Scalable, fault-tolerant NAS for Oracle—the next generation

Introducing the HP Enterprise File Services Cluster Gateway for Oracle database deployments

Table of contents

Introduction 2
HP Enterprise File Services Clustered Gateway 6
Proof of concept 9
Summary 19
Appendix 20

Abstract: Now there is scalable, modular, high performance NAS for Oracle®10*g* RAC and non-RAC with no single point of failure. This proof of concept shows that the new HP Enterprise File Server Clustered Gateway (EFS-CG) is currently the only scalable, highly available option when NAS is used as the storage architecture for Oracle. In fact, the Clustered Gateway allows customers to leverage existing SAN infrastructure such as HP, EMC, and Hitachi Data Systems.

This paper focuses on a proof of concept of Oracle10*g* R2 Real Application Clusters with the EFS-CG as storage for all database files, Oracle Clusterware files, Oracle Home, and External Tables. It also includes Oracle10*g* R2 performance results with I/O-intensive Oracle workloads. With the HP Enterprise File Server Clustered Gateway, the question is no longer SAN or NAS. Depending on application needs, the customer's choice is always the right choice.

Introduction

For several years Network Attached Storage (NAS) has been rapidly evolving into an acceptable storage option for Oracle® databases. With the advent of Gigabit Ethernet and software advancements in the NFS client space, reasonable performance and solid data integrity are realities on NAS. This is particularly the case when the NFS client is a Linux® system[1].

Oracle Corporation has been vocal about its adoption of NAS for the Oracle on Demand outsourcing business. Oracle also has established the Oracle Storage Certification Program (OSCP); whereby vendors can participate to prove that their NAS solutions are acceptable for Oracle databases. NAS is quite often the simplest, most cost-effective storage approach for Oracle databases.

The emerging storage demand of Grid Computing makes NAS essential. The fruition of Grid Computing will result in connectivity needs for clusters of servers numbering in hundreds of nodes. Building such a large cluster with a Fibre Channel Storage Area Network (SAN) would be a difficult task.

All technology has strengths and weaknesses. With NAS filers, the strong points for Oracle databases are ease of use and often cost. Architecturally speaking, however, NAS filers show weaknesses in the areas of availability and scalability. This is a bold statement given the wide acceptance of NAS for Oracle databases. The goal of this paper is to discuss these characteristics of the NAS filer model and the emerging technology that addresses the issues.

Without a doubt, the majority of Oracle databases deployed on NAS are in the same datacenters with formidable SAN configurations. Supporting two different storage architectures can be a cumbersome task for IT shops. This fact has ushered in the new wave of NAS gateway technology—another focus area of this paper.

Finally, choosing SAN, NAS, or both generally is heavily weighted on performance. To that end, this paper includes an analysis of a proof of concept in which both the availability and performance characteristics of Oracle 10*g*R2 RAC are tested in a new NAS gateway product from Hewlett-Packard called the HP Enterprise File Services Clustered Gateway (EFS-CG).

Goals for the reader

After reading this paper, the reader should have a deeper understanding of many aspects of deploying Oracle on NAS. Additionally, the reader will understand the differences between two traditional NAS architectural approaches:

- Single-headed NAS filers
- Asymmetrical multi-headed NAS gateways

Finally, through a description and analysis of a Proof of Concept test, the reader will learn about the EFS-CG, which provides:

- Multi-headed (scalable) architecture
- Fully symmetrical operations (all NAS heads can present all filesystems)
- Transparent NFS Client failover (highly available)

The primary goal for the reader is an architectural understanding of the technologies being discussed. This paper does not have direct marketing objectives and the only specific NAS technology discussed by name is the new technology that serves as the catalyst for the proof-of-concept testing—the HP EFS-CG.

[1] Network Appliance and Charles Lever in particular were instrumental in making modern NFS client software suitable for Oracle over NFS.

Perhaps the most attractive aspect of deploying Oracle databases on NAS is simplicity. This is especially the case when Real Application Clusters (RAC) are being deployed. System administrators of any Oracle database on NAS find it quite simple to request storage from the storage administration group and simply mount the filesystem on the database server.

NAS value proposition for Oracle database deployment

Simplicity

Industry-leading NAS providers have invested significantly in educating the industry on the value of deploying Oracle databases on NAS.

Perhaps the most attractive aspect of deploying Oracle databases on NAS is simplicity. This is especially the case when Real Application Clusters (RAC) are being deployed. System administrators of any Oracle database on NAS find it quite simple to request storage from the storage administration group and mount the filesystem on the database server. Once the NFS filesystem is mounted, the server administrator is completely out of the loop for storage issues. The space is given to the Oracle DBA team, and they use it as per the requirements of the database—no more interaction with the system or storage administration groups. Contrast this to the amount of system administrative overhead when deploying Oracle databases on raw partitions in a SAN.

Whether simple raw datafiles or ASM, administrative overhead is required. First, the database administrator has to determine the list of singleton LUNs needed, such as the Oracle clusterware files, raw datafiles, or ASM disk group partitions. The system administrator then requests these LUNs from the storage group and proceeds to work out connectivity, dealing with such issues as loading host bus adaptors, getting the LUNs presented as character special raw devices, permissions, and in the case of Linux, raw(8) binding and ASMLib configuration. This activity is much more complex than mounting filesystems.

With the NAS model, deploying RAC is much simpler for the DBA than using the typical SAN deployment model[2], which is a combination of OCFS2 for Oracle Clusterware and raw partitions for ASM (or simple raw datafiles), and Ext3 or OCFS2 for Oracle Home. With NAS the DBA is notified when the filesystem is mounted and work can begin. Simple files or large files as ASM disks, the choice is up to the DBA. The DBA can store everything associated with the Oracle database in the NFS filesystems and all RAC servers have complete shared read/write access. There, however, lies the potential performance bottleneck and availability concerns discussed later in this paper.

Cost reduction

Depending on the type of deployment, the NAS model can offer significant cost benefit compared to SAN. Certainly, test and development systems running Linux are less expensive when their storage connectivity is based on NFS instead of SAN volumes. No Fibre Channel HBAs to purchase, no Fibre Channel cabling, and most importantly, no expensive ports on a high-end Fibre Channel switch. NAS-based production systems also can be less expensive if they are RAC clusters with large node counts. A port on a 64-port Fibre Channel switch is much more expensive than a port on a small 8-port switch. Accordingly, configuring a RAC cluster for Grid Computing with large numbers of servers, each with multiple paths to storage, can be extremely cost prohibitive.

Some would argue that the NAS model has reduced administrative overhead. This is questionable, however,

[2] The scope of this paper is focused only on deployment options that fit under the Unbreakable Linux support model—a very short list limited to raw disk (e.g., ASM, raw datafiles) and OCFS. Clearly missing is the comparison to commercially available SAN-based, third-party Cluster filesystems, which have their own value propositions but do not fit within the constraints of Unbreakable Linux. Although third-party CFS are not included in the Unbreakable support model, they can be listed on Metalink as a "validated third-party CFS." None of this concern over Unbreakable Linux amounts to anything unless you are running Linux.

because most NAS-based RAC deployments are in IT shops that also have significant SAN investment. An increase in storage administration overhead occurs, as there are disparate storage systems to maintain. Unless, of course, the NAS device is a SAN gateway device. Without a SAN gateway, adding NAS into an environment with an established SAN creates the "storage sprawl" or "vendor sprawl" effect.

Traditional NAS

As already covered, NAS is an established, acceptable storage option for Oracle RAC and non-RAC alike. Acceptable, however, is the key word. No technology is perfect and the most popular NAS options being deployed today have weaknesses that should be of concern to an Oracle shop. The following sections cover two of the most common NAS architectures:

- Single-headed filer
- Asymmetrical multi-headed NAS device

Because this paper does not serve a marketing purpose, vendor names for various architecture types are not used.

Single-headed filers, the traditional NAS architecture

The most common NAS devices available on the market today are "single-headed" devices commonly referred to as filers. The term single-headed means that there is a single computer with a processor, memory, and I/O capability handling all accesses to a set of filesystems on disk. These filers are powerful but because of their architecture they possess two troubling characteristics:

- Single point of failure
- Performance bottleneck

Single point of failure

Having data that can only be accessed through a single filer is a single point of failure (SPOF) that cannot be overlooked. If that data is Oracle Clusterware files (e.g., Oracle10*g* RAC CRS, Oracle9*i* OCMS) or any Oracle SYSTEM tablespace, a single failure of the filer head will cause a total RAC outage. RAC is highly available, but the storage really is not. Some mitigating solutions for this single point of failure are presented later in this paper.

Performance bottleneck

Many Oracle databases are deployed today on single-headed NAS devices. However, if all data must pass through a single filer for a given set of data, the entire database performance is limited to the throughput of that single filer.

In the case of RAC, few nodes are needed to saturate completely a single-headed filer. The reason is that on the inside, the leading single-headed filers are based on the same Intel®-based servers typically used for Linux RAC. If you peel away the covers on today's leading NAS filer, you will see an Intel-based server. If you deploy a four-node RAC configuration with Xeon™-based servers and place the database in the filer, you may experience a bottleneck. The Xeon-based system in the filer cannot move more data across its system bus than any of the Xeon-based servers in the RAC configuration. So, given an I/O intensive operation such as any Parallel Query operation, the RAC nodes will be I/O constrained.

What if the database servers attached to the filer are significantly more capable of I/O than the filer? Consider a four-node RAC configuration of four-socket, dual-core AMD Opteron™ Servers. How scalable is that going to be? What about just a single large SMP from any manufacturer? The simple answer to these rhetorical questions is that the configuration will not scale.

A word about NVRAM cache:

The most common single-headed filers on the market offer an NVRAM cache that dramatically lowers the I/O latency for writes. These NVRAM cache cards are attached to the filer system bus. While they do allow writes to be serviced quickly, they do not address the throughput limitation of the filer at all. Quite the opposite is true. When the NVRAM cache is full, I/O requests will queue up while the cache is being checkpointed out to disk. Some productions sites must dedicate a filer solely for Oracle Redo Logging for this reason.

Mitigating solutions for single-headed filer issues

To address the inherent single point of failure issues with single-headed filers, some vendors offer clustered filers. The term "clustered filer" conjures up a lot of misconceptions. Cluster filers are just that, a cluster of single-headed filers. Neither can provide access to the other's data.

