Comparison of Performance of Competing Database Storage Technologies:

NetApp® Storage Networking vs. Hardware RAID

Dan Morgan | Network Appliance | June 2002 | TR-3145

TECHNICAL REPORT

Network Appliance, a pioneer and industry leader in data storage technology, helps organizations understand and meet complex technical challenges with advanced storage solutions and global data management strategies.

Table of Contents

1. Executive Summary

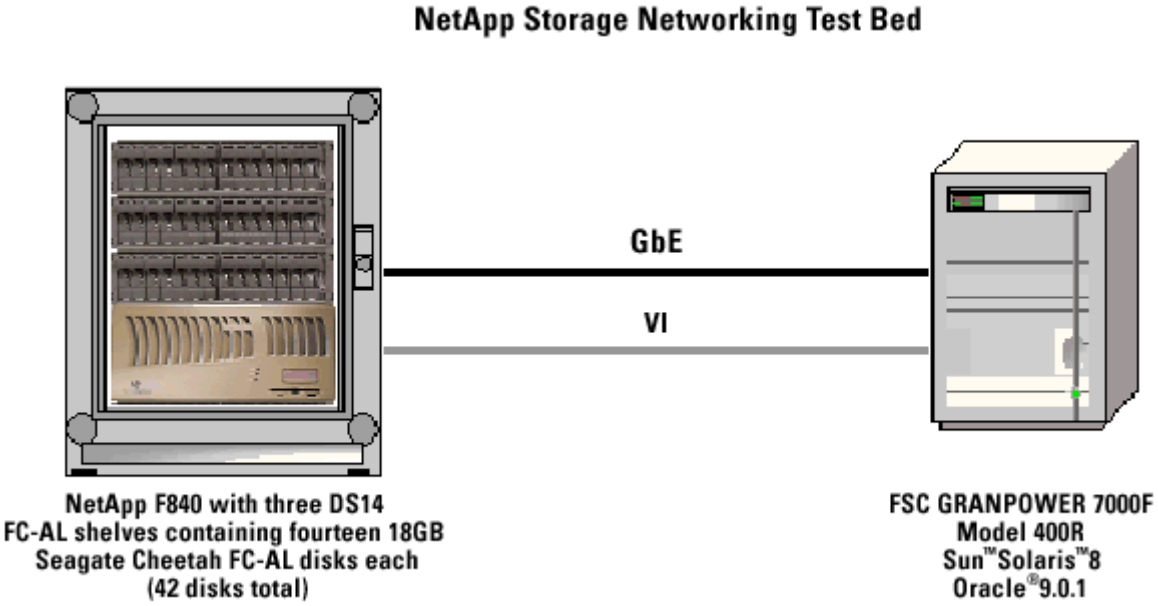
This is an update to the set of performance tests on competing database storage technology documented in TR3105. In addition to the two technologies that were the focus of the original testing,

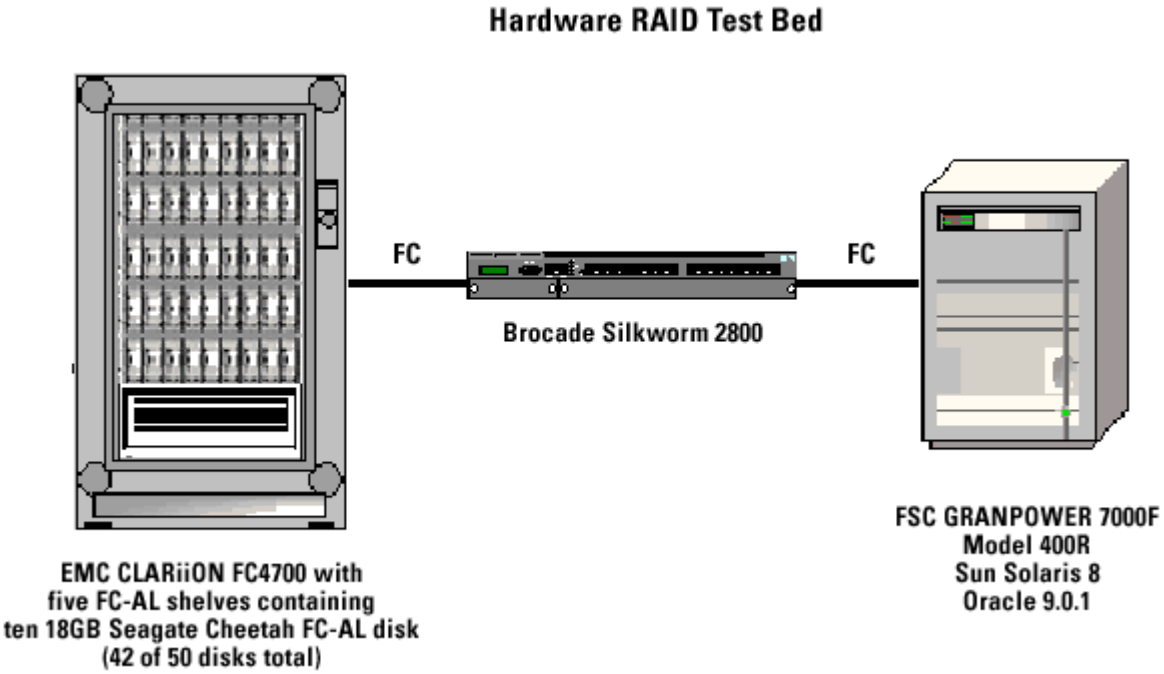
- Storage networking using Network Appliance™ filers
- Software RAID using Veritas with Quick I/O

