High Performance, Open Source, Dell Lustre™ Storage System Dell PowerVault™ MD3200 storage platform and QDR Mellanox Infiniband

Dell/Cambridge HPC Solution Centre

Wojciech Turek, Dr Paul Calleja

White paper structure

Abstract

The following paper was produced by the Dell Cambridge HPC Solution Centre and is the second in a series of detailed Lustre™ papers based on operational experience gained from using mass storage technologies within the University of Cambridge production HPC environment. The paper provides a detailed description of how to build a commodity 'Dell Lustre Storage Brick' together with comprehensive performance characteristics obtained from the Lustre storage brick when integrated into a QDR Infiniband network. At the time of going to press the solution described here represents a best-in-class commodity Lustre brick architecture in terms of both performance and availability. The paper also discusses the operational characteristics and system administration best practices derived from long-term large-scale production usage of the solution within the University of Cambridge HPC environment.

Each Dell Lustre Storage Brick is shown to provide 2GB/s read/write I/O bandwidth through the filesystem layer to networked clients, and this performance scales linearly with additional storage bricks. Over QDR Infiniband. each client is able to achieve an I/O bandwidth of 2GB/s, which scales linearly with successive clients until the backend bandwidth is saturated. The Cambridge HPC Service has constructed a three-brick solution that is housed in one rack cabinet and provides 280 TB of usable storage with 6GB/s back end I/O bandwidth. With new 3TB disks instead of the 2TB used here, the single rack capacity would rise to 420 TB. The solution described here has been used in the Cambridge HPC production environment for the last six months serving over 600 nodes and 500 users. The solution has proven to be a good fit for mid-range University HPC workload environments such as this with good application performance, availability and data security, demonstrating less than 0.5% unscheduled downtime in our 24/7 operational service.

In addition to bandwidth tests we present comprehensive IOPS and metadata benchmark results and it can be seen that the Dell Lustre Storage Brick exhibits good IOPS and metadata performance. We see IOPS sequential read/write multi-client maxima of over 500,000 and metadata performance of 50,000 for file stat and directory stat and 15,000 for file and directory remove and create. This is higher than seen with previous Lustre hardware and software builds and shows that the Dell Lustre Storage Brick is a good general-purpose filesystem. Lustre is well known as being optimised for large sequential I/O, typically seen with HPC scratch filesystems but less well known as a good IOPS/metadata solution. Here we demonstrate a Lustre solution with good IOPS and metadata performance suitable for many home filesystem use cases. In fact, the Cambridge/home filesystem is also run off a dedicated 'home' Lustre filesystem, and the performance is well balanced for our 500-user environment.

1.0 Introduction

The continued growth of clustered computing and the large computational power of even modest departmental and workgroup HPC systems has resulted in a storage architecture challenge in which traditional NFS, NAS, SAN and DAS-based storage solutions have not kept pace with the performance growth of a large segment of the HPC install base. This has led to many mid-range HPC users struggling to meet the I/O demands of their applications. It is widely recognised within the research computing community that cost effective, high performance, scalable and resilient centralised storage is the greatest challenge facing modern day HPC facilities.

Over recent years the HPC industry has seen a rise in the use of parallel filesystems such as Lustre™, GPFS™ (General Parallel File System™) and pNFS, as well as proprietary appliance-based systems in an attempt to overcome the HPC storage architecture problem.

This paper is the second in a series of Lustre papers describing in detail how to build and configure a Lustre parallel filesystem storage brick using Dell PowerVault™ systems. The solution uses current Dell MD3200/MD1200 arrays, Lustre 1.8.5 and a QDR Infiniband client network and as such represents an example of a current best-in-class commodity Lustre solution. The paper will also comment on operational characteristics and system administration best practices derived from over three years production usage of a large scale mass storage solution of the same architecture. The main focus of this paper is how to build Lustre on a commodity storage array, with the objective of allowing the reader to replicate the system described. This paper will provide detailed build instructions and comprehensive performance data on leading edge up-to-date commodity storage hardware.

This paper has been produced by the Dell Cambridge HPC Solution Centre, which is part of the wider University of Cambridge HPC Service. The Service runs one of the largest HPC facilities in the UK with a large user base and a high I/O workload. The Solution Centre works closely with Dell customer-facing staff as well as engineering, to provide solution development and a production test environment for Dell in which the operational experience of the Cambridge HPC service is combined with the customer requirement and the understanding, product and implementation knowledge of Dell to produce innovative, robust and fully tested HPC solutions. These solutions are then made openly available to the HPC community via white papers and technical bulletins. The Cambridge HPC Service has been successfully using the Lustre parallel filesystem within a large scale 300 TB central cluster filesystem for over three years. The Dell Lustre storage brick described here is an upgrade to our previous Lustre filesystem, doubling our capacity and trebling our performance. The Dell Lustre storage brick as described here has evolved to fill the operational requirements of a busy, working HPC centre and has been used in production in Cambridge for several years. Its development was critical in order to meet the pressing operational requirements caused by a growing demand for storage volume and the inability of our NFS filesystems to keep up with the I/O workload.

The Lustre filesystem has several key features which make it a good fit within the departmental and workgroup HPC centre as well as in the largest national level HPC centres. In fact Lustre is now used in over 50% of the top 50 supercomputers in the world. The key Lustre features that are responsible for Lustre's high level of usage within the HPC segment are:

- Lustre is a true parallel filesystem, allowing multiple nodes in a cluster to read and write from the same file at the same time. This can greatly increase the filesystem I/O for applications that support parallel I/O.

- Lustre serves files horizontally across any number of storage servers, with data striped across standard low-cost commodity storage arrays, providing the aggregate performance of all storage servers and storage arrays. In this way, Lustre can provide a large and scalable backend storage performance from low-cost hardware.

- The Lustre filesystem I/O can be delivered to cluster client nodes over a wide range of network technologies, gigabit, 10 gigabit or Infiniband. This enables high I/O performance to a single client, which can be advantageous for legacy HPC applications that do not take advantage of parallel I/O.

- The Lustre filesystem can be used to agglomerate many separate disk storage arrays into one single filesystem with a single global name space, which can grow dynamically simply by adding more storage elements – very useful when having to manage a large HPC central storage facility.

- Lustre^{TM} leverages Linux and is itself an open-source software effort. This, combined with its usage of commodity storage array hardware, drastically reduces the cost of Lustre filesystem installations. It also means that there is an active open-source community for future development, and mailing lists for trouble-shooting problems. Also, as there is a range of companies offering professional support of Lustre storage solutions, there is no vendor lock-in with Lustre.

- Lustre has many HA and resiliency features that can be used to configure the fault-tolerant and highly available storage solutions that are essential for use in large-scale HPC environments.

2.0 Lustre Filesystem Architecture

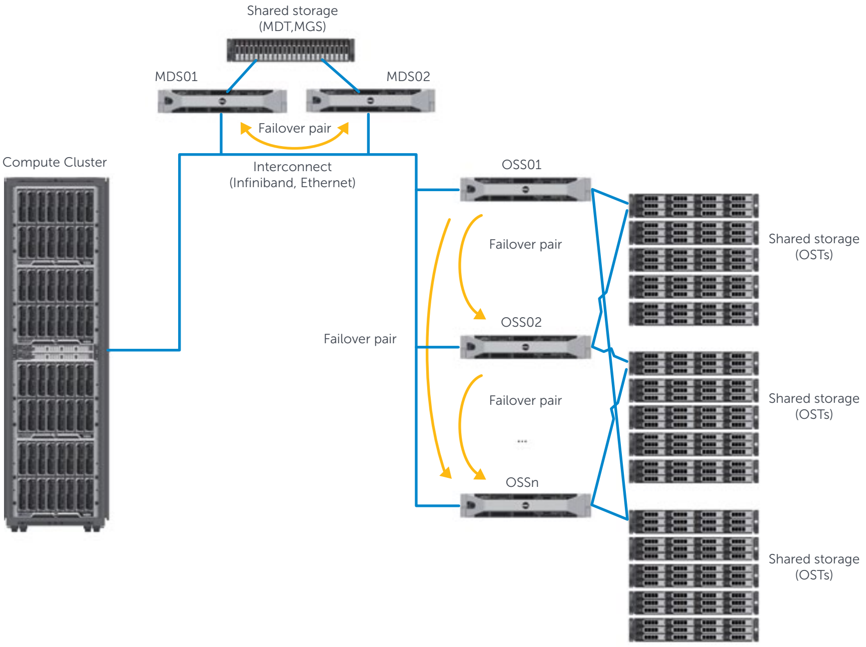
Lustre provides a storage architecture for clusters which allows significant freedom in hardware implementation. At the user level, the Lustre filesystem provides a POSIX-compliant UNIX filesystem interface. The main components of Lustre are the Metadata Server (MDS), Object Storage Server (OSS) and Lustre client. The Lustre filesystem uses an object-based storage model and provides several abstractions designed to improve both performance and scalability. At the filesystem level, Lustre treats files as objects which are located through the MDS. Metadata servers support all filesystem name space operations, such as file lookups, file creation, and file and directory attribute manipulation. This information is physically stored on the metadata target device (MDT). Lustre allows only one MDT per filesystem. File data is stored in objects on the object storage targets (OST), which are managed by OSSs. The MDS directs actual file I/O requests from a Lustre client to the appropriate OST, which manages the objects physically located on the underlying storage block devices. Once the MDS identifies the storage location of a file, all subsequent file I/O is performed between the client and the OSSs.

The Lustre clients are typically HPC cluster compute nodes running Lustre client software and communicating with Lustre servers over Ethernet or Infiniband. The Lustre client software consists of an interface between the Linux virtual filesystem and the Lustre servers. Each server target has a client counterpart: Metadata Client (MDC), Object Storage Client (OSC), and a Management Client (MGC). OSCs are grouped into a single Logical Object Volume (LOV), which is the basis for transparent access to the filesystem. Clients mounting the Lustre filesystem see a single, coherent, synchronised namespace at all times. Different clients can write to different parts of the same file at the same time, while other clients read from the file.

This design divides filesystem operation into two distinct parts: filesystem metadata operations on the MDS and file data operations on the OSSs. This approach not only improves filesystem performance but also improves other important operational aspects such as availability and recovery times.

As shown in Figure 1, the Lustre filesystem is very scalable and can support a variety of hardware platforms and interconnects.

Figure 1. High Availability Lustre^{TM} filesystem high-level architecture



3.0 Dell Lustre Storage System Overview

There are three main design goals of the Dell Lustre Storage System, namely, maximised HPC I/O performance, simplified implementation and configuration and a competitive price point. These goals are accomplished by using a modular approach with standardised commodity hardware components and open source software. This modular approach allows the HPC data centre to start with a number of storage modules that meet their workload requirements and easily grow capacity and throughput as needed by adding more OSS modules. The following design can utilise either Ethernet or Infiniband fast interconnects, making it able to meet most data centre deployment requirements.