So, to mitigate the impact of a failure of a NAS filer, the suggested remedy is to buy another equal filer and configure a cluster filer. Once the investment has been made in 100% overhead, the configuration can support failover in the event of a filer outage.

Failover, in the case of a cluster of single-headed filers, is a non-surgical procedure known as "sever and reattach" because the NFS handles are usually invalided on the client after a filer failure[3]. The client can, however, remount the filesystems from the other filer after some failover time—generally 45 to 180 seconds. Sever and reattach should immediately raise concern. In the case of RAC, Oracle treats this as any ordinary loss of storage—akin to a single array failure in a Storage Area Network (SAN). But unlike the single array SAN scenario, the clustered filer configuration should have no single point of failure. However, if the NAS filer presenting the most important application table happens to fail, the files will not be available until cluster failover is complete, and even then the instances will have to restart. While RAC supports multiple instances of a database, there is only one database. Loss of I/O connectivity to any of it is a single point of failure.

Clustered filers can help mitigate the performance bottleneck inherent in single-headed filers, to a degree. Clustered filers can be configured such that one filer serves a set of filesystems and the other a completely separate set of filesystems—basically, a partitioning effort. If I/O demand on a single directory is saturating a filer, no solution is available other than to move data manually between filers. What if the hot data is not the same hot data that saturates the filer six months from now?

Asymmetrical multi-headed NAS

Asymmetrical multi-headed NAS (AMHN) devices are not filers, per se. They are generally SAN-gateway devices. Because you probably have a SAN already, you simply request LUNs from the storage group and attach the LUNs to the gateway device. From there, the SAN gateway device presents filesystems as would any other NAS device.

Availability

Asymmetrical multi-headed NAS devices are similar to clustered single-headed filers but differ in one significant way—they support more than two NAS heads. All NAS heads in these devices do indeed have connectivity to all filesystems, but only one NAS head can present a filesystem at any given time. Therefore, this architecture is deemed asymmetrical. For instance, in a configuration with eight NAS heads and eight filesystems, each filesystem can be presented by only one NAS head at a time. In the event of a NAS head outage, the filesystem will fail over to one of the other heads. The similarities, conversely, between asymmetrical multi-headed NAS and clustered single-headed filers are easily seen.

Just like their clustered single-headed cousins, these NAS devices also suffer a sever and reattach impact in the event of a head failure. Failovers are no more transparent than they are with clustered single-headed filers. Failover times on these devices can be quite long. During failover, database files in the filesystem being failed-over are inaccessible.

Scalability

Asymmetrical multi-headed NAS devices have the same scalability characteristics as clustered single-headed filers. As mentioned above, only a single NAS head can present any given filesystem. As is the case with clustered single-headed filers, the game of moving hot data around to other filesystems to offload over-burdened NAS heads is very much the case.

From an architecture standpoint, the only difference between clustered single-headed filers and asymmetrical multi-headed NAS devices is that the latter supports more than two heads. They are more modular but still require physically partitioning data between filesystems to achieve a crude form of scalability.

[3] "Often, the transfer of data service is transparent to end users and applications" http://www.netapp.com/products/software/clustered.html

Table 1:
Key differences between the HP EFS-CG, clustered single-headed filers, and asymmetrical multi-headed NAS devices include:

Availability	The EFS-CG includes a specialized, proprietary NFS server implementation that combines with virtual host functionality to support completely transparent NFS client failover if a NAS head failure occurs or maintenance needs to be performed.
Modularity	The EFS-CG supports between two and 16 industry-standard servers for NAS heads.[4]
Scalability	The EFS-CG supports truly scalable NAS. Because of the built-in, fully symmetric, distributed cluster filesystem, all NAS heads can present any or all filesystems with fully coherent direct read/write access. Each NAS head can present filesystems over as many as three network interfaces for a system total of 48 Gigabit Ethernet interfaces for NFS traffic.
Standards	The NAS heads that comprise the EFS-CG are Intel-compatible Linux servers without proprietary hardware. Moreover, the NAS heads run Linux and, although there are proprietary kernel enhancements, administering the NAS heads is no different than any other Linux system.

HP Enterprise File Services Clustered Gateway

Overview

HP Enterprise File Services Clustered Gateway, or EFS-CG, is a new and radically different NAS technology from those already discussed in this paper. The key differences between the EFS-CG and both clustered single-headed filers and asymmetrical multi-headed NAS devices are explained in **Table 1** above.

The EFS-CG is a SAN gateway NAS device. Often this is seen as investment protection. After all, as mentioned previously in this paper, most sites that deploy Oracle today on NAS filers do so in the same datacenter where there is an existing, formidable SAN infrastructure. Such an environment is the perfect infrastructure for deploying a SAN gateway product like the EFS-CG. The EFS-CG plugs into the SAN (e.g., HP, EMC, Hitachi, IBM, etc.) and presents clustered filesystems via NFS. This is an entirely different scenario than carting in a pair of clustered NAS filers with their own internal storage and setting them next to the existing SAN equipment.

Presenting NAS storage through a gateway device from an established SAN infrastructure can be significantly less expensive than the same capacity in a set of NAS filers. This point deserves more attention.

Choosing a NAS model for RAC over SAN usually yields cost savings because the RAC nodes themselves do not require multiple Fibre Channel host bus adaptors with paths to multiple switches. Fewer servers connected to SAN switches can potentially alleviate the need to purchase extremely large switches. However, the net gain of these savings is canceled out if the storage itself is more expensive on a per-terabyte basis. With clustered NAS filers (single-headed), the cost of storage also must reflect the clustering architecture they support. With clustered single-headed filers, clustering is limited to "pair-wise" clusters. If both filers in a clustered filer configuration become saturated, adding a third filer actually means adding two more filers—if availability is a concern. Recent analysis of published list prices for industry-leading clustered NAS filers revealed customers pay approximately 300% premium as compared to equal capacity via EFS-CG technology, thus being threefold more costly and not scalable, nor truly available as explained earlier.

Availability characteristics

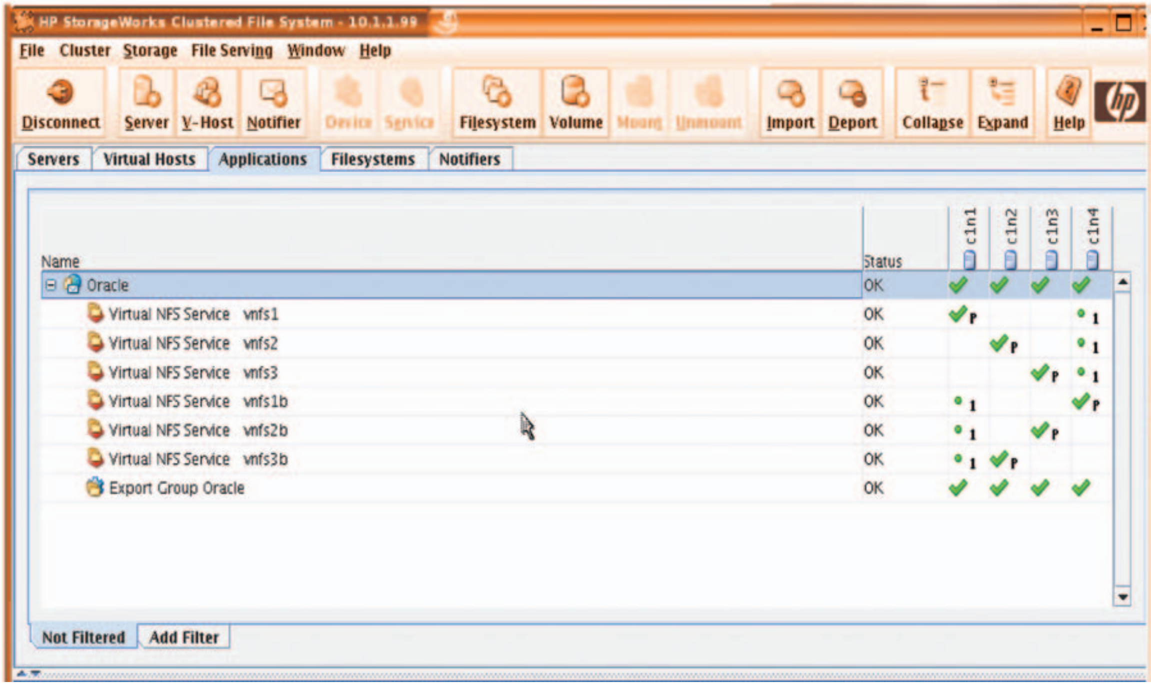
The EFS-CG provides a truly revolutionary NFS server implementation with special support for highly available NFS. All NFS export groups are presented to NFS clients via Virtual NFS Service (VNFS). The VNFS technology is important for two reasons:

- **Failover.** If a NAS head in the EFS-CG fails, the Virtual NFS Services running on the failed nodes will be transparently failed over to a backup NAS head. The NFS clients and the processes with open file handles on the filesystems involved will not be affected in any way.
- **Re-hosting.** Using the EFS-CG Management GUI or CLI, the administrator can move an active NFS service from one NAS head to the other for load balancing or maintenance. This operation also is fully transparent at the NFS client level and you do not need to stop applications—just a simple GUI drag and drop.

A VNFS is a combination of Virtual Host IP and a proprietary enhancement to the NFS server stack to support completely transparent NFS client failover. The filesystems remain accessible without re-mounting and, most importantly, processes with active file handles accessing files in the NFS filesystems are not impacted.

[4] In their current incarnation, EFS-CG heads are dual-processors AMD servers (e.g., HP Proliant DL-385).

Figure 1:
Enterprise File Services Cluster Gateway graphical user interface



The benefit of this technology in an Oracle deployment should be quite clear. As discussed earlier, placing an Oracle database in a NAS device other than the EFS-CG leaves a single point of failure. All other NAS technology completely disconnects all NFS clients in a failover event, whether on clustered filers or asymmetrical multi-headed filers. If a NAS head presenting datafiles should fail, these other NAS technologies will cause a global RAC meltdown. With RAC there is one copy of the database and suffering a “sever and reattach” sort of NAS head failure will impact every instance of the RAC cluster. This fact does not play well into the grid computing story. Imagine a 48-node RAC grid with 48 instances crashed because the NAS head presenting an essential tablespace has crashed.