this update includes the following new technologies:

- JBOD
- Hardware RAID
- Network Appliance DAFS Database Accelerator (DDA)

The following network diagrams contain the three primary configurations used in the tests:





The following matrix contains the results of this testing:

Metric	JBOD	DDA	NFS Jumbo	FC4700 RAID1+0 Raw	FC4700 RAID1+0 Vxfs QIO
Average Response Time (sec)	.12	.12	.15	.15	.16
Disk Capacity Utilized	13%	16%	16%	26%	44%
Disk Access Times (ms)	18	9	11	18	16
Disk Volume Creation	3.5 hrs	< 2 min	< 2 min	4 hrs	4 hrs
Transactions per Interval	47,895	48,215	39,123	38,235	38,569

The balance of this paper contains an overview of the technologies being compared, a more detailed description of the tests performed, and technical details of the configuration used in these tests.

2. Database Storage Technologies Background

2.1. JBOD

Traditionally, the database storage market was dominated by the use of JBOD (Just a Bunch Of Disks) technology. This consists of disks connected to a database server by means of a SCSI or Fibre Channel controller, without the use of RAID (Redundant Array of Inexpensive Disks). While JBOD technology is well understood, it has many disadvantages, including a lack of HA (High Availability) and difficult, expensive tuning. For this reason, the use of JBOD for storing database files is declining, and it is being replaced by RAID and storage networking.

2.2. RAID

RAID solutions address most of the deficiencies of JBOD. RAID systems balance load across a set of spindles, providing a measure of automatic load balancing. When a single disk drive in a RAID array fails, the RAID system can recover the data without any need to resort to restoring a backup. These advantages come at some cost in terms of storage efficiency and performance. The dominant forms of RAID are RAID5 and RAID0+1 (also known as mirrored striping).

The market for RAID storage products is undergoing rapid growth. This growth has been driven by three factors:

1. The growth of processor speed has outstripped the growth in disk speed. This imbalance transforms traditionally CPU-bound applications to disk I/O-bound applications. To obtain an improvement in application performance, disk I/O bandwidth must be increased. The most common way to do this is by increasing the number of disks used to work on the problem.

2. Arrays of small-diameter disks often have substantial cost, power, and performance advantages over larger disk drives.

3. Disk array subsystems can be made highly reliable by storing a small amount of redundant information in the array. Without this redundancy, large disk arrays have unacceptably low data reliability because of their large number of component disks. This is the reason RAID was developed.

The most common variant, RAID5, employs distributed parity. Data is striped over all disks so that large files can be fetched with high bandwidth. By distributing the parity, many random blocks can be written in parallel without creating a hot disk.

The other most common variant is RAID0+1, which combines nonredundant striping with mirroring. The principal disadvantage of this technology is cost, because the disk overhead is 100%.

While RAID5 disk arrays offer performance and reliability advantages for a wide variety of applications, they have at least one critical limitation: their throughput is penalized by a factor of four over nonredundant arrays for workloads of mostly small writes. This penalty arises because a small write request requires the following steps to be performed:

1. The old value of the user's targeted data must be read.

2. The old value must be overwritten with the new value.

3. The old value of the parity data must be read.

4. The old value of the parity data must be overwritten with the new value.

Since these four operations must be performed for every write, the burden is felt most strongly for loads involving many small writes, as the I/Os cannot be amortized over as large an amount of data.

In contrast, systems based on RAID0+1 simply write the user's data on two separate disks and are only penalized by a factor of two. This disparity, four accesses for small writes instead of two, is termed the "small write problem."

