Technical Report

Performance Study of iSCSI in an OLTP-Based Oracle 9i RAC Database Environment with NetApp iSCSI Storage Systems

Giovanni Brignolo and Sankar Bose, NetApp
Theodore Haining, Srinivas Maturi, and Xiaoping Li, Oracle Corporation
Robin Bhagat, Samdeep Nayak, and Robert Ortega, Adaptec, Inc.
Revised May 2009 | TR-3357

TABLE OF CONTENTS

1 INTRODUCTION 3
2 OVERVIEW OF THE SYSTEM ENVIRONMENT 5
SOFTWARE ENVIRONMENT 7
3 EXPERIMENTAL SETUP AND CONFIGURATION 8
DATA PLACEMENT AND OLTP WORKLOAD CHARACTERIZATION USING RAW DEVICES 8
STORAGE SYSTEM SETUP 9
INSTALLING THE ADAPTEC ISCSI DRIVER 10
ORACLE OLTP DATABASE DESIGN 13
4 ORACLE 9I RAC BEST PRACTICES FOR IA32 LINUX 17
MULTI-LUN SUPPORT IN THE LINUX KERNEL 17
DRIVER FOR THE ADAPTEC 7211C HBA 17
LINUX CONSIDERATIONS FOR ORACLE 9I RAC DATABASES USING RAW DEVICES 18
ASYNCHRONOUS I/O 18
LARGE SGA 19
CLUSTER MANAGER CONFIGURATION 19
CLUSTER INTERCONNECT 20
5 ANALYSIS OF ORACLE 9I RAC SCALABILITY 21
RESULTS 23
6 FUTURE WORK 25
7 SUMMARY AND CONCLUSIONS 26
TCO CONSIDERATIONS 26
8 ACKNOWLEDGEMENTS 27
9 REFERENCES 27
10 APPENDIX A: STORAGE SYSTEM SETUP 27
VOLUME CREATION 27
LUN CREATION 33
11 APPENDIX B: ISCSI DRIVER INSTALLATION 34
12 APPENDIX C: INITIATOR MAPPING EXAMPLE 35
USING ICONFIG TO GET ISCSI INITIATOR NAMES 37
MAPPING IGROUPS TO LUNS 38
13 REVISION HISTORY 40

1 INTRODUCTION

An increasing number of applications demand high-availability combined with online scalability and bounded response times. The growing fields of electronic commerce, world-wide business, and telecommunications all require databases that are highly available for all kinds of operations at any time. This makes continuous computing availability an operational necessity in today's global economy. Downtime is no longer an option, whether it is unplanned (due to hardware or software failure) or planned (due to scheduled backup or system upgrade). The problem of making information available at all times challenges IT managers with a series of seemingly conflicting objectives:

- Provide continuous application services and data access, even if applications or systems fail, or data centers become incapacitated.
- Provide secure universal access to enterprise data.
- Provide application performance that scales as the organization grows.
- Contain management costs as the number and complexity of the systems grow.

Clustering technologies help to meet these challenges by coordinating interconnected servers to enhance application and data availability, scalability and manageability. Clusters are attractive to users because of their potential to solve some of the most significant problems in computing:

- **Server failure**: If a server or application crashes, another server in the cluster can take over its application providing nearly continuous service to clients.
- **Network or I/O path failures**: If a network or I/O link fails, access to applications and data can continue through alternate paths.
- **Application growth**: If the application demands go beyond the ability of a single server, additional servers can be added and the workloads can be shared.
- **Management cost**: As the numbers of enterprise applications grow, clusters can provide a flexible platform for running multiple applications while allowing the entire cluster to be managed as a single system.

These requirements for high levels of availability, as well as scalability, have led companies to consider technologies such as Oracle® 9i Real Application Clusters (RAC).

Emerging storage protocols also help to meet these objectives. The iSCSI protocol specifies how to access SCSI storage devices over a TCP network. The protocol encapsulates disk access requests (in the form of SCSI commands and data blocks) into TCP packets, and transmits the SCSI commands and block-level data over IP networks. In this way, iSCSI encompasses two major protocols: for storage access, SCSI, and for networking transport, TCP. This allows IT managers to take advantage of ubiquitous Internet infrastructure, allows for greater flexibility when planning, deploying, and operating storage area networks, and facilitates remote storage, remote backup, and data mirroring because storage can now be accessed using low cost components like Gigabit Ethernet switches and Ethernet interface cards. This results in lower total cost of ownership (TCO) and increased availability, because the resulting unified storage and data networks provide additional service while greatly reducing management cost and employing common, reliable, and inexpensive hardware.

Recent developments have also supported the acceleration of iSCSI adoption: the ratification by the IETF of the iSCSI specification in February 2003 and the announcement by various vendors of iSCSI support in their Operating Systems.

Specialized hardware and software designed to leverage the advantages of the iSCSI protocol already exist. First generation iSCSI targets and initiators are currently available with iSCSI processing implemented in software. Vendors have already proposed iSCSI hardware adapters that benefit from hardware acceleration (such as TCP offload engines). These hardware adapters come in two forms: an iSCSI target mode Host Bus Adapter (HBA) which operates at the SCSI level (the host driver looks like an FCP-SCSI HBA driver), and an iSCSI TCP Offload Engine (TOE) which offloads data intensive portions of iSCSI communication to the TOE, thereby reducing the amount of processing that the host hardware and software must perform. For example, iSCSI Protocol Data Unit (PDU) identification, digest calculation, and verification. The host HBA interface is PDU-based, the hardware offloads the data intensive iSCSI digest and checksum calculation and leaves the PDU processing to the host software.

Current database management systems (DBMS) have widely varying requirements with respect to availability, reliability, scalability, and system performance. The combination of iSCSI and clustering technologies has significant potential to fulfill these requirements for systems used to access mission critical data, while offering reductions in total cost of ownership.

To determine the overall impact of using iSCSI and clustering technologies with DBMSs, this technical report presents the findings of a performance study of the iSCSI protocol in an Oracle 9i RAC database cluster on raw devices using an online transaction processing (OLTP) workload. This study had three goals:

- To understand how iSCSI storage performs and scales when used with the Oracle 9i RAC data sharing model in an OLTP environment,
- To evaluate the suitability and ease of use of iSCSI hardware and software for Oracle 9i RAC production environments, and
- To document best practices.

The cluster used in this study consisted of four Intel-based hosts and two NetApp® storage systems. The storage network in the cluster was configured using a specialized hardware implementation of the iSCSI Initiator that is commercially available from Adaptec® (the 7211C iSCSI HBA) and a NetApp software implementation of the iSCSI target on each storage system.

The results of this study show that iSCSI is eminently suitable for use with Oracle 9i RAC databases. The Adaptec 7211C HBA proved to be straightforward to configure and use. Scalability tests using the OLTP workload show near linear increases in relative transactions per minute as the size of the RAC cluster increased to four nodes. The stability of the I/O fabric using the Adaptec 7211C HBAs and the NetApp storage systems was very good during all tests. The overall iSCSI infrastructure demonstrated its suitability for production deployment.

The remainder of this report is organized in the following manner. The next section presents the general hardware layout, storage architecture, and software configuration of the cluster used for the performance study. The third section describes how the cluster in the study was set up. The fourth section describes the Oracle 9i RAC best practices for Linux and specific setup procedures that improve cluster reliability and performance. The fifth section of this report analyzes the performance of Oracle 9i RAC with respect to scalability. Finally, the last two sections discuss future work and present our conclusions.

2 OVERVIEW OF THE SYSTEM ENVIRONMENT

This section of the report explains the hardware and software components used for the performance study described in this document.

We built a 4-node high-availability cluster using Linux servers running Red Hat Advanced Server (AS) 2.1 to investigate Oracle 9i RAC performance and scalability. Each node of the cluster featured 2 Pentium-4 Xeon 2.4GHz processors with 4GB of memory and two Adaptec 7211C iSCSI HBAs. The cluster configuration includes two Gigabit switches (one for the cluster IPC and the other for the storage fabric) and two NetApp FAS960 storage systems configured with an iSCSI software target (ISWT) on each storage system. Figure 1 below shows how the four Oracle 9i RAC nodes are connected to the FAS960 storage systems using the iSCSI protocol for IP-SAN traffic through a Gigabit Ethernet switch (S1). The cluster interconnect (IPC) is implemented using another dedicated Gigabit Ethernet switch (S2). Table 1 below shows the IP-SAN and cluster IP addresses used for the iSCSI I/O fabric.

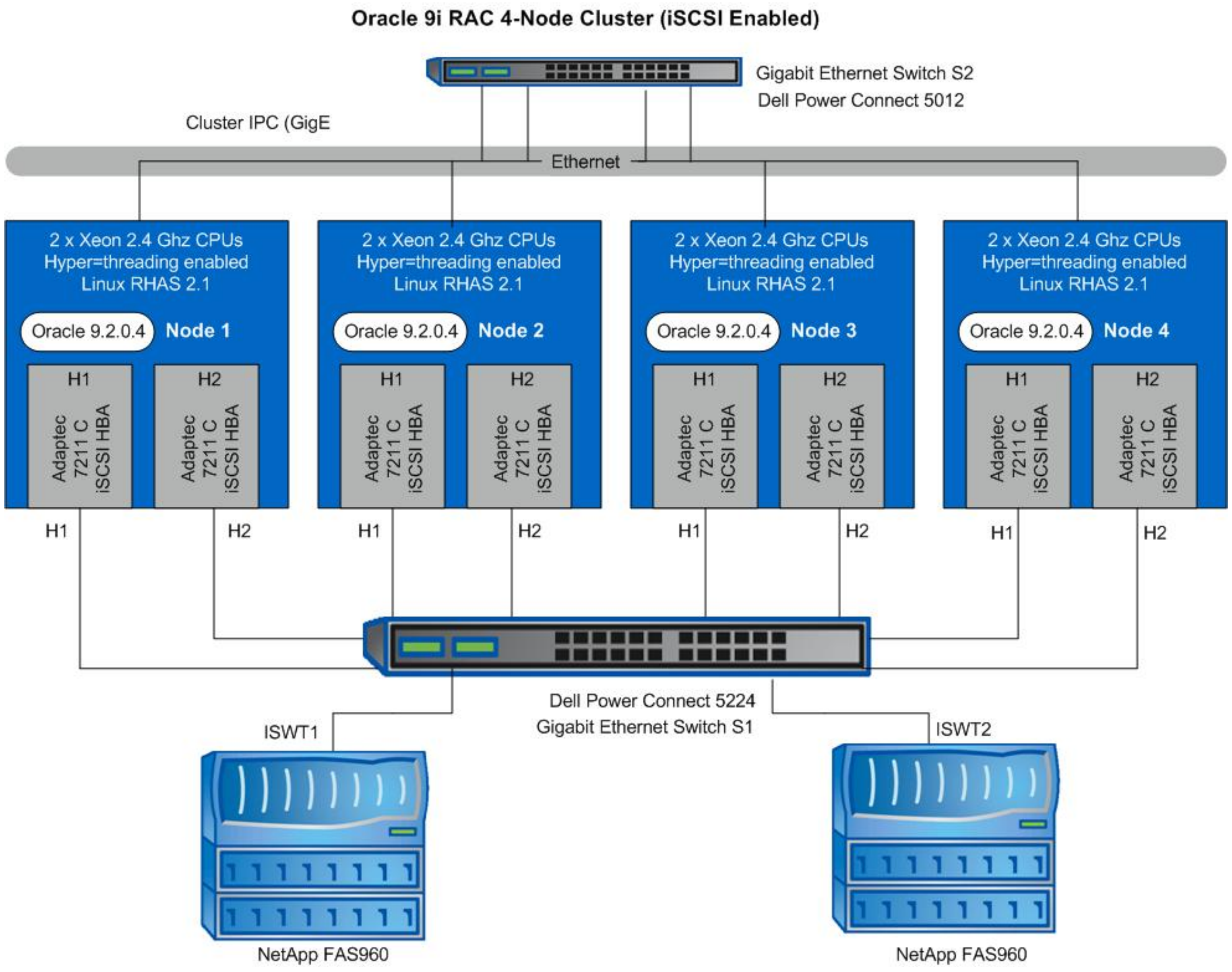

Figure 1) The network architecture of the four-node Oracle 9i RAC cluster.

IP-SAN ADDRESSES

Table 1) The IP addresses used for storage network communication within the four-node Oracle 9i RAC cluster.

Node Name	H1 IP Address	H2 IP Address
Node 1	192.168.101.201	192.168.101.202
Node 2	192.168.101.203	192.168.101.204
Node 3	192.168.101.205	192.168.101.206
Node 4	192.168.101.207	192.168.101.208
Storage System	**IP Address**	
ISWT1	192.168.101.101	
ISWT2	192.168.101.102	

Current iSCSI adapters work in two ways. Some adapters offload the TCP/IP stack into hardware and are called TOE cards (TCP Offload Engine). They present themselves as a Network Interface Card (NIC) to the Operating System; therefore, you need to add a software iSCSI initiator (the software driver that processes iSCSI commands). Other iSCSI HBAs, such as the Adaptec 7211C, offload TCP related processing from the host CPU and appear as a SCSI card to the Operating System providing their own SCSI Initiator in firmware (the iSCSI driver).