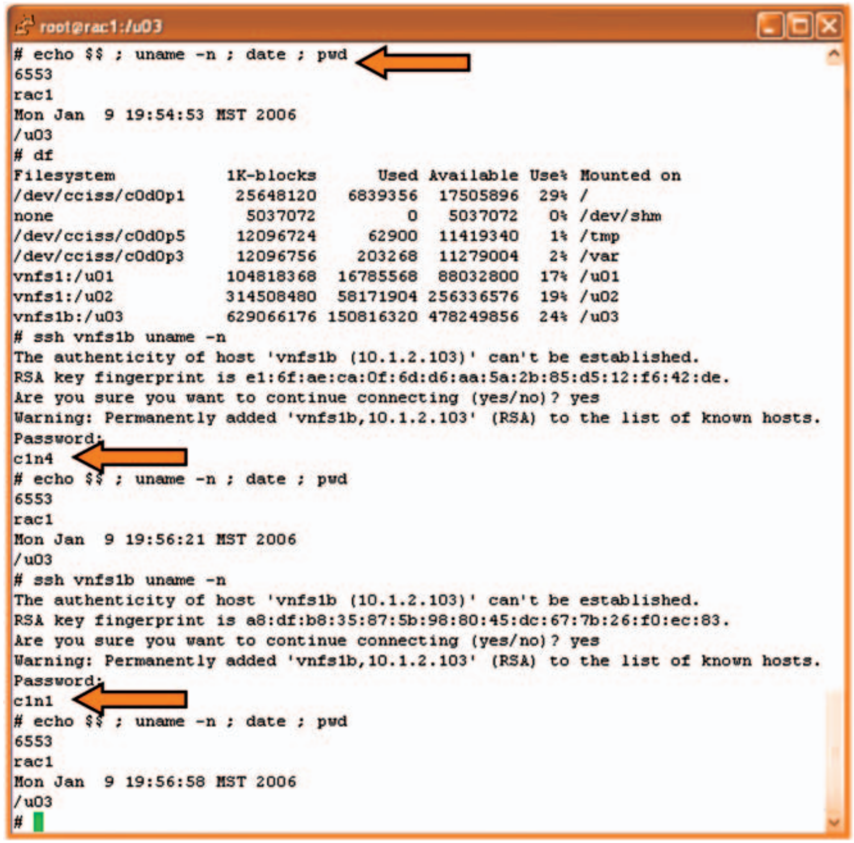
Figure 1 shows a portion of the EFS-CG management GUI. On the horizontal plane are a set of Virtual NFS Services. On the right-hand side along the vertical plane are the NAS heads by name. In this case there are four NAS heads (**c1n1 – c1n4**). This screen capture was taken from the Proof of Concept system described later in this paper and shows the status of the Virtual NFS Services.

For instance, the VNFS called **vnfs1**, and therefore all the filesystems being presented by **vnfs1**, are currently hosted by NAS head number 1 (**c1n1**). This status is established by the cell for that row, which shows P for primary, under the **c1n1** column. VNFS1 will fail over to **c1n4** because the numeral 1, short for first backup, appears under the **c1n4** column for that same row. Also, the VNFS called **vnfs3b** and **vnfs2** concurrently are hosted by NAS head number 2 (**c1n2**). Re-hosting a VNFS from one node to the other is a simple click, drag-and-drop operation.

Figure 2 depicts the transparent nature of the EFS-CG Virtual NFS Service technology. This Linux session shows three clear signs of transparent Virtual NFS failover/re-hosting.

Figure 2 ends by validating that the shell process still has a PID of 6553 and still is executing in /u03, which is an NFS mount. The shell process (6553) has at least one active file descriptor in its current working directory (CWD), which is how bash(1) works. The shell process was uninterrupted while the presentation of /u03 was moved from one NAS head to another. If /u03 were being served up by a cluster of single-headed filers or an asymmetrical multi-headed filer, the shell process 6553 would have died. Instead, the bash process was unaffected during the 37 seconds when /u03 moved from one NAS head to another.

Figure 2
Determining which EFS-CG NAS head is presenting a filesystem; re-hosting operations



```
root@rac1:/u03
# echo $$ ; uname -n ; date ; pwd
6553
rac1
Mon Jan  9 19:54:53 MST 2006
/u03
# df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/cciss/c0d0p1     25648120   6839356  17505896  29% /
none                   5037072         0   5037072   0% /dev/shm
/dev/cciss/c0d0p5     12096724     62900  11419340   1% /tmp
/dev/cciss/c0d0p3     12096756    203268  11279004   2% /var
vnfs1:/u01           104818368  16785568  88032800  17% /u01
vnfs1:/u02           314508480  58171904 256336576  19% /u02
vnfs1b:/u03          629066176 150816320 478249856  24% /u03
# ssh vnfs1b uname -n
The authenticity of host 'vnfs1b (10.1.2.103)' can't be established.
RSA key fingerprint is e1:6f:ae:ca:0f:6d:d6:aa:5a:2b:85:d5:12:f6:42:de.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'vnfs1b,10.1.2.103' (RSA) to the list of known hosts.
Password:
cln4
# echo $$ ; uname -n ; date ; pwd
6553
rac1
Mon Jan  9 19:56:21 MST 2006
/u03
# ssh vnfs1b uname -n
The authenticity of host 'vnfs1b (10.1.2.103)' can't be established.
RSA key fingerprint is a8:df:b8:35:87:5b:98:80:45:dc:67:7b:26:f0:ec:83.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'vnfs1b,10.1.2.103' (RSA) to the list of known hosts.
Password:
cln1
# echo $$ ; uname -n ; date ; pwd
6553
rac1
Mon Jan  9 19:56:58 MST 2006
/u03
#
```

- **First arrow.** The session first establishes that:
  - The shell Process Id (PID) is 6553
  - The shell is executing on the RAC node called **rac1** (NFS client)

  A df(1) command then shows that /u03 is presented by the EFS-CG via the Virtual NFS Service called **vnfs1b**.
- **Second arrow.** The **uname**(1) command is executed remotely (via the ssh(1)[5]) on the EFS-CG NAS head that is currently hosting the **vnfs1b** Virtual NFS Service. The **uname**(1) command shows that the /u03 filesystem is being presented from **cln4**, the fourth NAS head in the EFS-CG.
- **Third arrow.** Sometime between 19:56:21 and 19:56:58, the vnfs1b Virtual NFS service was either failed-over or re-hosted to cln1, the first NAS head in the EFS-CG.

Performance characteristics

Cluster Volume Manager

The EFS-CG includes an integrated Cluster Volume Manager. LUNs are presented to the EFS-CG by the SAN administrator and imported into the EFS-CG. Internal to the EFS-CG, these LUNs are then made into a striped (RAID 0) volume of user-defined stripe width. The LUNs are fault tolerant (e.g., RAID 1) at the SAN storage array level. The EFS-CG can support single Cluster Volumes of up to 16TB of redundant, high-performance (RAID 1+0) storage. The size of cluster volumes can be dynamically increased. Additionally, the EFS-CG supports large numbers of filesystems—up to a maximum theoretical limit of 512 16TB filesystems.

Cluster filesystem

All NAS devices available today present an internal filesystem of some sort via NFS. In the case of the EFS-CG, the filesystem is the PolyServe™ Matrix Server™ cluster filesystem, which is fully symmetric and distributed. This means that all NAS heads have equal, direct read/write access to all filesystems. Combining the cluster filesystem with the cluster volume manager is the foundation for the tremendous scalability this architecture offers.

Without the cluster filesystem, the EFS-CG would be no different than asymmetrical multi-headed NAS devices on the market today. The net effect would be that only a single NAS head would be able to present any given filesystem. The cluster filesystem is the glue that enables true scalability. Also, the size of EFS-CG cluster filesystems can be dynamically increased.

NAS head count

As mentioned above, the EFS-CG supports from 2 to16 NAS heads. When the cluster filesystem is combined with the cluster volume manager, a single filesystem (up to 16TB) can be presented by up to 16 NAS heads, each with coherent, direct read/write access.

Network interfaces

Each NAS head in the EFS-CG supports several Gigabit Ethernet network interfaces. Up to three interfaces can be used for serving NFS traffic. A fully configured EFS-CG with 16 nodes will support up to 48 GigE data paths to one or more filesystems.

The EFS-CG technology can be deployed on any industry standard server. No technical reason prevents the deployment of extremely powerful servers such as the HP Proliant DL-585 for use as NAS heads, which would support as many as 12 NFS I/O paths per NAS head.

[5] The NAS Heads are Linux servers and all Linux tools are at the administrator's disposal. This environment is much more flexible than that available to the administrator when connected to proprietary Operating Systems running on traditional NAS filers.

Bringing it all together—in an Oracle context

Perhaps the best way to elaborate on the potential for the EFS-CG in an Oracle deployment is to look at the extremes. Consider that a single Oracle10*g* BIGFILE tablespace, based upon a 4KB blocksize, can grow to 16TB. The caveat is that a BIGFILE tablespace is comprised of a single datafile. A single file must reside in a single filesystem. The EFS-CG is capable of supporting the 16TB BIGFILE tablespace, but it can do so with I/O scalability. Every single block of data in that one BIGFILE tablespace can be concurrently accessed through 48 network interfaces via 16 NAS heads.

Further, the EFS-CG can support a 48-node RAC cluster where each and every server has a dedicated GigE NFS data path with full read/write capability on the entire 16TB BIGFILE tablespace. If every instance in the 48-node RAC cluster were performing a full-table scan of a table in that BIGFILE tablespace, their I/O would be serviced at full GigE bandwidth. If all 48 nodes of the RAC cluster were participating in an index creation using Intra-node Parallel Query, nothing stands in the way of having the entire index reside in a single datafile.

This example should make the contrast clear between the EFS-CG and other NAS technology. Other NAS technology needs to cluster and split the tablespace across several filesystems to get multiple NAS heads to partake in the I/O load—each filesystem served by a single NAS head. However, a BIGFILE tablespace cannot be split into multiple datafiles anyway, so the comparison is moot. No solutions, other than the EFS-CG, can scale to 16 NAS heads at all, much less in a symmetrical manner.

This example is beyond the "normal" Oracle deployment and BIGFILE tablespaces are not the norm but sheds light on how ASM on NAS works.

A word about ASM on NAS

ASM on NAS is implemented by first creating large files on the NAS filesystem and then adding them to ASM as a "disk" in a disk group. The preceding BIGFILE example is the exact I/O profile of ASM on NAS. With other NAS architectures, any given ASM file can only be presented by a single NAS head. With the EFS-CG however, ASMcan be configured as a single file up to 16TB. Subsequently, up to 16 NAS heads can present full, direct read/write access to the ASM file.