Unfortunately, the performance of online transaction processing (OLTP) systems, a substantial segment of the database storage market, is largely dominated by the performance of small writes. Because of this limitation, many OLTP systems continue to employ the much more expensive

option of RAID0+1, which requires a disk overhead of 100%, as opposed to 20%, which is the typical level of parity overhead for most RAID5 systems.

Network Appliance's variant of RAID4 solves this problem by buffering the small writes into memory prior to writing them to disk. Effectively, many small writes are combined into a smaller number of large writes, thus avoiding the small write problem. An excellent discussion of Network Appliance's approach to this problem can be found in *A Storage Networking Appliance* by Dave Hitz and Mike Marchi (http://www.netapp.com/tech_library/3001.html). NetApp's solution combines the performance advantages of RAID0+1 with the cost advantages of RAID5. (Indeed, the parity overhead of NetApp RAID4 is even less than the typical RAID5 solution—on the order of 7% to 14%.) The results presented in this paper are an indication of the effectiveness of this approach.

2.3. Storage Networking

A further evolution in the area of database storage is storage networking. Historically, this has consisted of two areas: NAS (network-attached storage) and SAN (storage area networks). However, many consider these areas to be converging, and the distinction between these two markets is beginning to blur. For example, see "How Convergence Will End the SAN/NAS Debate" by Michael Alvarado and Puneet Pandit, DM Review, February 2001 (http://www.dmreview.com/master_sponsor.cfm?NavID=193&EdID=3016). For the purposes of this paper, both of these areas are included in the term "storage networking."

Storage networking is the combination of RAID and some type of networking, either TCP/IP or UDP/IP over Ethernet, Fibre Channel, or other proprietary network technology. (Please note that Network Appliance recommends TCP/IP rather than UDP/IP for use with databases due to the potential for packet loss with UDP/IP.) The use of networking allows the storage device to be shared by multiple database servers. The functionality advantages of storage networking are clear and have been documented in many places. For example, see the following paper for an explanation of the advantages of NetApp storage networking in the context of Oracle backup and recovery: Oracle9™ for UNIX®: Backup and Recovery Using a NetApp Filer by Jerry Liu, Sunil Mahale, and Jeff Browning (http://www.netapp.com/tech_library/3130.html).

Network Appliance storage networking over NFS uses TCP/IP or UDP/IP over Ethernet. For the purpose of the testing documented in this paper, the physical layer consisted of Gigabit Ethernet.

2.4. Direct Access File System (DAFS)

The Direct Access File System (DAFS) is a new, fast, and lightweight method of accessing data from a file server. Using the virtual interface (VI) architecture as its standard transport mechanism, DAFS allows clusters of application servers to efficiently share data while avoiding the overhead imposed by general-purpose operating systems.

The Local File-Sharing Environment

As the leading provider of network-attached file server solutions, Network Appliance deals with a wide spectrum of customer needs. One can describe these needs as being divided into two broad usage families: wide file sharing and local file sharing. The wide file-sharing environment is characterized as typically having many client machines that may be geographically dispersed and under the control of several different groups of administrators. The client machines can run different operating systems and application software, and usually provide services directly to end users. The client machines can access many different file servers and use them to share data occasionally—for example, so that one user can access another user's document.

The local file-sharing environment is characterized by a smaller number of client machines,

usually colocated in a data center. The client machines are typically themselves application servers providing services to their own set of client machines. The application servers generally run an identical set of operating system and application software, and are configured so that the ultimate users of the application see a single service. The application servers access a small set of file servers over a separate high-speed network and can share individual data files intensely. For example, in a scalable Web service, each application server might frequently reference a common home page as new users connect with the service. Local file-sharing architectures are attractive for several reasons. First, they allow scaling of both computing power and storage capacity. Computing resources can be scaled by simply adding another identically configured computing node. Storage can be scaled either by adding more storage to a file server or by adding more file servers. Local file-sharing architectures also simplify management since computing resources and storage can be scaled and managed independently. Finally, local file-sharing architectures can be made to be fault tolerant; an application server can take over the load of another failed application server, and the file servers can be configured to take over the data set of yet another failed file server.

The Virtual Interface Architecture

Keep in mind that it is the use of standard remote file access protocols that allows the data sharing that is critical to local file-sharing architectures. The alternative of attaching storage directly to each node data would require very specialized file system software to achieve consistent sharing of writable data and would be difficult to manage in any case. Currently, both network-attached storage and direct-attached storage require operating system support. Both require a significant amount of extra computing cycles and require the operating system to copy data from file system buffers into application buffers. This overhead plus any additional overhead from network protocol processing can be eliminated by using a new interconnection architecture called VI.