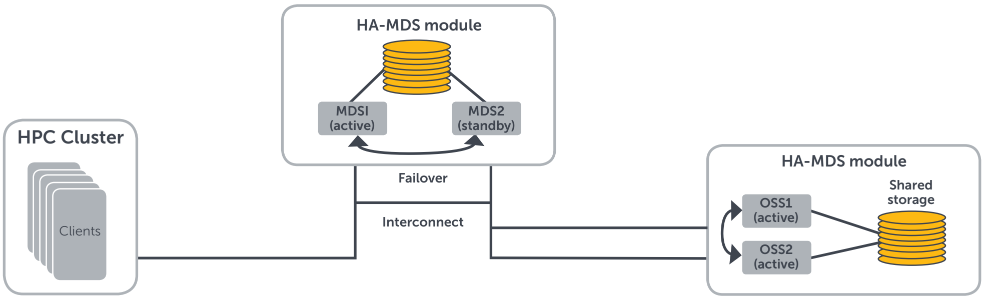
The following modules are used in the Dell Lustre Storage System design:

HA-MDS/MGS – High Availability Metadata Server and Management Server module.

HA-OSS – High Availability Object Storage Server module.

See figure 2 for the logical relationship between clients and the storage modules.

Figure 2. Overview of Dell Lustre Storage System design



4.0 Hardware Components

The Dell PowerEdge™ and Dell PowerVault™ product lines are the foundation of the Dell Lustre™ Storage System. Specifically the PowerEdge R710 server line and PowerVault MD3200 and MD1200 disk arrays are used as the building blocks of the Dell Lustre Storage System modules.

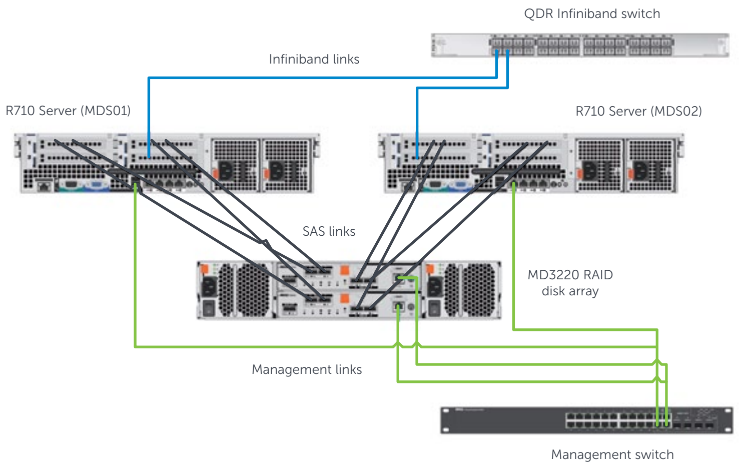
Table 1 below provides an overview of the core building blocks for the Dell Lustre Storage System.

HA-MDS brick	HA-OSS brick
• 3.2 TB of raw disk space. • 1.6 TB of RAID10 usable disk space • QDR InfiniBand • MDS/MGS Servers: Two Dell PowerEdge R710 servers, running CentOS-5.5 Linux and Lustre-1.8.5, equipped with dual socket motherboard, six core Intel CPUs and 32GB of RAM • Storage: One Dell PowerVault MD3220 with 24 136TB 15k SAS drives.	• 108.75 TB of raw space. • 87TB of RAID6 usable disk space • QDR InfiniBand. • OSS servers: Two Dell PowerEdge R710 servers, running CentOS-5.5 Linux and Lustre-1.8.5, equipped with dual socket motherboard, six core Intel CPUs and 24GB of RAM • Storage: One Dell PowerVault MD3200 and five MD1200 enclosures, each with 12 2TB NL SAS drives.

4.1 HA-MDS Module

The HA-MDS module comprises two Dell PowerEdge R710 servers, each equipped with two high-performance Intel processors, 32GB of RAM and six fast, low-latency, enterprise class SAS hard drives managed by an internal H200i RAID controller (figure 3). The internal drives provide fast and reliable disk space for the Linux OS. The Dell PowerVault MD3220 direct-attached external RAID storage array is designed to increase cluster I/O performance and functionality. Equipped with data-protection features to help keep the storage system online and available, the storage array is architected to avoid single points of failure by employing redundant RAID controllers and cache coherency provided by direct mirroring.

Figure 3. Dell Lustre HA-MDS module configuration



The HA-MDS module is designed to provide high IOPS and availability of data. The Metadata Server connects to the disk array via multiple redundant, fast, 6Gbps SAS connections. The system is built to avoid a single point of failure by using two dual port HBA cards, multiple SAS links and dual RAID controllers.

MD3220 storage array key features:

- 24 small factor 2.5inch 15k SAS disk drives (also support for SSDs).
- Mirrored data cache of up to 2048 MB on each RAID controller.
- Persistent cache backup on SD flash disk.
- Snapshots of virtual disks.
- High Performance Tier to optimise IOPS and throughput.
- In-band and out-of-band management using Dell Modular Storage Manager.
- Redundant, hot-swappable components and virtual disk failover.
- Transparent failover for clustered operating systems to maximise data protection.
- RAID-level migration, capacity expansion, consistency checks, and Self-Monitoring, Analysis and Reporting Technology (SMART).

4.1.1 Dell PowerVault^{TM} MD3220 RAID Configuration Overview

One of the biggest factors affecting overall Lustre™ performance is the speed of metadata storage, so the correct choice of hardware and configuration of the RAID disk groups and virtual disks plays an important role in the system configuration. Correctly configured RAID groups minimise the number of expensive read and write operations that are needed to complete a single I/O operation, and also increase the number of operations that can be completed per unit time.

Figure 4. Dell Lustre HA-MDS module RAID configuration

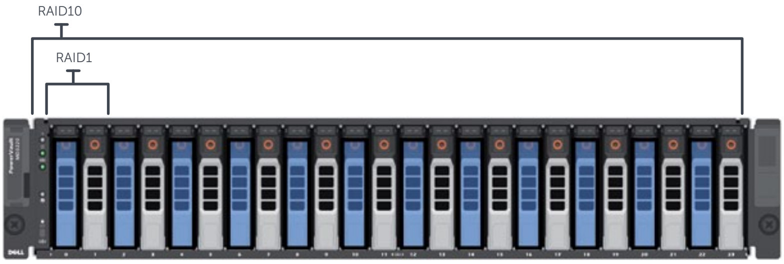


Figure 4 shows the disk layout of the Dell Lustre HA-MDS module. All 24 drives are configured into a RAID10 disk group with a chunk size of 64KB. Since Lustre metadata operations consist of many small I/O transactions, a 64KB chunk size is a good choice and provides satisfactory I/O performance. Another parameter that can make a significant impact on I/O performance of the array is cache block size, the value of which can be set between 4KB and 32KB. A small cache block size is most suitable to small transactional I/O patterns, so for MDS purposes we set it tunable to 4KB.

4.2 HA-OSS Module

The HA-OSS module consists of two Dell PowerEdge™ R710 servers attached to a single PowerVault™ MD3200 disk array, extended with four PowerVault MD1200 disk enclosures (figure 5 and figure 6). The PowerEdge servers are each equipped with two high-performance Intel® multicore processors, 24GB of RAM, five internal hard drives and an internal RAID controller. All disk enclosures are populated with fast 2TB NL SAS disks. The Dell PowerVault MD3200 direct-attached external RAID storage array is designed to increase cluster I/O performance and functionality. Equipped with data-protection features to help keep the storage system online and available, the storage array is designed to avoid single points of failure by employing redundant RAID controllers and cache coherency provided by direct mirroring. The Dell PowerVault disk array provides performance and reliability at the commodity price point required when constructing very large HPC central data storage facilities.

Figure 5. Dell Lustre™ HA-OSS module configuration

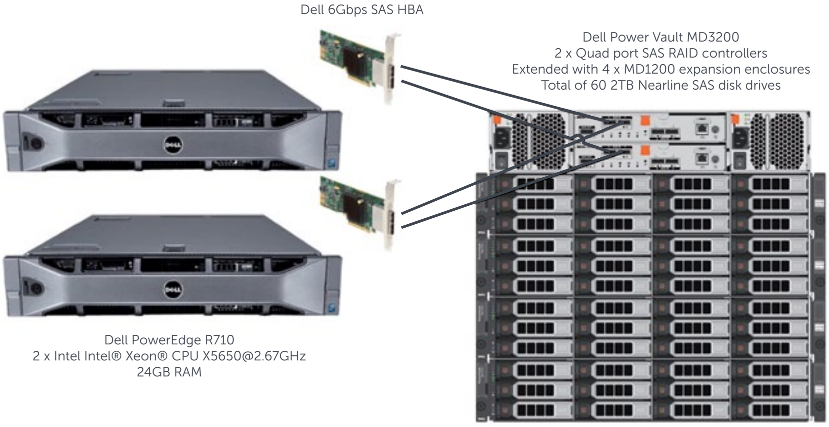
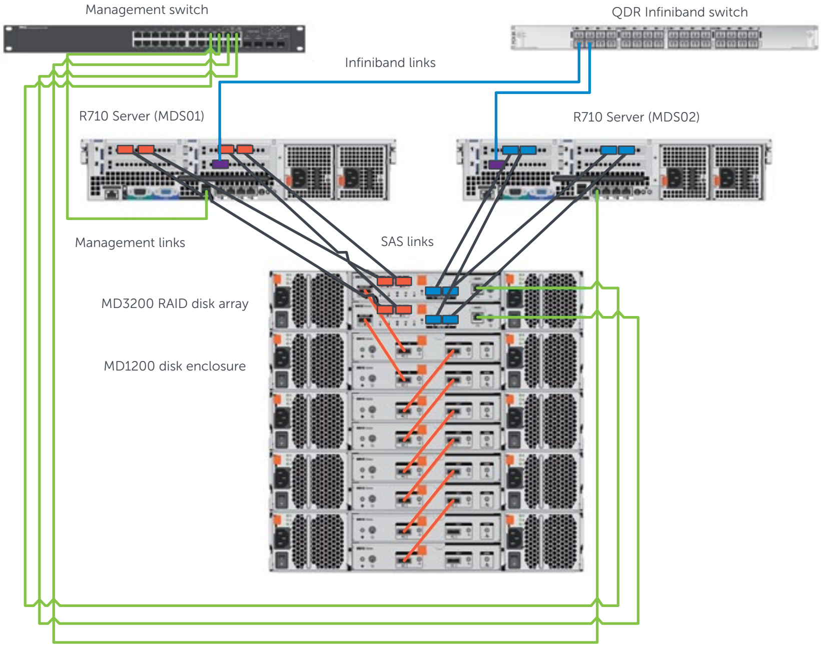

Dell 6Gbps SAS HBA
Dell Power Vault MD3200
2 x Quad port SAS RAID controllers
Extended with 4 x MD1200 expansion enclosures
Total of 60 2TB Nearline SAS disk drives
Dell PowerEdge R710
2 x Intel Intel® Xeon® CPU X5650@2.67GHz
24GB RAM

Figure 6. Dell Lustre HA-OSS module configuration – rear view



The HA-OSS module is configured to provide high availability of data. The Dell PowerEdge^{TM} Object Storage Servers connect to the Dell PowerVault^{TM} MD3200 via dual port SAS HBA cards. To ensure high availability failover, each OSS connects to both external controllers, providing four redundant links to the PowerVault MD3200 disk array per OSS. This design allows for either OSS to drive all the volumes on the Dell PowerVault disk arrays. During normal operation, each OSS exclusively owns a subset of volumes. In the event of failure the remaining server in the failover pair will take over the volumes from the failed server and subsequently service requests to all the volumes.