In the example the BIGFILE tablespace doesn't necessarily represent the norm, does that mean the EFS-CG is of no benefit for usual Oracle deployments?

Consider even a "small" Oracle Linux RAC cluster of four nodes. As described earlier, the RAC cluster would possess more I/O bandwidth than the largest single-headed NAS filer on the market[6]—about 400% more I/O bandwidth. You can split the data over multiple filesystems and go with a clustered NAS filer configuration and have "4 over 2" arrangement, but what happens when the fifth node is added? There is a risk of a LGWR and DBWR bottleneck also because of NVRAM checkpointing in the NAS filers.

ASM on NAS is implemented by first creating large files on the NAS filesystem and then adding them to ASM as a "disk" in a disk group. The preceding BIGFILE example is the exact I/O profile of ASM on NAS. With other NAS architectures, any given ASM file can only be presented by a single NAS head. With the EFS-CG however, ASM can be configured as a single file up to 16TB.

With the EFS-CG, should the RAC cluster grow from two to four and then eight nodes, you have the NAS modularity to simply add NAS heads and the scalability to actually benefit from them. No need to move data to different filesystems, simply add NAS heads—an operation that can be done without disruption. In the non-RAC case, if new database deployments are available, you do not need to buy more filers. Instead, create a new directory, place the new database in it, and present the filesystem through any, or all, of the NAS heads.

Proof of concept

The HP StorageWorks team partnered with PolyServe Matrix Server™ to perform a proof of concept test with Oracle on the EFS-CG. The goal of the testing was to answer three very important questions:

1. **Scalability** — Can the EFS-CG improve Oracle performance as NAS heads are added?
2. **Availability —** Can Oracle Real Application Clusters continue operations with complete transparency in the event of an EFS-CG NAS head failure?
3. **OLTP** — Can the EFS-CG perform under stressful OLTP workloads?

6 A single-headed filer with Intel Xeon technology serving a four-node RAC cluster.

Proof of concept test configuration description:

The test configuration used for the proof of concept consisted of six dual-processor AMD-based servers as NFS clients. Four servers were configured with Oracle10*g* Release 2 Real Application Clusters. The other two NFS client nodes were used for Oracle10*g* Release 1 and Oracle9*i* non-RAC databases to show that the EFS-CG is agnostic about database version. All of the Oracle servers were configured with 64-bit Red Hat Enterprise Linux® AS/ES 4.0. The NFS clients each had two GigE paths for NFS traffic.

The Enterprise File Services Cluster Gateway had four NAS heads. The EFS-CG presented four filesystems contained in cluster volumes that were a collection of LUNs in the SAN.

The four filesystems presented by the EFS-CG were:

- **/u01.** This filesystem contained all Oracle executables (e.g., $ORACLE_HOME)
- **/u02.** This filesystem contained the Oracle10*g*R2 clusterware files (e.g., OCR, CSS), datafiles, and External Tables for ETL testing
- **/u03.** This filesystem was lower-performance space used for miscellaneous tests such as backup disk-to-disk
- **/u04.** This filesystem resided on a high-performance volume spanning two storage arrays containing the main benchmark database

Both /u01 and /u02 were exported from the EFS-CG by separate NAS heads[7] as shown in **Figure 3**. The following lines were added to the /etc/fstab file for /u01 and /u02 on each RAC node.

```
vnfs1: /u01 /u01    nfs rw,bg,hard,nointr,tcp,vers=3,timeo=300,rsize=32768,wsize=32768,actimeo=0
vnfs3b:/u02 /u02    nfs rw,bg,hard,nointr,tcp,vers=3,timeo=300,rsize=32768,wsize=32768,actimeo=0
```

Because /u03 and /u04 contained hot objects such as External tables and databases, they were presented from the EFS-CG via two different Virtual NFS Services and two different NAS heads as shown in **Figure 3**. The first two RAC nodes, rac1 and rac2, used these /etc/fstab entries for mounting /u03 and /u04:

```
vnfs1b:/u03 /u03 nfs rw,bg,hard,nointr,tcp,vers=3,timeo=300,rsize=32768,wsize=32768,actimeo=0
vnfs1b:/u04 /u04 nfs rw,bg,hard,nointr,tcp,vers=3,timeo=300,rsize=32768,wsize=32768,actimeo=0
```

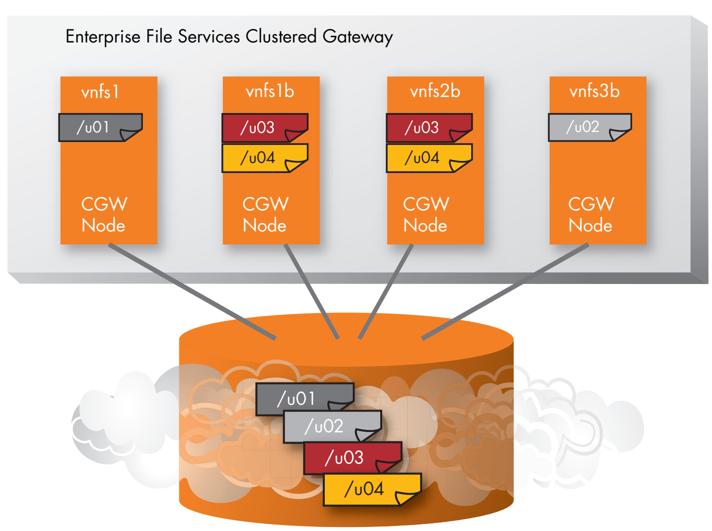
To route NFS traffic across a different NAS head for rac3 and rac4, those nodes were configured with the following /etc/fstab entries for mounting /u03 and /u04:

```
vnfs2b:/u03 /u03 nfs rw,bg,hard,nointr,tcp,vers=3,timeo=300,rsize=32768,wsize=32768,actimeo=0
vnfs2b:/u04 /u04 nfs rw,bg,hard,nointr,tcp,vers=3,timeo=300,rsize=32768,wsize=32768,actimeo=0
```

Configured in this fashion, all database I/O for rac1 and rac2 were serviced by one NAS head and an entirely different NAS head handled all database I/O requests for rac3 and rac4. All ORACLE_HOME traffic was serviced by a dedicated NAS head as was all Oracle Clusterware I/O. All of these Virtual NFS Services could have just as easily run on any, or even a single, NAS head.

In all cases, each VNFS can failover to any of the other NAS heads in the event of a head failure (e.g., hardware) and, as mentioned above, failover of any VNFS is completely transparent to the NFS client.

Figure 3
Proof of concept
EFS-CG configuration



[7]Note that it was not necessary to use two NAS heads to serve ORACLE_HOME and the Clusterware files. That choice was made in order to test wide degrees of functionality.

Figure 4
Associating filesystems with NAS heads

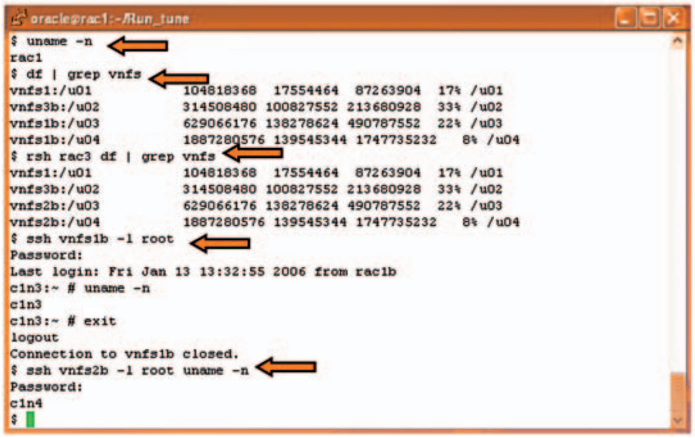
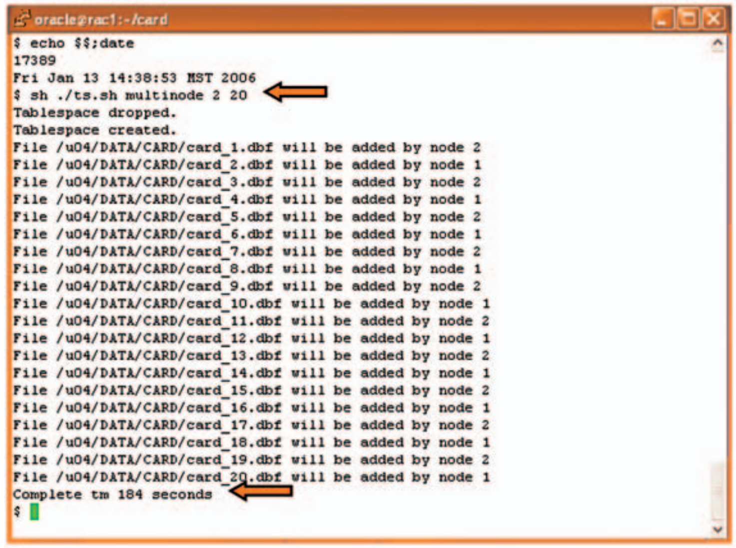


Figure 5
RAC tablespace creation throughput with single-headed EFS-CG



The screen capture in **Figure 4** explores this type of EFS-CG configuration. It shows the following:

- **First arrow.** The session starts by showing that it is a Linux shell on the RAC node called rac1
- **Second arrow.** A df(1) command on rac1 shows that /u04 is being presented to this RAC node via the Virtual NFS Service called vnfs1b
- **Third arrow.** A df(1) command, executed remotely via rsh(1), on the RAC node called rac3 shows that the /u04 filesystem is being presented to rac3 via the Virtual NFS Service called vnfs2b
- **Fourth arrow.** The ssh(1) command is used to log into the EFS-CG node presenting /u03 and /u04 via the VNFS called vnfs1b. The uname(1) command shows that the EFS-CG NAS head is a node called c1n3
- **Fifth arrow**. The ssh(1) command again is used to determine which EFS-CG NAS head is presenting /u03 and /u04 via the VNFS called vnfs2b. There, the uname(1) command reports that the node name is c1n4, which was the fourth NAS head in the EFS-CG

Scalability

The first set of scalability testing focused on multi-headed, single filesystem throughput. This set of tests included:

