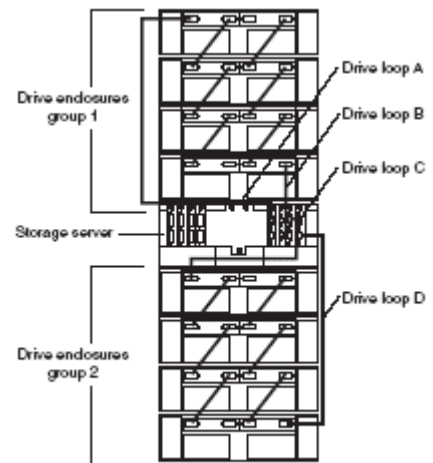


Figure 21. Rear view of the DS4400

For this implementation, the DS4400 had eight DS4000 EXP700 drive enclosures attached with a total of 110 15K rpm 36.4GB drives. The Storage Manager Software was used to automatically lay out each of the arrays across the multiple controllers and drive loops. The DS4000 allocation heuristic does a reasonable job of distributing I/O traffic across available resources so there was no need to manually define the arrays. Because the DS4400 can support two redundant drive loops, the drive enclosures were set up to take advantage of this redundancy. If one data path fails, the controller uses the other data path to maintain the connection to the drive group.

This back view of the storage server and drive enclosures shows two redundant drive loops. Loop A and Loop B make up one redundant pair of drive loops. Loop C and Loop D make up a second redundant pair.



The 110 drives were automatically distributed across five arrays of 22 drives each with a RAID-1 configuration and a 256K stripe size.

Managing the Storage System

The IBM TotalStorage DS4000 Storage Manager software simplifies management of extensive SAN installations. The Storage Manager software allows you to configure arrays and logical drives, to assign logical drives into storage partitions, to replace and rebuild failed disk drives, to expand the size of arrays and logical volumes, and to convert from one RAID level to another. It also gives you the ability to monitor performance. In addition to array/drive level control, the DS4000 Storage Manager can also be used to update the firmware and NVSRAM on the controllers.

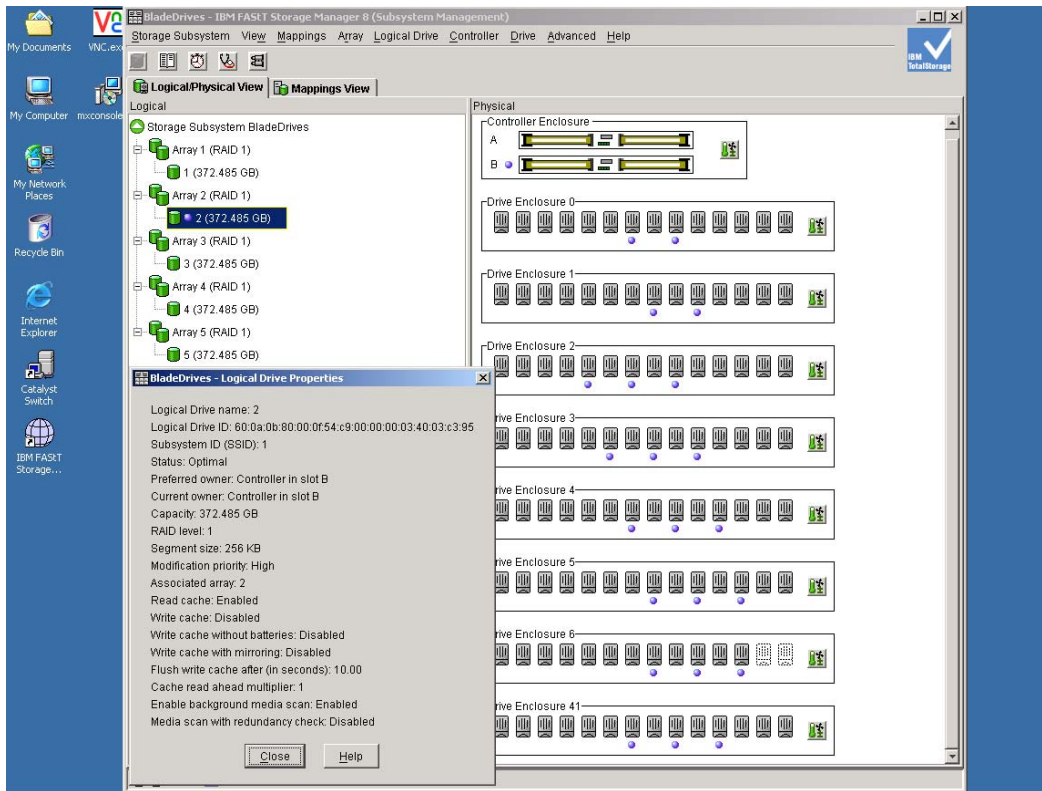


Figure 22. IBM DS Storage Manager 8 (Subsystem Management)

The DS4000 Storage Manager is supported in both Windows and Linux environments and provides two methods for managing storage subsystems:

- Host-agent (in-band) management through the Fibre Channel I/O path to the host
- Direct (out-of-band) management over the network

The performance monitor data can be used to make storage subsystem tuning decisions. There are many settings in the DS4400 that can have a large impact on performance. These include cache parameters, controller ownership, segment size, RAID levels, logical drive modification priority, and remote volume mirroring. To assist with overall storage tuning, the DS4000 Storage Manager has a built-in performance monitor that displays statistics for Total I/Os, Read Percentage, Cache Hit Percentage, Current KB/s, Maximum KB/s, Current I/Os and Maximum I/Os. The successful tuning of the DS4400, like other server components, is dependent on finding the right balance of availability and high performance. The performance monitor in the DS4000 Storage Manager is just one of the tools available to help end users improve SAN performance.

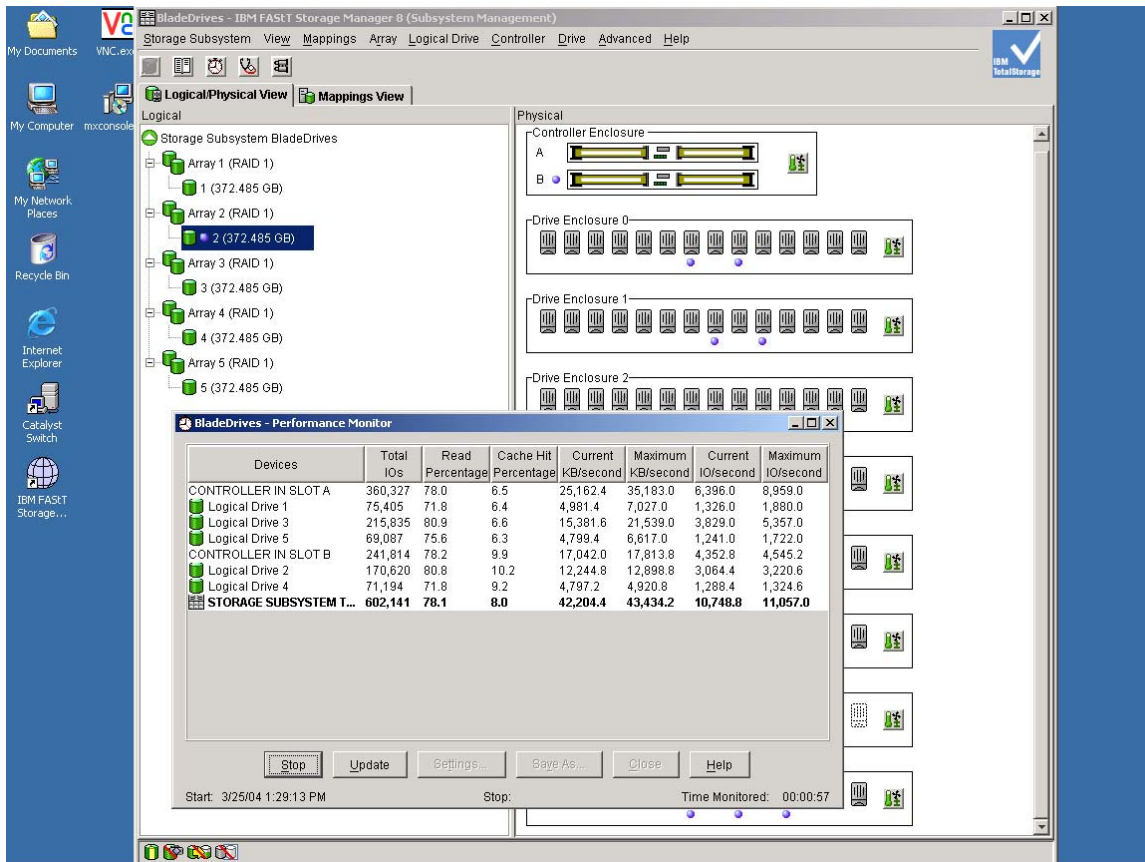


Figure 23. Performance Monitor

Overview of Databases

To test the Flexible Database Cluster architecture, three databases were created in the Matrix Server cluster filesystem using Oracle9i Release 2 version 9.2.0.4. The main databases were called OLTP and DSS. The third, smaller database was called DEV.

These databases were not intended to convey best practice for operations or performance—with the notable exception of the use of Oracle Managed Files (OMF). OMF is Oracle9i file management simplification, which deserves close consideration in any deployment scenario. The databases were intended only to be of realistic size and structure and usable to test the functionality and added value of FDC architecture.

OLTP Database (PROD)

The OLTP database schema is based on an order-entry system similar to but not compliant with that defined in the TPC-C¹⁰ specification. The default Oracle block size was 4KB with an additional buffer pool to support 16KB blocks from the Card table described below. At a high level, the database schema contains the following application tables:

Customers. The database contains more than 159 million customer rows in the “customer” table. This table contains customer-centric data such as a unique customer identifier, mailing address,

¹⁰ The Flexible Database Cluster proof of concept was in no fashion intended to comply with any specification of the Transaction Processing Performance Council. For more information on TPC-C, visit www.tpc.org.

e-mail contact information and so on. The customer table is indexed with a unique index on the custid column and a non-unique index on the name column.

Orders. The database contains an orders table with more than 200 million rows of data. The orders table has a unique composite index on the custid and ordid columns.