The Virtual Interface architecture was originally developed by Compaq, Intel, and Microsoft to serve as a standard paradigm for interconnecting computers in clusters. The VI architecture is intended to provide a single standard interface for clustering software, independent of the underlying networking technology. VI provides two fundamentally new capabilities that are not found on traditional interconnection networks. The first capability is direct memory-to-memory transfer. This allows bulk data to bypass the normal protocol processing and to be transferred directly between appropriately aligned buffers on the communicating machines. The second capability is direct application access; application processes can queue data transfer operations directly to VI-compliant network interfaces without operating system involvement. By eliminating protocol-processing overhead and providing a simple model for moving memory blocks from one machine to another, the VI architecture improves CPU utilization and drastically reduces latency. However, these improvements come at a cost: The VI architecture is optimized for communications within a controlled high-speed, low-latency interconnection network and not for general WAN communications via the Internet. Industry excitement over the VI architecture has encouraged the creation of many different implementations. The specification mandates a set of capabilities, not a specific physical implementation, so it can run over a variety of interconnection networks. Current implementations use Fibre Channel and proprietary interconnection networks. Implementations that utilize TCP/IP over Gigabit Ethernet (or even 10-Gigabit Ethernet) as well as Infiniband are being pursued. Though the details differ, VI implementations do share many performance-oriented details. VI host adapters perform all message fragmentation, assembly, and alignment in hardware and allow data transfer directly to or from application buffers in virtual memory.

VI-compliant interconnection networks allow a new file access paradigm: The Direct Access File System (DAFS) is a protocol that uses the underlying VI capabilities to provide direct application

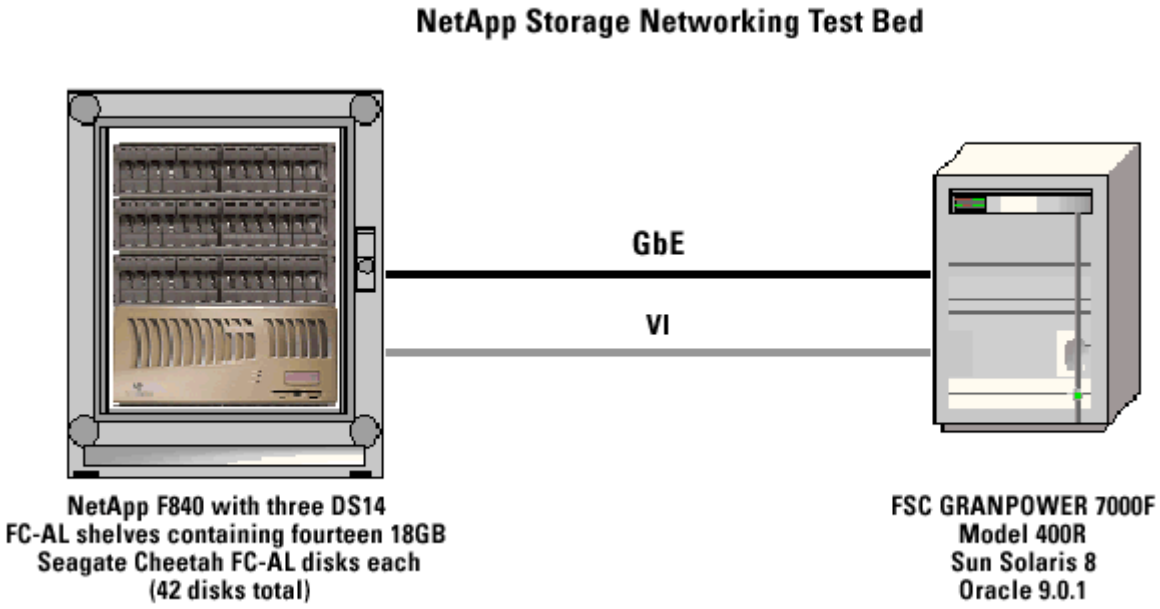
access to shared file servers. It is optimized for high-throughput, low -latency communication, and for the requirements of local file-sharing architectures.