The Adaptec 7211C iSCSI card was used for the Linux Host Initiators. It rapidly processes iSCSI packets by using the Adaptec Storage Accelerator (ASA), for complete TCP/IP offload, and an on-board CPU to process iSCSI functions. This Initiator optimizes iSCSI performance at the lowest possible host CPU utilization. To maximize performance across a wide range of applications, the ASA uses a pipelined architecture to provide high I/O throughput for both large and small block sizes.

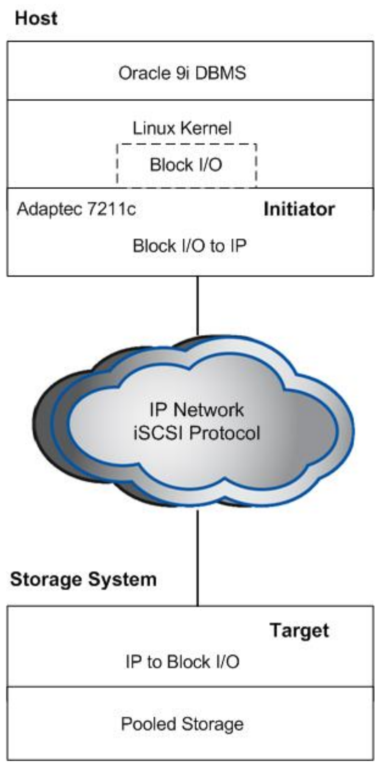

Figure 2) Protocol layers.

Figure 2 above shows the layers involved with the iSCSI initiator protocol, network transport, and target protocols processing. The IP network implemented in our study consisted of a dedicated Gigabit Ethernet Storage Area Network.

SOFTWARE ENVIRONMENT

We measured the scalability of Oracle 9i R2 (9.2.0.4) running on Red Hat Advanced Server 2.1 (kernel 2.4.9-e.25smp) with an OLTP workload. The workload used to exercise the Oracle 9i RAC on the iSCSI four-node system under test was provided by the Oracle OLTP workload generator module.

The OLTP database system was measured for transaction processing performance and scalability. All aspects of the system, including processor, memory, disk I/O and user input/output, were exercised. The software components and relative versions are listed in Table 2 below.

Table 2) Version information for software components used in the performance study.

Software Component	Version	Description	Comments
Oracle OLTP Workload Generator	N/A	OLTP transactional workload	Simulates order-entry wholesale business transactions
Oracle 9i RAC	9.2.0.4	Oracle Real Application Cluster	Used on one node then scaled up to four nodes
Linux	Red Hat Advanced Server 2.1 (kernel 2.4.9-e.25smp)	Operating Environment	Raw devices used for Oracle files
iSCSI Initiator	1.11	Adaptec iSCSI driver for Linux [Reference# 3. , 4.]	This driver was used for the iSCSI Initiator on the host CPU
iSCSI Target	6.5	iSCSI software driver (a feature of Data ONTAP® 6.5)	With this ISWT, a storage system appears as a single iSCSI Target Node with one iSCSI Node Name
Data ONTAP	6.5	Filer storage micro-kernel	Implements the ISWT driver in software

We created LUNs of appropriate sizes on the targets that were presented to the host as block level devices. This is described in the Filer Setup section.

We started with a one-node Oracle 9i R2 RAC database instance, and then we scaled the database up to two nodes, and then up to four nodes. We measured the RAC scalability by running an OLTP workload.

We performed an extensive performance study on the configuration described above. This study used an OLTP workload generated by processes simulating users who connect to a database and execute order-entry transactions that are representative of a wholesale business. These activities include entering and delivering orders, recording payments, checking the status of orders, and monitoring the level of stock. OLTP concepts are familiar to many industries, such as banking, airlines, mail-order, supermarkets, manufacturing, and financial services; therefore, OLTP technologies can be applied to solve the various data management challenges that are typical of such environments.

The performance was measured in new-order transactions per minute while running the mix of OLTP transactions. High transaction rates (bound to a response time constraint) were the primary performance objective in our experiments.

3 EXPERIMENTAL SETUP AND CONFIGURATION

In this section, we introduce the physical data model that supports the OLTP workload for the study using the hardware cluster described in the previous section. In the following sections, we describe the environment, the best practices used in the design and implementation of the database system, the function and setup of the iSCSI storage layer below Oracle 9i RAC, and the measurement methods and performance metrics used in the study.

DATA PLACEMENT AND OLTP WORKLOAD CHARACTERIZATION USING RAW DEVICES

The OLTP database design was optimized to support the principal activities of an order-entry system in a wholesale business environment. Our performance optimizations were obtained by monitoring throughput and performance of new-order transactions while running all the transaction combinations that are part of the OLTP working-set.

In this context, physical database design is important for query performance in a shared-everything parallel database system, in which data is shared among multiple independent loosely-coupled nodes. Our approach to data design partitions the workload across multiple I/O paths and storage volumes. We recommend that the candidate tables be partitioned according to a key-range or hashing scheme; this will benefit each workload query.

We exercised a 1.0 TB Oracle 9i RAC database on a four-node Linux cluster. The DBMS accessed the database through raw devices. The database spanned across 112 spindles using the data storage layout described in Table 3 below.

DATA STORAGE LAYOUT

Table 3) The storage system volume and LUN storage layout used with a 4-node RAC system.

FAS960 Storage System	WAFL® Volume	LUN	Size	Function
Storage System 1	V11	vol11_lun1	150GB	Datafiles and Control files
		vol11_lun2	150GB	Datafiles
		vol11_lun3	150GB	Datafiles
		vol11_lun4	150GB	Datafiles
	V12	vol12_lun1	150GB	Datafiles
		vol12_lun2	150GB	Datafiles
		vol12_lun3	150GB	Datafiles
		vol12_lun4	150GB	Datafiles
Storage System 2	V21	vol21_lun1	150GB	Datafiles and Control files
		vol21_lun2	150GB	Datafiles
		vol21_lun3	150GB	Datafiles
		vol21_lun4	150GB	Datafiles
	V22	vol22_lun1	150GB	Datafiles
		vol22_lun2	150GB	Datafiles
		vol22_lun3	150GB	Datafiles

		vol22_lun4	150GB	Datafiles
	log	log_lun1	100GB	Redo Logs
		log_lun2	100GB	Redo Logs

For the storage system volumes containing LUNs, the WAFL space reservation option was turned on.

Each LUN was presented to the Linux I/O subsystem as a block device. The RAC database requires access to character devices mapped to partitions of these block devices. To this end, we created partitions of appropriate sizes on the LUNs, and each of those partitions were mapped to a character special file using the procedures described in the section “Linux Considerations for Oracle 9i RAC Databases Using Raw Devices”.

STORAGE SYSTEM SETUP

LUNs reside on WAFL volumes on the storage system, and one must specify the volume name and the number of disks to be used to create each volume. The capacity and number of disks we specify determine the size of the volume.

The volume was sized to accommodate all of the database objects that will reside on the volume with some extra capacity used for space reservation. From the storage system’s perspective, a LUN is a logical representation of a physical unit of storage. In other words, LUNs appear as local disks attached to the host. As a result, it is imperative that LUNs behave as attached disks, which means that when you create a LUN of a specified size, the application storing data in the LUN must always have access to that amount of allocated space.

For all the WAFL volumes that we created for storing the database files, a RAID group size of eight (seven for data, one for parity) is recommended. For a database workload, a dedicated WAFL volume for storing the Oracle redo logs is recommended.

When we create LUNs, they are automatically striped across many physical disks. Based on our experience, the following recommendations should be considered when creating volumes for LUN allocation:

1. Turn on the Data ONTAP space reservation option (this is the default).
2. Do not create any LUNs on `/vol/vol0`. This is the root volume.
3. Create volumes large enough to accommodate LUNs that will be stored on the volume by taking into consideration the Snapshot™ copy and space reservation overhead. In our case, we turned off the automatic Snapshot copy schedule (that is, `nosnap=on`).
4. Turn off aggressive read ahead, by setting `minra=on`, for volumes that will store LUNs used as database files. Using the minimum read ahead (that is, `minra=on`), we have observed a 10-15% performance improvement for database OLTP workloads.

   Note: Historically, NetApp had recommended disabling aggressive storage readahead for OLTP (Online Transaction Processing) database workloads by setting the Data ONTAP parameter "`minra`" to "`on`". Data ONTAP 6.5.1, however, introduced significant changes to the readahead algorithm, making it more intelligent and efficient. Hence, disabling readahead for database workloads is no longer recommended. Recent experience indicates that in fact, enabling `minra` may lower the overall database performance. As a result, NetApp now recommends that the `minra` setting be left in the default "`off`" state unless explicit guidance to do otherwise is given by the NetApp Global Support Organization.

In our configuration, we created two WAFL volumes on one storage system and three WAFL volumes on a second storage system for a total of five WAFL volumes. For a detailed description on how to setup the volumes as per the above listed best practices, see “Appendix A: Storage System Setup”.

INSTALLING THE ADAPTEC ISCSI DRIVER

The Linux iSCSI driver for the Adaptec 7211C HBAs acts as an iSCSI protocol initiator to transport SCSI requests and responses over an IP network between the client and an iSCSI enabled target device. This driver is available from Adaptec and it must be installed with the Linux kernel. It is downloadable using the URL provided in Reference [3]. See Appendix B for a detailed description of the driver installation procedures.

MAPPING THE LINUX ISCSI INITIATORS TO THE FILER TARGETS

After installing the Adaptec iSCSI adapter and driver, the logical connectivity between the host adapters and the storage system targets must be established. Each Adaptec iSCSI initiator can be mapped either to a single target (one-to-one mapping) or to multiple targets (one-to-many mapping). We observed that by mapping one initiator to multiple targets (one-to-many), the OLTP working-set ramp up time took longer to reach its steady state than when using a configuration with two initiators, each mapped to a single target storage system (one-to-one mapping). The increase in ramp-up time was the result of using only one I/O channel on the host mapped to two storage system targets (only one I/O channel is established from the initiator and mapped to the two storage system targets) as opposed to using two I/O channels (one from each of the two initiators to each of the target storage systems). The details of the two different mappings are described in Figure 3 and Figure 4 below.

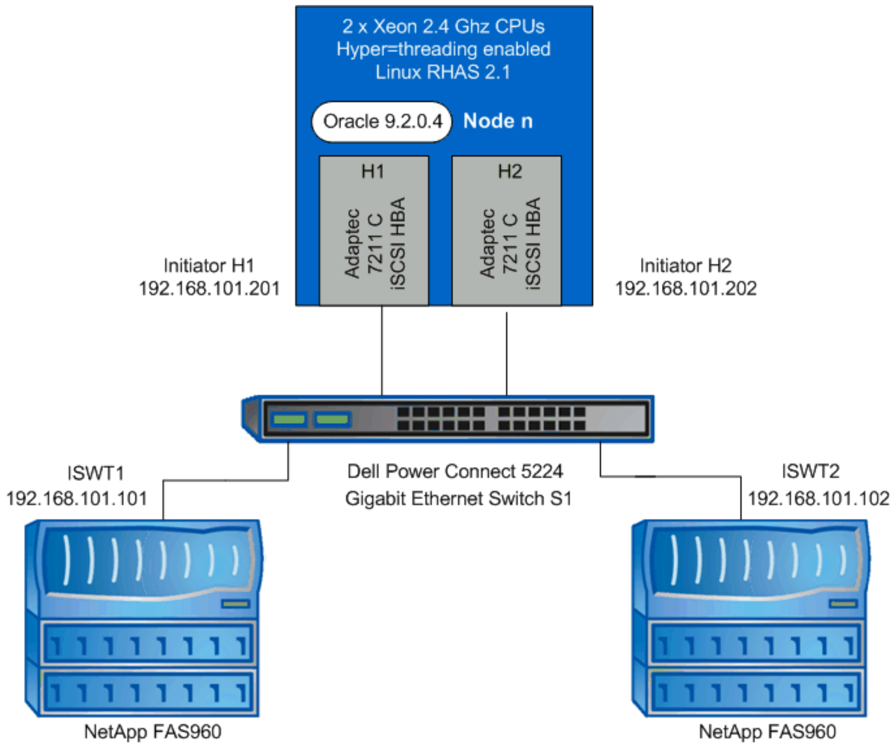

Figure 3) Two Linux iSCSI initiator with each initiator mapped to a single storage system (one-to-one initiator to target mapping).