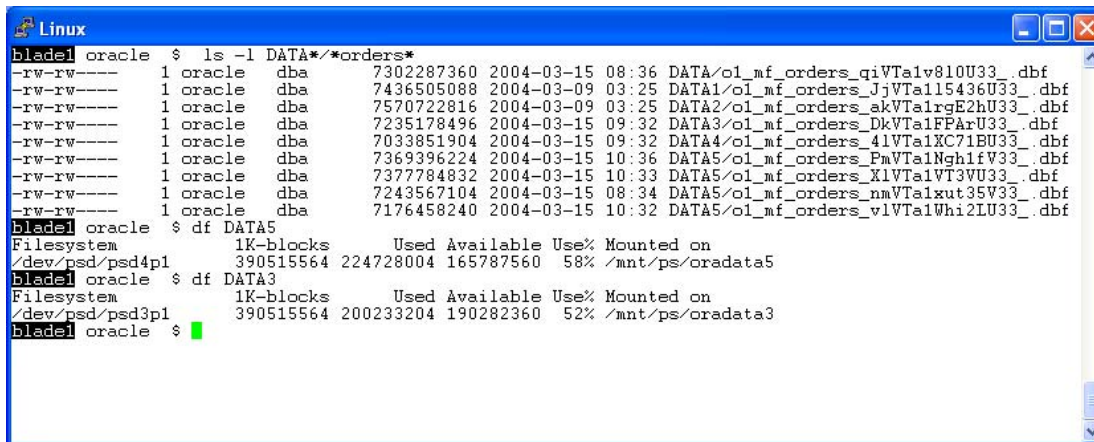
Line Items. Simulating a customer base with complex transactions, the Line Item table contains as many as 15 line items per order for a total of nearly 1.7 billion rows. The item table has a three-way unique composite index on custid, ordid and itemid. Creating this index was one of the timed Parallel Query tasks described in the DSS performance section later in this paper.

Product. This table describes products available to order. Along with such attributes as price and description, there are up to 140 characters available for a detailed product description. There are more than 140 million products. The product table is indexed with a unique index on its prodid column.

Warehouse. This table maintains product levels at the various warehouse locations as well as detailed information about warehouses. This table is crucial in order fulfillment. The warehouse table is indexed with a unique composite index of two columns.

Card. The card table holds credit card transaction detail. This table is also used by the DSS database instances as a transportable tablespace. The Card table is created in a tablespace of 16KB Oracle blocks. The card table has exactly 2 billion rows of data. The table is partitioned into 10 tablespaces by hash value on the vendor_id column of which there are roughly 1 million distinct values.

The database was created using the simplified Oracle Managed Files method. The FDC proof of concept was meant to prove manageability in a complex environment, so it made little sense to use complex tablespace definitions. In fact, OMF works extremely well. Tablespaces created by OMF are optimized for the norm; however, specialized tuning may still be required in certain cases. To help illustrate the simplicity of this type of file management, Figure 24 shows a simple `ls(1)` command that displays all of the data files associated with the ORDERS tablespace. The data files are named automatically by OMF, which enables unique naming.



```
bladel oracle $ ls -l DATA*/orders*
-rw-rw---- 1 oracle dba 7302287360 2004-03-15 08:36 DATA/ol_mf_orders_qiVTalv810U33_.dbf
-rw-rw---- 1 oracle dba 7436505088 2004-03-09 03:25 DATA1/ol_mf_orders_JjVTa115436U33_.dbf
-rw-rw---- 1 oracle dba 7570722816 2004-03-09 03:25 DATA2/ol_mf_orders_akVTalrgE2hU33_.dbf
-rw-rw---- 1 oracle dba 7235178496 2004-03-15 09:32 DATA3/ol_mf_orders_DkVTa1FFarU33_.dbf
-rw-rw---- 1 oracle dba 7033851904 2004-03-15 09:32 DATA4/ol_mf_orders_4lVTa1XC71BU33_.dbf
-rw-rw---- 1 oracle dba 7369396224 2004-03-15 10:36 DATA5/ol_mf_orders_PmVTa1Ngh1fV33_.dbf
-rw-rw---- 1 oracle dba 7377784832 2004-03-15 10:33 DATA5/ol_mf_orders_XlVTa1VT3VU33_.dbf
-rw-rw---- 1 oracle dba 7243567104 2004-03-15 08:34 DATA5/ol_mf_orders_nmVTa1xut35V33_.dbf
-rw-rw---- 1 oracle dba 7176458240 2004-03-15 10:32 DATA5/ol_mf_orders_vlVTa1Whi2LU33_.dbf
bladel oracle $ df DATA5
Filesystem      1K-blocks      Used Available Use% Mounted on
/dev/psd/psd4p1 390515564 224728004 165787560  58% /mnt/ps/oradata5
bladel oracle $ df DATA3
Filesystem      1K-blocks      Used Available Use% Mounted on
/dev/psd/psd3p1 390515564 200233204 190282360  52% /mnt/ps/oradata3
bladel oracle $
```

Figure 24. `ls(1)` and `df(1)` commands

Since all databases were created in the PolyServe Matrix Server cluster filesystem, it is quite simple to ascertain the amount of free space available for the databases. Figure 24 also shows a simple `df` command that lists free and used space for some of the files in the databases.

All database files were created in the `oradata[1-5]` filesystems. The `mount(1)` command in Figure 25 shows that they are all mounted with the Matrix Server `dboptimized` mount option, which provides cache-coherent direct I/O. This means that database accesses by Oracle are rendered direct, and all other tools (e.g., backup tools) can also get direct I/O without recompiling with code changes to perform the `open(2)` system call with the `O_DIRECT` flag. With the `dboptimized` mount

option, if I/O can be rendered direct, it will be. I/O that cannot be rendered direct is serviced through the buffered I/O path, and all the while cache coherency is maintained.

```

bladel1 oracle $ df
Filesystem            1K-blocks      Used Available Use% Mounted on
/dev/hda2              38032716  12982080  25050636   35% /
shmfs                 1033972      0    1033972    0% /dev/shm
/dev/psd/psd8p1       6264216    5447748    816468   87% /mnt/ps/shared_apps
/dev/psd/psd1p2      389712392  243283936  146428456   63% /mnt/ps/oradata1
/dev/psd/psd8p2      384218500  188128744  196089756   49% /mnt/ps/oradata2
/dev/psd/psd3p1      390515564  200233204  190282360   52% /mnt/ps/oradata3
/dev/psd/psd5p1      390515564  195607524  194908040   51% /mnt/ps/oradata4
/dev/psd/psd4p1      390515564  224728004  165787560   58% /mnt/ps/oradata5
bladel1 oracle $ mount
/dev/hda2 on / type reiserfs (rw)
proc on /proc type proc (rw)
devpts on /dev/pts type devpts (rw,mode=0620,gid=5)
shmfs on /dev/shm type shm (rw)
usbdevfs on /proc/bus/usb type usbdevfs (rw)
/dev/psd/psd8p1 on /mnt/ps/shared_apps type psfs (rw,logtotty,shared)
/dev/psd/psd1p2 on /mnt/ps/oradata1 type psfs (rw,logtotty,dboptimize)
/dev/psd/psd8p2 on /mnt/ps/oradata2 type psfs (rw,logtotty,dboptimize)
/dev/psd/psd3p1 on /mnt/ps/oradata3 type psfs (rw,logtotty,dboptimize)
/dev/psd/psd5p1 on /mnt/ps/oradata4 type psfs (rw,logtotty,dboptimize)
/dev/psd/psd4p1 on /mnt/ps/oradata5 type psfs (rw,logtotty,dboptimize)
bladel1 oracle $

```

Figure 25. mount(1) options

To illustrate that the FDC proof of concept was much more than a typical benchmark, Figure 26 shows a query against the Oracle GV\$ virtual tables. The query indicates that the database instances had been active non-stop at one point during the testing and had amassed nearly 3 billion physical I/O transfers. The `io.sql` script also reveals that the database instances had collectively transferred roughly 11TB of data without rebooting.

To put this amount of I/O in perspective, even at the observed peaks of 14,000 IO/sec that the FDC demonstrated, it would take that I/O rate every second of every day for two and a half days to reach this amount. That is not a benchmark! In reality, this test was specifically set up to execute the OLTP test on all 14 blades for an entire weekend.

```

SQL> @now
LOCAL_TIME
-----
03/30/2004 11:41:45

SQL> !cat io.sql
select sum(PHYRDS) reads,sum(PHYBLKRD * 4 )/1024 readMB,
       sum(PHYWRTS) writes,sum(PHYBLKWRT * 4 )/1024 writeMB
from dba_data_files,gv$filestat
where dba_data_files.file_id = gv$filestat.file#;
REM exit;

SQL>
SQL> @io
      READS      READMB      WRITES      WRITEMB
-----
2332139171  9113542.17  541058580  2113524.43

SQL>

```

Figure 26. Query showing I/O activity

DSS Database: Decision Support

The DSS database was used to perform analytical queries about customer credit. The table used for this decision support was the CARD table from the OLTP schema described above. The card tablespace was simply set up as a transportable tablespace and accessed directly—without copy—by the DSS database. This is the power of a large cluster on a SAN. It becomes very efficient to take data from one database to another with no copies across a network.

DEV Database: Development

The DEV database is a simple insert engine designed to test scalability while inserting records 2KB in size. The database is approximately 10GB total. Only two threads are defined; therefore, only two instances can access this database at one time.

Workload Descriptions

The workloads chosen for the Flexible Database Cluster proof of concept were not as important as the fact that there were three of them. As stated earlier, the databases were called PROD, DSS and DEV. Following is a description of the type of processing each database sustained during the test. The goal was to have a realistic mix of processing running on the system while testing the manageability of FDC architecture.

OLTP Workload

Simulating an order-entry system, the application accessing the PROD database consists of connecting 100 users per node via the PROD service defined in the *tnsnames* service definition shown earlier. The nodes under test are evenly loaded due to the load balancing attribute of the PROD SQL*Net service. Each user cycles through a set of transactions described below. At the end of each transaction, the client process sleeps for a small random period of time to simulate human interaction. To that end, this testing is not the typical benchmark style where all processors are 100% utilized¹¹. Such a condition is not desirable in a datacenter scenario; therefore, testing manageability of a proposed architecture under those conditions was not deemed realistic.