- Adding space to the database
- Parallel query scans

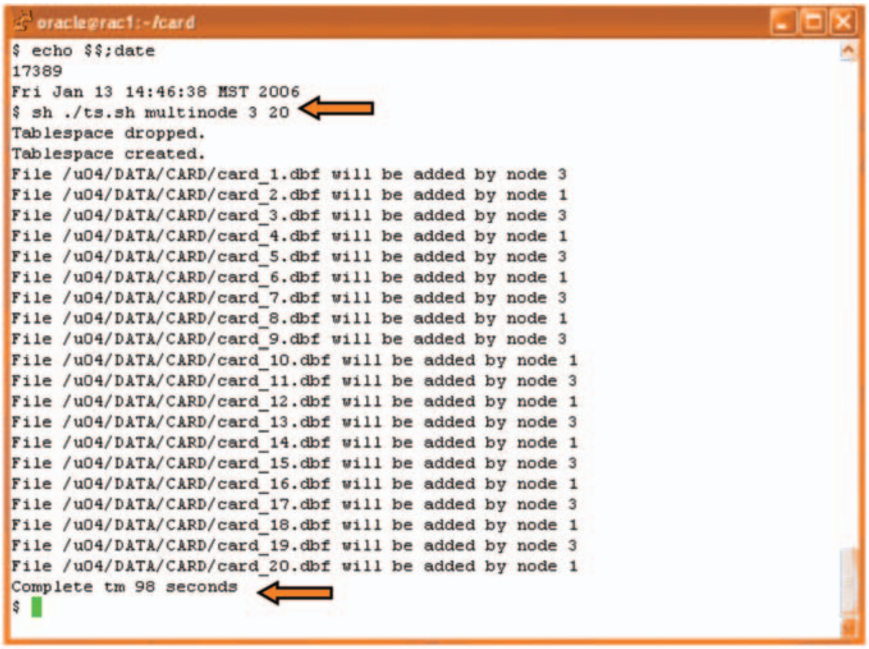
Adding space to the database

With the ALTER TABLESPACE ADD DATAFILE command, space can be added to the database in parallel. This is a 100% write operation that Oracle performs with large multiblock writes to initialize all blocks in the file.

A test was set up to initialize a 20GB tablespace. To drive up the parallelism at the Oracle end, the test consisted of 20 concurrently executed ALTER TABLESPACE ADD DATAFILE statements, each adding a 1GB datafile to a newly created tablespace with a small, 8M, primary datafile. Because timing of the test starts after the tablespace is created, the only timed portion is the concurrent ALTER TABLESPACE ADD DATAFILE operations.

The ts.sh script, listed in the appendix, was used for this test. This script uses promiscuous rsh (1) to execute SQL*Plus commands on the local node and another node as per the second argument of the program. **Figure 5** shows an example of this testing. The first arrow shows that the command was executed with the second argument

Figure 6
RAC tablespace creation throughput with multi-headed EFS-CG single filesystem



```
oracle@rac1:~/card
$ echo $$;date
17389
Fri Jan 13 14:46:38 MST 2006
$ sh ./ts.sh multinode 3 20
Tablespace dropped.
Tablespace created.
File /u04/DATA/CARD/card_1.dbf will be added by node 3
File /u04/DATA/CARD/card_2.dbf will be added by node 1
File /u04/DATA/CARD/card_3.dbf will be added by node 3
File /u04/DATA/CARD/card_4.dbf will be added by node 1
File /u04/DATA/CARD/card_5.dbf will be added by node 3
File /u04/DATA/CARD/card_6.dbf will be added by node 1
File /u04/DATA/CARD/card_7.dbf will be added by node 3
File /u04/DATA/CARD/card_8.dbf will be added by node 1
File /u04/DATA/CARD/card_9.dbf will be added by node 3
File /u04/DATA/CARD/card_10.dbf will be added by node 1
File /u04/DATA/CARD/card_11.dbf will be added by node 3
File /u04/DATA/CARD/card_12.dbf will be added by node 1
File /u04/DATA/CARD/card_13.dbf will be added by node 3
File /u04/DATA/CARD/card_14.dbf will be added by node 1
File /u04/DATA/CARD/card_15.dbf will be added by node 3
File /u04/DATA/CARD/card_16.dbf will be added by node 1
File /u04/DATA/CARD/card_17.dbf will be added by node 3
File /u04/DATA/CARD/card_18.dbf will be added by node 1
File /u04/DATA/CARD/card_19.dbf will be added by node 3
File /u04/DATA/CARD/card_20.dbf will be added by node 1
Complete tm 98 seconds
$
```

Figure 7
Tablespace size validation

```
oracle@rac1:~/card
SQL> @tsinfo

TS_SIZE_GB
----------
20.0078125

SQL>
```

set to 2. This means that 10 of the sqlplus execution streams will execute on rac1 and the other 10 on rac2. The files were initialized with zeros using the dd (1) command, and the REUSE verb was used in the ALTER TABLESPACE command to ensure that the cost for initialization was the same under both executions of ts.sh. As seen in the code listing in the appendix, timing of the ALTER TABLESPACE statements is reported before the program exits. The first time ts.sh was executed, it took 184 seconds to add 20G of space to the tablespace.

Without interruption, the script was re-run and took 98 seconds, as shown in **Figure 6**, yet the filesystem was the same and the files created were the same. This represents a speedup from 111 to 208MB/s (94% scalability) of large multiblock writes—a clear example of serving a single filesystem via multiple NAS heads.

The explanation for the increased performance is transparent, scalable multi-headed NAS. The first ts.sh execution used RAC servers rac1 and rac2. The second used rac1 and rac3. As shown in **Figure 3**, the /u04 filesystem was presented by two EFS-CG NAS heads. Because /u04 was served by a single head to nodes rac1 and rac2, they were limited to the bandwidth of the network interface associated with the Virtual NFS service called vnfs1b. Note, the limit of one GigE NFS path to each EFS-CG head presenting /u04 was an artificial limit imposed for the purpose of the Proof of Concept. The EFS-CG heads support three GigE data paths per head by default.

The second time ts.sh was executed **(Figure 6),** it used RAC nodes rac1 and rac3. In this case, both vnfs1b and vnfs2b were involved because rac3 uses vnfs2b as its VNFS path to /u04. To get this level of throughput with any other NAS technology, the tablespace would have to consist of datafiles from multiple directories, each in filesystems presented by a different NAS heads.

Figure 7 shows the tsinfo.sql script reporting the tablespace size as 20GB after the ts.sh script was executed.

Write-intensive workload summary

To summarize the scalability and throughput test of adding space to the database, **Figure 8** shows that adding EFS-CG NAS heads to a write-intensive workload yielded nearly linear scalability on the Proof of Concept configuration. The scalability is limited only to the throughput limit of /u04. With the EFS-CG architecture, it is simple to make a filesystem perform better. Simply add another LUN to the cluster volume and grow the filesystem—an operation that puts more spindles under the NFS filesystems without disruption to the NFS clients.

Figure 8
RAC tablespace creation, single and multi-headed EFS-CG

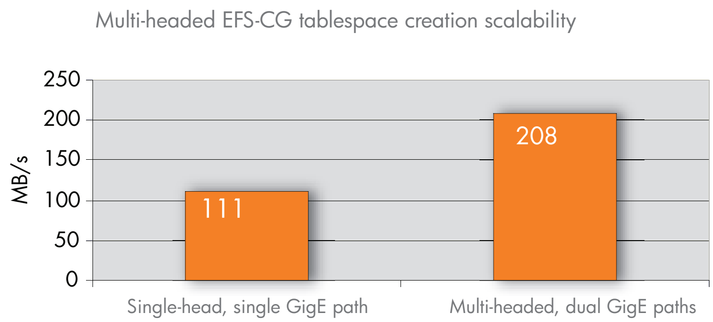
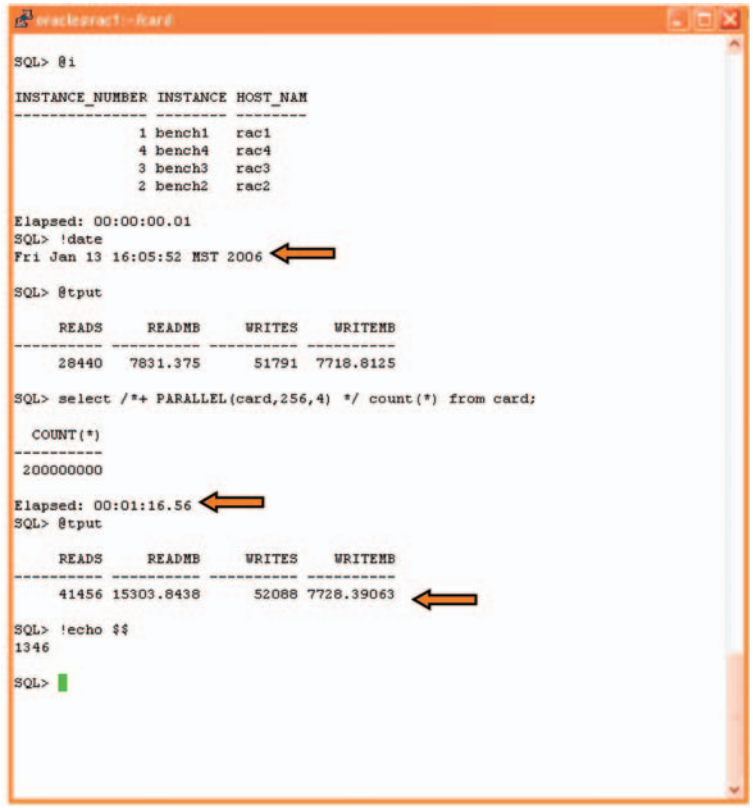


Figure 9
RAC full-table scan throughput, single-headed EFS-CG



```
SQL> @i

INSTANCE_NUMBER INSTANCE HOST_NAM
--------------- -------- --------
              1 bench1   rac1
              4 bench4   rac4
              3 bench3   rac3
              2 bench2   rac2

Elapsed: 00:00:00.01
SQL> !date
Fri Jan 13 16:05:52 MST 2006

SQL> @tput

     READS     READMB     WRITES    WRITEMB
---------- ---------- ---------- ----------
     28440   7831.375      51791  7718.8125

SQL> select /*+ PARALLEL(card,256,4) */ count(*) from card;

  COUNT(*)
----------
 200000000

Elapsed: 00:01:16.56
SQL> @tput

     READS     READMB     WRITES    WRITEMB
---------- ---------- ---------- ----------
     41456 15303.8438      52088 7728.39063

SQL> !echo $$
1346

SQL>
```

Parallel query scans

Using Oracle Intra-node Parallel Query with RAC, full-table scans can utilize all of the available I/O bandwidth that a system has to offer. After the ts.sh script was executed, a table was created to hold simulated credit card transaction records—200,000,000 of them. The table required 7.49GB space in the CARD_TS tablespace created with the ts.sh script.