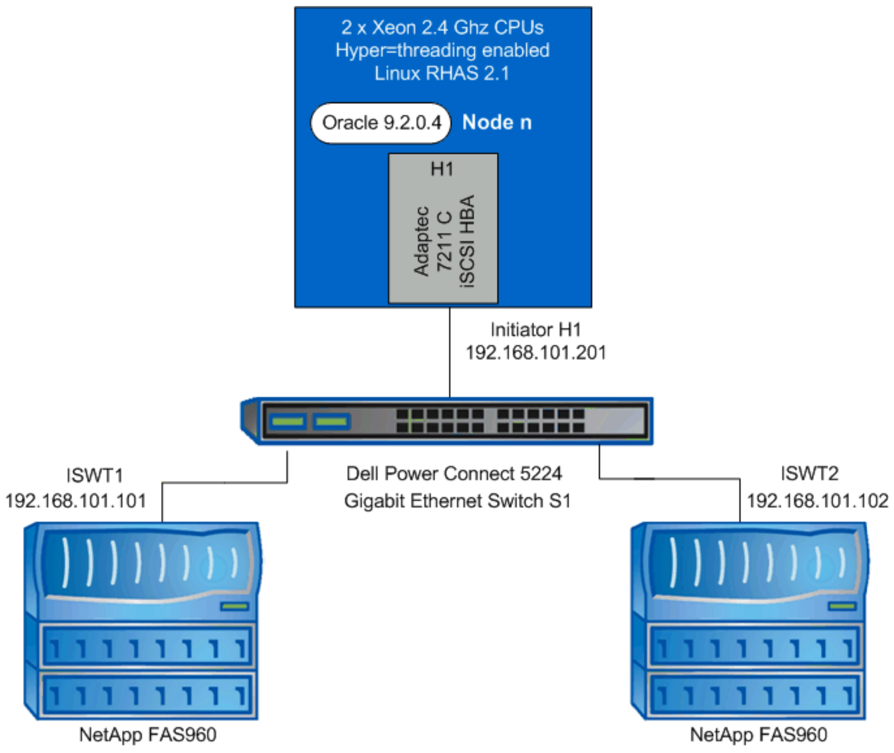

Figure 4) One Linux iSCSI initiator mapped to two storage systems (one-to-many initiator to target mapping).

We used the Adaptec iconfig utility on Linux to create the mappings. See the *Adaptec's User's Guide iConfig Utility* [Reference 4] for a detailed description on how to use the iconfig utility. The example in Appendix C demonstrates using the iconfig utility to map an initiator with an IP address of 192.168.101.201 to an iSCSI target with an IP address of 192.168.101.101.

The physical connectivity established using iconfig is described in Table 4 below.

Table 4) Physical connectivity established using iconfig.

Node Name	Initiator	Initiator IP Address	Target IP Address	
Node 1	H1	192.168.101.201	Storage System 1 ISWT1	192.168.101.101
Node 2	H1	192.168.101.203		
Node 3	H1	192.168.101.205		
Node 4	H1	192.168.101.207		
Node 1	H2	192.168.101.202	Storage System 2 ISWT2	192.168.101.102
Node 2	H2	192.168.101.204		
Node 3	H2	192.168.101.206		
Node 4	H2	192.168.101.208		

For our configuration, the iSCSI Initiator Login parameters are set as follows:

```
Max Outstanding R2T        : 1
```

```
Data PDU In Order          : Yes
Data Sequence In Order     : Yes
Error Recovery Level        : 0
DefaultTime2Retain         : 3 Secs
Initial R2T                : Yes
Immediate Data             : No
MaxRecvDataSegmentLength   : 65536 bytes
FirstBurstLength           : 65536 bytes
MaxBurstLength             : 65536 bytes
Header Digest              : No
Data Digest                : No
Authentication Method      : None
```

We also had the iSCSI Discovery Parameters set as follows:

```
SLP Discovery = DISABLED
iSNS Discovery = DISABLED
SLP Discovery Parameters
IsBroadcast = FALSE
Security = OFF
Scopes = DEFAULT
Multicast TTL = 8
Multicast Max Timeout (in msec) = 6000
Multicast Timeout Periods (in msec)= 1000, 2000, 3000, 4000, 8000
iSNS Discovery Parameters
HeartBeat = DISABLED
```

CREATING AND MAPPING INITIATOR IGROUPS TO LUNS

An igroup is a collection of iSCSI node names of initiators in an iSCSI network. The iSCSI node name includes the host's node name. When we map a LUN on a storage system to the igroup, we grant all the initiators in that igroup access to that LUN. If a host's name is not in an igroup that is mapped to a LUN, that host will not have access to the LUN. This effectively masks access to the LUN. Every iSCSI node must have a node name. The two formats, or type designators, for iSCSI node names are IQN (iSCSI Qualified Name) and EUI (Enterprise Unique Identifier). The Adaptec iSCSI initiators use the IQN format for the node names.

The procedures for mapping initiator igroups to LUNs are described in Appendix C: Initiator Mapping Example.

ORACLE OLTP DATABASE DESIGN

The 4-node RAC (Oracle Real Application Cluster) OLTP database used for this performance study simulated a real-world wholesale business with 4000 warehouses. The system sizing and configuration took into account the storage system I/O capability, the client CPU processing power, the RAM size on each node, and the number of Linux nodes in the RAC system.

The OLTP workload centers on the principal activities of an order-entry system that includes entering and delivering orders, recording payments, checking the status of orders, and monitoring the level of stock. By varying the number of users, we controlled the amount of work, the CPU load, and the I/O subsystem load.

Disk and database configurations were optimized to get the best possible performance for the OLTP database. All of the Oracle database datafiles (tables and indexes, but not log files) were striped across the available disks. Some disks were reserved for the Oracle log files. We used locally managed extent management for all tablespaces. Manual segment space management was used for the Oracle data objects that were partitioned across different nodes, while auto segment space management was used for all the other data objects. The physical data layout for the RAC database is described in Table 5 below.

Table 5) Data storage layout.

FAS960 Storage System	Number of Disks	RAID Group	WAFL Volume	LUN	Size	LUN Partitions Mapped to Linux Devices
Storage System 1	8	rg0	v11	vol11_lun1	150GB	sda1..sda15
	8	rg1		vol11_lun2	150GB	sdb1..sdb14
	8	rg2		vol11_lun3	150GB	sdc1..sdc12
				vol11_lun4	150GB	sdd1..sdd12
	8	rg0	v12	vol12_lun1	150GB	sde1..sde15
	8	rg1		vol12_lun2	150GB	sdf1..sdf15
	8	rg2		vol12_lun3	150GB	sdg1..sdg15
				vol12_lun4	150GB	sdh1..sdh13
Storage System 2	8	rg0	v21	vol21_lun1	150GB	sdi1..sdi15
	8	rg1		vol21_lun2	150GB	sdj1..sdj12
	6	rg2		vol21_lun3	150GB	sdk1..sdk12
				vol21_lun4	150GB	sdl1..sdl14
	8	rg0	v22	vol22_lun1	150GB	sdm1..sdm15
	8	rg1		vol22_lun2	150GB	sdn1..sdn13
				vol22_lun3	150GB	sdo1..sdo12
	6	rg2		vol22_lun4	150GB	sdp1..sdp14
	8	rg0	log	log_lun1	100GB	sdq1..sdq4
				log_lun2	100GB	sdr1..sdr4

We configured two FAS960 storage systems for this test. Each storage system had four disk shelves. Each disk-shelf had fourteen 72GB disks. Therefore, we used a total of 112 disks for the whole test environment. Four WAFL volumes, using 92 disks, were created on the storage systems for the Oracle datafiles. In addition, we created one WAFL volume, using eight disks, for the Oracle redo logs files. Therefore, we dedicated a total of five volumes on the two storage systems to store the Oracle OLTP database.

On each volume allocated to the Oracle datafiles, we created four 150GB LUNs. On the volume allocated to the redo log files, we created two 100GB LUNs. Therefore, we created 16 LUNs for the Oracle datafiles, and two LUNs for the redo log files, for a total of 18 LUNs on the two storage systems.

For an Oracle RAC database on Linux using raw devices, LUNs have to be partitioned, the partitions mapped to raw devices on the Linux host, and the raw devices associated with the individual Oracle datafiles, control files, and log files.

For example, using the volume allocated to the Oracle log files, we created two LUNs. These two LUNs were mapped to the Linux host as two block devices. On these two block devices, which can provide a maximum of 28 partitions, we created eight partitions. These partitions were mapped to the eight redo log files used by RAC database instances on the four nodes.

In summary, a total of 210 usable raw partitions were allocated for the creation of the Oracle 9i RAC database on a four-node cluster.

During the process of mapping raw devices to partitions, we followed the OLTP database design principle that we outlined earlier, that is, every Oracle object should make use of every available disk. Additionally, in designing the OLTP database, we partitioned the database tables across the RAC instances on the four nodes using key-range and hashing table partitioning strategies.

PARTITION ALIGNMENT

Partition tables describe a disk in terms of the number of bytes per sector, `B`, the number of sectors per track, `S`, the number of heads, `H`, and the number of cylinders, `C`. Described in this way, the disk contains `C×H×S×B` bytes. Especially with modern disks and LUNs, this is a logical description of a disk's geometry, unrelated to the physical characteristics of the underlying media. The partition table is stored in the first `S×B` bytes.

On each invocation of the Linux fdisk utility, the Adaptec iSCSI driver reports a LUN's disk geometry as 512-byte sectors with 255 heads and 63 sectors per track. The number of cylinders depends upon the size of the LUN.

For best performance, when using a NetApp storage system, partitions should start on a 4096-byte boundary. Since 4096 = 8×512, and the first partition starts at byte address S×B, partitions will be aligned on 4096-byte boundaries if the number of sectors per track, S, is divisible by eight. To that end, before allocating partitions, we instructed the Linux fdisk utility to decrease the number of sectors per track to 56 and to proportionally increase the number of cylinders.

The following example illustrates this process for a 250GB LUN with one existing 500MB primary partition, `/dev/sda1`. The default geometry, as reported by fdisk, for this LUN is 32635 cylinders, 255 heads, 63 sectors per track, and 512-byte sectors. In order to maintain the same size for the LUN, when we decrease the number of sectors per track from 63 to 56, we must increase the number of cylinders from 32635 to (32635×63)÷56 = 36714.

Note: Every time you invoke fdisk for use on a LUN on a NetApp storage system, you should decrease the number of sectors per track and increase the number of cylinders.

```
# fdisk /dev/sda
The number of cylinders for this disk is set to 32635.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
(e.g., DOS FDISK, OS/2 FDISK)
```

```
Command (m for help): **m**
Command action
  a toggle a bootable flag
  b edit bsd disklabel
  c toggle the dos compatibility flag
  d delete a partition
  l list known partition types
  m print this menu
  n add a new partition
  o create a new empty DOS partition table
  p print the partition table
  q quit without saving changes
  s create a new empty Sun disklabel
  t change a partition's system id
  u change display/entry units
  v verify the partition table
  w write table to disk and exit
  x extra functionality (experts only)

Command (m for help): p
Disk /dev/sda: 255 heads, 63 sectors, 32635 cylinders
Units = cylinders of 16065 * 512 bytes
  Device Boot    Start     End      Blocks     Id     System
  /dev/sda1      1         64       514052     83     Linux
Partition 1 does not end on cylinder boundary:
phys=(71, 254, 56) should be (71, 254, 63)
Command (m for help): x
Expert command (m for help): m
Command action
  b move beginning of data in a partition
  c change number of cylinders
  d print the raw data in the partition table
  e list extended partitions
  f fix partition order
  g create an IRIX partition table
  h change number of heads
  m print this menu
  p print the partition table
  q quit without saving changes
  r return to main menu
  s change number of sectors/track
  v verify the partition table
  w write table to disk and exit

Expert command (m for help): s
Number of sectors (1-63, default 63): 56
Warning: setting sector offset for DOS compatibility
Expert command (m for help): c
Number of cylinders (1-1048576, default 32635): 36714
The number of cylinders for this disk is set to 36714.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
(e.g., DOS FDISK, OS/2 FDISK)
Expert command (m for help): r
Command (m for help): p
Disk /dev/sda: 255 heads, 56 sectors, 36714 cylinders
Units = cylinders of 14280 * 512 bytes
  Device Boot    Start     End      Blocks     Id     System
  /dev/sda1      1         72       514052     83     Linux

Command (m for help):
```

The fdisk session is now ready to create another partition that will be aligned on a 4096-byte boundary. We create a second primary partition, `/dev/sda2`, containing 250MB.

```
Command (m for help): n
Command action
e extended
p primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (73-36714, default 73):
Using default value 73
Last cylinder or +size or +sizeM or +sizeK (73-36714, default 36714): +250M
Command (m for help): p
Disk /dev/sda: 255 heads, 56 sectors, 36714 cylinders
Units = cylinders of 14280 * 512 bytes
  Device Boot    Start     End      Blocks     Id     System
  /dev/sda1      1         72       514052     83     Linux
  /dev/sda2      1         108      257040     83     Linux

Command (m for help): w
The partition table has been altered!
Calling ioctl() to re-read partition table.
```

```
WARNING: If you have created or modified any DOS 6.x
partitions, please see the fdisk manual page for additional
information.
Syncing disks.
#
```

4 ORACLE 9I RAC BEST PRACTICES FOR IA32 LINUX

This section highlights specific configuration settings and procedures used to setup the Oracle 9i RAC cluster that merit special attention as recommended best practices. Material in this section will cover configuration of the Linux operating system, installation and configuration of Oracle 9i RAC, and installation and configuration of the Adaptec 7211C HBA.