A key attribute of this testing is that it was completely void of traditional cluster-aware tuning. Most cluster-centric database tests have possessed at least some form of application-level partitioning. For example, nearly all TPC-C benchmarks executed on a cluster use a method called “data-dependent request routing” at a bare minimum. This method uses a transaction monitor to route all requests for a given transaction to a node in the cluster provisioned for servicing requests that modify data within a certain key range. For instance, node 1 in the cluster accepts new order transaction requests only from customers with customer identifiers in the 1-1,000,000 range; node 2 services the 1,000,001-2,000,000 range; and so on.

The limitation of such partitioning schemes is that they require changes to the application. Applications should not have to change in order to scale horizontally in a clustered environment, and with Oracle9i Real Application Clusters, they don't. Oracle9i Real Application Clusters is a radical departure from typical clustered database technology. With its Cache Fusion Technology and shared disk architecture, “off-the-shelf” applications can fully exploit clustered systems—without cluster-centric tuning or application code modifications. To that end, the Flexible Database Cluster proof of concept used an application test that did not require any cluster-centric tuning.

¹¹ The value of traditional benchmarks is not being questioned. Hardware vendors need a fair playing field to establish the capability of their offerings.

The pseudo-users of the test application connect to any Oracle Instance in the cluster and execute transactions as if they were running on a legacy SMP system. In fact, this test application has been used in the past to test SMP scalability.

The transaction details are as follows:

Orders Query. This transaction accounts for 16% of the activity. It provides top-level detail on existing orders for the customer and provides such detail in a most-recent to least-recent order.

Customer Attribute Update. This transaction represents 26% of the workload. It offers the ability to update information such as the phone number, address, and credit card information.

Orders Report. This transaction differs from Orders Query in that it offers full order detail for a customer to include shipment status. This transaction is executed 4% of the time.

Product Update. This transaction occurs as the result of a change to a product description; 26% of all transactions are a Product Update.

New Items. This transaction accounts for 11% of the mix; it adds items into stock on hand.

New Orders. This transaction simulates taking an order from an existing customer for stock on hand and accounted for 17% of the total transactions.

The physical I/O mix for the workload was 79% reads and 21% writes. On average, each transaction has the following cost associated with it:

Oracle Statistics	Average per Transaction
SGA Logical Reads	46.9
SQL Executions	2.7
Physical I/O	8.2
Block Changes	5.9

The Oracle statistics show that the workload is quite formidable. We encourage comparisons of these statistics to those taken from production systems with similar CPU count. This is a very realistic workload.

DSS Workload

The DSS workload consisted of three tests which focused on data loading, sorting and index creation using the Parallel Query Option. The tests will be described in greater detail later in this paper.

DEV Workload

The DEV workload is simply a zero think time program that inserts 2K rows via pipe to SQL*Loader. The streams of loader processes execute on up to two nodes when DEV is being tested along with the other workloads.

Measurement Results

High Availability and Manageability

While Oracle9i RAC is known for scalability, it also offers unprecedented availability. When combined with the power of SQL*Net, RAC provides that there will almost always be an instance of a database to which to connect. The full capability of RAC for high availability is limited,

however, in small clustered environments. Once again the Flexible Database Cluster adds tremendous architectural and operational value.

Consider an application that is sized to require a 4-node cluster. If a node suffers hardware failure, the application is now serviced at no better than 75%. Until the hardware is either repaired or replaced, this condition will persist. Of course there is still 100% uptime for the application due to the power of RAC, but there would likely be users that can detect the performance hit related to the reduced node count.

Having a spare node fully loaded with Oracle¹² and ready to be cabled and booted is one form of protection. The question is whether it will have the right “personality” to replace the failed node. That is, with Oracle Cluster Management Services (OCMS), the *cmcfg.ora* parameter called *PrivateNodeNames* contains a list of hostnames or IP addresses that cannot be changed without rebooting OCMS on all nodes in the cluster. Of course all database instances must be down to reboot OCMS. Similarly, *tnsnames.ora*, *listener.ora*, and many other configuration files likely expect the replacement node to possess at a minimum the same IP address and hostname. While these issues are not insurmountable, they do tax the continuity of operations in the face of node failure. Every minute counts when your database is running at 75% capacity.

In contrast, this same application serviced by four nodes in a FDC environment can rapidly ramp back up to 100% capacity in light of not one, but potentially several concurrent hardware failures. We should never be so unlucky as to suffer multiple hardware failures, but it can happen and this is one of the reasons RAC is the right choice for mission-critical deployments.

During the FDC proof of concept, a test was set up to measure the added availability the architecture provides. Figure 27 has a time-line graphic that depicts the events that occurred during the test. In summary, the test consisted of the following:

- 2,000 users were connected to a 10-node cluster executing OLTP
- One of the 10 nodes was powered off
- All users connected to the nine remaining nodes maintained their connection to Oracle
- 71 seconds after the node was powered off, a replacement node was up with an instance of Oracle accepting connections

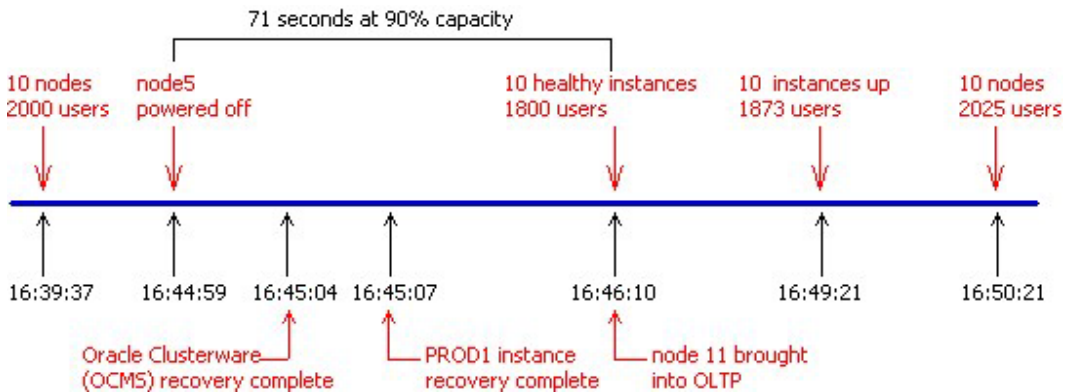
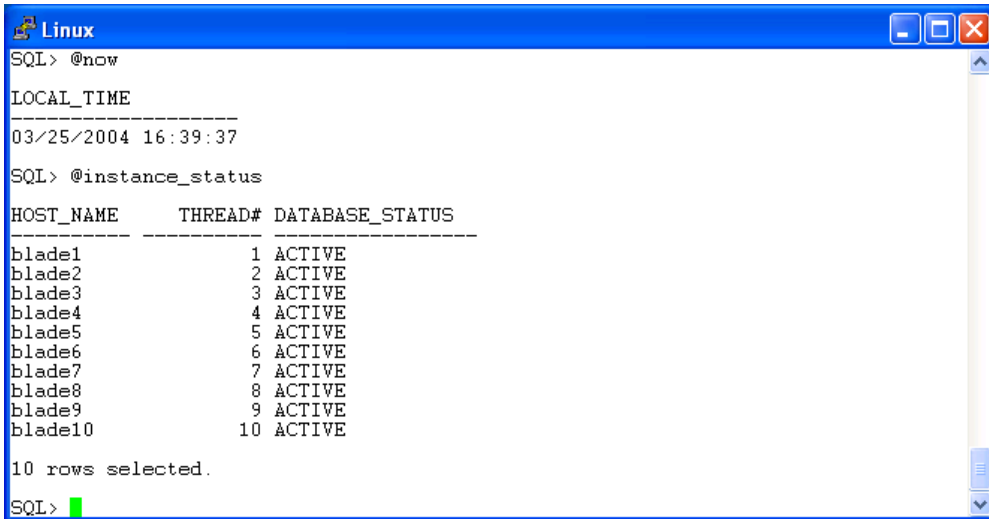


Figure 27. Availability time line

¹² A requirement in the absence of a Cluster Filesystem-based Shared Oracle Home such as that configured for the Flexible Database Cluster testing.

The OLTP workload was set up to execute at 200 users per node on blades 1-10. A light DSS database workload was executing on the remaining four blades. Figure 28 shows that at 16:39:37, the **instance_status.sql** script reported that there were 10 active instances on blades 1-10.



```
Linux
SQL> @now
LOCAL_TIME
-----
03/25/2004 16:39:37
SQL> @instance_status
HOST_NAME      THREAD#  DATABASE_STATUS
-----
blade1         1  ACTIVE
blade2         2  ACTIVE
blade3         3  ACTIVE
blade4         4  ACTIVE
blade5         5  ACTIVE
blade6         6  ACTIVE
blade7         7  ACTIVE
blade8         8  ACTIVE
blade9         9  ACTIVE
blade10        10 ACTIVE
10 rows selected.
SQL>
```

Figure 28. Active instances on blades 1-10

In Figure 29, the **users.sql** script shows that there were 200 user connections per node. Note that the script also counted 12 Oracle background processes per node.



```
Linux
SQL> @now
LOCAL_TIME
-----
03/25/2004 16:40:42
SQL> @users
MACHINE      COUNT(*)
-----
blade1       223
blade10      212
blade2       212
blade3       212
blade4       212
blade5       212
blade6       212
blade7       212
blade8       212
blade9       212
10 rows selected.
SQL>
```