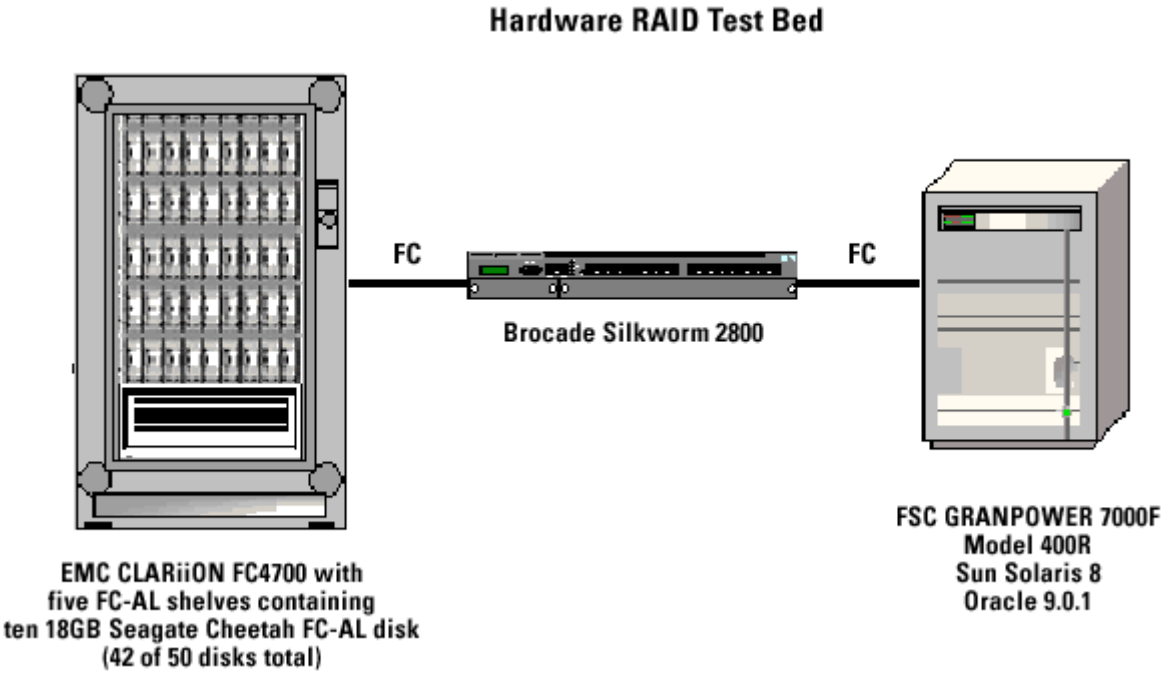
DAFS will be implemented as a file access library that can be linked into local file-sharing applications. The DAFS library will in turn require a VI provider library implementation appropriate to the selected VI-compliant interconnect. Once an application is modified to link with DAFS, it is independent of the underlying operating system for its data storage requirements.

The Direct Access File System enables a new paradigm for shared data access that goes beyond the current capabilities of network-attached storage, direct-attached storage, and Fibre Channel SAN architectures. DAFS is specifically designed to enhance local file-sharing architectures, which in turn enables a platform to deploy scalable and reliable services based on cost-effective commodity server hardware.

3. Test Description

In all storage networking configurations, NFS and DDA, the same disk controller, disk shelves, and disks were used. The primary difference in the storage networking configurations is the network controller. In the NFS configuration, a Gigabit Ethernet controller that supports Jumbo frames was used, the SysKonnect 9843. For DDA, an Emulex VI controller was used. The FC4700 was set up in a RAID1+0 configuration with the same number of FC disk spindles used in all the other comparisons. Further, these were all currently available, state-of-the-art components. Also, the same database server, with nearly identical settings, was used in all cases. In areas where the settings differed, these were changed in order to achieve the best results on both test beds. The technical details contained later in this paper provide these settings. The following network diagrams contain an overview of the three primary configurations used.





The benchmark's simulated users entered a random mix of five different transactions. These transactions simulated a complete, albeit simple, order-entry and order-fulfillment process. The transactions were processed by an Oracle9i relational database containing records for about 30,000,000 customer accounts, 100,000 distinct part numbers, and 1,000 warehouses. Although the benchmark was simple, it generated a load on the system that was typical of most order-entry systems. Key features were the use of substantial amounts of CPU time for each transaction and an intense I/O load that mixed random-access reads and writes with sequential I/O and performance-critical recovery logging. A more complete set of benchmark specifications can be obtained through your Network Appliance sales representative.

4. Test Results

4.1. Statement of Metrics

The following table restates the results of these experiments:

Metric	JBOD	DDA	NFS Jumbo	FC4700 RAID1+0 Raw	FC4700 RAID1+0 Vxfs QIO
Average Response Time (sec)	.12	.12	.15	.15	.16
Disk Capacity Utilized	13%	16%	16%	26%	44%
Disk Access Times (ms)	18	9	11	18	16
Disk Volume Creation	3.5 hrs	< 2 min	< 2 min	4 hrs	4 hrs
Transactions per Interval	47,895	48,215	39,123	38,235	38,569

4.2. Explanation of Metrics

4.2.1. Average Response Times

This metric is the time in seconds that it takes to complete a transaction. The lower the number, the better. Response times of 2 to 3 seconds are usually considered reasonable. A response time below one second is exceptional.