MULTI-LUN SUPPORT IN THE LINUX KERNEL

The default Linux kernel settings will not probe for LUNs other than LUN 0. To enable multi-LUN support, we used the following procedure:

1. Install the kernel along with its source:

   ```
   # rpm -ivh kernel-smp-2.4.9-e.40.i686.rpm
   # rpm -ivh kernel-source-2.4.9-e.40.i386.rpm
   ```

2. Verify the installation of the packages:

   ```
   # rpm -qa | grep e.40
   kernel-source-2.4.9-e.40
   kernel-smp-2.4.9-e.40
   #
   ```

3. Add the following parameter to the `/etc/modules.conf` file:

   ```
   options scsi_mod max_scsi_luns=15
   ```

   With this setting, each target will be able to see a maximum of 15 LUNs.

4. Regenerate the RAM disk:
5. Run lilo to update the bootloader's data structures.

   ```
   # mkinitrd /boot/initrd-2.4.9-e.40smp.img 2.4.9-e.40smp
   ```

DRIVER FOR THE ADAPTEC 7211C HBA

The initial release of Adaptec’s iSCSI driver was not enabling block coalescing (varyio) and it was not avoiding bounce buffers (highmemio) for the raw layer. As a result, the I/O throughput was sub-optimal for the raw interface.

To understand the behavior of the I/O sub-system with the raw interface, we collected the kernel profile using the Linux utility readprofile. To this end, we added the following boot time parameters to the `/etc/lilo.conf` file and reran the **`/sbin/lilo`** command to enable kernel profiling during subsequent boots:

```
append=“profile=1 nmi_watchdog=1”
```

We rebooted the system. Once the system had rebooted, we cleared the profiler buffer and then ran the read or write I/O test using the Linux dd utility:

```
# /usr/sbin/readprofile –r
```

Upon completion of the test, we collected a kernel profile and put it into a text file using the following command:

```
# /usr/sbin/readprofile -m <mapfile> > <filename.txt>
```

We measured the I/O throughput using the Linux utilities vmstat and iostat on the host system and using the Data ONTAP sysstat utility on the storage system target. The `sysstat -i` option provides iSCSI performance statistics on the storage system.

After enabling varyio and highmemio in the iSCSI driver (through `can_do_varyio` and `can_dma_32` bits), the I/O throughput with raw interface improved significantly.

LINUX CONSIDERATIONS FOR ORACLE 9I RAC DATABASES USING RAW DEVICES

The Linux operating system limits the number of SCSI disks that can be connected to a machine running an Oracle RAC instance to 128. It also limits the number of partitions on each disk to a maximum of 15. Of these 15 partitions per disk, only 14 can generally be used. This causes inflexibility when creating a RAC database on shared raw devices, because each partition can only be associated with one raw device. The maximum number of raw devices the Linux kernel can access is 255. This inevitably limits the maximum size of a RAC database and maximum number of RAC instances that can participate in a shared database cluster. Configuring shared raw devices involves two steps: disk partitioning and raw device binding. The disk partitioning can easily be accomplished by using the fdisk utility. If required, use the mknod utility to create the raw devices as follows:

```
# mknod /dev/raw/<raw#> c <major#> <minor#> (as user root) and:
```

```
# chown oracle:dba /dev/raw/<raw#>
```

Raw device binding must be performed for every RAC node that uses the shared partitions. Once the disk partitions of the correct size are created, the device to raw mapping can be achieved by editing the `/etc/sysconfig/rawdevices` file on every RAC node as follows:

```
/dev/raw/<raw-device> /dev/<partition>
```

Or, manually by binding as follows:

```
# raw /dev/raw/<raw#> /dev/<partition> (as user root)
```

Then, create the symbolic links for database access, as follows:

```
# ln –s /dev/raw/<raw#> /<fs-mntpt>/…/<db-file-name>
```

The actual size of the partitions and the status of raw device bindings can be determined using the following commands, respectively:

```
# fdisk -l
```

```
# raw -qa
```

ASYNCHRONOUS I/O

To enable asynchronous I/O, Oracle program binaries must first be re-linked. Make sure to backup the existing Oracle program binaries before re-linking. Use the following command to backup and re-link Oracle software:

1. Shutdown all running instances of Oracle

   ```
   # cd $ORACLE_HOME/lib # cp –a oracle oracle_org# cd $ORACLE_HOME/rdbms/lib #
   make -f ins_rdbms.mk async_on # make -f ins_rdbms.mk ioracle
   ```

   ```
    (backup Oracle) (enable async_io) (relink Oracle binary)
   ```

2. In addition to re-linking, the parameter `disk_asynch_io` has to be set to true in the Oracle initialization parameter file (`init<sid>.ora`).

3. If the Oracle datafiles reside on filesystems instead of raw devices, the following parameter also has to be set:

```
filesystemio_options = asynch
```

4. For better I/O throughput, the `/proc/sys/fs/aio-max-size` has to be increased from the default of 131072 bytes to at least 1MB. This can be done by executing the following command as `root` user:

```
# echo 1048576 > /proc/sys/fs/aio-max-size
```

LARGE SGA

There are several ways to extend the Oracle SGA beyond the default of approximately 1.7GB on 32-bit Linux. Since each node was configured with 4GB of memory, the approach chosen was to lower the Oracle base address. This requires re-linking of the Oracle binaries with a lowered SGA base address as follows:

1. Shutdown all running instances of Oracle:

```
# cd $ORACLE_HOME/lib

# cp -a libserver9.a libserver9.a.org (backup libserver9.a)

# cd $ORACLE_HOME/bin

# cp -a oracle oracle.org (backup Oracle)

# cd $ORACLE_HOME/rdbms/lib

# genksms -s 0x12000000 > ksms.s (lower SGA base address)

# make -f ins_rdbms.mk ksms.o

# make -f ins_rdbms.mk ioracle (relink Oracle binary)

and, as user root, set the shmmax kernel parameter to 3GB, as follows:

# echo 3000000000 > /proc/sys/kernel/shmmax
```

   or:

```
# sysctl -w kernel.shmmax=3000000000
```

2. Additionally, as user root, lower the mapped base for the processes running Oracle, as follows:

```
# echo $$ (to get the process id of the shell)

# echo 0x10000000 > /proc/<shell-pid>/mapped_base
```

3. Startup the Oracle instance and other Oracle processes from this shell.

With these settings, the maximum size of SGA would be approximately 2.7 GB.

CLUSTER MANAGER CONFIGURATION

A kernel module called hangcheck-timer provides cluster reliability and high-availability for the Oracle 9i RAC Cluster Manager. It employs a kernel-based timer to periodically verify that the system task scheduler is functioning correctly and resets the node immediately when a system hang has been observed. It has two parameters:

- `hangcheck-tick:` The period of time between checks of system health by the module (recommended setting: 30 seconds).
- `hangcheck-margin:` The maximum hang delay that the module will tolerate before resetting the node during the time between ticks (recommended setting: 180 seconds).

The hangcheck-timer kernel module is loaded as follows:

```
/sbin/insmod hangcheck-timer hangcheck_tick=30 hangcheck_margin=180
```

The use of the hangcheck-timer module requires coordination between the settings of hangcheck-tick and hangcheck-margin and the MissCount parameter of the Cluster Manager. This is done to make sure that the Cluster Manager does not declare an instance to be dead, therefore forcing a cluster reconfiguration, and starting a RAC node recovery until a hung node is guaranteed to be reset. The MissCount parameter for the Cluster Manager (in the cmcfg.ora) must be set using this formula:

```
MissCount > hangcheck-tick + hangcheck-margin
```

The hangcheck-timer introduces the following configuration parameter used in `cmcfg.ora` to allow the Cluster Manager to know the name of the hangcheck-timer kernel module so it can determine if it is correctly loaded:

```
KernelModuleName=hangcheck-timer
```

CLUSTER INTERCONNECT

CACHE FUSION

Cache Fusion is an important component of Oracle's Real Application Cluster configuration. It is a shared cache that extends in a transparent manner database applications from a single system to a multi-node shared-disk cluster.

Each Oracle instance inside a RAC system has its own buffer cache of disk buffers, and taken together, these local caches form a global buffer cache. To maintain cache coherency in this global cache, a global resource control service is required. This control mechanism is called Global Cache Service (GCS). For additional details on RAC and GCS, see Reference [2].

GCS controls the location and access modes of all cache resources (data blocks) in the global cache. It synchronizes global cache accesses, allowing only one instance at a time to modify a cache resource. The GCS uses a distributed architecture whereby each instance shares the responsibility of managing a subset of the global cache.

By maintaining a global view of all data blocks, GCS can direct a read or write request to the instance that has the current cached buffer for that block. The instance that is the current holder will transfer the cached buffer to the requestor's instance directly, and GCS will update the holder data structures to reflect that the requesting instance is now the holder.

Using Cache Fusion, data blocks are shipped directly from one node to another using interconnect messaging, eliminating the need for extra I/Os to facilitate sharing. Cache Fusion exploits recent advances in clustering technology by using a low latency network protocol. Oracle instances, in this way, can directly share the contents of their buffer caches, resulting in a shared-cache clustered database architecture.

Critical to the scalability and efficiency of a clustered database is the efficiency of internode messaging. There are three factors to be considered that contribute to the efficiency of inter-node messaging:

- Latency of inter-node messaging
- Number of nodes involved in servicing a request
- Frequency of inter-node synchronization events

For maximum interconnect performance, we have configured the inter-node messaging transport with one Gigabit Ethernet network interface card (NIC) on a dedicated LAN, obtaining high-efficiency. In our experience with the OLTP workload we used for stress testing, we were able to saturate the host CPUs without saturating the Gigabit inter-node messaging LAN. We recommend the use of two Gigabit interconnect cards to eliminate potential single point of failures for the GCS services.

A dedicated switch and network for the interconnect traffic should be used. Other network traffic to the host server (such as client sessions and server responses) should go through a separate NIC and network.

For efficient inter-process communication in a RAC environment, the UDP send and receive buffer sizes have to be adjusted as follows. UDP is used as the default protocol on Linux. This can be accomplished by invoking the following commands as the root user:

```
# sysctl -w net.core.rmem_max=262144

# sysctl -w net.core.wmem_max=262144

# sysctl -w net.core.rmem_default=262144
# sysctl -w net.core.wmem_default=262144
```

RECOVERY IN A RAC ENVIRONMENT

The RAC architecture guarantees availability of the database as long as at least one instance is alive. Recovery cost is also proportional to the number of failures, not the total number of nodes in the cluster since only redo logs from the failed node are read and applied. Recovery is also sped-up by the fact that disk reads for recovery are eliminated for blocks that are present in a surviving node's cache. Parallel recovery mechanisms make use of all surviving nodes, parallelizing disk I/Os for log reads and data blocks reads across multiple recovering instances.

5 ANALYSIS OF ORACLE 9I RAC SCALABILITY

The database block size we used was 4K. This block size was chosen to achieve a high level of transactional concurrency for the type of workload (OLTP) used for this study. We tuned the memory for a single RAC instance and used those settings for the two-node and four-node tests. Of the Oracle memory caches, namely, shared pool, log buffer, and buffer cache; shared pool and log buffer were sized based on their usage. The rest of the SGA was assigned to the buffer cache. The system memory usage was monitored using the `/proc/meminfo` and the Linux utility free.

We chose Oracle's Statspack utility to monitor performance problems in the Oracle 9i RAC database system. The top five wait events listed in the statspack report provide some detail with regards to potential system bottlenecks. To identify which of these are major issues, we investigated the granular reports within other sections of the statspack report.

Initial statspack reports had lots of free buffer waits indicating that DBWR could not write fast enough to free up database buffers for foreground processing. To reduce the free buffer waits, we increased the number of DBWR processes to two.

We experienced some problems with Oracle 9i RAC cache fusion resulting in a significant drop in performance. Initially the average wait time of global cache cr request was approximately 20 milliseconds with CPU idle of approximately 40% on one node and approximately 5 milliseconds with CPU idle of less than 3% on the other node during the two node tests.

The CPU utilization was monitored using the Linux utilities vmstat and top during these tests. Since UDP is the default protocol used for cluster interconnects, we monitored for any collisions, errors, and lost packets on this interface using the Linux utility netstat and noticed no significant errors.