Figure 29. users.sql script

The workload was allowed to come to a steady state executing transactions. After some time, blade5 was powered off via the BladeCenter management console to simulate a node failure. Figure 30 shows the Oracle Cluster Management Services (OCMS) *cm.log* file from blade1 logging the event that node 4 (counting from zero) was no longer responding at 16:44:59.

```

>WARNING: PollingThread(): node(4) missed(3) checkin(s), tid = PollingThread:51
26 file = nmmember.c, line = 846 {Thu Mar 25 16:44:56 2004 }^M
>WARNING: PollingThread(): node(4) missed(4) checkin(s), tid = PollingThread:51
26 file = nmmember.c, line = 846 {Thu Mar 25 16:44:57 2004 }^M
>WARNING: PollingThread(): node(4) missed(5) checkin(s), tid = PollingThread:51
26 file = nmmember.c, line = 846 {Thu Mar 25 16:44:58 2004 }^M
>WARNING: PollingThread(): node(4) missed(5) checkin(s), tid = PollingThread:51
26 file = nmmember.c, line = 846 {Thu Mar 25 16:44:59 2004 }^M
>WARNING: RecvMsg: socket closed for node(5), tid = CMNodeListener:14351 file
= cmipc.c, line = 1036 {Thu Mar 25 16:44:59 2004 }^M

```

Figure 30. *cm.log* file from blade1

After the Oracle instance on blade5 was powered off, OCMS needed to perform cluster membership recovery. Figure 31 shows that the *cm.log* file registered OCMS recovery complete at 16:45:04. This represents a period of time lasting only 15 seconds in which both OCMS and PolyServe Matrix Server were able to respond to node 5 leaving the cluster. For a cluster of 14 nodes, this is remarkable.

```

Successful reconfiguration, 13 active node(s) node 0 is the master, my node num
is 0 (reconfig 2) {Thu Mar 25 16:45:04 2004 }^M
>WARNING: RecvMsg: socket closed for node(1), tid = CMNodeListener:48140 file
= cmipc.c, line = 1036 {Thu Mar 25 17:02:33 2004 }^M
>WARNING: RecvMsg: socket closed for node(1), tid = CMNodeListener:48140 file
= cmipc.c, line = 1036 {Thu Mar 25 17:02:33 2004 }^M
>WARNING: RecvMsg: socket closed for node(1), tid = CMNodeListener:48140 file
= cmipc.c, line = 1036 {Thu Mar 25 17:02:33 2004 }^M

```

Figure 31. *cm.log* file showing recovery

But what about the state of the database instances? It is also important to analyze whether instances of Oracle on nodes other than blade5 suffered an outage.

This test proved the remarkable cluster recovery of Oracle9i RAC. Because Oracle executables are stored in the Matrix Server cluster filesystem, Oracle cannot commence instance recovery until the Matrix Server recovery is complete. In the test, this occurred very quickly, in 15 seconds. Matrix Server and OCMS recovery completed at 16:45:04 and the OLTP instance on blade1 (PROD1) started reconfiguration just 2 seconds later. According to the alert log information shown in Figure 32, the PROD1 instance finished with reconfiguration just one second later.

```

Linux
Thu Mar 25 16:45:06 2004
Reconfiguration started
List of nodes: 0,1,2,3,5,6,7,8,9,
Global Resource Directory frozen
Communication channels reestablished
Master broadcasted resource hash value bitmaps
Non-local Process blocks cleaned out
Resources and enqueues cleaned out
Resources remastered 6488
72142 GCS shadows traversed, 192 cancelled, 5555 closed
48733 GCS resources traversed, 0 cancelled
33283 GCS resources on freelist, 77706 on array, 77706 allocated
set master node info
Submitted all remote-enqueue requests
Update rdomain variables
Dwn-cvts replayed, VALBLKs dubious
All grantable enqueues granted
72142 GCS shadows traversed, 5844 replayed, 5747 unopened
Submitted all GCS remote-cache requests
1 write requests issued in 49352 GCS resources
6 PIs marked suspect, 2 flush PI msgs
Thu Mar 25 16:45:07 2004
Reconfiguration complete
2370,12 17%

```

Figure 32. PROD1 instance reconfiguration

The manageability of the FDC architecture proved invaluable. Having such a large cluster in support of various applications provides the ability to “repurpose” a node. What better time to repurpose a node than in response to a server failure as was simulated by powering off blade5.

Figure 33 shows that reconfiguration finished for OCMS nodes 0-3 and 5-10 (counting from zero) by 16:46:10. There were nine healthy instances at 16:45:07 and 63 seconds later there were 10 instances. The 63 seconds spanned the time required to shut down the DSS instance that was executing on Blade11 and then to start a PROD11 instance on that node. All the while, OLTP was executing with 200 sessions per node on the nine remaining original PROD instances!

```

Linux
Reconfiguration started
List of nodes: 0,1,2,3,5,6,7,8,9,10,
Global Resource Directory frozen
Communication channels reestablished
Master broadcasted resource hash value bitmaps
Non-local Process blocks cleaned out
Resources and enqueues cleaned out
Resources remastered 6453
72142 GCS shadows traversed, 2 cancelled, 7904 closed
49280 GCS resources traversed, 2 cancelled
33506 GCS resources on freelist, 77706 on array, 77706 allocated
set master node info
Submitted all remote-enqueue requests
Update rdomain variables
Dwn-cvts replayed, VALBLKs dubious
All grantable enqueues granted
72143 GCS shadows traversed, 6338 replayed, 7906 unopened
Submitted all GCS remote-cache requests
Thu Mar 25 16:46:10 2004
Completed redo application
Thu Mar 25 16:46:10 2004
0 write requests issued in 44200 GCS resources
0 PIs marked suspect, 0 flush PI msgs
16:45:07 2409,12 17%

```

Figure 33. OCMS reconfiguration

Figure 34 shows that throughout the recovery period, 100% of the user count on nodes 1-4 and 6-10 remained connected. Figure 34 also shows that 62 of the users from blade5 were failed over through Net Services to nodes 2, 3, 4 and 8. These users were executing applications enabled for Transparent Application Failover.

```

Linux
SQL> @now

LOCAL_TIME
-----
03/25/2004 16:49:21

SQL> @users

MACHINE      COUNT(*)
-----
blade1        223
blade10       212
blade11        12
blade2        223
blade3        234
blade4        224
blade6        212
blade7        212
blade8        229
blade9        212

10 rows selected.

SQL>
  
```

Figure 34. User count during recovery

The remaining users connected to blade11 once it was online. Figure 35 shows that in the one-minute time from Figure 33 to Figure 34, 150 users had connected to PROD11. At that point, 100% of the original user community was back online.

```

Linux
SQL> @now

LOCAL_TIME
-----
03/25/2004 16:50:21

SQL> @users

MACHINE      COUNT(*)
-----
blade1        223
blade10       212
blade11       164
blade2        223
blade3        234
blade4        224
blade6        212
blade7        212
blade8        229
blade9        212

10 rows selected.

SQL>
  
```

Figure 35. Users connecting to PROD11

The summary impact to the pseudo-users in this test was a reduction in server bandwidth of 10% for only 71 seconds from 16:44:59 when blade5 went offline to 16:46:10 when blade11 was ready to accept connections. No total outage, no long duration service brown-outs, no troublesome action required on behalf of the administrative and operational staff in spite of a server failure in this large cluster environment establishes the FDC architecture as a natural fit for today's demanding IT requirements.

Scalability in an OLTP Environment

OLTP

The OLTP workload described earlier was used to test the technology combination of IBM eServer BladeCenter, PolyServe Matrix Server and Oracle9i RAC. The test method consisted of connecting 100 pseudo-users who were executing the Pro*C benchmark code and connected via SQL*Net to dedicated servers.

“Scalability on Demand” was a continual focus during all of the FDC testing. Data points collected at each node count were not preceded by a global reboot. Instead, once a test execution completed, additional servers were brought online to satisfy the next test node count requirement. That is, there were no database shutdowns after the test began at 100 users on one node. After that test completed, an instance of the PROD (OLTP) database was started and the 2-node test commenced, and so on. This method clearly supports the position that the FDC architecture is a truly flexible pool of server resources that can be added to a workload without interruption. Since SQL*Net is set up to support connectivity to a PROD instance on any node in the BladeCenter, all that has to transpire to support such connections is starting up an instance—a seamlessly dynamic utilization of the server pool.

In the following graphs, each data point represents an average taken from three executions of the benchmark of 30-minutes duration. All told, a full test suite from one to 14 nodes represented 12 hours of solid OLTP processing.

Generally, scalability is the first data point to examine in a benchmark scenario. The FDC proof of concept was much more than just a benchmark. Scalability for Oracle9i RAC on PolyServe Matrix Server and IBM eServer BladeCenter was very good. Scalability was limited under this workload, as it turns out, by not having enough disk subsystem bandwidth to satisfy the tremendous bandwidth available in the BladeCenter nodes.

Older systems architectures were much more prone to performance bottlenecks than the combined technology of BladeCenter and Oracle9i RAC on PolyServe Matrix Server. In the not so distant past, running this OLTP workload on the same number of disk drives would have certainly driven 28 processors to saturation. However, the 28 CPUs in the BladeCenter under test never peaked above 82% utilization. With Oracle9i RAC on BladeCenter, it seems the challenge for building balanced, scalable systems running this workload may not be a processor or memory-level concern, but may instead be a storage subsystem issue. That is, this testing has established that Oracle9i RAC and BladeCenter with PolyServe Matrix Server can scale as much as the underlying disk subsystem can handle.

The BladeCenter was attached to a SAN configured with 110 disk drives. The database was placed evenly across 106 of those disks with a Stripe and Mirror Everything (SAME) methodology. Modern drives such as those used for the OLTP database should be expected to service roughly 100 random 4K transfers with acceptable latency. That latency, however, is not static. As the drives approached critical service levels, the latencies seen during this testing approached 40ms per transfer as should be expected.

As seen in Figure 36, Oracle9i RAC was able to drive the physical 4K random I/O rate up to steady states of roughly 12,600 transfers per second. Peaks of nearly 14,000 per second did occur during checkpoints and other burst-related activity. This is a tremendous amount of random OLTP disk transfers. These I/O rates are not from a simple test program. For Oracle to demand this much I/O, there is an incredible amount of processor- and memory-intensive activity in the System Global Area (SGA). For example, the table on page 33 shows that for every physical transfer (8.2 per TPS), there are also 5.7 logical reads of buffers in the SGA. Every logical read has associated Oracle internal overhead such as acquiring/freeing latches (e.g., cache buffers chains, cache buffers lru, etc). The peak sustained “I/O fallout” of 12,600 transfers per second is only indicative of just how stressed Oracle instances were on each node.