4.2.2. Disk Capacity Utilized

This is the amount of disk space currently in use by the database across all three volumes used in the configuration. A lower number indicates that you have more usable space for future growth.

4.2.3. Disk Access Times

This is the amount of time measured in milliseconds that the disks take to respond to an I/O request. A higher number means that the disk group is responding slowly and will increase the average response times the end user will experience. A lower number represents just the opposite: faster disk access with lower transaction response times, which are good for the end user.

4.2.4. Disk Volume Creation

The disk volume creation metric is the amount of time needed to create and bring online a fully functional disk RAID group. This time includes the amount of time needed to enter and execute the commands necessary to create the disk RAID group volumes in question and have them fully ready to accept data.

4.2.5. Transactions per Interval

This is the number of OLTP transactions that were completed during the measurement interval. The benchmark runs span a total of ten minutes each. The results listed here are the average results for a five-minute transaction window.

5. Technical Details

This section provides more detail into the specifics surrounding this benchmark comparison.

5.1. Database Server

The database server was a Fujitsu GP7000F Model 400R. This machine was a four-way 296MHz SPARC system equipped with 4GB of physical memory. This system was running Solaris8 with the following patches:

- Kernel patch level 108528-11
- 109524-06
- 110383-02
- 111293-03

In the JBOD configurations, a Qlogic 2200 Fibre Channel controller was used in the database server to direct connect the system to the disks. The disk shelves used in the tests were Eurologic FC9 shelves. The disks housed in the Eurologic shelves were Seagate 18GB Cheetah disks. Similarly, the F840 filer uses a Qlogic 2200 Fibre Channel controller to connect to its disk shelves. The disk shelves used in the storage networking configurations, both NFS and DDA, were DS14 shelves. The disks were also Seagate 18GB Cheetah disks. The disk configuration in both environments is identical. The primary exception in the NFS and DDA configurations is the network technology and controller; gigabit ethernet and VI respectively.

In the hardware RAID configurations, a Qlogic 2200 Fibre Channel controller was connected to the EMC FC4700 via a Brocade Fibre Channel switch in a switch fabric point to point configuration.

Here are the parameters that were used in the database server's /etc/system file:

```
* Begin Oracle specific changes
* Semaphores
*------------------
set shmsys:shminfo_shmmax=8589934592
set shmsys:shminfo_shmmin=1
set shmsys:shminfo_shmmni=900
set shmsys:shminfo_shmseg=300

set semsys:seminfo_semmap=600
set semsys:seminfo_semmni=1000
set semsys:seminfo_semmns=1400
set semsys:seminfo_semmnu=800
set semsys:seminfo_semume=400
set semsys:seminfo_semmsl=1400
set semsys:seminfo_semopm=400

*--------------------
* Message Queue
*--------------------
set msgsys:msginfo_msgmap=1024
set msgsys:msginfo_msgmax=65535
set msgsys:msginfo_msgmnb=65535
set msgsys:msginfo_msgmni=1024
set msgsys:msginfo_msgssz=2048
set msgsys:msginfo_msgtql=1024
* End Oracle specific changes
*
*Increases the size of STREAMS synchronization
set sq_max_size = 1600
*
set maxusers = 2048
set nfs:nfs3_max_threads = 48
set nfs:nfs3_nra = 10
* vxvm_START (do not remove)
forceload: drv/vxdmp
forceload: drv/vxio
forceload: drv/vxspec
* vxvm_END (do not remove)

* vxfs_START -- do not remove the following lines:
*
* VxFS requires a stack size greater than the default 8K.
* The following values allow the kernel stack size
* for all threads to be increased to 16K.
*
set lwp_default_stksize=0x4000
set rpcmod:svc_run_stksize=0x4000
* vxfs_END


A script called S99netperf was placed in the /etc/rc2.d directory to
configure various networking parameters. The script is executed upon
```

reboot and its contents are listed below:

```
case "$1" in
          'start')

               echo "Setting local kernel parameters...\c"
               ndd -set /dev/udp udp_recv_hiwat 65535
               ndd -set /dev/udp udp_xmit_hiwat 65535
               ndd -set /dev/tcp tcp_recv_hiwat 65535
               ndd -set /dev/tcp tcp_xmit_hiwat 65535
               ndd -set /dev/ge instance 0
               ndd -set /dev/ge adv_pauseTX 1
               ndd -set /dev/ge adv_1000autoneg_cap 1
               ndd -set /dev/ge adv_1000fdx_cap 1
               echo " "
               ;;

          'stop')

               echo "$0: No parameters changed."
;;

          *)

               echo "Usage: $0 (start|stop)"
               ;;

          esac
          exit 0
```