MD3200 storage array key features:

- 12 x 3.5 inch disk slots.
- Mirrored data cache of up to 2048 MB on each RAID controller.
- Persistent cache backup on SD flash disk.
- Snapshots of virtual disks.
- High Performance Tier to optimise IOPS and throughput.
- In-band and out-of-band management using Dell Modular Storage Manager.
- Redundant, hot-swappable components and virtual disk failover.
- Transparent failover for clustered operating systems to maximize data protection.
- RAID-level migration, capacity expansion, consistency checks, and Self-Monitoring, Analysis and Reporting Technology (SMART).

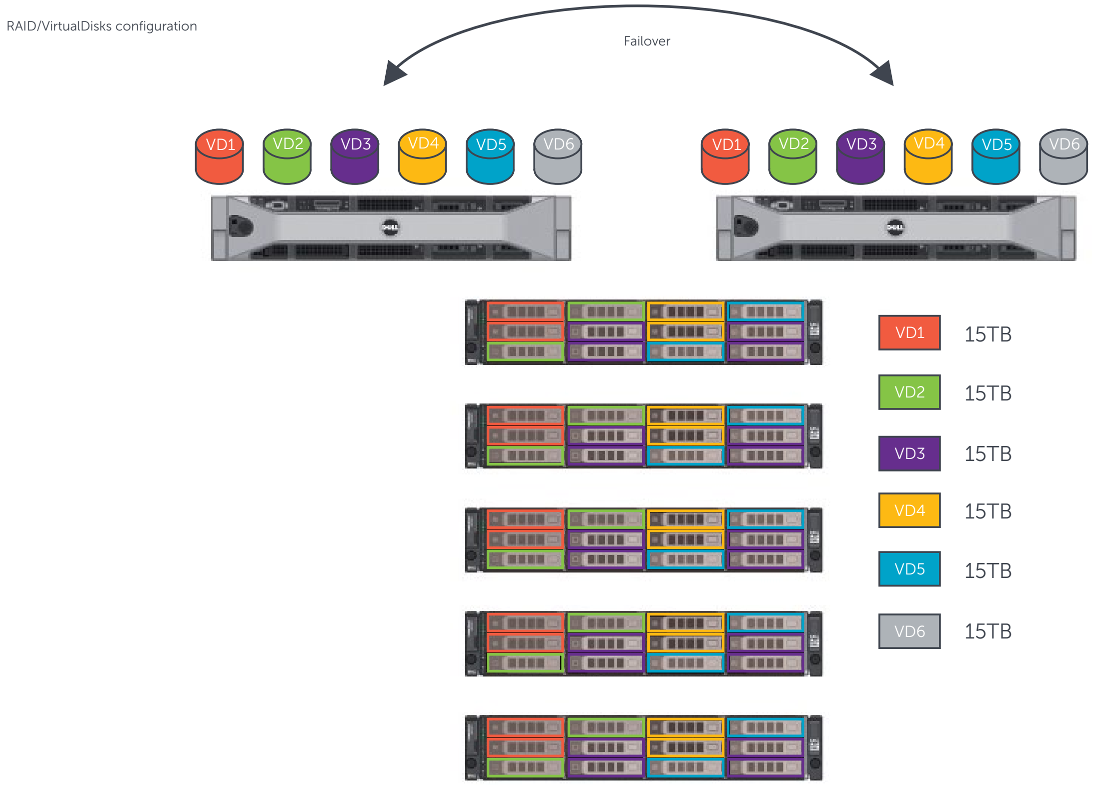
The storage array configuration described here of one MD3200 extended by four MD1200’s was designed to provide a balanced solution in terms of performance, disk capacity and cost. We used 2TB 7.2K NL SAS disks to increase storage capacity and reduce the cost per TB of the total solution.

4.2.1 Dell PowerVault MD3200 Configuration

One of the biggest factors affecting overall Dell PowerVault MD3200 storage array performance is the correct choice and configuration of the RAID disk groups and virtual disks, especially when using RAID5 or RAID6 disk groups. Correctly configured RAID groups minimise the number of expensive read and write operations needed to complete a single I/O operation.

Figure 7 shows the disk configuration of the Dell Lustre^{TM} HA-OSS module. There are six RAID6 disk groups configured. Each RAID6 disk group consists of eight data disks and two parity disks. The segment size (chunk size) of the disk group is 128KB so the stripe size is 1024KB. Such a large segment size minimises the number of disks that need to be accessed to satisfy a single I/O operation. The Lustre typical I/O size is 1MB, therefore each I/O operation requires only one RAID stripe. Disks in each disk group are spanned across all five disk arrays, which leads to faster disk access per I/O operation. For optimal performance only one virtual disk per disk group is configured, which results in more sequential access of the RAID disk group and which may significantly improve I/O performance. The Lustre manual suggests using the full device instead of a partition (sdb vs sdb1). When using the full device, Lustre writes well-aligned 1 MB chunks to disk. Partitioning the disk removes this alignment and will noticeably impact performance. In this configuration, a RAID disk group distributed across all enclosures has the additional advantage of not having more than two disks per disk enclosure, meaning that in the event of enclosure outage, data integrity is still protected by RAID6 redundancy.

Figure 7. RAID groups and virtual disk layout of the Dell Lustre™ HA-OSS module



The Dell MD3200 controller configuration requires some adjustment in order to optimise performance of the Lustre filesystem. After performing a number of tests exercising various controller parameters the following choice of configuration parameters seems to be best suited to the Lustre filesystem where the requirement is high performance under large I/O workload:

```
Summary:
   cacheBlockSize=32
   segmentSize=128
   readCacheEnabled=true
   cacheReadPrefetch=true
   modificationPriority=lowest
```

The Linux® OS can also improve performance by means of various system parameters. The default setting of the read-ahead algorithm in RedHat® Enterprise Linux® is 128KB per read. The default value is too small for sequential I/O and increasing this value to 8192 KB significantly improves read performance. In addition, the Lustre™ 1.8 OSS is equipped with a read cache feature, which also has a positive impact on storage system read performance. Some extra write performance boost can be gained by disabling the write cache mirroring feature. Making this change however needs to be carefully considered because it decreases data integrity safety, so it is only recommended for non-sensitive scratch data filesystems.

```
Summary of OSS tuning options:

Linux I/O scheduler – parameter set to “deadline”
echo deadline > /sys/block/<blk_dev>/queue/scheduler

max_sectors_kb – parameter set to 2048
echo 2048 > /sys/block/<blk_dev>/queue/max_sectors_kb

read_ahead_kb – parameter set to 2048
echo 2048 > /sys/block/<blk_dev>/queue/read_ahead_kb
```

5.0 Software Components Overview

The Dell Lustre Storage System software stack installed on the Metadata Servers and Object Storage Servers includes the following software components:

- Linux operating system.
- Lustre filesystem.
- Heartbeat failover.
- Dell configuration and monitoring tools.

The Dell Lustre Storage system uses the CentOS-5.5 Linux operating system distribution. This is a rebuild of the RedHat® Enterprise Linux® distribution (RHEL5.5) This Linux distribution is an Enterprise-level operating system which provides the industry’s most comprehensive suite of Linux solutions. RHEL5.5 delivers a high performance platform with most recent Linux features and the highest levels of reliability. The Dell Lustre Storage system uses Lustre version 1.8.5, which is the latest stable version of Lustre, incorporating several key features when compared to older versions of Lustre.

Key features of Lustre 1.8.5:

- **Adaptive timeout**s – offers simplified management for small and large clusters, automatically adjusts RPC timeouts as network conditions and server loads change and reduces server recovery time.

- **OSS read cache** – provides read-only caching of data on the OSS side, allows OSTs to cache read data more frequently and improves repetitive reads to match network speeds instead of disk speeds.

- **OST pools** – allows the system administrator to create a named group of OSTs for file striping purposes, easier disk policy implementation and ease of disk management.

- **Version-based recovery (VBR)** – improves the speed of client recovery operations and allows Lustre™ to recover even if multiple clients fail at the same time as the server. VBR permits clients not to be evicted if some miss recovery and unrecovered clients may try to recover after the server recovery window closes.

The Dell Lustre Storage system offers a failover solution based on ClusterLabs Heartbeat/Pacemaker software. ClusterLabs Heartbeat software is responsible for detecting failure of the primary Lustre servers and controlling the resource transition to secondary nodes. The Dell Lustre Storage high availability modules use the Heartbeat 'version 1' Cluster Resource Manager, which provides reliable failover in a two node cluster. To ensure full data protection by forcing the server to disconnect from the shared storage, Heartbeat software is able to completely power-off the failed server. The process of automatic power control and management is called STONITH (Shoot The Other Node In The Head). Heartbeat-stonith is configured to control and manage power via the server's IPMI interface.

6.0 System Software Configuration

This section describes the steps needed to install the required software packages and configure the Dell Lustre™ Storage system.

6.1 Dell Storage Configuration Prerequisites

The Dell PowerVault™ MD3200 physically connects both its controllers to each OSS node via a dual port 6Gbps SAS HBA. LUNs active on one controller are passive on the other controller. Until all the required software is properly installed, Linux operating system errors such as the following may occur in the system log while booting:

```
Buffer I/O error on device sdc, logical block 1
end_request: I/O error, dev sdc, sector 8
```

Those errors are caused by the Linux SCSI layer trying to access LUNs via passive paths. It is necessary to install and load a special SCSI device handler module, which enables the Linux SCSI layer to handle passive paths. It is safe to ignore such errors until installation of all software has been completed. It is recommended to disconnect the MD3200 from the server prior to Linux® OS and RDAC SCSI handler driver installation to avoid the lengthy boot time caused by the above messages flooding the system logs.