Figure 36 shows that the OLTP workload drove the physical I/O rates from 2,073 per second at one node to 12,656 per second at 14 nodes. Analysis of I/O latency data revealed that the I/O service times remained acceptable through the 8-node test, but beyond that point, the drives were saturated. Since the sustained I/O rate at eight nodes was 9,829 transfers per second, or 93 per drive, the resulting increase in transfer times was expected. The key point, however, was that the amount of hardware available to allocate to this test held scalability to the level achieved. Oracle9i RAC on PolyServe Matrix Server suffered no scalability problems. As was demonstrated by the peak clusterwide CPU utilization of 82%, the BladeCenter surely had more bandwidth.

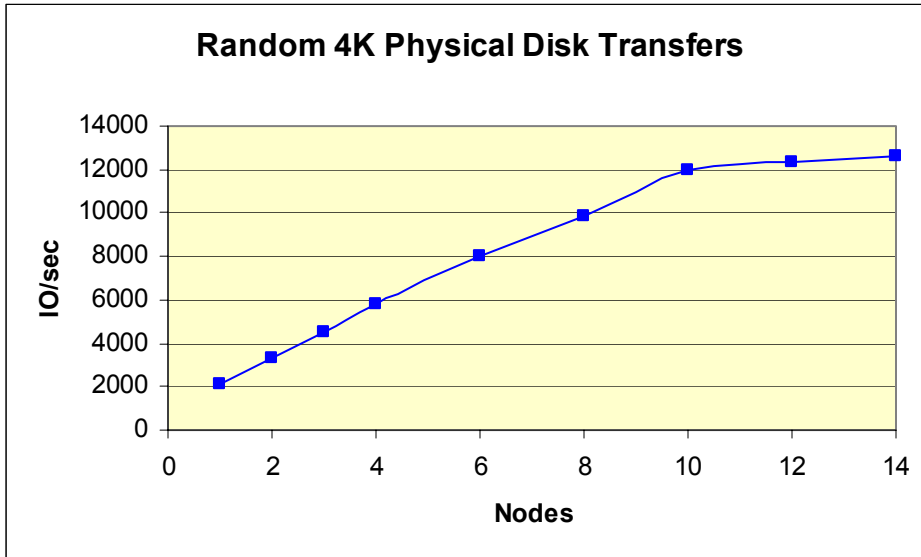


Figure 36. Physical disk transfers

The throughput graph in Figure 37 clearly indicates that scalability was directly related to I/O throughput. This is actually really good news. Unlike older OLTP system architectures, where the bottleneck was usually at the system level, resolving this performance issue is a snap. With the DS4000 architecture, you simply add disks to the array or upgrade to newer generation controllers as they become available.

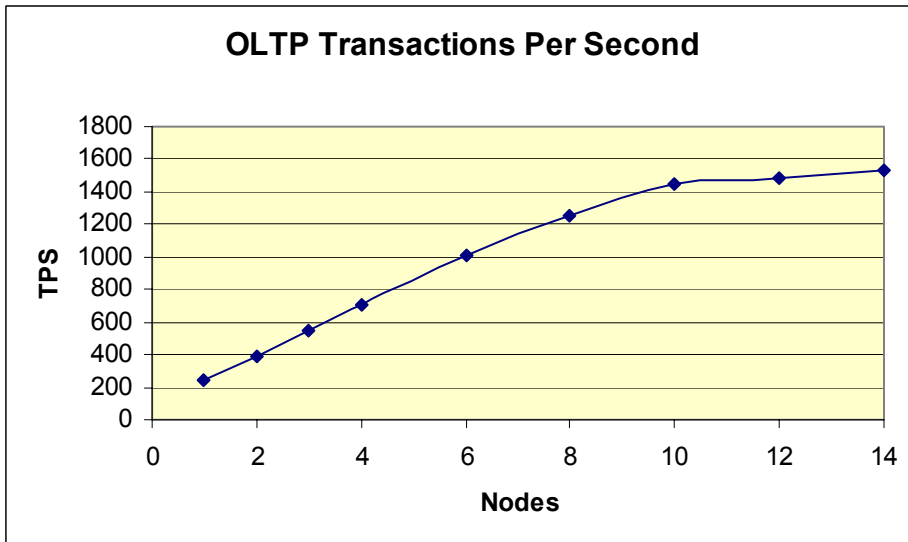


Figure 37. Scalability and I/O throughput

Scale on Demand, Dynamically and Transparently

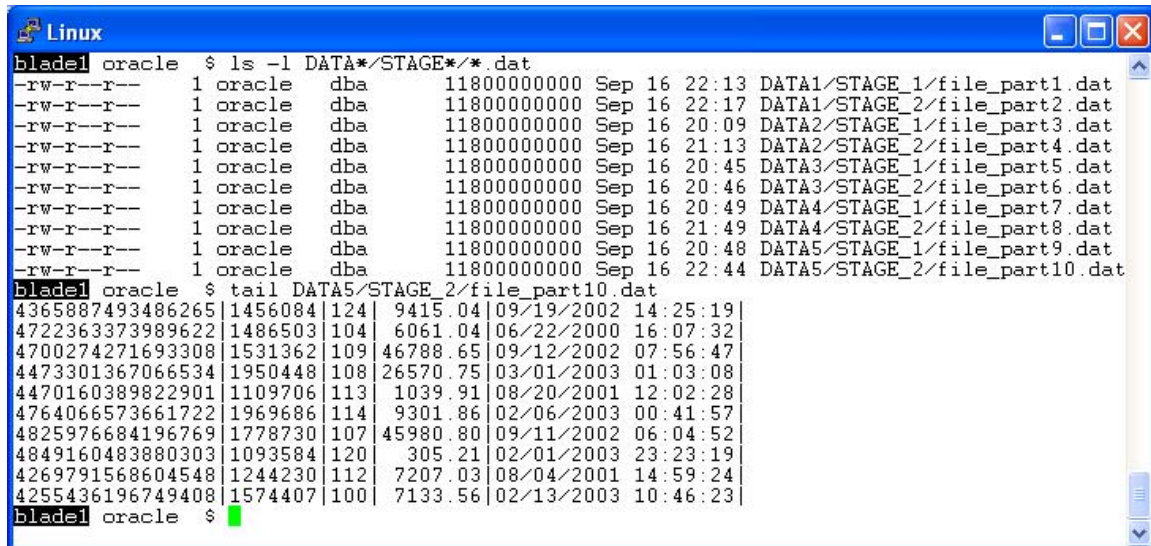
DSS

The FDC architecture has the advantage of being able to use the Oracle Parallel Query Option (PQO) for DSS workloads. PQO has the ability to decompose DSS-style queries and large data maintenance functions into work units executed by some or all of the CPUs in a cluster.

The DSS-style testing performed on the FDC also proved the “Scale on Demand” theme of Oracle9i RAC. The testing essentially consisted of adding nodes to speed up a DSS query or large administrative task such as index creation. The DSS database was not shut down during the testing. Instead, as was the case in the OLTP testing, additional servers were simply allocated and the test was executed again. The DSS database was architected to run on the top eight nodes of the cluster while OLTP was running on nodes 1 through 6, which is not a very high load.

Three main tests were executed at the various node count levels. The tests consisted of the following:

DSS Test 1. This test was a timed loading of the 2 billion pipe-delimited records of data from flat files into the card table. The test consisted of 4 streams of SQL*Loader per node each loading an equal amount of flat file data. The SQL*Loader direct path load feature was used. The records were roughly 54 bytes in length. The flat files required 110GB filesystem space in aggregate. Figure 38 shows a cluster filesystem listing of the 10 flat files as well as a display of the last few records in one of the files.



```
blade1 oracle $ ls -l DATA*/STAGE*/*.dat
-rw-r--r-- 1 oracle dba 11800000000 Sep 16 22:13 DATA1/STAGE_1/file_part1.dat
-rw-r--r-- 1 oracle dba 11800000000 Sep 16 22:17 DATA1/STAGE_2/file_part2.dat
-rw-r--r-- 1 oracle dba 11800000000 Sep 16 20:09 DATA2/STAGE_1/file_part3.dat
-rw-r--r-- 1 oracle dba 11800000000 Sep 16 21:13 DATA2/STAGE_2/file_part4.dat
-rw-r--r-- 1 oracle dba 11800000000 Sep 16 20:45 DATA3/STAGE_1/file_part5.dat
-rw-r--r-- 1 oracle dba 11800000000 Sep 16 20:46 DATA3/STAGE_2/file_part6.dat
-rw-r--r-- 1 oracle dba 11800000000 Sep 16 20:49 DATA4/STAGE_1/file_part7.dat
-rw-r--r-- 1 oracle dba 11800000000 Sep 16 21:49 DATA4/STAGE_2/file_part8.dat
-rw-r--r-- 1 oracle dba 11800000000 Sep 16 20:48 DATA5/STAGE_1/file_part9.dat
-rw-r--r-- 1 oracle dba 11800000000 Sep 16 22:44 DATA5/STAGE_2/file_part10.dat
blade1 oracle $ tail DATA5/STAGE_2/file_part10.dat
4365887493486265|1456084|124| 9415.04|09/19/2002 14:25:19|
4722363373989622|1486503|104| 6061.04|06/22/2000 16:07:32|
4700274271693308|1531362|109| 46788.65|09/12/2002 07:56:47|
4473301367066534|1950448|108| 26570.75|03/01/2003 01:03:08|
4470160389822901|1109706|113| 1039.91|08/20/2001 12:02:28|
4764066573661722|1969686|114| 9301.86|02/06/2003 00:41:57|
4825976684196769|1778730|107| 45980.80|09/11/2002 06:04:52|
4849160483880303|1093584|120| 305.21|02/01/2003 23:23:19|
4269791568604548|1244230|112| 7207.03|08/04/2001 14:59:24|
4255436196749408|1574407|100| 7133.56|02/13/2003 10:46:23|
blade1 oracle $
```

Figure 38. Flat file load data for DSS Test 1

DSS Test 2. This query tested the ability of the FDC architecture to scan and sort a large amount of data. The query consisted of a `select count(distinct(account_num))` from the card table described above. The 2 billion rows of data stored in 16KB Oracle datablocks totaled roughly 112GB space in the database. Therefore, the query processing for this test required a good deal of sorting and merging to eliminate duplicates. The sort key, being a credit card number, was 16 Bytes in length. The sorting could not be performed entirely in memory.