The test setup needs explanation. In this test all RAC nodes were used. Before the test was executed, the EFS-CG GUI was used to re-host both vnfs1b and vnfs2b on the fourth NAS head. Configured as such, the RAC instances were limited to the throughput of a single NAS head, but more importantly to a single GigE network adaptor. Again, the EFS-CG supports three paths per NAS head, but the test configuration was set up in this manner to reduce the test hardware requirement and to make the scalability case clear.

Figure 9 shows the following:

- **First arrow:** The HOST date command was used to establish that the test started at 16:05:52hrs.
- **Second arrow:** The parallel full-table scan completed in 1m17s.
- **Third arrow:** The tput.sql script was used to determine the cumulative global I/O. This output, combined with the output from the previous tput.sql command, shows that the full-table scan read in 7,472MB of data.

Figure 10
Multi-headed EFS-CG RAC full-table scan throughput

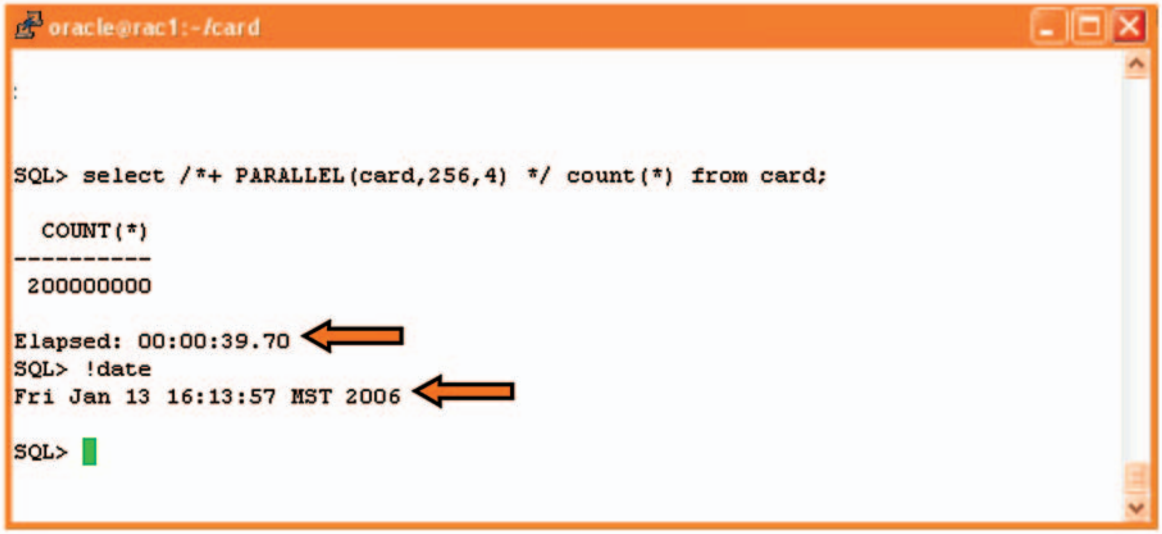
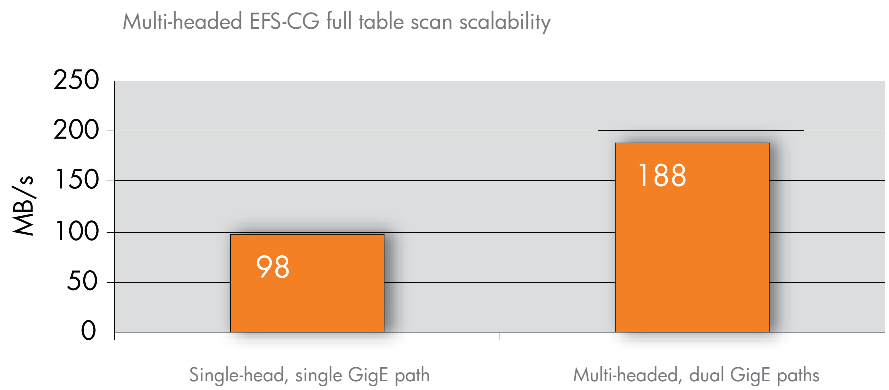


Figure 11
RAC full-table scan throughput increase by adding an EFS-CG NAS head



Continuing the test, **Figure 10** shows the following:

- **First arrow:** The second time the query was executed, it completed in only 39.7 seconds.
- **Second arrow:** The time of day at the end of the second full-table scan was 16:13:57.

In a period of 8 minutes, the 200,000,000 row table was scanned once with 98MB/s throughput and then again at 188MB/s. How did performance increase from 98 to 188/MBs? The answer is VNFS re-hosting. Before the full-table scan was executed the second time, the EFS-CG GUI was used to re-host the vnfs1b Virtual NFS Service from the fourth NAS head to the third. Doing so doubled the NFS data path bandwidth—without disruption to Oracle. Because this tablespace resides entirely in a single filesystem, employing more than one NAS head in this test is clear validation of multi-headed NAS scalability.

Read-intensive workload summary

Using Parallel Query on four RAC instances, the full-table scan test established that a linear increase in throughput can be achieved by adding NAS heads. By re-hosting a Virtual NFS Service from one EFS-CG NAS head to another—without disruption to the Oracle database instances—incremental I/O bandwidth can be added. NAS scalability was limited only by the bandwidth capability of the /u04 filesystem. In a production environment, customers always configure more than one NFS data path from each RAC node to multiple heads in the EFS-CG, and the filesystems being presented can be easily configured or changed to meet Oracle I/O demand.

OLTP testing

For database I/O, the primary concern in Online Transaction Processing (OLTP) environments is I/O service times. Many have been concern over the added service time for Oracle random I/O when accessing datafiles in NAS. During the proof of concept, OLTP testing was conducted on the four-node RAC cluster to gauge the ability of the EFS-CG to handle OLTP-style I/O patterns. This entire section of the testing was serviced by a single EFS-CG NAS head. There was no requirement to scale out the NAS end to satisfy the I/O requirements for this testing. However, as mentioned in this paper, an OLTP-only Oracle deployment does not exist. A single-headed NAS device cannot satisfy the I/O requirements needed for most parallel query operations (e.g., index creation, ad hoc query, ETL) performed by a four-node RAC cluster. Any other NAS offering would require the partitioning of some data into other filesystems to present the data through other NAS heads. If intentions were to partition the data, Real Application Clusters would not have been chosen. RAC is a scalable database architecture that scales by adding nodes, not by partitioning data—the same scaling paradigm as the EFS-CG.

The database in this proof of concept was not partitioned into several filesystems. This was a single filesystem in a single EFS-CG volume being presented by a single EFS-CG NAS head. As described above, a simple EFS-CG management GUI mouse drag-and-drop operation is all that is needed to present this database through more, or all, NAS heads to support high-bandwidth I/O operations. Any other NAS head in the EFS-CG can transparently failover VNFSes in the event of a NAS head outage.

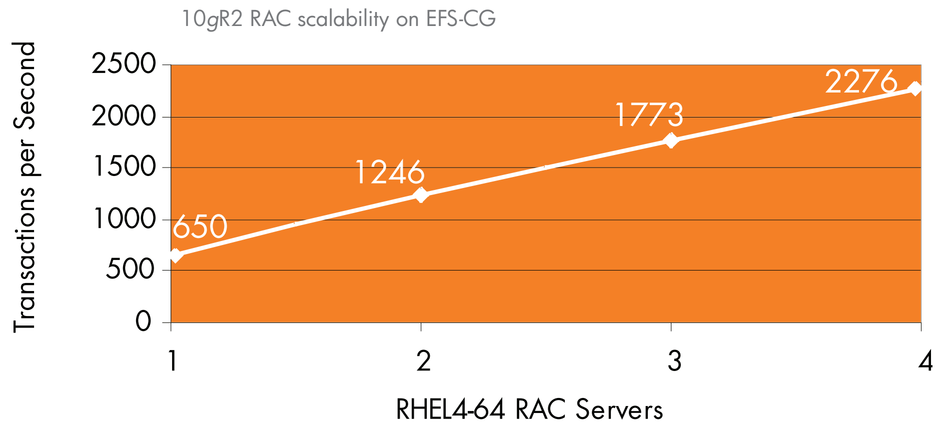
OLTP test database description

The OLTP database schema was based on an order entry system similar to, but not compliant with, that defined in the TPC-C[8] specification. At a high level, the database schema contained the following application tables:

- **Customers.** The database contained over 4 million customer rows in the customer table. This table contained customer-centric data such as a unique customer identifier, mailing address, email contact information, and so on. The customer table was indexed with a unique index on the custid column and a non-unique index on the name column.
- **Orders.** The database contained an orders table with over five million rows of data at the initial load time. The orders table grew throughout the workload execution. The orders table had a unique composite index on the custid and ordid columns.
- **Line items.** Simulating a customer base with complex transactions, the line item table contained as many as eight line items per order, or an initial level of over 40 million rows. The line item table had a unique three-way composite index on custid, ordid, and itemid.
- **Product.** The product table described products available to order. Along with such attributes as price and description, up to 140 characters were available for a detailed product description. Over one million products were in the product table. The product table was indexed with a unique index on its prodid column.
- **Warehouse.** The warehouse table maintained product levels at the various warehouse locations and had over 10 million rows. This table was crucial in order fulfillment. The warehouse table was indexed with a unique composite index of two columns.

[8] The tests conducted for this proof of concept were not compliant with the TPC-C specification. While similar to TPC-C, the workload did not comply with the specification as detailed at www.tpc.org

Figure 13
Oracle10*g*R2 OLTP scalability with database stored in the EFS-CG



Transaction descriptions

Because the transactions serviced customers at random, the I/O pattern for this database renders storage-array cache rather ineffective. The EFS-CG architecture includes an amount of cache because it is a SAN gateway device. However, because the test database was significantly larger than the array cache in the SAN, the random I/O pattern forced the highest majority of the I/O operations to access physical disk. The database was spread evenly across 140 disk drives using S.A.M.E methodology.