It turned out that only one of the nodes would experience faster response times from the database, and thereby it would become too busy, and it could not service global cache requests to the other node. This resulted in an unbalanced workload distribution as the node with lots of CPU idle time was not getting serviced fast enough for its global cache requests. This condition was avoided by increasing the scheduling priority of the LMS processes. This allowed LMS processes to service the incoming messages faster and process them at a much higher rate. To increase the process scheduling priorities, we followed these steps as user root:

`# ps auxw | grep lms` (Get process IDs of LMS processes)

`# renice -20 -p <pid1, pid2`,.. of LMS processes> (Increase scheduling priority)

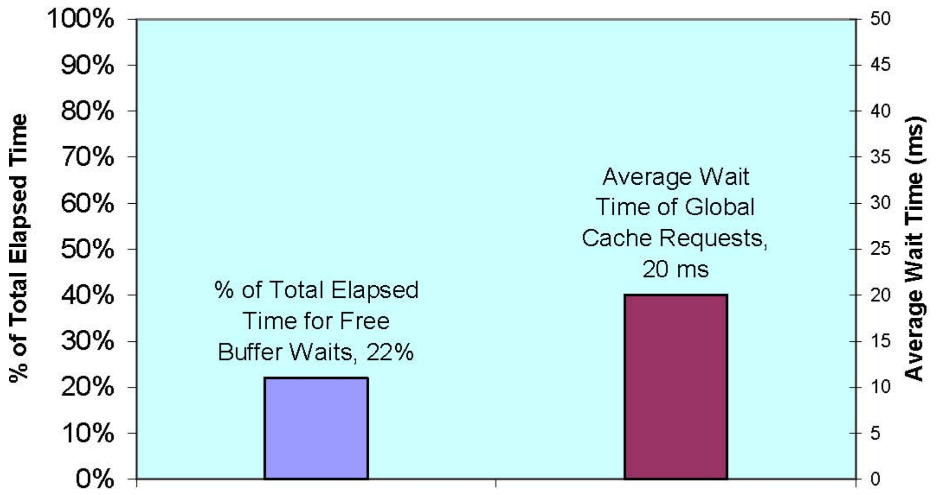

Figure 5) Before tuning—free bugger and GCS requests waits.

After `renicing` the LMS processes, the average wait time of global cache `cr` request was reduced to 4 milliseconds with CPU idle of less than 3% on both nodes. These statistics remained similar up to 4 nodes. Figure 5 and Figure 6 shows the measured trends.

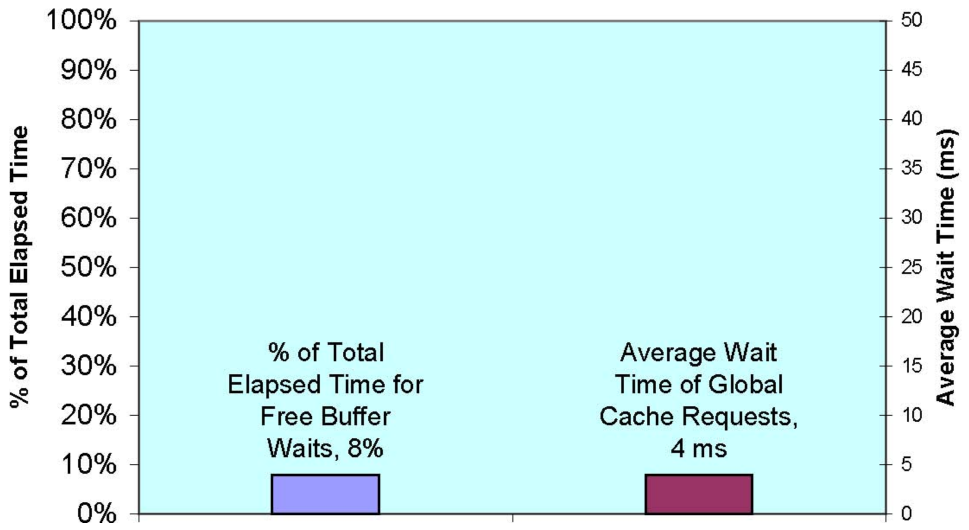

Figure 6) After tuning—free buffer and GCS requests waits.

The average wait for log file sync was still 40 milliseconds while the average wait for log file parallel write was 1 millisecond in the final tuned version of the RAC instance. We also observed that the average wait for log file sync increased as the load increased. This indicates that there were no significant disk I/O problems, but there were some performance issues with inter-process communication efficiency between the LGWR and the foreground processes. We also noticed that Red Hat Advanced Server 2.1 has some issues with scalability of the semaphore operations. These problems have been addressed and fixed in Red Hat Enterprise Linux 3.0; our planned tests on Red Hat Enterprise Linux

3.0 with 10g RAC will verify and validate that the semaphore improvements will actually fix the IPC performance issues with LGWR.

A tabular form of the vmstat output collected for every 30 seconds interval is shown in Figure 7 below for one of the nodes of a tuned Oracle 9i RAC instance. This shows CPU utilization close to 97%, indicating the configuration was CPU bound.

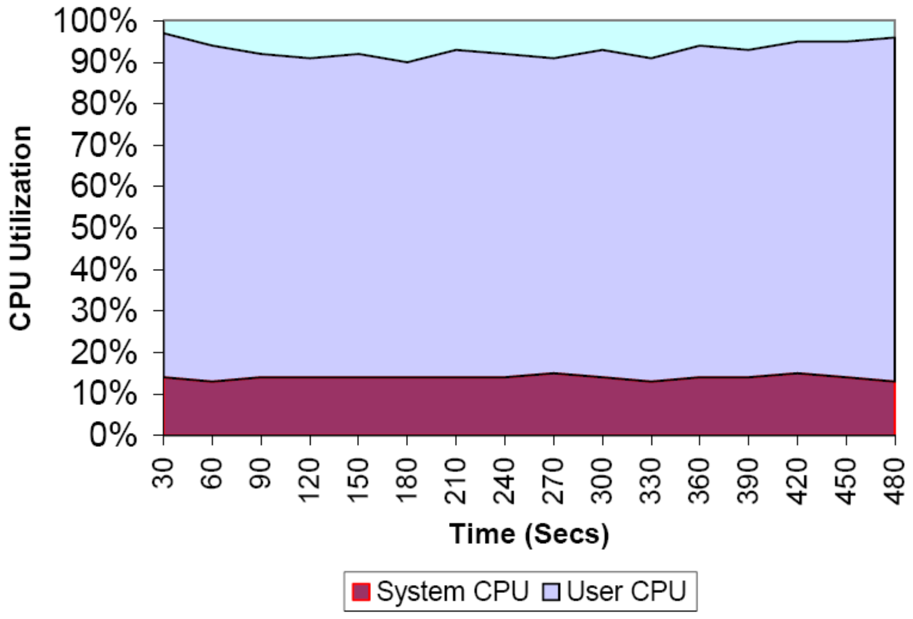

Figure 7) CPU stats taken on Node 1 of the 4-Node RAC System (Using vmstat).

Due to the characteristics of the OLTP workload, a large amount of activity is centered on searching tables. Because of the complex relationships existing between the entities of the data model, indexing optimizes read/write time. Since an OLTP environment has a large amount of relationships, much of the lookup activity is done via primary and foreign keys.

To increase efficiency, concurrency, and performance of OLTP applications based on Oracle 9i RAC, we used indexing structures based on range partitioning and hashing. Oracle 9i RAC physically restructures the records in a given table to conform to the index structure.

Over time, insertions and deletions may cause allocations and de-allocations of index pages. Index pages may become less than half full causing a drop in the space utilization and an increase in the number of disk reads required to read the same number of index keys. An index may also become declustered (that is, index keys within a key range will not be in contiguous disk space) resulting in performance degradation during queries. To restore the clustering of data, we can drop and recreate the index. However, this operation requires holding a shared table lock on the table, thereby making the table unavailable to OLTP transactions, which is not acceptable for performance measurement. We used the storage system Snapshot capability to recreate the database under test, at regular intervals, by restoring the database from a previously materialized Snapshot copy.

RESULTS

Scalability is a measure of how well the database workload performs cluster-wide as nodes are added. We ran an OLTP workload and scaled the load to saturate the Host CPUs. We obtained scalability results by measuring the OLTP throughput in transactions per minutes (TPM) on single Oracle 9i RAC node first to establish a baseline. We then increased the number of nodes in the cluster to two and four nodes, while scaling the database size and the OLTP workload.

For two nodes, we calculated the throughput results as a percent increase from the baseline of one node. For four nodes, the baseline used was the two-node throughput. In this configuration we were able to achieve an 80% scalability factor when scaling from one node to two Oracle 9i RAC nodes, and of 90% when scaling from two to four Oracle 9i RAC nodes.

Our measurements are summarized in Figure 8, Figure 9, and Figure 10.

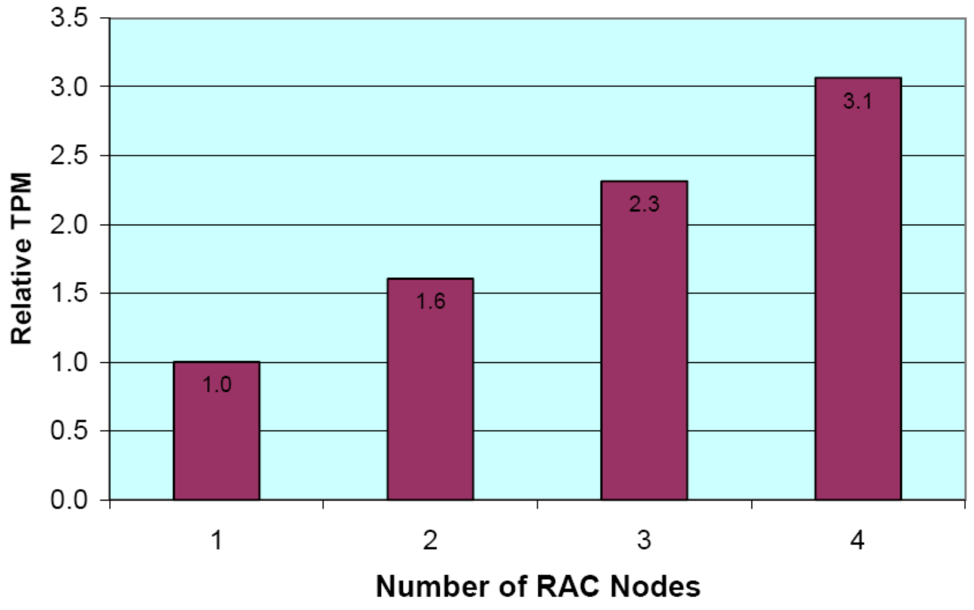

Figure 7) Scalability metrics.

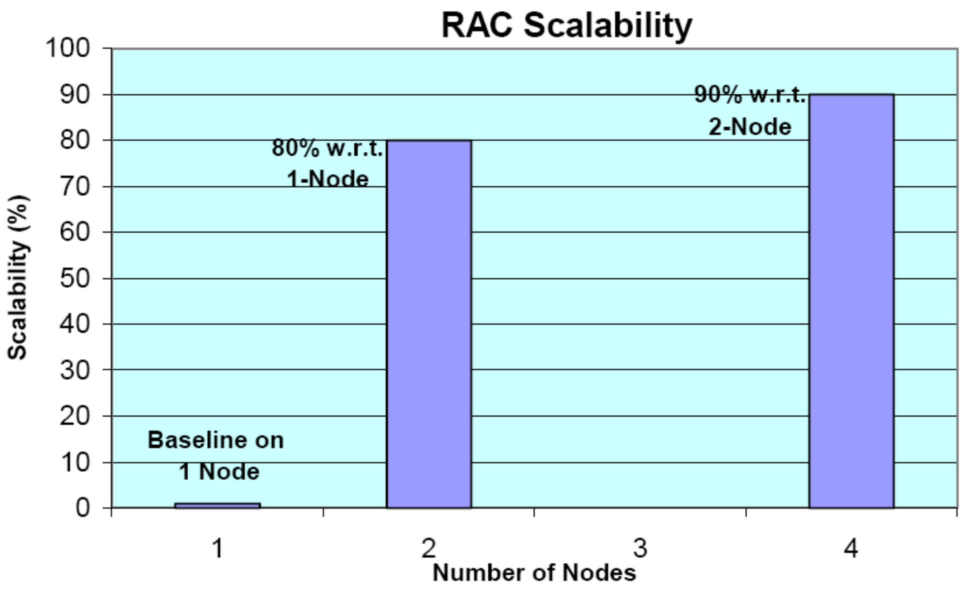

Figure 8) RAC scalability.