All necessary Linux host software is available on the Dell PowerVault MD32xx resource CD image. The latest resource CD image can be downloaded from the Dell Support website at support.dell.com

The downloaded image can be mounted as a loop device. It is recommended to place the mount point on – or copy the contents of the MD32xx resource CD to – a shared NFS location in order to make it available to all OSS servers.

```
[root]# mkdir /mnt/md32xxiso
[root]# mount -o loop md32xx_resource_cd.iso /mnt/md32xxiso
```

The MD32xx installation process involves the following software modules:

- DKMS - some software modules are provided in a form of a Dynamic Kernel Module Support (DKMS) package. It is necessary to install the dkms rpm prior to installation of other software modules.
- SCSI RDAC device handler - since each OSS server has multiple paths to the same Virtual Disk (VD) via both controllers it is necessary to install a special device handler driver that enables Linux to detect and handle active and passive paths.
- MPT2 SAS Driver - provides latest driver for the 6Gbps SAS HBA.
- Device Mapper Multipath - a special driver that consolidates all paths to a VD into a single pseudo block device. If path failure occurs the DMM driver automatically switches paths, and the OSS server continues to access data through the same pseudo block device but via the alternate path.
- Dell Configuration and Management - the MD32xx resource CD provides the Modular Disk Storage Manager and Command Line Interface. This software tool provides a means of accessing the MD32xx storage for configuration and monitoring purposes.

DKMS installation:

```
[root]# cd /mnt/md32xxiso/linux/dkms
[root]# rpm -ivh dkms-{version}.rpm
```

SAS HBA driver installation:

```
[root]# cd /mnt/md3000iso/linux/RPMS/rhl5
[root]# rpm -ivh mpt2sas-{version}.rpm
```

SCSI RDAC device handler driver installation:

```
[root]# cd /mnt/md32xxiso/linux/linux/dm/rhel55
[root]# rpm -ivh scsi_dh_rdac-{version}.rpm
```

If the installation process completes without error, a reboot of the server is necessary to ensure that the new initial ramdisk with the updated modules is loaded. Additional installation and troubleshooting instructions can be found in the README file located on the Dell PowerVault™ MD32xx resource CD:

When installing the PowerVault Modular Disk Storage Manager and Host Utilities the System administrator has the option of using an interactive installer, which requires the X Windows System and Java™, or using text mode, which can be performed either interactively or non-interactively in a Linux terminal. The second method is a good choice for the cluster environment where software has to be installed on many nodes and allows automation of the installation process for all available servers.

To install the Modular Disk Storage Manager (MDSM) in silent mode, use one of the provided install scripts located in the same directory as the MDSM installer. Depending on the role of the server in the Dell Lustre^{TM} Storage System, use one of the three provided scripts. Use mgmt.properties on a server which is a dedicated management node. Use host.properties on a server which is either an OSS or MDS (and is directly connected to a MD32xx enclosure).

OSS/MDS servers – host.properties file contents:

```
INSTALLER_UI=silent
CHOSEN_INSTALL_FEATURE_LIST=SMruntime,SMutil,SMagent
AUTO_START_CHOICE=0
USER_REQUESTED_RESTART=NO
REQUESTED_FO_DRIVER=mpio
```

Management servers – mgmt.properties file contents:

```
INSTALLER_UI=silent
CHOSEN_INSTALL_FEATURE_LIST=SMclient,Java Ac
AUTO_START_CHOICE=0
USER_REQUESTED_RESTART=NO
```

Once the appropriate properties file is chosen, the software can be installed with the following commands:

```
[root]# cd /mnt/md32xxiso/linux/mdsm/
[root]# ./SMIA-LINUX-{version}.bin -i silent -f host.properties
```

The above process can be streamlined to use a single shell script which will run on the server to install all necessary software - an example script is shown in appendix A.

6.2 Dell PowerVault^{TM} MD3200 Disk Array Initial Configuration

Before attempting MD3200 array configuration, care should be taken to ensure that all storage arrays are correctly connected (check Figure 6 for details) and the necessary drivers and software tools described in the previous section have been successfully installed.

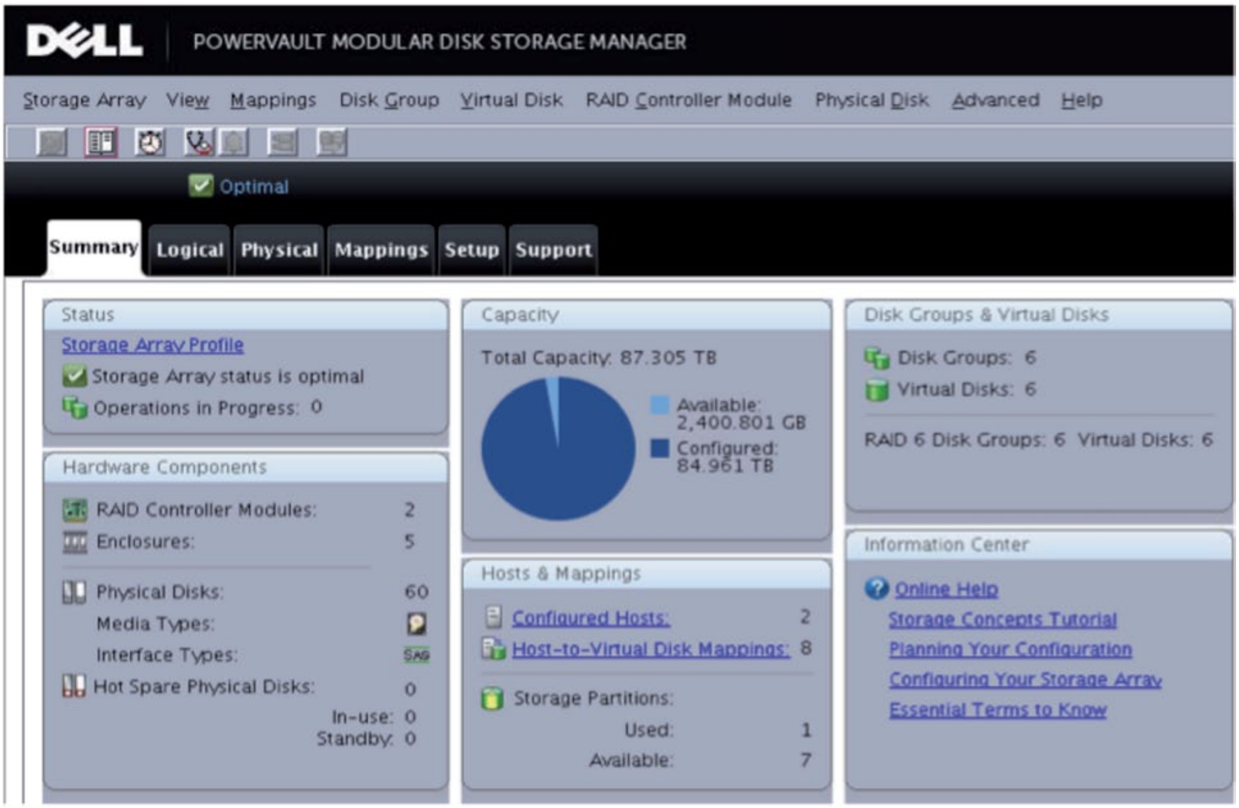
Storage arrays can be configured using either a command line interface (CLI) or modular disk storage manager graphical interface (MDSM GUI). For example, in order to view details about a storage array the following command line can be used:

```
SMcli -n scratch_brick -c “show storageArray;”
```

The Dell website provides detailed guidance on using the CLI, see
http://support.dell.com/support/edocs/systems/md3200/en/CLI/CLI.pdf.

The MDSM GUI can be accessed by issuing the SMclient command on the management node. Below is a screenshot of a main MDSM GUI window, which provides summary information about the storage array. The GUI interface is an easy way to access and control all aspects of the storage array. The Dell website provides a detailed manual describing usage of the MDSM GUI, see
http://support.dell.com/support/edocs/systems/md3000/en/1stgen/UG/PDF/UGA00MR.pdf.

Figure 8 – Module Disk Storage Manager GUI



Status
Storage Array Profile
Storage Array status is optimal
Operations in Progress: 0

Hardware Components
RAID Controller Modules: 2
Enclosures: 5
Physical Disks: 60
Media Types:
Interface Types: SAS
Hot Spare Physical Disks: 0
In-use: 0
Standby: 0

Capacity
Total Capacity: 87.305 TB
Available: 2,400.801 GB
Configured: 84.961 TB

Hosts & Mappings
Configured Hosts: 2
Host-to-Virtual Disk Mappings: 8
Storage Partitions:
Used: 1
Available: 7

Disk Groups & Virtual Disks
Disk Groups: 6
Virtual Disks: 6
RAID 6 Disk Groups: 6 Virtual Disks: 6

Information Center
Online Help
Storage Concepts Tutorial
Planning Your Configuration
Configuring Your Storage Array
Essential Terms to Know

On the OSS servers the SMagent service should be started prior to attempting any configuration. The SMagent service can be configured in the Linux OS to start automatically at boot time:

```
[root]# chkoconfig SMagent on
[root]# service SMagent start
```

To start the configuration process, run the MDSM application installed on the management node:

```
/opt/dell/mdstoragemanager/client/SMclient
```

Perform the initial setup tasks as follows:

```
In MDSM main window click Tools at the top left corner of the application
window, and then click Automatic Discovery… and allow MDSM to scan for
MD3200 storage arrays.
```

Once the scan is complete, all connected MD3200 storage arrays should be visible. It is recommended to start by upgrading the storage array components' firmware. Detailed instructions can be found in the User's Guide on the Dell PowerVault™ MD3200 documentation site:

```
http://support.dell.com/support/edocs/systems/md3200/
```

The next step of the configuration process is to enable OSS Linux server host access to the MD3200 storage array. Host access can be configured either automatically or manually. In the automatic method all hosts running the SMagent service will be automatically detected by the MDSM.

Follow these steps to configure host access automatically:

```
In MDSM main window

1. Right Click on the Storage Array Name and choose Manage Storage Array
option and a new window will open

2. In the new window click on Storage Array -> Configuration -> Automatic…

3. Follow instruction to complete the task
```

Configure storage array name:

```
[root]# SMcli oss01 -c “set storageArray userLabel=\”dell01\”;”
```

Example configuration of storage array host access and host group:

```
SMcli oss01 -c “create hostGroup userLabel=\”dell01\”;”

SMcli oss01 -c “create host userLabel=\”oss01\” hostType=1

              hostGroup=\”dell01\”;”

SMcli oss01 -c ”create hostPort host=\”oss01\” userLabel=\”oss010\”

              identifier=\”{SASportAddress}\” interfaceType=SAS;”
```

The most efficient way to configure is via a configuration script:

```
[root]# SMcli oss01 -f storageArray.cfg
```

A more detailed script for configuring a Dell MD3200 HA-OSS module can be found in Appendix B.

7.0 Lustre^{TM} Installation and Configuration

This section describes the steps needed to install the software for and configure the Lustre filesystem.