5.1.1. NFS Mount Options

Here is an example of the file system mount options used for the NFS test. Note the use of **forcedirectio**. With Solaris8, this mount option is essential for database performance. Direct I/O essentially bypasses the Solaris file system cache. So when you read a block of data from the disk, it is read directly into the database's buffer cache, and not in the file system cache. Without direct I/O, a block of data is read into the file system cache, and then into the database's buffer cache. This is essentially double buffering the data, and the database server does not reference the file system cache. This generates a lot of overhead in context switches and system CPU utilization increases resulting in degraded database performance. It should be noted that Direct I/O should only be used on mount points that house database files. You should not use direct I/O on nondatabase files, executables, etc., or when doing normal file I/O operations such as 'dd'. Normal file I/O operations benefit from caching at the system level. You'll also notice the use of TCP instead of UDP. As previously stated, TCP is strongly recommended over UDP due to UDP's potential for packet loss.

```
filer:/vol/vol1 - /mnt/datavol1 nfs - yes rw,bg,hard,intr,proto=tcp,
vers=3,rsize=32768,wsize=32768,forcedirectio
```

5.2. Oracle Settings

Oracle 9.0.1.0.0 for Solaris (32-bit) was used in all tests. For the most part, the same initialization parameters were used for all tests as well. Several exceptions are noted below.

5.2.1. Common

```
_db_file_noncontig_mblock_read_count = 1
_db_writer_max_writes = 640
_db_writer_chunk_writes = 100
_spin_count = 3000
```

```
compatible = 9.0.1.0.0
control_files = $ctrl/ctrl_1,$ctrl/ctrl_2
cursor_space_for_time = TRUE
db_block_buffers = 920000
db_block_size = 4096
db_files = 2000
db_file_multiblock_read_count = 1
db_name = oltp
distributed_transactions = 0
dml_locks = 200
enqueue_resources = 2000
hash_area_size = 0
hash_join_enabled = false
java_pool_size = 4k
lock_sga = false
log_buffer = 1048576
log_checkpoint_interval = 0
log_checkpoints_to_alert = true
max_rollback_segments = 400
open_cursors = 80
open_links = 1
parallel_automatic_tuning = false
parallel_execution_message_size = 4096
parallel_max_servers = 40
parallel_min_servers = 0
parallel_threads_per_cpu = 8
pre_page_sga = true
processes = 225
recovery_parallelism = 40
replication_dependency_tracking = false
session_cached_cursors = 40
sessions = 225
shared_pool_size = 42000000
sort_area_size = 8192
timed_statistics = true
transactions = 275
transaction_auditing = false
transactions_per_rollback_segment = 1
```

5.2.2. NetApp Storage Networking

The Sun implementation of ASYNC_IO for file systems uses a number of lightweight processes (LWPs). In our testing environment, we found that the kernel overhead of these LWP's was higher than the Oracle overhead for using multiple DB Writers(DBWR). Therefore, we disabled DISK_ASYNC_IO for NFS–mounted databases on Solaris.

```
disk_async_io = false
db_writer_processes = 4
```

5.2.3. Hardware RAID

Asynchronous I/O allows the Oracle DBWR process to schedule multiple I/Os without waiting for the I/O to complete. When the I/O completes, the kernel notifies the DBWR using an interrupt. Enabling the following parameter lets Oracle take advantage of asynchronous I/O and avoids having to configure multiple DBWR processes:

```
disk_async_io = true
db_writer_processes = 1
```

5.3. Network Settings

During the software RAID tests, and for JBOD, there was no need for network connectivity since all tests were performed on the database server machine. However, during the storage networking tests, networking was required between the filer and the database server.

For the hardware RAID tests utilizing the FC4700, the database server and the FC4700 were connected using a Qlogic Fibre Channel controller on the database server and a Brocade Fibre Channel switch in a switch fabric point-to-point configuration.

For NFS, the database server and F840 filer were direct-connected via one Gigabit Ethernet controller in each system. To enable jumbo frame support on the database server, the SysKonnect 9843 controller was used. On the filer, jumbo frames is supported on the Gigabit Ethernet controller.

For the DDA tests, the database server and F840 filer were direct-connected via one Emulex/VI controller in each system.