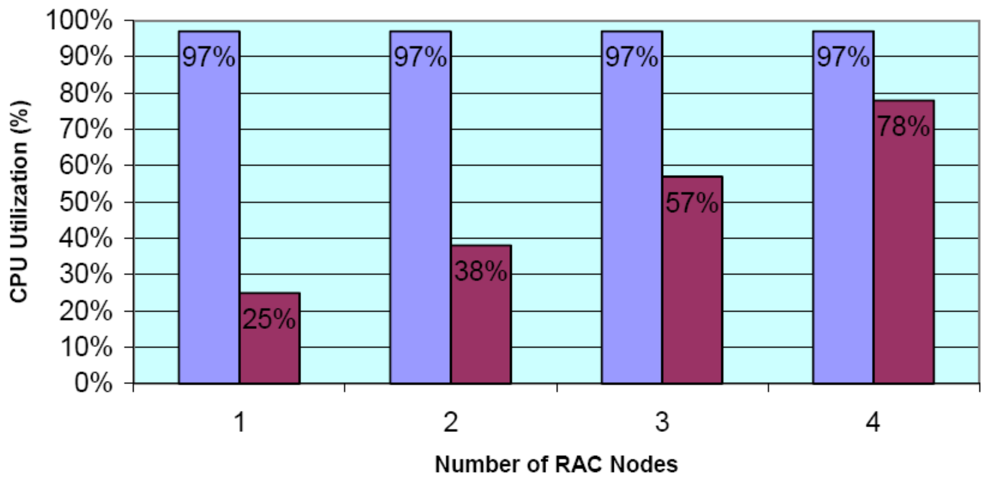

Figure 9) Storage system and Host CPU utilization.

Our test results and experiences have shown that the performance of the iSCSI transport and fabric used in our system under test have proven to be at the high-end of our expectations. The stability of the I/O fabric, as well as of the overall system, has been very good and suitable for a production deployment. These observations are based on similar experiences with equivalent systems using Direct-Attached Disks (DAS).

The requirement of full coherency, that the GCS component of Oracle 9i RAC imposes, was easy to meet with iSCSI, since the iSCSI stack, even when using a host-based Cluster File System, guarantees atomicity and a strong cache coherency model.

6 FUTURE WORK

We will continue this study by investigating a Oracle 9i RAC 4-node cluster using an OLTP workload with the Oracle Cluster File System (OCFS), as well as measuring performance characteristics for a decision support workload (DSS). We also plan to exercise iSCSI for a 10g RAC database using raw devices and OCFS with OLTP and DSS workloads.

Adaptec will continue to enhance and improve both the iSCSI HBA offering and the driver. The next software version of the driver for the Adaptec 7211C iSCSI HBA is Palomar 1.2. It will provide a firmware and driver upgrade. There will be no hardware changes to the current 7211 HBA single port TOE ASIC. The major features of the new Linux driver will be:

- iSCSI boot (also called remote boot or BIOS boot)
- Target redirection support

The follow-on product from Adaptec will be based on the next generation of ASIC. It will support dual 1Gigabit Ethernet ports, and in addition to the TCP/IP offload, it will also have the iSCSI and IPSec offload.

7 SUMMARY AND CONCLUSIONS

We have presented our experiments and observations, as well as the best practices we implemented, that have been derived from our stress and scalability tests. Following these best practices a user can successfully run his Oracle 9i RAC applications with good performance and scalability on Linux with iSCSI.

iSCSI leverages the convergence of two very mature technologies, TCP/IP networking technology and SCSI storage technology, enabling remote storage over ubiquitous IP networking infrastructure.

In this study, we have examined an iSCSI storage organization under an OLTP database workload and analyzed its performance. We conducted block-level performance measurement using raw devices for the database tablespaces. We show that iSCSI storage with Gigabit connections can achieve reasonable performance for a commercial mission-critical OLTP environment. This study helps us better understand the characteristics of iSCSI storage in an OLTP database transactional environment and can be used to identify performance bottlenecks.

TCO CONSIDERATIONS

The iSCSI protocol essentially encapsulates a SCSI subsystem within a TCP/IP connection. This allows for greater flexibility within a Storage Area Network (SAN), as they can now be implemented using less expensive components like Gigabit Ethernet Interfaces and Gigabit Ethernet switches.

The iSCSI protocol can offer performance and scalability characteristics that compare quite favorably with an equivalent FCP-SAN topology, with iSCSI offering a lower Total Cost of Ownership (TCO). A protocol such as iSCSI offers a lower TCO, because an organization can leverage the same network infrastructure for:

- Normal non-storage network traffic
- For its NAS traffic
- For its SAN traffic

Thus, the combination of Gigabit Ethernet and TCP/IP networks will eventually become the dominant storage protocol combination. Additionally, iSCSI overcomes the drawback that Fibre Channel cannot be used over a Wide Area Network (WAN), since Fibre Channel is not truly WAN-enabled. More importantly, an organization’s investment in training their network administrators to acquire skills in IP-based management tools such as IPSec for security, SLP for discovery, SNMP for monitoring and configuration, OSPF for routing, and DiffServ/IntServ for QoS cannot be transferred to Fibre Channel. Also, Ethernet host adapters and IP switches can be reused for iSCSI networks. Given that an organization’s investment in an IP network cannot be carried over to Fibre Channel networks, the iSCSI protocol has been created to leverage the management advantages of IP networks and the low cost advantages of Ethernet cards and IP routers due to high volume manufacturing.

The iSCSI protocol can be implemented completely in software (iSCSI and TCP/IP layers are executed by the host CPU), or completely offloaded to an adapter card such as the Adaptec 7211C (both the iSCSI and TCP/IP layers execute on a HBA).

The Adaptec 7211C iSCSI Adapter Card, which provides a TCP Off-Load Engine (TOE), offloads the TCP/IP stack from the host. It also minimizes the number of host interrupts from the network card by interrupting the host once for each TCP segment. When the TCP layer is in software on the host, then a network card will interrupt the host more frequently, once for each Ethernet frame. Although some Ethernet Network Cards have some form of interrupt coalescing, they are not as effective as TCP segment level interrupt coalescing that is possible with a TOE card.

The lower TCO realized with iSCSI is particularly interesting when deploying a scalable cluster with many host clients having to interconnect and share a common storage subsystem like Oracle 9i RAC.

8 ACKNOWLEDGEMENTS

We wish to acknowledge the contributions of: Kotaro Ono, Stefan Pommerenk, and Stefan Roesch from Oracle Technology and Business Solutions for their consulting help in tuning the OLTP workload and Oracle 9i RAC. We also wish to acknowledge Dan Morgan from the NetApp Performance Group for his contributions in designing an optimal data layout and aggregation scheme for the storage system storage used in this study.

The authors would like to thank all the reviewers for their many helpful comments and suggestions.

9 REFERENCES

1. Roland Knapp, RAC on Linux best practices, Oracle Metalink Note#: 240575.1.
2. Oracle Corporation. Oracle 9i Real Application Clusters Concepts Release 1 (9.0.1), Part Number A89867-01.
3. Adaptec, 7211C iSCSI driver version 1.11, which can be downloaded from

   http://www.adaptec.com/worldwide/support/drivers_by_product.jsp?sess=no&language=English+US&cat=/Product/ASA-7211C
4. Adaptec, iConfig Utility User's Guide, Part Number 513194-06, Version AA, February 2003.
5. Adaptec, ASA-7211C/F iSCSI Adapter for Initiator Applications Installation Guide, Part Number 513193-06, Version AB, June 2003.

10 APPENDIX A: STORAGE SYSTEM SETUP

VOLUME CREATION

We used two FAS960 storage systems with four DS-14 shelves per head. We created two WAFL volumes per storage system with the RAID group size of eight. These volumes were created on storage systems fas960c-svl1 (Filer 1) and fas960c-svl2 (Filer 2) as follows:

FILER 1

```
fas960c-svl1*> vol create v11 -d 2a.18 2a.19 2a.20 2a.21 2a.22 2a.23 2a.34 2a.35 2a.36
                                  2a.37 2a.38 2a.39 3a.17 3a.18 3a.19 3a.20 3a.21 3a.22
                                  3a.32 3a.33 3a.34 3a.35 3a.36 3a.37
fas960c-svl1*> vol create v12 -d 2a.24 2a.25 2a.26 2a.27 2a.28 2a.29 2a.40 2a.41 2a.42
                                  2a.43 2a.44 2a.45 3a.23 3a.24 3a.25 3a.26 3a.27 3a.28
                                  3a.38 3a.39 3a.40 3a.41 3a.42 3a.43
```

FILER 2

```
fas960c-svl2*> vol create v21 -d 2a.16 2a.18 2a.19 2a.20 2a.21 2a.22 2a.33 2a.34 2a.35
                                  2a.36 2a.37 2a.38 3a.24 3a.25 3a.26 3a.32 3a.33 3a.34
                                  3a.35 3a.36 3a.37 3a.38
fas960c-svl2*> vol create v22 -d 2a.23 2a.24 2a.25 2a.26 2a.27 2a.28 2a.39 2a.40 2a.41
                                  2a.42 2a.43 2a.44 3a.27 3a.28 3a.29 3a.39 3a.40 3a.41
                                  3a.42 3a.43 3a.44 3a.45
fas960c-svl2*> vol create log -d 3a.16 3a.17 3a.18 3a.19 3a.20 3a.21 3a.22 3a.23
```

The volume details after the creation on the two storage systems are shown as follows:

```
fas960c-svl1*> vol status

Volume       State      Status        Options

v11          online     raid4         nosnap=on, minra=on

v12          online     raid4         nosnap=on, minra=on

vol0         online     raid4         root

fas960c-svl1*>

fas960c-svl2*> vol status

Volume       State      Status        Options

vol0         online     raid4         root

v21          online     raid4         nosnap=on, minra=on

v22          online     raid4         nosnap=on, minra=on

log          online     raid4         nosnap=on, minra=on

fas960c-svl2*>
```

Note: NetApp does not recommend using the "`minra`" option to restrict the storage system's read ahead feature.[1]

[1] Historically, NetApp had recommended disabling aggressive storage readahead for OLTP (Online Transaction Processing) database workloads by setting the Data ONTAP parameter "`minra`" to "`on`". Data ONTAP 6.5.1, however, introduced significant changes to the readahead algorithm, making it more intelligent and efficient. Hence, disabling readahead for database workloads is no longer recommended. Recent experience indicates that in fact, enabling `minra` may lower the overall database performance. As a result, NetApp now recommends that the `minra` setting be left in the default "`off`" state unless explicit guidance to do otherwise is given by the NetApp Global Support Organization.

Examples of the WAFL volumes v11, v12, v21 and v22 are shown below:

FILER 1-VOLUME V11

```
fas960c-svl1*> vol status v11 -r
Volume v11 (online, raid4) (block checksums)
   Plex /v11/plex0 (online, normal, active)
   RAID group /v11/plex0/rg0 (normal)
   RAID Disk  Device   HA   SHELF  BAY    CHAN  Used (MB/blks)    Phys(MB/blks)
   ---------------------------------------------------------------------------
   parity     3a.37    3a    2     5      FC:A  68000/139264000   68552/140395088
   data       2a.19    2a    1     3      FC:A  68000/139264000   68444/140174232
   data       2a.20    2a    1     4      FC:A  68000/139264000   68444/140174232
   data       2a.21    2a    1     5      FC:A  68000/139264000   68444/140174232
   data       2a.22    2a    1     6      FC:A  68000/139264000   68444/140174232
   data       2a.23    2a    1     7      FC:A  68000/139264000   68444/140174232
   data       2a.34    2a    2     2      FC:A  68000/139264000   68444/140174232
   data       2a.35    2a    2     3      FC:A  68000/139264000   68444/140174232
   RAID group /v11/plex0/rg1 (normal)
   RAID Disk  Device   HA   SHELF  BAY    CHAN  Used (MB/blks)    Phys(MB/blks)
   ---------------------------------------------------------------------------
   parity     2a.36    2a    2     4      FC:A  68000/139264000   68444/140174232
   data       2a.37    2a    2     5      FC:A  68000/139264000   68444/140174232
   data       2a.38    2a    2     6      FC:A  68000/139264000   68444/140174232
   data       2a.39    2a    2     7      FC:A  68000/139264000   68444/140174232
   data       3a.17    3a    1     1      FC:A  68000/139264000   68444/140174232
   data       3a.18    3a    1     2      FC:A  68000/139264000   68444/140174232
   data       3a.19    3a    1     3      FC:A  68000/139264000   68444/140174232
   data       3a.20    3a    1     4      FC:A  68000/139264000   68444/140174232

   RAID group /v11/plex0/rg2 (normal)
   RAID Disk  Device   HA   SHELF  BAY    CHAN  Used (MB/blks)    Phys(MB/blks)
   ---------------------------------------------------------------------------
   parity     3a.21    3a    1     5      FC:A  68000/139264000   68444/140174232
   data       3a.22    3a    1     6      FC:A  68000/139264000   68444/140174232
   data       3a.32    3a    2     0      FC:A  68000/139264000   68552/140395088
   data       3a.33    3a    2     1      FC:A  68000/139264000   68552/140395088
   data       3a.34    3a    2     2      FC:A  68000/139264000   68552/140395088
   data       3a.35    3a    2     3      FC:A  68000/139264000   68552/140395088
   data       3a.36    3a    2     4      FC:A  68000/139264000   68552/140395088
   data       2a.18    2a    1     2      FC:A  68000/139264000   68444/140174232
```