7.1 Installing Lustre RPMs

The easiest way to install Lustre is to obtain Lustre RPM packages from the Lustre maintainer website, currently http://wiki.lustre.org/index.php/Download.

In order to determine which Lustre packages should be downloaded, please check the Lustre support matrix:

http://wiki.lustre.org/index.php/Lustre_Release_Information#Lustre_Support_Matrix

Please note that if OSTs bigger than 8TB will be used, then the lustre-ldiskfs-ext4 package must be installed.

The Lustre servers require the following RPMs to be installed:

- kernel-lustre: Lustre patched kernel
- lustre-modules: Lustre kernel modules
- lustre: Lustre utilities package
- lustre-ldiskfs-ext4: Lustre-patched backing filesystem kernel module
- e2fsprogs: utilities package for the ext4 backing filesystem
- lustre-test: various test utilities

The Lustre client requires these RPMS to be installed:

- lustre-modules: Lustre kernel modules
- lustre: Lustre utilities package
- lustre-test: various test utilities

Additionally, the following packages may be required from your Linux distribution, or Infinband vendor:

- kernel-ib: OFED

Lustre installation is completed by installing the above RPMs and rebooting the server.

7.2 Lustre network configuration

In order for the Lustre network to operate optimally, the system administrator must verify that the cluster network (e.g. Infiniband) has been set up correctly and that clients and servers can communicate with each other using standard network protocols for the particular network type used. An Infiniband network can be tested using tools provided in the perftest package. Also a ping utility can be used to check that the TCP/IP configuration of the IB interfaces functions properly.

In a cluster with a Lustre filesystem, servers and clients communicate with each other using the Lustre networking API (LNET). LNET supports a variety of network transports through Lustre Network Drivers (LNDs). For example, ksocklnd enables support for Ethernet and ko2iblnd enables support for Infiniband.

End points of the Lustre^{TM} network are defined by Lustre network identifiers (NID). Every node that uses the Lustre filesystem must have a NID. Typically a NID looks like address@network where address is the name or IP address of the host and network is the Lustre network, e.g. tcp0 or o2ib.

When configuring a Lustre Infiniband network in RHEL5 the system administrator needs to add the following lines to the file /etc/modprobe.conf:

```
# lnet configuration for Infiniband
options lnet networks=o2ib(ib0)
```

This will configure the appropriate NID and load the correct LND for the Infiniband interconnect. More information about Lustre networking can be found at http://wiki.lustre.org/manual/LustreManual18_HTML/UnderstandingLustreNetworking.html.

7.3 Lustre Configuration for HA-MDS Module

Before Lustre can be used, a system administrator must format Lustre targets with the ldiskfs filesystem. During the format procedure some Lustre configuration options can be set. Lustre configuration can be changed later if required using the nondestructive command tunefs.lustre.

In order to configure and start the Lustre management server, run the following commands on one of the failover pair, mds01 and mds02:

```
[root]# mkfs.lustre --mgs —-fsname=testfs          /
                  --failnode=mds02_ib@o2ib    /
                  /dev/mpath/mgs
# Start Lustre management service
[root]# mount -t lustre /dev/mpath/mgs /lustre/mgs
```

Similarly, to configure and start the Lustre metadata server (note that typically the MGS and MDS run on the same host):

```
[root]# mkfs.lustre --reformat --mdt –fsname=testfs       /
  --mkfsoptions=’-O flex_bg -G 256 -E stride=16,stripe-width=64’  /
  --failnode= mds02_ib@o2ib                               /
  --mgsnode= mds01_ib@o2ib                          /
  --mgsnode= mds02_ib@o2ib                          /
  /dev/mpath/testfs_mdt00
# Start metadata service
[root]# mount -t lustre /dev/mpath/testfs_mdt00 /lustre/testfs_mdt00
```

7.4 Lustre^{TM} Configuration for HA-OSS Module

To configure Lustre Object Storage Servers, run the following commands on one server of the OSS failover pair:

```
# on oss01 server
[root]# mkfs.lustre --ost -–fsname=testfs                 /
  --mkfsoptions='-O flex_bg -G 256 –E                    /
stride=32,stripe-width=256'
  --failnode=oss02_ib@o2ib        /
  --mgsnode=mds01_ib@o2ib   /
  --mgsnode=mds02_ib@o2ib   /
 /dev/mpath/testfs_ost00
```

```
[root]# mkfs.lustre --ost -–fsname=testfs                 /
  --mkfsoptions='-O flex_bg -G 256 –E                    /
stride=32,stripe-width=256'
  --failnode=oss02_ib@o2ib        /
  --mgsnode=mds01_ib@o2ib   /
  --mgsnode=mds02_ib@o2ib   /
 /dev/mpath/testfs_ost01
```

```
[root]# mkfs.lustre --ost -–fsname=testfs                 /
  --mkfsoptions='-O flex_bg -G 256 –E                    /
stride=32,stripe-width=256'
  --failnode=oss02_ib@o2ib        /
  --mgsnode=mds01_ib@o2ib   /
  --mgsnode=mds02_ib@o2ib   /
 /dev/mpath/testfs_ost02
```

```
[root]# mkfs.lustre --ost -–fsname=testfs                 /
  --mkfsoptions='-O flex_bg -G 256 –E                   /
stride=32,stripe-width=256'
  --failnode=oss02_ib@o2ib        /
  --mgsnode=mds01_ib@o2ib   /
  --mgsnode=mds02_ib@o2ib   /
 /dev/mpath/testfs_ost03
```

```
[root]# mkfs.lustre --ost -–fsname=testfs                 /
  --mkfsoptions='-O flex_bg -G 256 –E                   /
stride=32,stripe-width=256'
  --failnode=oss02_ib@o2ib        /
  --mgsnode=mds01_ib@o2ib   /
  --mgsnode=mds02_ib@o2ib   /
 /dev/mpath/testfs_ost04
```

```
[root]# mkfs.lustre --ost -–fsname=testfs                 /
  --mkfsoptions='-O flex_bg -G 256 –E                   /
stride=32,stripe-width=256'
  --failnode=oss02_ib@o2ib        /
  --mgsnode=mds01_ib@o2ib   /
  --mgsnode=mds02_ib@o2ib   /
 /dev/mpath/testfs_ost05
```

Start object storage services:

```
[root]# mount -t lustre /dev/mpath/testfs_ost00 /lustre/testfs_ost00
#repeat above mount command for remaining OSTs, mount 6 OSTs per OSS
```

Start Lustre^{TM} on the client node

```
# Mount clients
[root]# mount -t lustre mds01@o2ib;mds02@02ib:/lustre /lustre
```

8.0 Lustre^{TM} Administration Best Practices

Lustre was designed with large sequential I/O in mind and its early usage was confined to scratch filesystems at large datacentres. Nowadays, Lustre is considered not only one of the best cluster filesystems, but it finds its place in more general purpose applications. Lustre comes with a good range of features and tools and learning how to use them effectively can help to run Lustre-based filesystems in a smooth and efficient way.

8.1 Small Files Lustre Performance

Although in general Lustre is designed to work with large files, it is possible to tune Lustre client parameters to improve small file performance. It is especially useful to perform this tuning on nodes that are interactively accessed by users, for example login nodes:

- **Disable LNET debug on client nodes:** By default Lustre records many types of debug messages. Disabling them may increase client performance.

```
cat /proc/sys/lnet/debug
ioctl neterror warning error emerg ha config console
# disable debug messages
sysctl -w lnet.debug=0
```

- **Increase dirty cache on client nodes:** By default Lustre will have 32Mbytes of dirty cache per OST.

```
lctl get_param osc.*.max_dirty_mb
osc.scratch2-OST0000-osc-fff8102266f1000.max_dirty_mb=32
...
osc.scratch2-OST0003-osc-fff8102266f1000.max_dirty_mb=32
# increase dirt cache per OST
lctl set_param osc.*.max_dirty_mb=256
```

- **Increase number of RPCs in flight:** By default Lustre have 8 RPCs in flight. Increasing this number up to 32 can improve performance of both normal data and metadata.

```
lctl get_param osc.*.max_rpcs_in_flight
osc.scratch2-OST0000-osc-fff8102266f1000.max_rpcs_in_flight=8
...
osc.scratch2-OST0003-osc-fff8102266f1000.max_rpcs_in_flight=8
# increase number of RPCs in flight
lctl set_param osc.*. max_rpcs_in_flight=32
```

- **Lustre striping:** Lustre performance degrades if small files are striped on many OSTs. It is good practice to set the stripe count for /home directories to 0 or 1

```
lfs setstripe -c 0 /home/user
```

8.2 Managing OST Free Space

Lustre™ provides many tools and methods for managing free space on the OSTs. It is important to monitor and keep OSTs' free space in balance to stop OSTs becoming full. Making use of OST pools enables administrators to create default file allocation policies which can improve file access and I/O performance.

- **Working with OST pools:** The OST pool feature enables administrators to group OSTs together to make object placement more flexible. A "pool" is a name associated with an arbitrary subset of OSTs. Use the lctl command to create/destroy a pool, add/remove OSTs in a pool, list pools and OSTs in a specific pool. The lctl command MUST be run on the MGS node. Pools can be used to group OSTs with the same technology or performance (slower or faster) or those that are preferred for certain jobs. Examples are SATA OSTs versus SAS OSTs

```
# create OST poll <poolname>
lctl pool_new <fsname>.<poolname>

# add even number of OSTs to newly created pool
lctl pool_add <fsname>.<pool name> OST[0-10/2]

# list pools in the named filesystem
lctl pool_list <fsname>

# list OSTs in the named pool
lctl pool_list <fsname>.<poolname>

# remove named OSTs from the pool
lctl pool_remove <fsname>.<poolname> <ost_list>

# destroy pool
lctl pool_destroy <fsname>.<poolname>
```

- **Handling Full OSTs:** If an OST becomes full and an attempt is made to write more information to the filesystem, an error occurs. The procedures below describe how to deal with a full OST.

```
lfs df

UUID                       1K-blocks       Used              Available
Use % Mounted on

scratch2-MDT0000_UUID
2% /scratch2[MDT:0]        237804280       6958248           217255916

scratch2-OST0000_UUID
99% /scratch2[OST:0]       7687338532      7405477732        281860800

scratch2-OST0001_UUID
84% /scratch2[OST:1]       7687338532      6499283628        797560416

scratch2-OST0002_UUID
84% /scratch2[OST:2]       7687338532      6477962704        818872512

scratch2-OST0003_UUID
84% /scratch2[OST:3]       7687338532      6483592540        813251428
```

OST:0 is almost full and attempts to write to this OST may fail as follows:

```
writing ‘/scratch2/user/test’ : No space left on device
```

To enable continued use of the filesystem, the full OST has to be deactivated using the lctl command. This is done on the MDS, since the MSD allocates space for writing.

```
[root@mds01 ~]# lctl dl | grep OST0000
 5 UP osc scratch2-OST0000-osc scratch2-mdtlov_UUID 5
[root@mds01 ~]# lctl --device 5 deactivate
# Check that OST is INactive
[root@mds01 ~]# lctl dl | grep OST0000
 5 IN osc scratch2-OST0000-osc scratch2-mdtlov_UUID 5
```

In order to free up some space on the full OST some data can be manually migrated off that OST. This can be accomplished by copying data located on the deactivated OST to a new location. New files will be created using only active OSTs. Then original data can be deleted and copies of the data can be moved to their original locations.

```
[root@client01 ~]# cp /scratch2/user/file /scratch2/user/file.tmp
[root@client01 ~]# rm scratch2/user/file
[root@client01 ~]# mv /scratch2/user/file.tmp scratch2/user/file
Use the lfs find command to find files located on the particular OST.
lsf find --obd scratch2-OST0000_UUID /scratch2/user/somefiles > file.list
```

The Lustre^{TM} manual contains a script which helps to automate the migration task.
Also Lustre BUG 22481 contains the latest version of that script.