The following is a summarization of the transactions:

- **New order.** This transaction accounted for 18% of the workload mix. It consists of the traditional elements of taking a new order—customer validation, credit check, check stock on hand, etc.
- **Orders query.** This transaction accounted for 45% of the activity. This query provides detail on existing orders for the customer and provides such detail in a most recent to least-recent order but only top-level detail.
- **Customer and product attribute updates.** These transactions accounted for 10% of the workload and perform updates of such information as phone number, address, credit card info, price, warranty information, recall information, product description, etc.
- **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 8% of the time.
- **New items.** This transaction accounts for 11% of the mix and adds items into stock on hand.
- **Miscellaneous transactions.** The remaining 8% of the transactions perform such tasks as deleting items, adding customers, and product.

[9]The physical I/O per transaction varies due to the effect of Cache Fusion. While the average was 6.9, the range varied from 8 at one node to 6 at four nodes.

The I/O mix for this workload is 60% read and 40% write. The test harness was written in Proc*C and has think time built in between transactions. Processor utilization leveled out at roughly 75% at all node counts. **Figure 12** shows the average Oracle-related cost associated with each transaction.

Figure 12
OLTP transaction server statistics

Oracle Statistics	Average per Transaction
SGA Logical Reads	33
SQL Executions	5
Physical I/O	6.9[9]
Block Changes	8.5
User Calls	6
GCS/GES Messages Sent	12

Performance measurements

The goal of the tests was to establish that the EFS-CG supports random I/O sufficiently to support the I/O requirements of Real Application Clusters. The claim is not that the EFS-CG architecture somehow is capable of improving RAC scalability, but instead that it will not hinder scalability as other NAS architectures do. At each node count, the test workload was executed three times for 600 seconds and results from the third run were used for the analysis. **Figure 13** shows a graph of the results collected from the OLTP test executed at one through four RAC nodes.

With this workload, **Figure 13** shows that Oracle10*g*R2 on the EFS-CG was able to achieve 87% scalability. Because the I/O rate tracked throughput consistently, any significant I/O bottleneck would have prevented this level of scalability.

Figure 14
OLTP I/O scalability with Oracle10*g*R2 on the EFS-CG

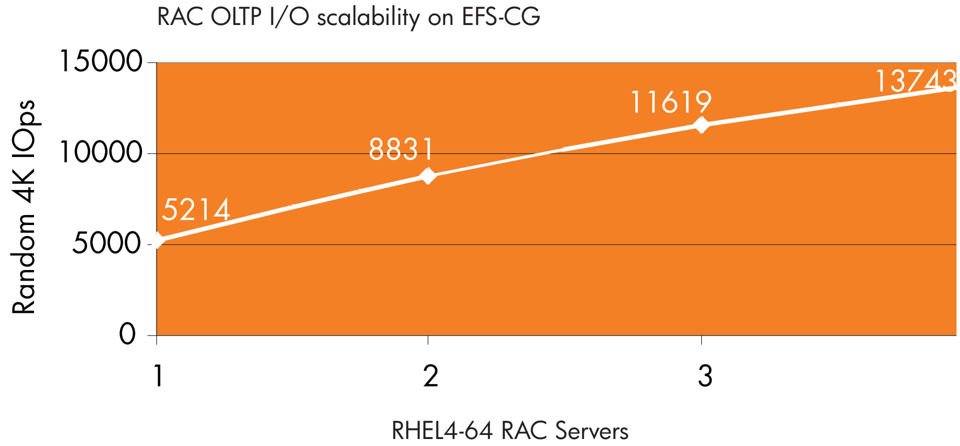


Figure 15
EFS-CG I/O rates for redo and datafile read/write operations with four RAC nodes

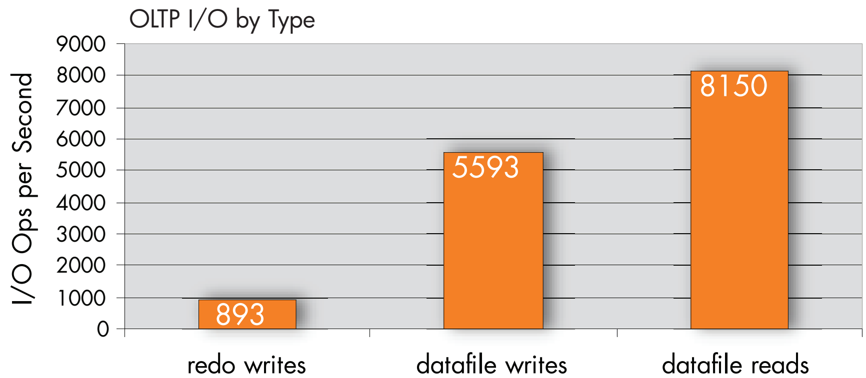


Figure 14 shows the I/O scalability curve. It appears as though scalability from one to four RAC nodes is only approximately 65%. How did the I/O and transaction rate scale differently when the workload was consistent? The answer is in RAC itself. This is a typical cache fusion effect, as sometimes when a data block is needed it is in the SGA buffer pool of another instance. That was exactly the case during this testing. For instance, during the four-node testing, the global count of global cache blocks received was 3,738 per second in addition to the physical disk I/O.

The true test in I/O efficiency for OLTP is in the session statistics for I/O service times. Perhaps the most relevant statistic is db file sequential read wait times. Analysis of the STATSPACK reports at the four-node level showed these random 4K synchronous reads were serviced with 5ms latency on average. The NFS overhead at more than 13,000 random 4K transfers per second was negligible. A typical SAN configuration with an intelligent array would be expected to service this I/O load with 5ms latency—over Fibre Channel Protocol. As this testing has shown, the EFS-CG presents files over NFS with respectable service times.

No OLTP performance analysis would be complete without considering the cost of transaction logging. The OLTP workload used for the EFS-CG proof of concept exercises a respectable rate of LGWR writing. **Figure 15** shows the ratio of redo writes to datafile I/O. At 893 redo writes per second, the four-node RAC test exhibited an approximate 1:9 ratio of redo writes to db file sequential read operations.

Redo capability is a critical aspect of OLTP and is considered a potential problem when deploying Oracle on NAS. So how does this workload compare to typical production sites? It is much more stressful. Over 1,000 STATSPACK reports[10] from commercial production Oracle sites were analyzed to determine whether or not the right level of stress was being applied during the proof of concept. Not only is this workload more stressful than a typical Oracle site but its orders of magnitude are also more stressful. At 893 LGWR writes per second, this workload is over 10 times more logging-intensive than 98% of typical Oracle production databases. Of the 1,000 STATSPACK reports, only 2% showed LGWR activity within even 25% of what was exercised in this proof of concept. Because the EFS-CG performed admirably under this level of OLTP duress, the proof of concept makes the case that the EFS-CG is more than capable of handling production Oracle I/O requirements.

[10] Special thanks to Anjo Kolk of OraPerf.com for the STATSPACK reports used to compare the proof of concept workload to typical Oracle production workloads.

Figure 16
EFS-CG I/O rates for redo and datafile read/write operations with four-node RAC Stress Test

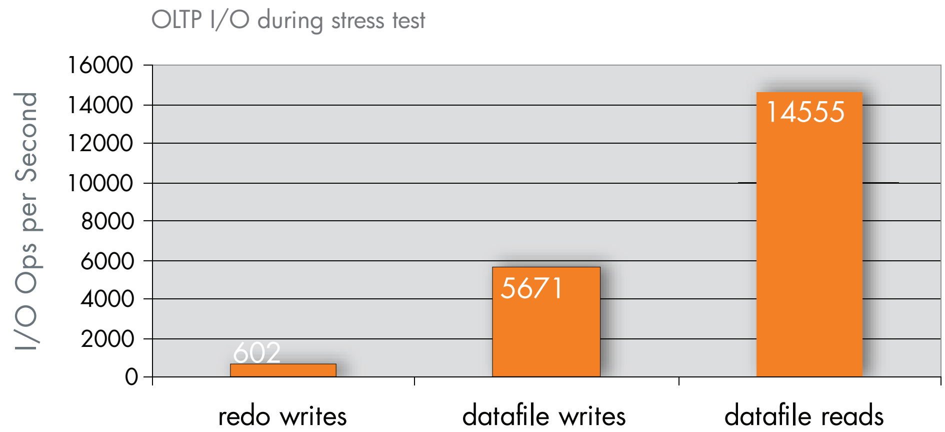
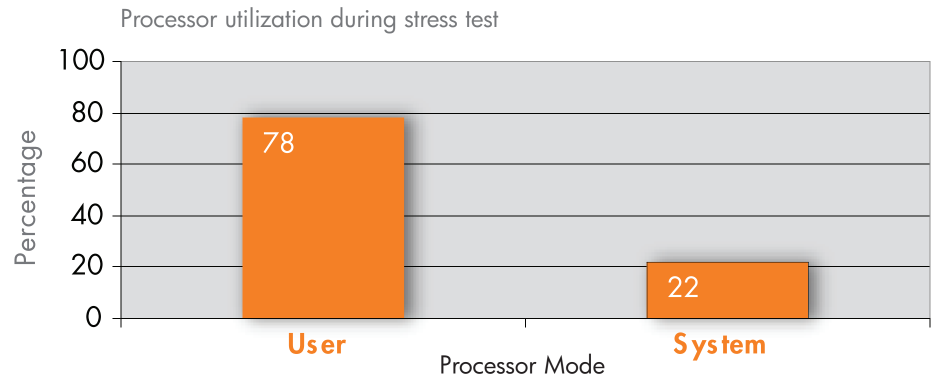


Figure 17
Processor break-out between User and System mode for RAC at 20,226 EFS-CG IOps



Extreme stress testing

After performing the scalability analysis, a test was conducted for any load-related pathologies with RAC on the EFS-CG. Unlike the scalability test, the stress test was run without any think time to drive up the system load. This test was executed repeatedly with sustained run times of eight hours. The workload mix was changed slightly to prevent any comparison of this specific test result to those shown in **Figure 14**. Whereas the read-to-write ratio was 60:40, the Stress Test workload was 72:28 and LGWR writes were lowered to 602/s from 893/s by different transactions. **Figure 16** shows the I/O throughput measured during the Stress Test phase of the proof of concept.