FILER 1-VOLUME V12

```
fas960c-svl1*> vol status -r v12
Volume v12 (online, raid4) (block checksums)
  Plex /v12/plex0 (online, normal, active)
   RAID group /v12/plex0/rg0 (normal)
   RAID Disk   Device   HA   SHELF   BAY   CHAN   Used (MB/blks)      Phys(MB/blks)
   ---------------------------------------------------------------------------------
   parity      3a.43    3a    2      11    FC:A   68000/139264000    68444/140174232
   data        2a.25    2a    1      9     FC:A   68000/139264000    68444/140174232
   data        2a.26    2a    1      10    FC:A   68000/139264000    68444/140174232
   data        2a.27    2a    1      11    FC:A   68000/139264000    68444/140174232
   data        2a.28    2a    1      12    FC:A   68000/139264000    68444/140174232
   data        2a.29    2a    1      13    FC:A   68000/139264000    68444/140174232
   data        2a.40    2a    2      8     FC:A   68000/139264000    68444/140174232
   data        2a.41    2a    2      9     FC:A   68000/139264000    68444/140174232
   RAID group /v12/plex0/rg1 (normal)
   RAID Disk   Device   HA   SHELF   BAY   CHAN   Used (MB/blks)      Phys(MB/blks)
   ---------------------------------------------------------------------------------
   Parity      2a.42    2a    2      10    FC:A   68000/139264000    68444/140174232
   data        2a.43    2a    2      11    FC:A   68000/139264000    68444/140174232
   data        2a.44    2a    2      12    FC:A   68000/139264000    68444/140174232
   data        2a.45    2a    2      13    FC:A   68000/139264000    68444/140174232
   data        3a.23    3a    1      7     FC:A   68000/139264000    68444/140174232
   data        3a.24    3a    1      8     FC:A   68000/139264000    68444/140174232
   data        3a.25    3a    1      9     FC:A   68000/139264000    68444/140174232
   data        3a.26    3a    1      10    FC:A   68000/139264000    68444/140174232

   RAID group /v12/plex0/rg2 (normal)
   RAID Disk   Device   HA   SHELF   BAY   CHAN   Used (MB/blks)      Phys(MB/blks)
   ---------------------------------------------------------------------------------
   parity      3a.27    3a    1      11    FC:A     68000/139264000  69536/142410400
   data        3a.28    3a    1      12    FC:A   68000/139264000    68444/140174232
   data        3a.38    3a    2      6     FC:A   68000/139264000    68552/140395088
   data        3a.39    3a    2      7     FC:A   68000/139264000    68552/140395088
   data        3a.40    3a    2      8     FC:A   68000/139264000    68552/140395088
   data        3a.41    3a    2      9     FC:A   68000/139264000    69536/142410400
   data        3a.42    3a    2      10    FC:A   68000/139264000    69536/142410400
   data        2a.24    2a    1      8     FC:A   68000/139264000    68444/140174232
fas960c-svl1*>
```

FILER 2-VOLUME V21

```
fas960c-svl2*> vol status -r v21

Volume v21 (online, raid4) (block checksums)

  Plex /v21/plex0 (online, normal, active)

    RAID group /v21/plex0/rg0 (normal)

    RAID Disk  Device  HA  SHELF  BAY  CHAN  Used (MB/blks)     Phys(MB/blks)
    ----------------------------------------------------------------------------
    parity     3a.38   3a   2     6    FC:A  68000/139264000    68444/140174232
    data       2a.18   2a   1     2    FC:A  68000/139264000    68552/140395088
    data       2a.19   2a   1     3    FC:A  68000/139264000    68552/140395088
    data       2a.20   2a   1     4    FC:A  68000/139264000    68552/140395088
    data       2a.21   2a   1     5    FC:A  68000/139264000    68552/140395088
    data       2a.22   2a   1     6    FC:A  68000/139264000    68552/140395088
    data       2a.33   2a   2     1    FC:A  68000/139264000    68444/140174232
    data       2a.34   2a   2     2    FC:A  68000/139264000    68444/140174232

    RAID group /v21/plex0/rg1 (normal)

    RAID Disk  Device  HA  SHELF  BAY  CHAN  Used (MB/blks)     Phys(MB/blks)
    ----------------------------------------------------------------------------
    parity     2a.35   2a  2      3    FC:A  68000/139264000    68444/140174232
    data       2a.36   2a  2      4    FC:A  68000/139264000    68444/140174232
    data       2a.37   2a  2      5    FC:A  68000/139264000    68444/140174232
    data       2a.38   2a  2      6    FC:A  68000/139264000    68444/140174232
    data       3a.24   3a  1      8    FC:A  68000/139264000    68444/140174232
    data       3a.25   3a  1      9    FC:A  68000/139264000    68444/140174232
    data       3a.26   3a  1      10   FC:A  68000/139264000    68444/140174232
    data       3a.32   3a  2      0    FC:A  68000/139264000    68444/140174232

    RAID group /v21/plex0/rg2 (normal)

    RAID Disk  Device  HA  SHELF  BAY  CHAN  Used (MB/blks)     Phys(MB/blks)
    ----------------------------------------------------------------------------
    parity     2a.16   2a  1      0    FC:A  136000/278528000   137104/280790184

    data       3a.34   3a  2      2    FC:A  68000/139264000    68444/140174232
    data       3a.35   3a  2      3    FC:A  68000/139264000    68444/140174232
    data       3a.36   3a  2      4    FC:A  68000/139264000    68444/140174232
    data       3a.37   3a  2      5    FC:A  68000/139264000    68444/140174232
    data       3a.33   3a  2      1    FC:A  68000/139264000    68444/140174232

fas960c-svl2*>
```

FILER 2-VOLUME V22

```
fas960c-svl2*> vol status -r v22

Volume v22 (online, raid4) (block checksums)

  Plex /v22/plex0 (online, normal, active)

    RAID group /v22/plex0/rg0 (normal)

    RAID Disk  Device  HA  SHELF  BAY  CHAN  Used (MB/blks)     Phys(MB/blks)
    ----------------------------------------------------------------------------
    parity     3a.45   3a  2      13   FC:A  68000/139264000    68444/140174232
    data       2a.24   2a  1      8    FC:A  68000/139264000    68552/140395088
    data       2a.29   2a  1      13   FC:A  68000/139264000    68552/140395088
    data       2a.26   2a  1      10   FC:A  68000/139264000    68552/140395088
    data       2a.27   2a  1      11   FC:A  68000/139264000    68552/140395088
    data       2a.28   2a  1      12   FC:A  68000/139264000    68552/140395088
```

```
   data         2a.39    2a   2      7       FC:A  68000/139264000    68444/140174232
   data         2a.40    2a   2      8       FC:A  68000/139264000    68444/140174232

   RAID group /v22/plex0/rg1 (normal)

   RAID Disk    Device   HA   SHELF  BAY     CHAN  Used (MB/blks)     Phys(MB/blks)
   -------------------------------------------------------------------------------------------

   parity       2a.41    2a   2      9       FC:A  68000/139264000    68444/140174232
   data         2a.42    2a   2      10      FC:A  68000/139264000    68444/140174232
   data         2a.43    2a   2      11      FC:A  68000/139264000    68444/140174232
   data         2a.44    2a   2      12      FC:A  68000/139264000    68444/140174232
   data         3a.27    3a   1      11      FC:A  68000/139264000    68444/140174232
   data         3a.28    3a   1      12      FC:A  68000/139264000    68444/140174232
   data         3a.29    3a   1      13      FC:A  68000/139264000    68444/140174232
   data         3a.39    3a   2      7       FC:A  68000/139264000    68444/140174232

   RAID group /v22/plex0/rg2 (normal)

   RAID Disk    Device   HA   SHELF  BAY     CHAN  Used (MB/blks)     Phys(MB/blks)
   -------------------------------------------------------------------------------------------

   Parity       3a. 40   3a   2      8       FC:A  68000/139264000    8444/140174232
   data         3a.41    3a   2      9       FC:A  68000/139264000    68444/140174232
   data          3a.42   3a   2      10      FC:A  68000/139264000    68444/140174232
   data         3a.43    3a   2      11      FC:A  68000/139264000    68444/140174232
   data         3a.44    3a   2      12      FC:A  68000/139264000    68444/140174232
   data         2a.23    2a   1      7       FC:A  68000/139264000    68552/140395088

fas960c-svl2*>
```

FILER 2-VOLUME LOG

```
fas960c-svl2*> vol status -r log
Volume log (online, raid4) (block checksums)

  Plex /log/plex0 (online, normal, active)

   RAID group /log/plex0/rg0 (normal)

   RAID Disk    Device   HA   SHELF  BAY     CHAN  Used (MB/blks)     Phys(MB/blks)
   -------------------------------------------------------------------------------------------

   parity       3a.23    3a   1      7       FC:A  68000/13926400068444/140174232

   data         3a.17    3a   1      1       FC:A  68000/13926400068444/140174232

   data         3a.18    3a   1      2       FC:A  68000/139264000    68444/140174232

   data         3a.19    3a   1      3       FC:A  68000/139264000    68444/140174232

   data         3a.20    3a   1      4       FC:A  68000/139264000    68444/140174232

   data         3a.21    3a   1      5       FC:A  68000/139264000    68444/140174232

   data         3a.22    3a   1      6       FC:A  68000/139264000    68444/140174232

   data         3a.16    3a   1      0       FC:A  68000/139264000    68444/140174232

fas960c-svl2*>
```

LUN CREATION

We created the LUNs on the WAFL volumes as follows:

```
fas960c-svl1*> lun create –s 150g –t linux /vol/v11/vol11_lun1
fas960c-svl1*> lun create –s 150g –t linux /vol/v11/vol11_lun2
fas960c-svl1*> lun create –s 150g –t linux /vol/v11/vol11_lun3
fas960c-svl1*> lun create –s 150g –t linux /vol/v11/vol11_lun4

fas960c-svl1*> lun create –s 150g –t linux /vol/v12/vol12_lun1
fas960c-svl1*> lun create –s 150g –t linux /vol/v12/vol12_lun2
fas960c-svl1*> lun create –s 150g –t linux /vol/v12/vol12_lun3
fas960c-svl1*> lun create –s 150g –t linux /vol/v12/vol12_lun4

fas960c-svl2*> lun create –s 150g –t linux /vol/v21/vol21_lun1
fas960c-svl2*> lun create –s 150g –t linux /vol/v21/vol21_lun2
fas960c-svl2*> lun create –s 150g –t linux /vol/v21/vol21_lun3
fas960c-svl2*> lun create –s 150g –t linux /vol/v21/vol21_lun4

fas960c-svl2*> lun create –s 150g –t linux /vol/v22/vol22_lun1
fas960c-svl2*> lun create –s 150g –t linux /vol/v22/vol22_lun2
fas960c-svl2*> lun create –s 150g –t linux /vol/v22/vol22_lun3
fas960c-svl2*> lun create –s 150g –t linux /vol/v22/vol22_lun4

fas960c-svl2*> lun create –s 100g –t linux /vol/log/log_lun1
fas960c-svl2*> lun create –s 100g –t linux /vol/log/log_lun2
```

The LUN details are shown below for both the storage systems:

```
fas960c-svl1*> lun show
    /vol/v11/vol11_lun1   150g   (161061273600)   (r/w, online, mapped)
    /vol/v11/vol11_lun2   150g   (161061273600)   (r/w, online, mapped)
    /vol/v11/vol11_lun3   150g   (161061273600)   (r/w, online, mapped)
    /vol/v11/vol11_lun4   150g   (161061273600)   (r/w, online, mapped)
    /vol/v12/vol12_lun1   150g   (161061273600)   (r/w, online, mapped)
    /vol/v12/vol12_lun2   150g   (161061273600)   (r/w, online, mapped)
    /vol/v12/vol12_lun3   150g   (161061273600)   (r/w, online, mapped)
    /vol/v12/vol12_lun4   150g   (161061273600)   (r/w, online, mapped)
fas960c-svl1*>

fas960c-svl2*> lun show
    /vol/log/log_lun1     100g   (107374182400)   (r/w, online, mapped)
    /vol/log/log_lun2     100g   (107374182400)   (r/w, online, mapped)
    /vol/v21/vol21_lun1   150g   (161061273600)   (r/w, online, mapped)
    /vol/v21/vol21_lun2   150g   (161061273600)   (r/w, online, mapped)
    /vol/v21/vol21_lun3   150g   (161061273600)   (r/w, online, mapped)
    /vol/v21/vol21_lun4   150g   (161061273600)   (r/w, online, mapped)
    /vol/v22/vol22_lun1   150g   (161061273600)   (r/w, online, mapped)
    /vol/v22/vol22_lun2   150g   (161061273600)   (r/w, online, mapped)
    /vol/v22/vol22_lun3   150g   (161061273600)   (r/w, online, mapped)
    /vol/v22/vol22_lun4   150g   (161061273600)   (r/w, online, mapped)
fas960c-svl2*>
```

11 APPENDIX B: ISCSI DRIVER INSTALLATION

The Adaptec 7211C iSCSI driver is available from Adaptec. It must be installed with the Linux kernel (see Reference [3]).