5.4. Disk/Volume Settings

5.4.1. NetApp Storage Networking

The filer used one Qlogic FC-AL controller to connect to the 42 disks that were attached to it. These were then configured into volumes using the following command examples:

```
vol create oltpnfs1 -r 14 -d 7.0 7.1 7.2 7.3 7.4 7.5 7.6 7.8 7.9 7.10
7.11 7.12 7.13 7.14

vol create oltpnfs2 -r 14 -d 7.16 7.17 7.18 7.19 7.20 7.21 7.22 7.24
7.25 7.26 7.27 7.28 7.29 7.30

vol create oltpnfs3 -r 14 -d 7.32 7.33 7.34 7.35 7.36 7.37 7.38 7.40
7.41 7.42 7.43 7.44 7.45 7.46
```

The following volume options were set:

```
vol options oltpnfs1 minra on
vol options oltpnfs2 minra on
vol options oltpnfs3 minra on
vol options oltpnfs1 nosnap on
vol options oltpnfs2 nosnap on
vol options oltpnfs3 nosnap on
vol options oltpnfs1 nosnapdir on
vol options oltpnfs2 nosnapdir on
vol options oltpnfs3 nosnapdir on
vol options oltpnfs1 no_atime_update on
vol options oltpnfs2 no_atime_update on
vol options oltpnfs3 no_atime_update on
```

5.4.2. Hardware RAID

Three Raid Groups consisting of a single LUN were created on the FC4700. Each LUN was comprised of 14 disks in a RAID1+0 configuration.

To create a RAID 1+0 LUN, seven disks on one FC-AL loop were mirrored and striped on seven

disks on a second FCAL loop. This process was repeated two more times until all 42 disks were configured. The final result was three RAID1+0 LUNs.

These configurations gave each disk farm three mount points over which the database could be built. All the Oracle tablespace files were spread evenly across all three mount points.

5.5. Cache Configuration

5.5.1. CLARiiON FC4700

The FC4700 has 2 storage processors (SPs). In addition to other functions, the SP controls access to the FC disks. Each SP has 904MB of cache memory. Of the 904MB of cache, 56MB were configured as read cache and 756MB were configured as write cache.

5.6. Direct Access File System (DAFS) Database Accelerator (DDA) Setup

5.6.1. Server Side

The DDA server configuration is beyond the scope of this report.

For information about managing the DAFS server, see "File Access Using DAFS" in the *Data ONTAP™ 6.2 System Administrator's Guide*. You can obtain this guide from the NetApp on the Web (NOW) site at the following URL: http://now.netapp.com/

Included here is a listing of the DAFS options in use on the filer.

Issue `options dafs` at the filer prompt to obtain this listing.

```
dafs.auth_none.enable on
dafs.checksums.enable off
dafs.default_gid 100
dafs.default_uid 100
dafs.enable on
dafs.idle_timeout 7200
dafs.max_disconnected_sessions 32
dafs.max_pending_requests_server 512
dafs.max_request_size 69632
dafs.max_requests 50
dafs.max_requests_server 1024
dafs.max_response_size 69632
dafs.num_requestd 2
```

5.6.2. Client Side

The DDA installation and configuration procedure is beyond the scope of this report.

You can obtain DDA documentation from the NetApp on the Web (NOW) site at the following URL: http://now.netapp.com/

Included here is an example `ddafs.conf` file located in `/usr/kernel/drv`.

Aside from the default settings, we were required to add the last line, which increased the number of files per DAFS server from 200 to 1,024.

```
#ident "@(#)ddafs.conf 1.2 93/09/09"

# ddafs is a pseudo driver, hence there is only one instance
```

```
name="ddafs" parent="pseudo" instance=0;

max_nics=4;
max_number_of_servers=8;

# This is the max nReq, i.e., the maximum number of dafs operations
# that will be outstanding on one connection at any time. This value
# might also be reduced by the DAFS server.
max_dafs_requests=80;

# Provider name strings - the following name must match the driver
# module names that are reported by the Solaris `modinfo' command. In
# the order specified, dDAFS will search for and use the following
# loadable driver modules with its Kernel VI Providers interface.
#
pnames="orvi";

# ddafs cache enable: turn on/off the ddafs driver cache. Enabling
# this cache will enhance performance in the io data path by optimizing
# IOMMU loads and unloads (aka enabling dvma). Default is enabled.
ddafs_cache_enable=1;

# Oracle DB has several files
max_rawfs_per_svr=1024;
```

NetApp®

Network Appliance, Inc.
495 East Java Drive
Sunnyvale, CA 94089
www.netapp.com