At 20,226[11] physical datafile I/O operations per second, the stress test proved that the EFS-CG can deliver a substantial rate of physical I/O and do so with excellent stability. At 140 total disk drives, this test performed roughly 144 physical disk transfers per disk/second—a significant sustained I/O rate.

The last big question to address is processor utilization. The typical presumption about Oracle on NAS is that the NFS overhead is so significant as to leave Oracle insufficient cycles for internal processing. The results provided up to this point in the paper refute that notion.

[11] This was not a reflection of storage-level cache effect. The database was so much larger than the storage cache that random I/O was rendered to physical disk.

[12] A 10 Billion is 10^9, 10 Billion is 10^10

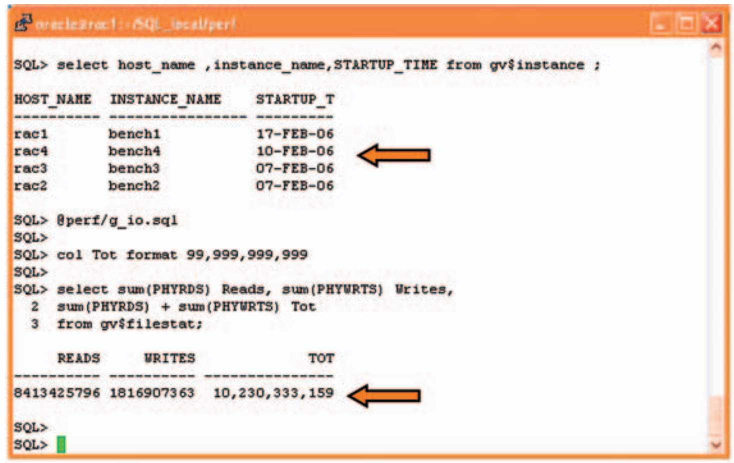
Because the NFS clients used as RAC servers in the proof of concept were configured with Red Hat® Enterprise Linux®, the I/O was rendered through the direct I/O code path. The NFS I/O protocol is not free but is not as bad as most would presume. The processor utilization was recorded during the stress test phase of the proof of concept to ensure that a reading was taken at peak I/O levels. **Figure 17** shows that using Oracle10*g*R2 with datafiles in the EFS-CG leaves 78% of the processor cycles for user-mode processing—even at 20,226 I/Os per second. As shown in all of the results in this paper, 78% of modern processors for Oracle yields superior performance.

Long duration stress test

The proof of concept was not a typical benchmark. After the extreme stress test analysis was conducted, the next phase of testing was the long duration stress test. The goal of this test was to stress four instances of RAC with the proof of concept workload described above for a substantial period of time. However, execution time was not the primary metric of the test. Instead, physical database I/O operations were measured in the Oracle gv$filestat view. The goal of this test was to push four instances of RAC to perform 10 billion[12] physical disk transfers.

When Oracle makes a system call for physical I/O, it does not actually know whether or not the I/O was satisfied in cache at one of the many possible levels. For instance, an

Figure 18
Physical disk transfers during long duration stress test


```
SQL> select host_name ,instance_name,STARTUP_TIME from gv$instance ;

HOST_NAME  INSTANCE_NAME    STARTUP_T
---------- ---------------- ---------
rac1       bench1           17-FEB-06
rac4       bench4           10-FEB-06
rac3       bench3           07-FEB-06
rac2       bench2           07-FEB-06

SQL> @perf/g_io.sql
SQL>
SQL> col Tot format 99,999,999,999
SQL>
SQL> select sum(PHYRDS) Reads, sum(PHYWRTS) Writes,
  2  sum(PHYRDS) + sum(PHYWRTS) Tot
  3  from gv$filestat;

     READS     WRITES             TOT
---------- ---------- ---------------
8413425796 1816907363  10,230,333,159

SQL>
SQL>
```

Oracle single block read (e.g., db file sequential read) might actually be satisfied with an OS-level logical I/O if the block is cached in the OS page cache. Conversely, if the downwind storage configuration has cache (e.g., a SAN intelligent storage array), the block might be cached at that level, eliminating a physical disk transfer. Neither of these possibilities was true during this test. The test workload exhibits a random I/O pattern as described above. The main table being accessed randomly was over 200GB and the storage array cache was only 4GB, so activity against this table alone renders the storage array cache obsolete. Because the RAC instances were running on Linux servers with a 2.6 kernel, Oracle was accessing the files in the EFS-CG filesystem through the direct I/O code path. Essentially, no cache existed other than the SGA. The physical disk transfers counted in the gv$filestat table were indeed physical disk transfers.

The screen capture in **Figure 18** was taken on February 19, 2006. The first arrow shows that of the four instances, two had been up for 12 days, one for nine days and one for two days. A query against the gv$filestat view returns the aggregate values from each of the instances for the columns being queried. Servers rac1 and rac4 were taken down for maintenance reasons during the long duration stress test, resulting in the loss of the physical disk I/O they had performed because the test started. The architecture of RAC accommodates taking instances down for maintenance because no other instances are still active. The instances were brought back online and the workload commenced once again and continued to execute on all four RAC nodes until the test had performed over 10 billion physical disk transfers. The second arrow in **Figure 18** points to a query against gv$filestat showing that this RAC database had sustained 10,230,333,159 physical transfers of which 18% were writes. Missing from this I/O figure is redo log writer I/O transfers.

This test was concluded within approximately 12 days. The EFS-CG NAS filesystem sustained around 10,000 physical I/Os every second of every hour during the test. This stress test represents orders of magnitude more physical I/O than the typical Oracle production database. Most Oracle databases do not generate this number of physical disk transfers in a full year.

Claims of storage subsystem stability are often based upon MTBF[13] analysis and projections. Seldom is so much hardware submitted to such a rigorous stress test for this duration. Most industry benchmarks, such as TPC-C[14], are executed for short periods of time, after which results are analyzed. A single database seldom is subjected to long duration, highly-contentious OLTP workloads with this sort of physical I/O load.

Completing this proof of concept with a long-duration stress test made the project results complete. The architectural differences between the EFS-CG and other NAS architectures has been shown and demonstrated. The durability of this type of storage proves its fit for the demanding I/O requirements of today's Oracle databases.

Summary

When was the last time a lack of choice helped you solve an IT problem? Oracle customers should feel comfortable making the choice between NAS or SAN for their Oracle deployments. If SAN is established in your datacenter but it makes sense for you to deploy Oracle on NAS, why not leverage the SAN? This paper has briefly covered the compelling reasons why a SAN gateway product is a good way to present filesystems via NFS to your Oracle servers. This paper also has covered the caveats associated with different NAS architectures. Bearing those caveats in mind, the best path forward to Oracle on NAS should be clearer. Only the HP Enterprise File Services Cluster Gateway possesses the right technology for highly available, scalable NAS for Oracle.

[13] Mean Time Between Failure. Instead of executing a long stress test, many manufactures will project how their storage would handle a workload based on how many disks and how often they are projected to fail.

[14] Depending on the version of the specification, the measurement period is as short as 30 minutes after some cache warm-up.

Appendix

ts.sh script:

```
#!/bin/bash

function drop_and_recreate() {
# Drops and recreates tablespace and throws away unnecessary
# text returns from sqlplus via grep
sqlplus -S '/ as sysdba' <<EOF | grep -i tablespace
REM drop tablespace card_ts including contents and datafiles;
drop tablespace card_ts including contents;
create tablespace card_ts
datafile '/u03/DATA/CARD/card_0.dbf'
SIZE 8190K REUSE
AUTOEXTEND OFF
EXTENT MANAGEMENT LOCAL
SEGMENT SPACE MANAGEMENT AUTO
BLOCKSIZE 16K;
exit
EOF
}
```

```
function add_file() {
# Uses promiscuous rsh to throw the execution over to node $NODE

local NODE=$1
local FPATH=$2
local FILE=$3
(rsh rac${NODE} "cd ~oracle;. .bash_profile> /dev/null 2>&1;sqlplus '/ as sysdba' <<EOF >
/dev/null 2>&1
alter tablespace card_ts add
datafile '${FPATH}/card_${FILE}.dbf'
SIZE 1024M REUSE;
exit
EOF" ) &
}
```

```
### Main Program Body

MODE=$1
OTHER_NODE=$2
NUM_FILES=$3
FPATH=/u03/DATA/CARD
cnt=1
drop_and_recreate   #function
B=$SECONDS
until [  $cnt -gt $NUM_FILES ]
do
FILE=$cnt
(( x = $FILE % 2 ))
       if [  "$MODE" = "multinode" ]
       then
            [[  $x -eq 0 ]]  && NODE=1  || NODE=$OTHER_NODE
       else
            NODE=1
       fi
echo "File $FPATH/card_${FILE}.dbf will be added by node $NODE"
add_file $NODE $FPATH $FILE    #function
(( cnt = $cnt + 1 ))
done
wait
(( TOT = $SECONDS - $B ))
echo "Complete tm ${TOT} seconds"
```

tsinfo.sql:

```
select sum(blocks) * 16 / (1024 *1024 ) TS_SIZE_GB
from dba_data_files where TABLESPACE_NAME = 'CARD_TS';
```

i.sql:

```
col INSTANCE_NUMBER format 9
col INSTANCE_NAME   format a8
col HOST_NAME        format a8

select INSTANCE_NUMBER,INSTANCE_NAME,HOST_NAME
from gv$instance;
```

tput.sql:

```
select sum(PHYRDS) reads,sum(PHYBLKRD * 16 )/1024
readMB,
sum(PHYWRTS) writes,sum(PHYBLKWRT * 16 )/1024 writeMB
from dba_data_files
```

Authors

Kevin Closson, Chief Architect
Oracle Database Solutions, PolyServe

Special recognition is given to the following individuals for their knowledge and impeccable support of this effort:

Dennis Miyoshi
Head Solutions Engineer
NAS StorageWorks Division, HP

John Dupuis
WW Technical-Sales Marketing
NAS StorageWorks Division, HP

Janelle Adams
Technical Publications, PolyServe

Special thanks are given to the HP StorageWorks organization for sponsoring this proof of concept.

For more information about HP Enterprise File Services Cluster Gateway for Oracle Database Deployments and PolyServe Matrix Servers software, visit www.hp.com or www.polyserve.com.

4AA0-4746ENW, March 2006

invent

ORACLE