DSS Test 3. This index create test was by far the most data-intensive. It created a three-way unique composite index on the Line Item table described above. The table had roughly 1.7 billion

rows that could not be sorted in memory. This index creation consisted of reading rows and writing to both sort segments and the target index.

Once again, the key point about this testing was that at each step, adding nodes to speed up the workload was a non-intrusive effort. Simply start an instance of the DSS database on other blades, run the task again, and completion times will improve.

DSS Test Results Analysis

Perhaps one of the most critical functions of Data Warehousing is bulk data loading. To ascertain the ability of the FDC architecture to satisfy high-rate data loading, DSS Test 1 was used. As described above, DSS Test 1 was a timed loading of the flat file data depicted in Figure 38. In order to maximize the processor utilization, 4 concurrent streams of SQL*Loader were executed on each node tasked with loading equal portions of the flat file data. Figure 39 shows the scalability of bulk data loading in the Flexible Database Cluster architecture. Loading just short of 2 million rows per second into an Oracle database would be a respectable feat in its own right. However, this rate was achieved with only 8 of the 14 nodes physically present in the cluster.

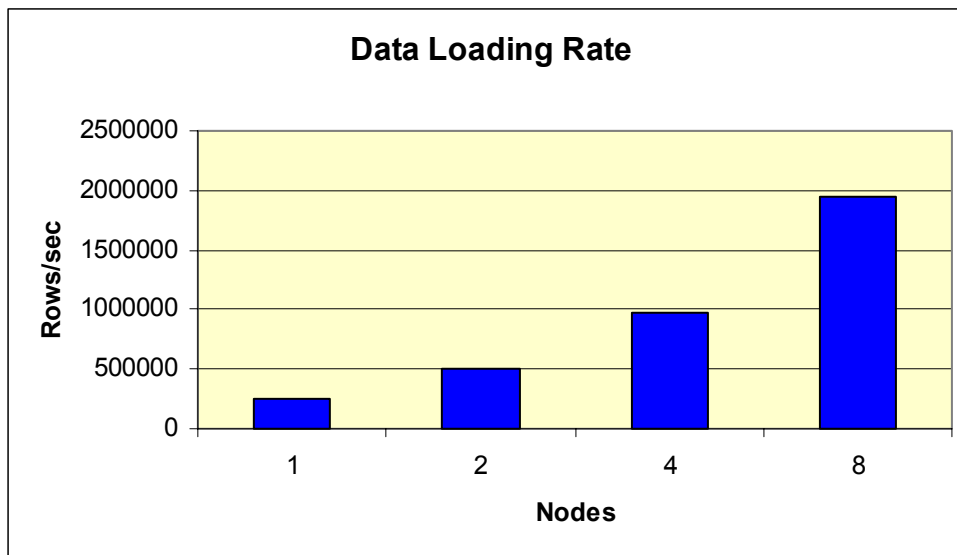


Figure 39. Data loading rate for DSS Test 1

While Figure 39 depicts the row insertion rate into the Card table, Figure 40 shows the I/O throughput presented in Megabytes per second.

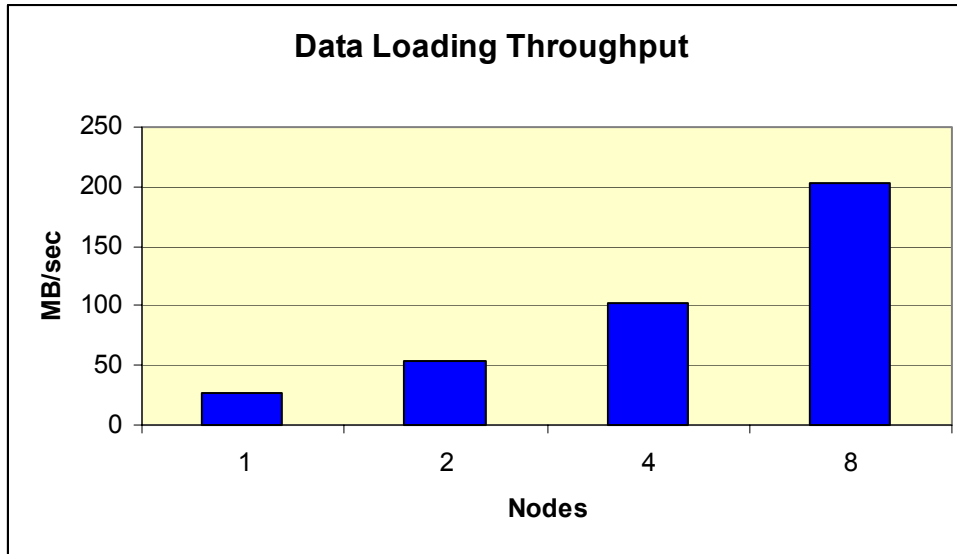


Figure 40. Data loading throughput for DSS Test 1

As described above, DSS Test 2 consisted of counting distinct credit card numbers in the 2 billion row Card table. When executed on a single node, the query completion time was 112 minutes. Without interruption, another instance of Oracle was added and the completion time improved with 93% linear scalability. Further proving the “Scalability on Demand” feature of the FDC architecture, Oracle instances were booted and the test was rerun in succession for 4-node, 6-node, and 8-node counts.

The 8-node completion time for the test was 24 minutes. To put this in perspective, returning this query in 24 minutes means Oracle9i RAC scanned and sorted roughly 87,000 rows per second per CPU or 1.4 million rows per second clusterwide. Figure 41 shows the scalability of this query.

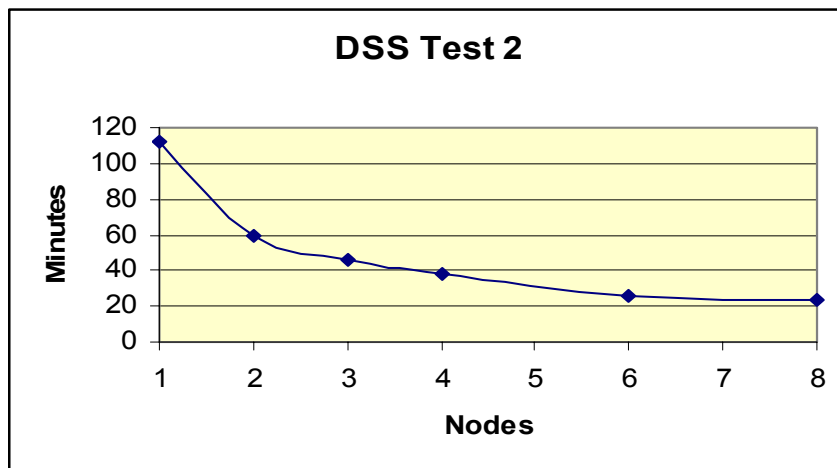


Figure 41. Timed Card Table Sorted Full Scan

DSS Test 2 was both processor and I/O intensive. The sort key for the query was a 16-digit credit card number so the key space was roughly 30GB. The nodes were configured with 2GB main memory and of that, 800MB was allocated on each node to the Oracle PGA aggregate target for use as sort area. There is a short analysis of the effect of doubling the sort area memory allocation later in this paper.

The I/O load for DSS Test 2 warranted closer analysis. The scanning load alone was roughly 112GB as the data was stored in Oracle 16KB data blocks with a small percentage of free space per block to facilitate any space requirement for row updates. Added to this 112GB was the sort overflow, since even at 8 nodes only 6.4GB global memory was available to sort the approximated 30GB key space. The sort I/O totaled approximately 24GB written to temporary sort segments and, given the RAID 1+0 layout of the DS storage subsystem, the array-level I/O load for DSS Test 1 was roughly 160GB (112 + (24 * 2)) over 24 minutes execution time, or 114MB per second. Figure 42 depicts the array-level I/O for DSS Test 1 at various node counts.

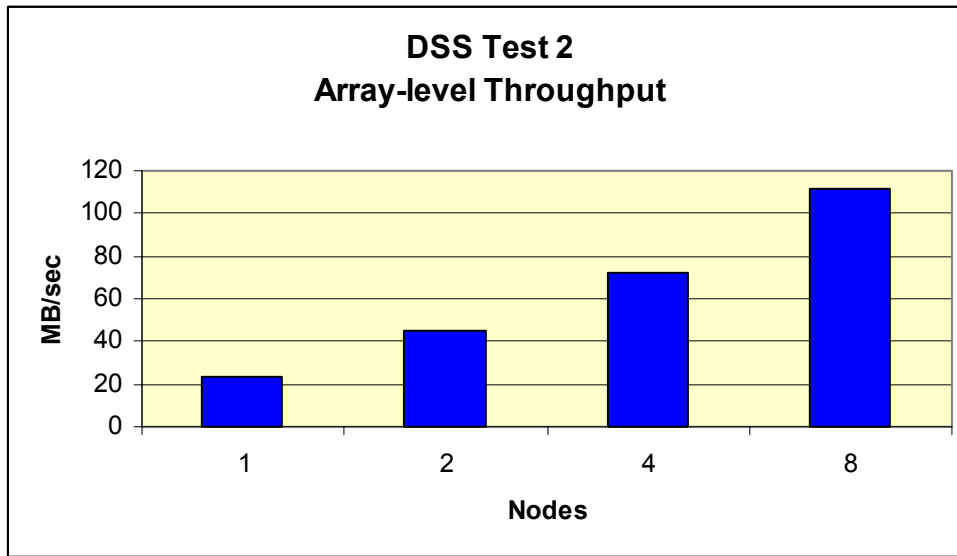


Figure 42. Card Table Array-level I/O Throughput

After executing DSS Test 2, the Oracle instances on the second through eighth nodes were shut down in preparation for DSS Test 3. This test created a 3-way unique composite index on the 1.7 billion row Line Item table described above.