8.3 Recreating Lustre Configuration Logs

If the Lustre configuration logs are in a state where the filesystem cannot be started, the *writeconf* command can be used to erase them. After the *writeconf* command is run and the servers restarted, the configuration logs are regenerated and saved on the MGS.

The *writeconf* command should only be used if configuration logs were corrupted and the filesystem cannot start, or a server NID has changed. Running the *writeconf* command will erase pool information and conf_param settings so please make sure that those parameters are recorded in the recovery_script that can be run after *writeconf* execution.

Before running *writeconf* the Lustre filesystem should be stopped on all clients and servers.
The *writeconf* command must be run on the MDT first and then on all OSTs.

```
[root@mds01 ~]# tunefs.lustre --writeconf <mdt_device>
[root@oss01 ~]# tunefs.lustre --writeconf <ost_device>
...
[root@oss10 ~]# tunefs.lustre --writeconf <ost_device>
Restart the filesystem in this order:
mount MGS
mount OSTs
mount MDT
Mount clients
```

8.4 Recovering from Errors or Corruption on a Backing ldiskfs Filesystem

When an OSS, MDS, or MGS server crash occurs, it is not generally necessary to run e2fsck on the filesystem, unless the journal is corrupted. Ext4 journaling usually ensures that the filesystem remains consistent. Also the backing filesystems are never accessed directly from the client, so client crashes are not relevant. Sometimes however, an event may occur which the ext4 journal is unable to handle. This can be caused by hardware failure or I/O error. If ext4 detects corruption, the filesystem is remounted read-only, and this is reported in the syslog as error -30. In such a case e2fsck must be run on the bad device. Once e2fsck fixes the problems, the device can be put back into service. If a serious problem is suspected, it is recommended that first e2fsck is run with the -n switch. Also, it is useful to record the output of the e2fsck run to a file if the information might be needed later:

```
e2fsck -fn /dev/<device> #don’t fix anything, just check for corruption
e2fsck -fp /dev/<device> #fix filesystem using prudent answers
```

8.5 Recovering from Corruption in the Lustre Filesystem

In a situation where the MDS or an OST becomes corrupt, the lfsck distributed filesystem checker can be run to determine how serious the corruption is and what sorts of problem exist. This process consists of many steps and may take a very long time to complete.

- run e2fsck -f to fix any local backing filesystem errors.
- build the mdsdb database – it is quicker to write the database file to a local filesystem. Depending on the number of files, this step can take several hours to complete:

```
e2fsck -n -v --mdsdb /tmp/mdsdb /dev/{mdsdev}
```

- Make this file accessible on all OSSs and use it to generate a OST database file per OST:

```
e2fsck -n -v --mdsdb /tmp/mdsdb --ostdb /tmp/{ostNdb} /dev/{ostNdev}
```

- Make the mdsdb file and all ostdb files available on a mounted client and run lfsck to examine the filesystem. Optionally, correct the defects found by lfsck:

```
lfsck -n -v --mdsdb /tmp/mdsdb --ostdb /tmp/{ost1db} /tmp/{ost2db} ... /
lustre/mount/point
```

9.0 Heartbeat

Lustre™ does not offer a complete failover solution. It must be combined with high-availability (HA) software to enable full failover support. Linux-HA Heartbeat software provides all necessary functionality to provide a complete Lustre failover solution. HA-Linux Heartbeat is very portable and runs on many Linux platforms. It requires the following packages to be installed:

- Heartbeat: the Heartbeat subsystem for HA-Linux.
- Heartbeat-stonith: provides an interface to Shoot The Other Node In The Head (STONITH).
- Heartbeat-pils: provides a general plugin and interface loading library.

Heartbeat can be installed via the yum package manager using the following commands:

```
[root]# yum install Heartbeat Heartbeat-stonith Heartbeat-pils
```

Heartbeat configuration involves three configuration files which need to be identical on both failover nodes.

The three configuration files are:

authkeys – This file contains keys for mutual node authentication, it must have root only readable permissions set.

ha.cf – Global cluster configuration, this file is read by the Heartbeat daemon on startup. It lists the communication facilities enabled between nodes, enables or disables certain features, and optionally lists the cluster nodes by host name.

haresources – This file specifies the resources for the cluster and the default owner. The haresources file is one of the most important files to configure when using Heartbeat.

Heartbeat configuration parameters:

The Dell Lustre HA-MDS module Heartbeat parameters	
MDS node host name	mds01, mds02
MGS device	/dev/mpath/mgs
mount point	/lustre/mgs
MDT device	/dev/mpath/testfs_mdt00
mount point	/lustre/testfs_mdt00
IPMI device names	mds01_ipmi, mds02_ipmi

The Dell Lustre™ HA-OSS module Heartbeat parameters	
First OSS node parameters	
OSS node host name	oss01
OST device	/dev/mpath/testfs_ost00
mount point	/lustre/testfs_ost00
OST device	/dev/mpath/testfs_ost01
mount point	/lustre/testfs_ost01
OST device	/dev/mpath/testfs_ost02
mount point	/lustre/testfs_ost02
IPMI device names	oss01_ipmi

The Dell Lustre HA-OSS module Heartbeat parameters	
Second OSS node parameters	
OSS node host name	oss02
OST device	/dev/mpath/testfs_ost03
mount point	/lustre/testfs_ost03
OST device	/dev/mpath/testfs_ost04
mount point	/lustre/testfs_ost04
OST device	/dev/mpath/testfs_ost05
mount point	/lustre/testfs_ost05
IPMI device names	oss02_ipmi

The complete configuration files can be found in Appendix C.

Once the configuration is complete, the Heartbeat service can be started. It has to be started on both nodes at the same time to work properly. The role of the service is to start Heartbeat resources, which mount the Lustre filesystem and manage the resource ownership. The Heartbeat service on the first node monitors the status of the partner node. If the partner node fails, the Heartbeat service can move resources mounted on that node to the failover node. In addition, the STONITH subsystem can power-off or reboot the failed node to ensure integrity of the Lustre filesystem data. This keeps the Lustre filesystem availability as high as possible.

The following command starts Heartbeat service on the server node.

```
root]# service Heartbeat start
```

It is recommended to configure the Heartbeat service to start at server boot time.

The system administrator can issue a failover request manually by using `hb_takover` and `hb_standby` tools. Both tools can use one of the following options: `all|foreign|local|failback`

For example:

```
[root]# /usr/lib64/Heartbeat/hb_takover local
```

More detailed information about configuration and usage of HA-Linux Heartbeat with Lustre^{TM} can be found at http://wiki.lustre.org/manual/LustreManual18_HTML/Failover.html

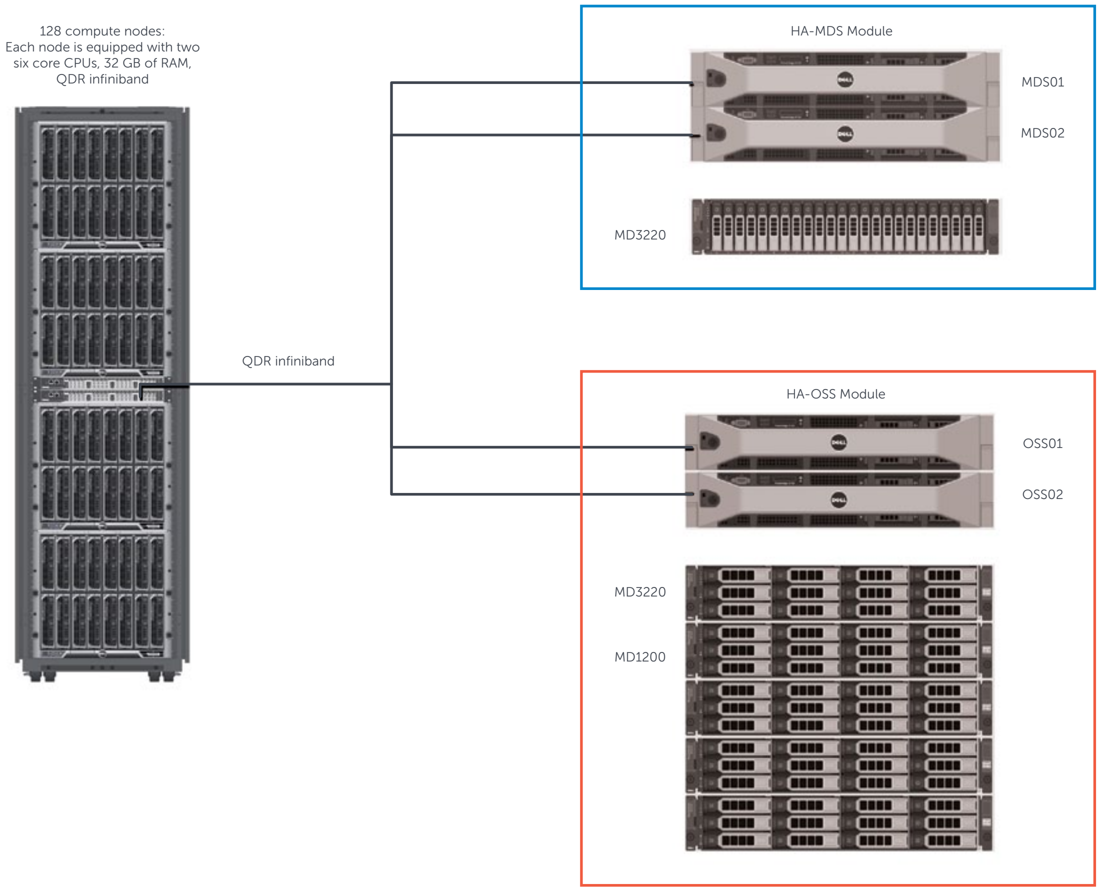
10.0 Performance Tests and Results

This section presents the results of performance benchmarking on the Dell Lustre Storage Solution. A variety of performance tests were run to evaluate the following common I/O workloads:

- Throughput of large sequential reads and writes
- IOPS of sequential reads and writes
- Metadata operations

The Iozone, IOR and mdtest benchmarks were used for this study. The I/O network used for all tests is a QDR Infiniband network and the test system configuration is shown in Figure 9.