1. Refer to *Installation Guide ASA-7211C/F iSCSI Adapter for Initiator Applications* (see Reference [5]), to learn more about the installation process. The following steps were executed to install the ASA72XX iSCSI module from Adaptec:
2. Install the Linux kernel source on the system before installing the ASA72XX module. Most distributions have the kernel source on the first disc of the distribution CDs.
3. Create a directory asa72XX under `/usr/src`, copy the `.tar.gz` package to

   `/usr/src/asa72xx, and extract the tar file.`
4. Run the configure script:

   ```
   # ./configure
   ```

   Note: If the kernel source is not under `/usr/src/linux`, run the configure script with the following option:

   ```
   # ./configure --tree=<path to source tree>
   ```

   For example:

   ```
   # ./configure --tree=/usr/src/linux
   ```
5. Build the ASA72XX module by typing:

   ```
   # make
   ```
6. Load the module. You may load it automatically or manually, depending on your preference. There is a script file provided on the CD-ROM (S06asa72xx). If you want to start the module automatically, type:

   ```
   # make install
   ```

   ```
   # ./S06asa72xx
   ```

   Note: If you want to start the module manually, type:

   ```
   # insmod asa72xx.o asa72xxc="MachineName:xxxxx"
   ```

   Where *xxxxx* is the system hostname. To display the system hostname, type:

   ```
   # hostname
   ```
7. To check if the ASA72XX module is loaded, type:

   ```
   # lsmod
   ```

   If the ASA72XX module is loaded, `asa72xx` will be displayed in the module table list.
8. Reboot the system.

12 APPENDIX C: INITIATOR MAPPING EXAMPLE

```
# iconfig
Select which iSCSI Adapter to be used:
Adaptec iSCSI Adapter (/dev/asa72xx0)
Adaptec iSCSI Adapter (/dev/asa72xx1)
Select: 1
```

```
ASA72XX Initiator Configuration Utility (iConfig) v1.01.82
```

```
Copyright (c) Adaptec, Inc. 2003 iSCSI v.20

User Name LOGIN: admin
Password: *******
Other Initiator(s) information:
Initiator #2:
      Start ISID = 1 Num. of Sessions = 39
      iSCSI Name : iqn.1992-08.com.adaptec.0-0-d1-26-4-95
Conflict with Initiator #2's Start ISID = 1,
Number of sessions = 39 and your Start ISID input: 1
WARNING:
You have to set the Start ISID to a non overlapping value from
"Main Menu" -> "Setup Properties" -> "Controller Properties" -> "Set Properties"
=================== Set IP Address =======================
Enter <IP Address> (ex: xxx.xxx.xxx.xxx): 192.168.101.201
Current Subnet Mask: 0.0.0.0
(ENTER KEY = 255.255.255.0)
Enter <Subnet Mask> (ex: xxx.xxx.xxx.xxx) :(Enter key)
Current Gateway Address: None
================ Set IP Address ==========================
(ENTER KEY = Set IP Address as Gateway)
Gateway Address> (ex: xxx.xxx.xxx.xxx): (Enter key)
Changes sent to the Controller.
============ Set Initiator iSCSI Name =========================
The current
Initiator iSCSI Name is: iqn.1992-08.com.adaptec.0-0-d1-26-5-95
(ENTER KEY = Keep Current Value)
Enter the iSCSI Name desired: (Enter key)
iSCSI Name unchanged!
======== Set Discovery Parameters ==========================
Do you want to use SLP and/or iSNS to discover targets automatically? (y/n): n
You can also add/edit/delete targets or modify Discovery and Login parameters
```

```
under the Main menu "Targets and Devices"
============== Add Targets =================================
Add target? (y/n): y
Enter Target DNS name (press Enter if not known): (Enter key)
No Target DNS name was entered. An IP address will be required.
Enter <IP Address> (ex: xxx.xxx.xxx.xxx): 192.168.101.103
(ENTER KEY = 3260)
Enter <Port Number> (no less than 1024):
Port Number: 3260
Enter iSCSI Target Name: iSCSI
Add another target? (y/n): n
The Following Data was sent successfully to the controller:
Target # Port# IP Address Target Type Target Name
1 3260 192.168.101.103 configured iSCSI
Target DNS Name:
-----------------
=============== Save settings ================================
Do you want the changes to remain in effect after REBOOT (y/n): y
Saving Settings. Please Wait.....
================== MAIN MENU ==================================
Display Properties
Setup Properties
Device and Targets
Routing, DNS and ARP
Statistics, MIB and Event Log
Firmware and ROM
User Logout and Login Password
IMMEDIATE SAVE SETTINGS
or ESCape Key to QUIT
Select: 9
Do you really want to exit the program? (y/n): y
Do you want to save the settings into a text file? (y/n): n
EXITING iConfig ...
```

USING ICONFIG TO GET ISCSI INITIATOR NAMES

Using the Adaptec iconfig utility we obtain the initiator iSCSI name as follows:

```
# iconfig
Select which iSCSI Adapter to be used :
Adaptec iSCSI Adapter (/dev/asa72xx0)
Adaptec iSCSI Adapter (/dev/asa72xx1)
Select :1
__________________________________________________________
ASA72XX Initiator Configuration Utility (iConfig) v1.01.82
__________________________________________________________
Copyright (c) Adaptec, Inc. 2003 iSCSI v.20
User Name LOGIN :admin
Password : *******
================== MAIN MENU ==================================
Display Properties
Setup Properties
Device and Targets
Routing, DNS and ARP
Statistics, MIB and Event Log
Firmware and ROM
User Logout and Login Password
IMMEDIATE SAVE SETTINGS
or ESCape Key to QUIT
Select:1
================== Display Properties =============================
Controller Properties
Ethernet Properties
Target List
Device Mapping Info
iSCSI Connection Info
Discovery Parameters
Login Parameters
or ESCape Key to QUIT
Select:1
    Machine Name             : adaptec-3.iol-svl.eng.netapp.com
    Controller Model Number  : ASA72XX
    Initiator iSCSI Name     : iqn.1992-08.com.adaptec.0-0-d1-25-fc-af
    Controller Description   : Adaptec iSCSI Host Bus Adapter
    Manufacturer Name        : Adaptec Inc. U.S.A
    Controller Serial Number : 00d125fcaf
    Firmware Version         : v1.01.91
    ROM Version              : 2.08
    Number of Ports          : 01
    Interface Type           : ETHERNET PORT
    IP Version Supported     : IPV4
    PCI Bus Number          : 0003
    PCI Device Number       : 000a
    PCI Function Number     : 0001
    PCI Base Address        : fbe00000
    PCI Interrupt Vector    : 001c
    PCI Vendor ID           : 9005
    PCI Device ID           : 564a
    PCI Subsystem Vendor ID : 9005
    PCI Subsystem Device ID : 7211
    Driver Version          : v1.01.91ntap_asa72xx
    Driver Name             : asa72xx
    Start Session ID  (hex) : 4000031b0001
    Max Sessions  (decimal) : 41     (PRESS ANY KEY TO CONTINUE)
============================== Display Properties ================
    1. Controller Properties
    2. Ethernet Properties
    3. Target List
    4. Device Mapping Info
    5. iSCSI Connection Info
    6. Discovery Parameters
    7. Login Parameters
    8. or ESCape Key to QUIT
    Select:8
```

MAPPING IGROUPS TO LUNS

`iqn.1992-08.com.adaptec.0-0-d1-25-fc-af` is the initiator iSCSI name in the IQN format for a specific Adaptec initiator card on one of the nodes as shown above. In our configuration, we have four nodes with two Adaptec initiator cards per node. Each initiator card on a node is mapped to one storage system target. So we will have eight initiator iSCSI names, of which four will map to the first storage system target and the remaining four will map to the second storage system target. So we create an igroup on each storage system as follows:

```
fas960c-svl1*> igroup create -i -t linux igroup_svl1 iqn.1992-08.com.adaptec.0-0-d1-25-fc-e7
                                                     iqn.1992-08.com.adaptec.0-0-d1-25-fc-c5
                                                     iqn.1992-08.com.adaptec.0-0-d1-25-fc-af
                                                     iqn.1992-08.com.adaptec.0-0-d1-26-4-87
fas960c-svl2*> igroup create -i -t linux igroup_svl2 iqn.1992-08.com.adaptec.0-0-d1-25-fc-8d
                                                     iqn.1992-08.com.adaptec.0-0-d1-26-5-95
                                                     iqn.1992-08.com.adaptec.0-0-d1-26-5-29
                                                     iqn.1992-08.com.adaptec.0-0-d1-26-5-9f
```

To verify the existence of the igroups that we just created on the storage systems, issue the following command on the storage system:

```
fas960c-svl1*> igroup show
   igroup_svl1 (iSCSI) (ostype: linux):
      iqn.1992-08.com.adaptec.0-0-d1-25-fc-e7
      iqn.1992-08.com.adaptec.0-0-d1-25-fc-c5
      iqn.1992-08.com.adaptec.0-0-d1-25-fc-af
      iqn.1992-08.com.adaptec.0-0-d1-26-4-87
```

```
fas960c-svl2*> igroup show
    igroup_svl2 (iSCSI) (ostype: linux):
        iqn.1992-08.com.adaptec.0-0-d1-25-fc-8d
        iqn.1992-08.com.adaptec.0-0-d1-26-5-95
        iqn.1992-08.com.adaptec.0-0-d1-26-5-29
        iqn.1992-08.com.adaptec.0-0-d1-26-5-9f
```

Then, we map the igroup that we just created on each storage system to the LUNs as follows:

```
fas960c-svl1*> lun map /vol/v11/vol11_lun1 igroup_svl1 0
fas960c-svl1*> lun map /vol/v11/vol11_lun2 igroup_svl1 1
fas960c-svl1*> lun map /vol/v11/vol11_lun3 igroup_svl1 2
fas960c-svl1*> lun map /vol/v11/vol11_lun4 igroup_svl1 3
fas960c-svl1*> lun map /vol/v12/vol12_lun1 igroup_svl1 4
fas960c-svl1*> lun map /vol/v12/vol12_lun2 igroup_svl1 5
fas960c-svl1*> lun map /vol/v12/vol12_lun3 igroup_svl1 6
fas960c-svl1*> lun map /vol/v12/vol12_lun4 igroup_svl1 7

fas960c-svl2*> lun map /vol/v21/vol21_lun1 igroup_svl2 0
fas960c-svl2*> lun map /vol/v21/vol21_lun2 igroup_svl2 1
fas960c-svl2*> lun map /vol/v21/vol21_lun3 igroup_svl2 2
fas960c-svl2*> lun map /vol/v21/vol21_lun4 igroup_svl2 3
fas960c-svl2*> lun map /vol/v22/vol22_lun1 igroup_svl2 4
fas960c-svl2*> lun map /vol/v22/vol22_lun2 igroup_svl2 5
fas960c-svl2*> lun map /vol/v22/vol22_lun3 igroup_svl2 6
fas960c-svl2*> lun map /vol/v22/vol22_lun4 igroup_svl2 7

fas960c-svl2*> lun map /vol/log/log_lun1 igroup_svl2 8
fas960c-svl2*> lun map /vol/log/log_lun2 igroup_svl2 9
```

Finally, in order to verify the LUN mappings, we issue the following commands on the storage systems:

```
fas960c-svl1*> lun show -m
LUN path                 Mapped to                 LUN ID
/vol/v11/vol11_lun1      igroup_svl1               0
/vol/v11/vol11_lun2      igroup_svl1               1
/vol/v11/vol11_lun3      igroup_svl1               2
/vol/v11/vol11_lun4      igroup_svl1               3
/vol/v12/vol12_lun1      igroup_svl1               4
/vol/v12/vol12_lun2      igroup_svl1               5
/vol/v12/vol12_lun3      igroup_svl1               6
/vol/v12/vol12_lun4      igroup_svl1               7
fas960c-svl1*>
fas960c-svl2*> lun show -m
LUN path                 Mapped to                 LUN ID
/vol/log/log_lun1       igroup_svl2                8
/vol/log/log_lun2       igroup_svl2                9
/vol/v21/vol21_lun1     igroup_svl2                0
/vol/v21/vol21_lun2     igroup_svl2                1
```

```
/vol/v21/vol21_lun3     igroup_svl2               2
/vol/v21/vol21_lun4     igroup_svl2               3
/vol/v22/vol22_lun1     igroup_svl2               4
/vol/v22/vol22_lun2     igroup_svl2               5
/vol/v22/vol22_lun3     igroup_svl2               6
/vol/v22/vol22_lun4     igroup_svl2               7
fas960c-svl2*>
```

13 REVISION HISTORY

Date	Author	Comments
November 2004	Giovanni Brignolo and Sankar Bose	Original draft
May 2009	Esther Smitha	Updated changes to the readahead setting recommendation.