The design of the FDC architecture naturally provides increased server bandwidth. Each two-CPU blade has its own bus, memory controller, and I/O adapter, making it more unlikely to encounter a traditional bus or memory controller bottleneck. These are very balanced servers. This architecture will fully utilize all disk I/O subsystem bandwidth given to it. This architecture is optimized for growth. If you want more server bandwidth, simply add a server. If more disk subsystem bandwidth is needed, simply add it to the SAN.

FDC architecture enables the simple addition of BladeCenter nodes to these index-create tasks. Each data point was collected without interruption. Additional instances of Oracle were simply started on the additional nodes and the index-create task was run again. Figure 43 represents the scalability of intra-node Parallel Query Option when executing in the Flexible Database Cluster architecture.

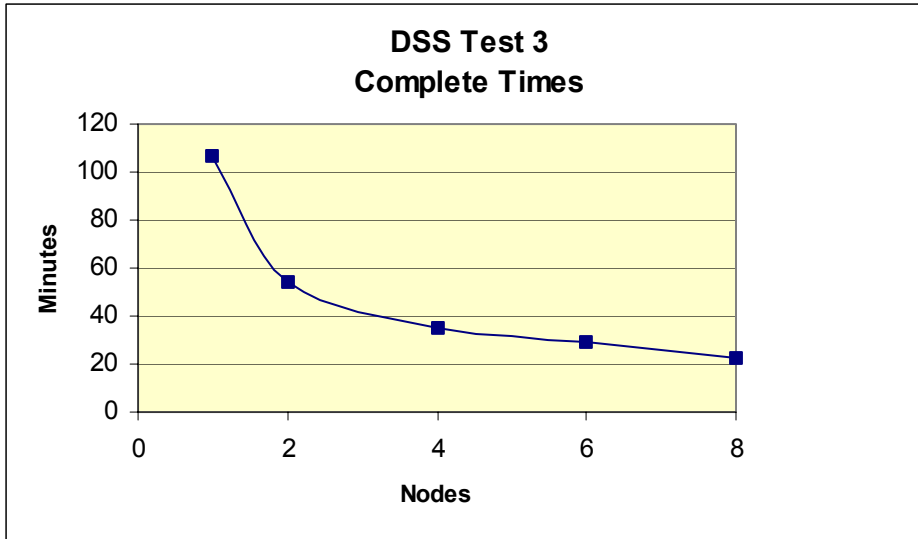


Figure 43. Index-create times

Analysis of the disk I/O performed by DSS Test 3 was conducted using the `ioDSS.sql` script provided in the appendix to this paper. The physical disk transfer rate from the Parallel Query processes scaled 100% from one to two nodes, 70% from two to four nodes, and 74% from four to eight nodes.

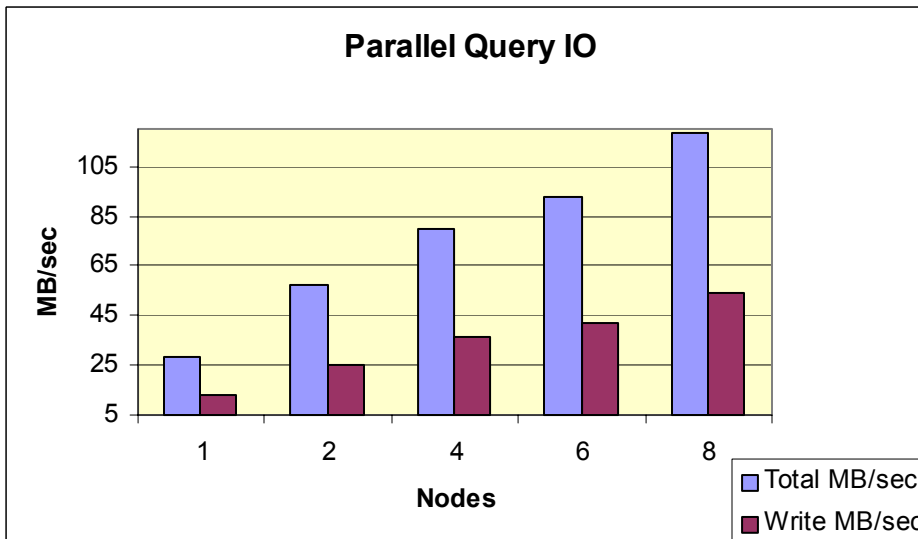


Figure 44. Disk I/O reported in gv\$ tables

One important aspect of this I/O profile is missing from the data provided in the Oracle GV\$ performance virtual tables. Since the storage was striped and mirrored, there were substantially more disk transfers at the array level. At eight nodes, the array-level I/O ratio was actually 37% reads and 63% writes. This is attributable to the fact that DSS Test 3 generated 54MB per second of data written to a mixture of sort segments and index blocks.

The GV\$ performance virtual tables do not present the 100% overhead associated with the mirror write. While the total server-level I/O was 118 MB/sec, the write component of that (54 MB/sec)

has an associated cost of 108 MB/sec in the storage array. Therefore, the total array-level I/O at eight nodes was 172 MB/sec. The breakout is as follows:

$$(64\text{MB/s server-level read}) + (54\text{MB/s server level write}) + (54\text{MB/s array-level write overhead}) = 172\text{MB/s}$$

Figure 45 shows a graphical representation of this DSS array-level I/O effect.

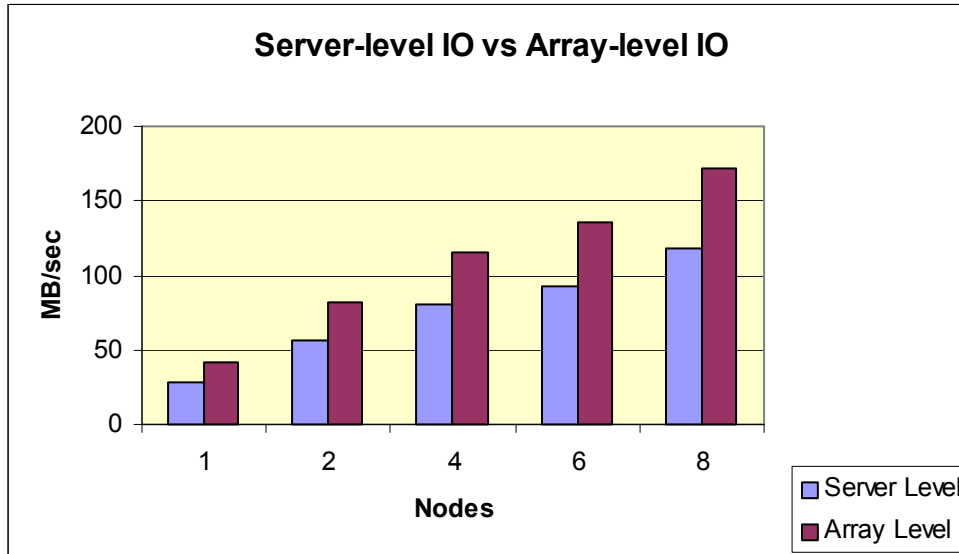


Figure 45. Server-level I/O vs. array-level I/O

The BladeCenter nodes possess a very favorable match of bus bandwidth to processor speed. Even when completely processor bound, there is still headroom for additional throughput should additional memory be dedicated to the Oracle sort area (init.ora pga_aggregate_target). To determine the effect of doubling the amount of allowable memory sort space, a test was run using DSS Test 2. At four nodes, the query complete time with pga_aggregate_target set to 800MB was 38 minutes. Even though the processor utilization on all four nodes was 100% in both cases, the query complete time improved by 24% when the sort area allowance increased from 800MB to 1600MB as depicted in Figure 46. This is a tremendous testament to the bus bandwidth of the BladeCenter nodes since, in general, systems running at complete processor saturation do not have headroom for a 24% performance increase.

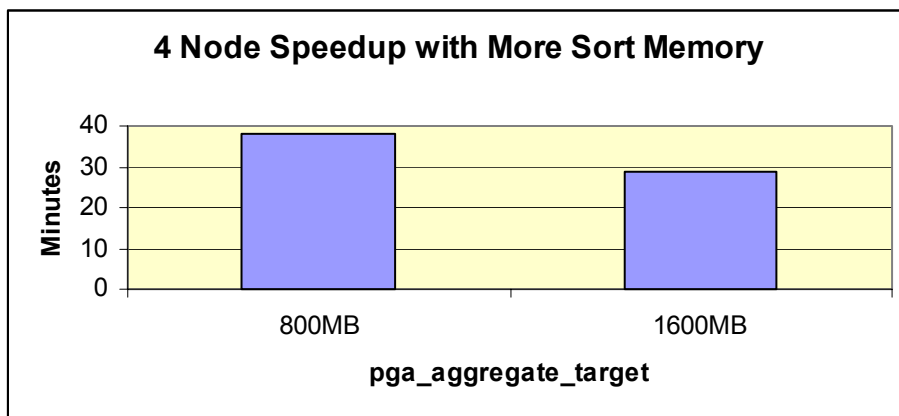


Figure 46. Memory Allocation compliments Server Bandwidth

Hardware Upgrades

FDC Architecture Eases Hardware Upgrades

The architecture of the Flexible Database Cluster lends itself nicely to introducing updated hardware components with expected performance improvement. Indeed, nodes of the cluster can easily be replaced with newer servers. Taking one node out to replace it does not severely impact production when deployed on large clusters. Even the disk subsystem can be updated. Late in the FDC project, the SAN attached to the FDC was upgraded by replacing the DS4400 array controllers with the new DS4500 model. The data remained as-is on disk. This was a simple component swap.

After configuring the DS4500 array controllers, select tests of the DSS test suite were executed to determine what improvement the storage upgrade offered. Note that all of these tests were completely processor bound and I/O intensive; a performance increase was not a sure thing.

To put full stress on the upgraded SAN, DSS Test 1 was run on 10,12 and 14 nodes. Prior to the introduction of the DS4500, the performance for this test flattened out at 10 nodes. The DS4500, on the other hand, yielded scalability through all 14 nodes. The improved handling of write requests in the DS4500 provides for much quicker draining of I/O queues at the server level. Doing so provides for more proficient usage of processor and memory bandwidth and thus, the DSS Test 1 throughput was improved. In fact, the improvement at 14 nodes was 40% with a data loading rate of 3.2 million rows per second as depicted in Figure 47.