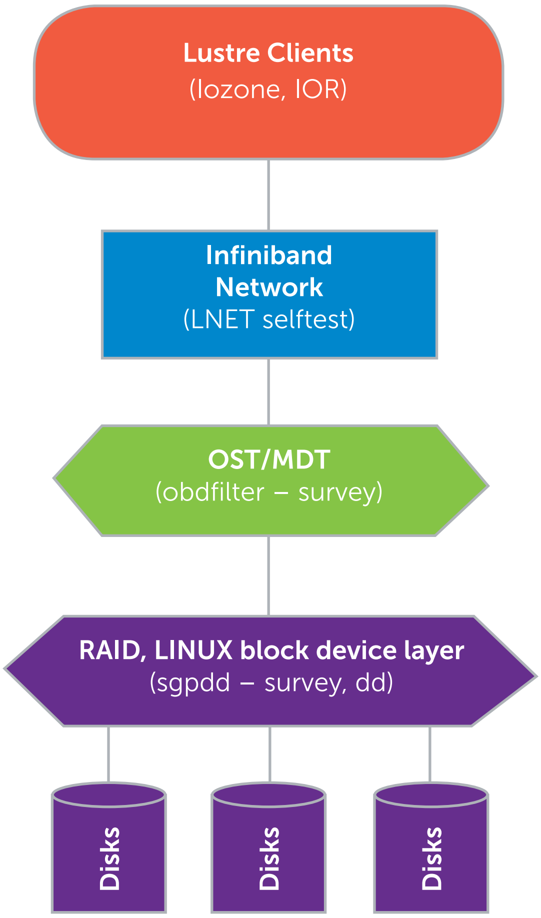
Figure 9. Lustre Test Setup



10.1 Lustre^{TM} Storage Profiling

It is good practice to obtain baseline performance figures for all storage system elements involved in tests. Figure 10 shows a high level representation of the testing path that should be followed to properly profile a storage system. Lustre provides a handful of profiling tools which can be used to obtain baseline figures for the backend storage system hardware. These tools allow testing the hardware platform from the very bottom of the benchmarking stack by measuring raw performance data with minimal software overhead. Moving up the stack Lustre provides tools to measure the server's local filesystem, the network connecting client nodes and the clients themselves.

Figure 10. Dell Lustre storage benchmarking stack



The purpose of the performance testing is to provide accurate Lustre filesystem throughput measurements of the Dell Lustre Storage System as configured in this paper. The results can then be compared with the expected theoretical performance of the tested storage system. Performance tests were conducted using one HA-MDS module and one HA-OSS module.

10.2 HA-OSS Module Raw Speed (sgpdd-survey)

The Lustre^{TM} I/O kit included in the lustre-test rpm package provides a set of tools which help to validate the performance of various hardware and software layers. One of the I/O kit tools called sgppd_survey can test bare metal performance of the back-end storage system, while bypassing as much of the kernel as possible. This tool can be used to characterise the performance of a storage device by simulating an OST serving files consisting of multiple stripes. The data collected by this survey can help to determine the performance of a Lustre OST exporting the device.

sgpdd-survey is a shell script included in the Lustre IOKit toolkit. It uses the sgp_dd tool provided by the sg3 utilities package, which is a scalable version of the dd command and adds the ability to do multi-threaded I/O with various block sizes and to multiple disk regions. The script uses sgp_dd to carry out raw sequential disk I/O. It runs with variable numbers of sgp_dd threads to show how performance varies with different request queue depths. The script spawns variable numbers of sgp_dd instances, each reading or writing to a separate area of the disk to demonstrate performance variance within a number of concurrent stripe files. The purpose of running sgppdd_survey is to obtain a performance baseline and to set expectations for the Lustre filesystem.

Below are the results of the sgpdd_survey for tests using all six LUNs. Two sets of tests were run, first with the cache mirroring feature disabled and second with cache mirroring feature enabled. Figures 11, 12, 13, 14 show results from those tests and there is clear evidence that disabling cache mirroring provides significant boost to write performance at a cost of decreased data safety (no protection for controller failures). Since the performance is much better with cache mirroring feature turned off, for the tests to follow, this feature will be disabled.

Command line used:

```
size=”24576” crglo=1 crghi=16 thrlo=1 thrhi=64  scsidevs=”/dev/sdk /dev/sdw
/dev/sdaa /dev/sdo /dev/sdi /dev/sdu /dev/sdm /dev/sdy /dev/sdg /dev/sds /
dev/sdac /dev/sdq” ./sgpdd-survey
```