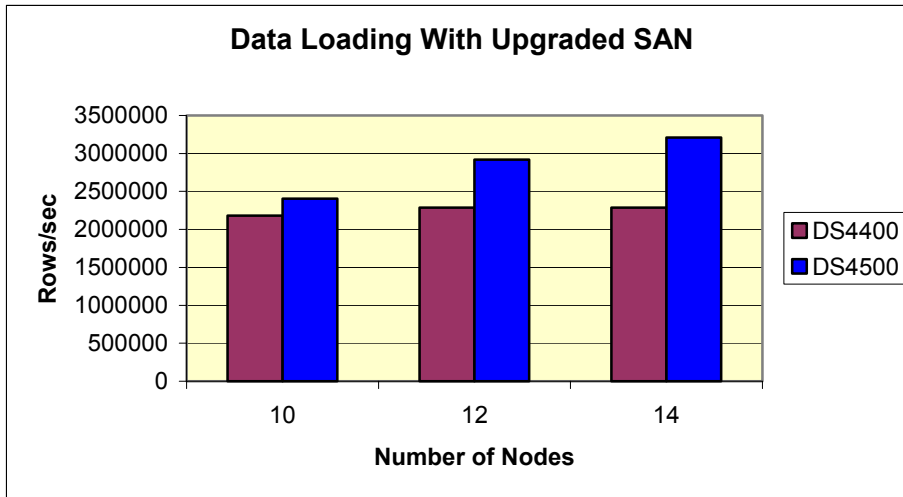


Figure 47. Loading rate with upgraded SAN

Not only did data loading times improve, the query times also improved. All this without redistributing the data on disk, just a simple array controller replacement. DSS Test 2 was executed at node counts ranging from 1 to 14 nodes with job complete time improvements from 9 to 13 percent.

An additional test was run to test the sheer throughput capability of the DS4500. Using a simple Oracle select statement such as `select count(*) from card` eliminates most processor overhead and therefore is a pure test of I/O capability. Performing a full scan of the 2-billion row Card table with this query improved dramatically with the upgraded disk subsystem. The two-node complete time for this lightweight scan improved 45% as shown in Figure 48.

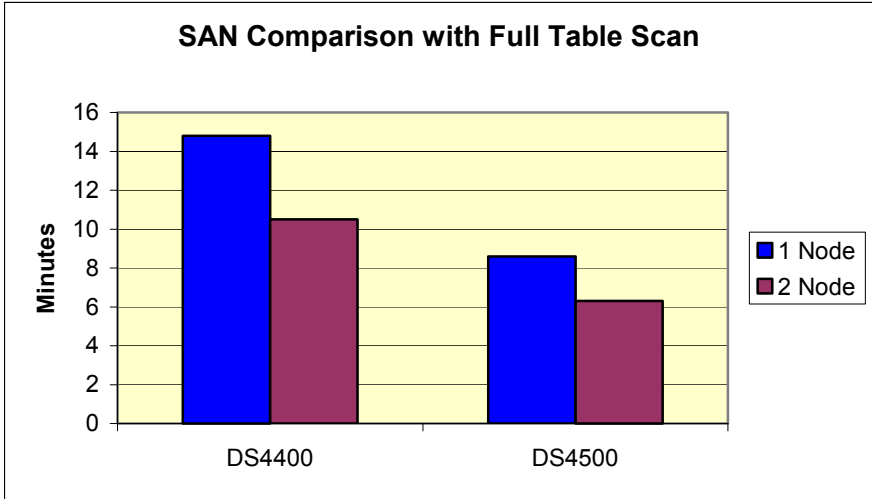


Figure 48. Full scan performance after SAN upgrade

Using the BladeCenter performance monitor, the I/O throughput for the full table scan peaked at 416 MB/sec (426,473KB) as show in Figure 49.

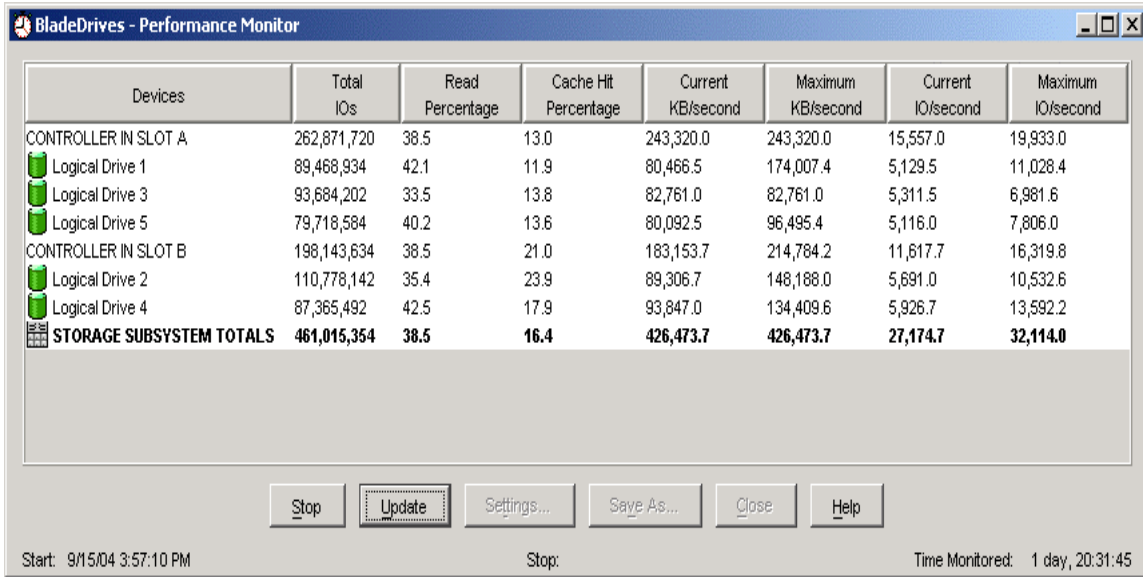


Figure 49. Full scan performance after SAN upgrade – 416MB/sec

Summary

The Flexible Database Cluster architecture, with the IBM eServer BladeCenter and PolyServe Matrix Server, clearly enhances the value of Oracle9i Real Application Clusters, providing:

- A means to consolidate and deploy multiple databases and associated applications in a single, easily managed cluster environment.
- Simplified management of large database clusters, made possible by the PolyServe Matrix Server clustered filesystem.
- Dynamic “scalability on demand” architecture, enabling near-linear speedup to running applications in some cases—with little or no interruption.
- Dynamic repurposing of server resources on demand to quickly and easily move processing capacity to where it is most needed.
- The ability to adopt improved hardware at the server and SAN level, made simple by the modular architecture.
- An autonomic, always-on operating environment with fast or even immediate self-healing and little or no performance degradation (and therefore increased utilization rates).
- Dramatic incremental TCO benefits from improved manageability, scalability, expandability, availability and asset utilization.

Appendix

users.sql

```
Column machine format a8
select machine,count(*) from gv$session
group by machine ;
```

instance_status.sql

```
column host_name format a10
select host_name,thread#,database_status from gv$instance
order by thread# ;
```

now.sql

```
select to_char(sysdate,'MM/DD/YYYY HH24:MI:SS') "LOCAL_TIME" from dual
```

io.sql

```
select sum(PHYRDS) reads,sum(PHYBLKRD * 4 )/1024 readMB,
       sum(PHYWRTS) writes,sum(PHYBLKWRT * 4 )/1024 writeMB
from dba_data_files,gv$filestat
where dba_data_files.file_id = gv$filestat.file#;
REM exit;
```

ioDSS.sql

```
select sum(PHYRDS) reads,sum(PHYBLKRD * 16 )/1024 readMB,
       sum(PHYWRTS) writes,sum(PHYBLKWRT * 16 )/1024 writeMB
from dba_data_files,gv$filestat
where dba_data_files.file_id = gv$filestat.file#;
REM exit;
```



© IBM Corporation 2004

IBM Systems and Technology Group

Department 23U

Research Triangle Park, NC 27709

Produced in the USA.

6-04

All rights reserved.

Visit www.ibm.com/pc/safecomputing periodically for the latest information on safe and effective computing. Warranty Information: For a copy of applicable product warranties, write to: Warranty Information, P.O. Box 12195, RTP, NC 27709, Attn: Dept. JDJA/B203. IBM makes no representation or warranty regarding third-party products or services including those designated as ServerProven or ClusterProven.

IBM, the eight bar logo, the eServer logo, eServer, xSeries, BladeCenter, ServerProven, and TotalStorage are trademarks or registered trademarks of International Business Machines Corporation in the U.S. and other countries. For a list of additional IBM trademarks, please see <http://www.ibm.com/legal/copytrade.shtml>

Intel and Xeon are trademarks or registered trademarks of Intel Corporation.

Oracle and Oracle9i are trademarks or registered trademarks of Oracle Corporation.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

PolyServe and the PolyServe logo are trademarks of PolyServe, Inc.

Other company, product, and service names may be trademarks or service marks of others.

IBM reserves the right to change specifications or other product information without notice. References in this publication to IBM products or services do not imply that IBM intends to make them available in all countries in which IBM operates. IBM PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied

warranties in certain transactions; therefore, this statement may not apply to you.

This publication may contain links to third party sites that are not under the control of or maintained by IBM. Access to any such third party site is at the user's own risk and IBM is not responsible for the accuracy or reliability of any information, data, opinions, advice or statements made on these sites. IBM provides these links merely as a convenience and the inclusion of such links does not imply an endorsement.

This document is for informational purposes only and does not set forth any warranty, expressed or implied, concerning any software, software feature, or service offered or to be offered by PolyServe, Inc. PolyServe, Inc., reserves the right to make changes to this document at any time, without notice, and assumes no responsibility for its use. This informational document describes features that may not be currently available. Contact PolyServe corporate headquarters for information on feature and product availability. The PolyServe Matrix Server product uses software developed by Spread Concepts LLC for use in the Spread toolkit. For more information about Spread, see <http://www.spread.org>.