Figure 11 sgpdd_survey with 6 LUNs, cache mirroring disabled – write test

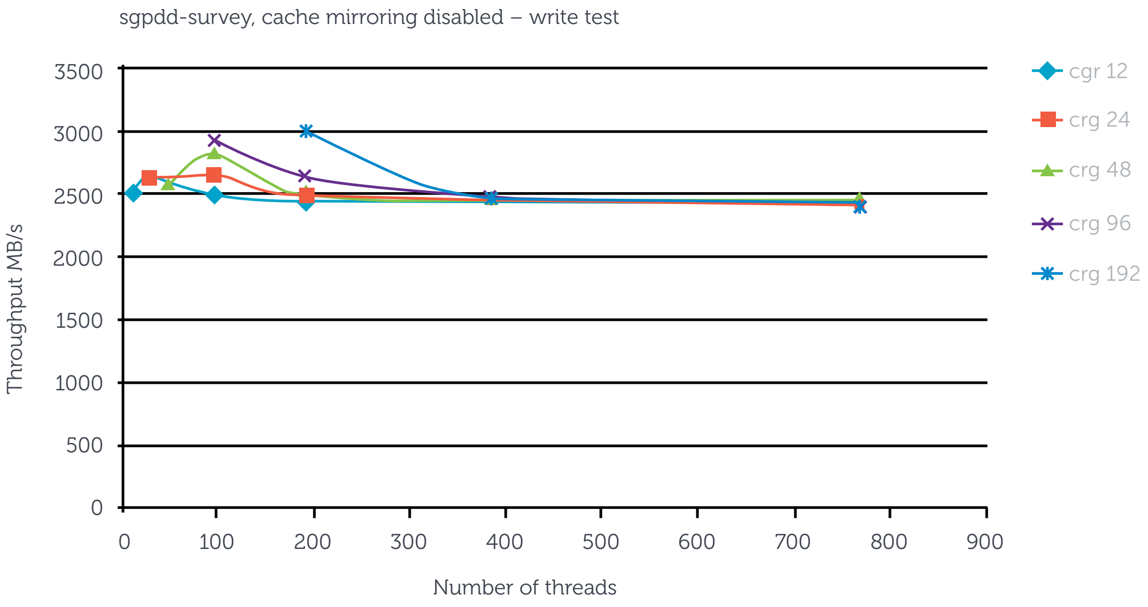


Figure 12 sgpdd_survey with 6 LUNs, cache mirroring disabled – readtest

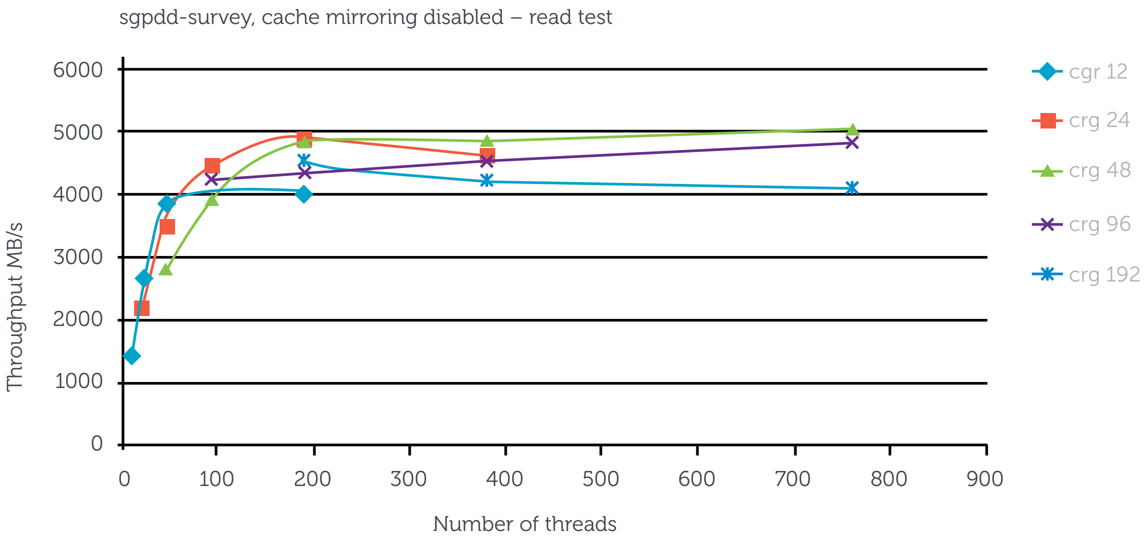


Figure 13 sgpdd_survey with 6 LUNs, cache mirroring enabled – write test

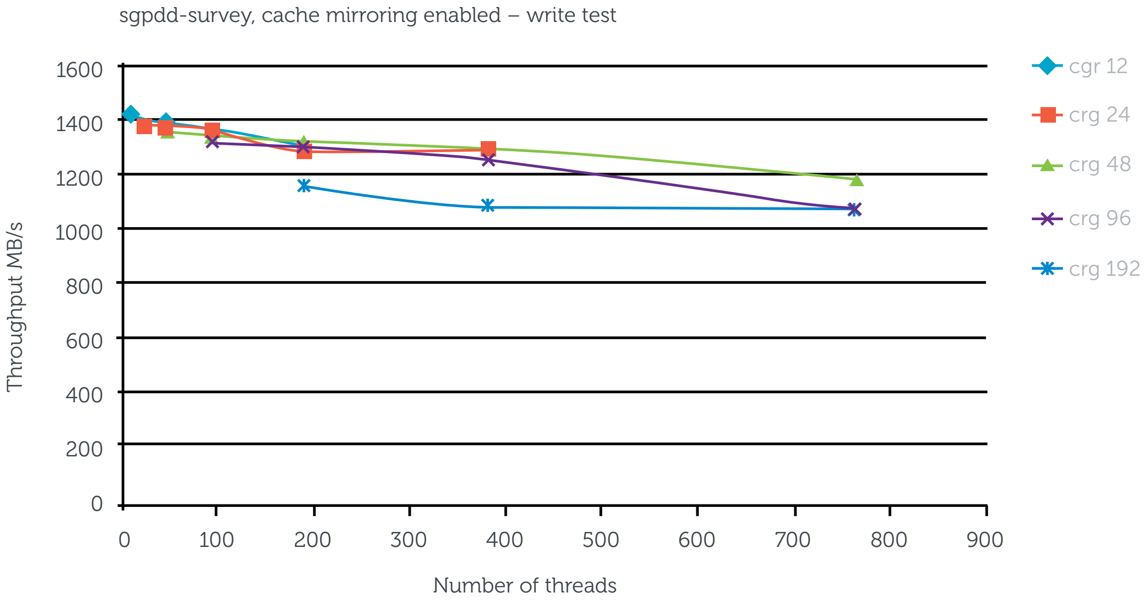
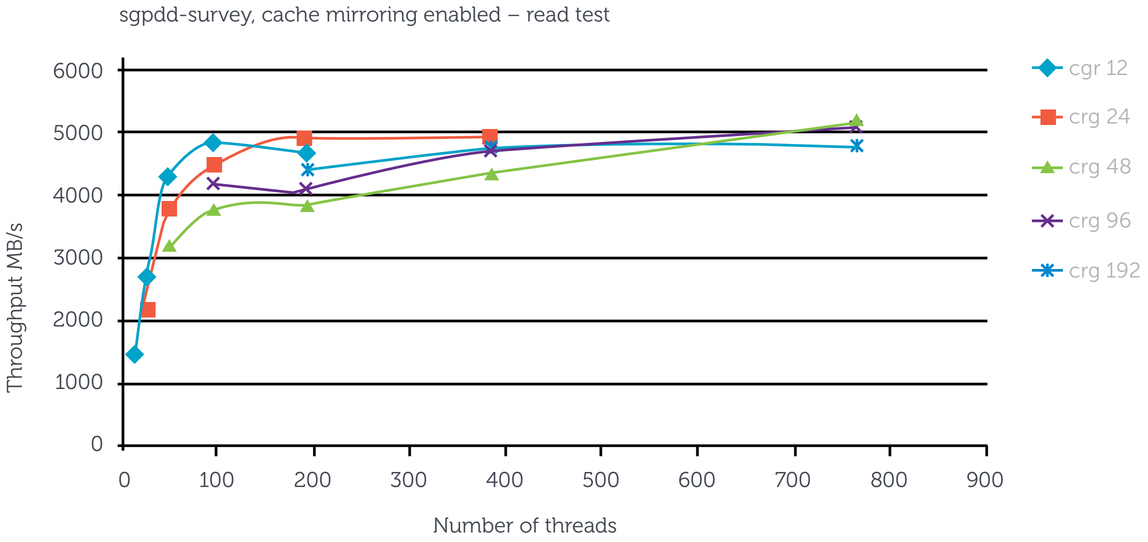


Figure 14 sgpdd_survey with 6 LUNs, cache mirroring enabled – read test



Understanding sgpdd-survey
crg (concurrent regions) – describes how many regions on the disk are written to or read by sgp_dd. crg simulates multiple Lustre clients accessing an OST. As crg grows, performance will drop due to more disk seeks (especially for reads).

thr (number of threads) – this parameter simulates Lustre™ OSS threads. More OSS threads can do more I/O, but if too many threads are in use, the hardware will not be able to process them and performance will drop.

The above results show that the Dell PowerVault™ MD3200 storage can provide near maximum theoretical read and write performance when tested with the sgpdd_survey tool. This demonstrates that the RAID configuration used is optimal. Summarizing the above test results it is clear that the PowerVault MD3200 array as configured provides close to the maximum performance expected from the raw device and sets a good baseline for further filesystem performance tests.

10.3 Testing Dell Lustre^{TM} OSS Module with obdfilter-survey

Once the block devices are formatted with a Lustre filesystem, the next step is to benchmark the OSTs and their underlying ldiskfs filesystem. This can be done using a tool provided by the Lustre IOKit called obdfilter-survey. This script profiles the overall throughput of the storage hardware by applying a range of workloads to the OSTs. The main purpose of running obdfilter-survey is to measure the maximum performance of a storage system and to find the saturation points which cause performance drops.

Command line used:

```
nobjlo="2" nobjhi="16" thrlo="2" thrhi="64" size="24576" targets="testfs-OST0000 testfs-OST0001 testfs-OST0002 testfs-OST0003 testfs-OST0004 testfs-OST0005" ./obdfilter-survey
```

Figure 15. obdfilter – survey write

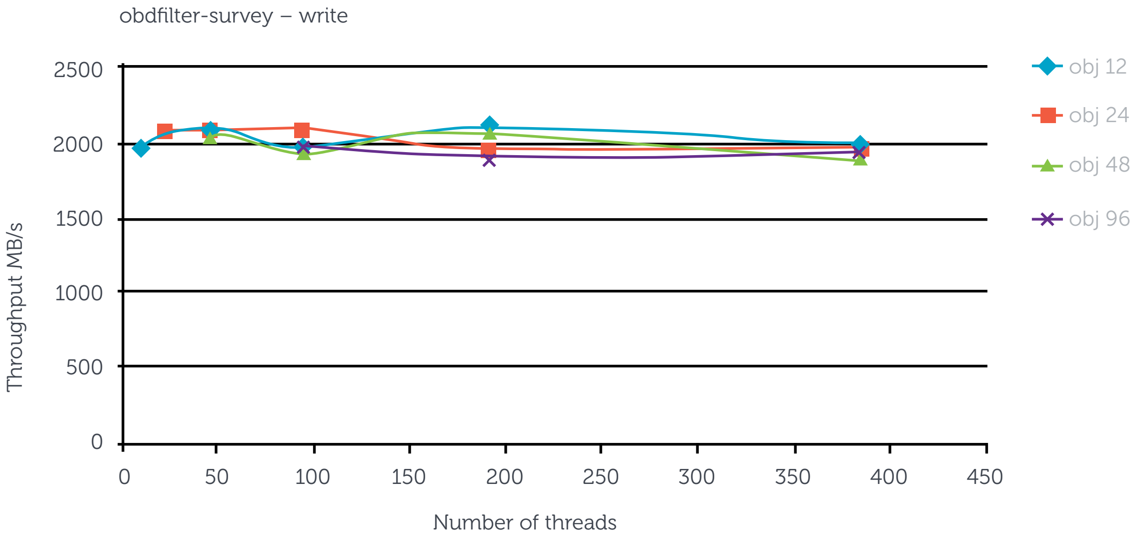
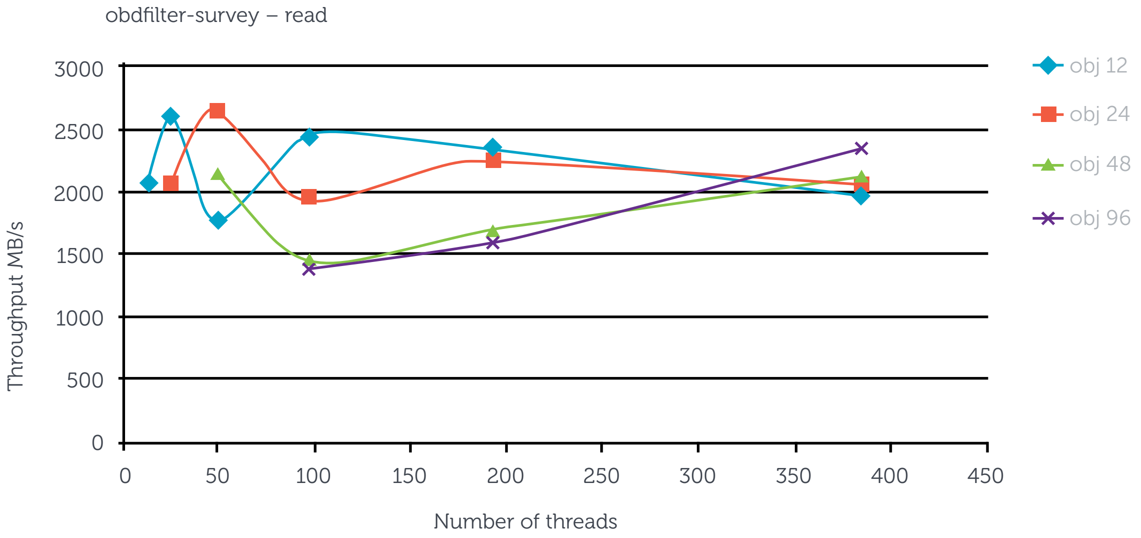


Figure 16. obdfilter – survey read



Understanding obdfilter-survey

obj (Lustre objects) - describes how many Lustre™ objects are written or read. obj simulates multiple Lustre clients accessing the OST and reading/writing multiple objects. As the number of obj grows, performance will drop due to more disk seeks (especially for reads).

thr (number of threads) - this parameter simulates Lustre OSS threads. More OSS threads can do more I/O, but if too many threads are in use, the hardware will not be able to process them and performance will drop.

10.4 Measuring Lustre Infiniband Network Performance with LNET Selftest

The LNET selftest (lst) utility helps to test the operation of a Lustre network between servers and clients. It allows verification that the Lustre network was properly configured and that performance meets expected values. LNET selftest runs as a kernel module and has to be loaded on all nodes that take part in the test. A utility called lst is used to configure and launch Lustre network tests. Below is an example lnet_selftest script which can be used to reproduce data presented in this paper.

```
#!/bin/bash

dist=”$1”
conc=”$2”
cli=”$3”
action=”$4”
export LST_SESSION=$$
echo ====================================
echo clients: 10.144.9.${cli}@o2ib
echo distribution: $dist
echo concurency: $conc
echo action: $action
echo ====================================
lst new_session rw
lst add_group clients 10.144.9.${cli}@o2ib
lst add_group servers 10.144.245.17@o2ib
lst add_batch bulk_rw
lst add_test --batch bulk_rw --distribute ${dist}:1 --concurrency ${conc}
--from clients --to servers brw ${action} size=1M
#lst add_test --batch bulk_rw --distribute 1:1 --concurrency 8 --from
clients --to servers brw read size=1M
# start running
lst run bulk_rw
# display server stats for 180 seconds
#lst stat 10.44.245.17@o2ib 10.44.245.18@o2ib & sleep 10
lst stat clients servers & sleep 30
lst stop bulk_rw
# tear down
lst end_session
pkill lst
```

Figures 17 and 18 show results from tests run on a range of Lustre™ client nodes from 1 to 16 and one OSS server with variable numbers of concurrency (RPCs in flight). More RPCs in flight seem to have a positive effect on the performance thus for the further client tests max_rpcs_in_flight will be set to 32.

Figure 17 LNET selftest

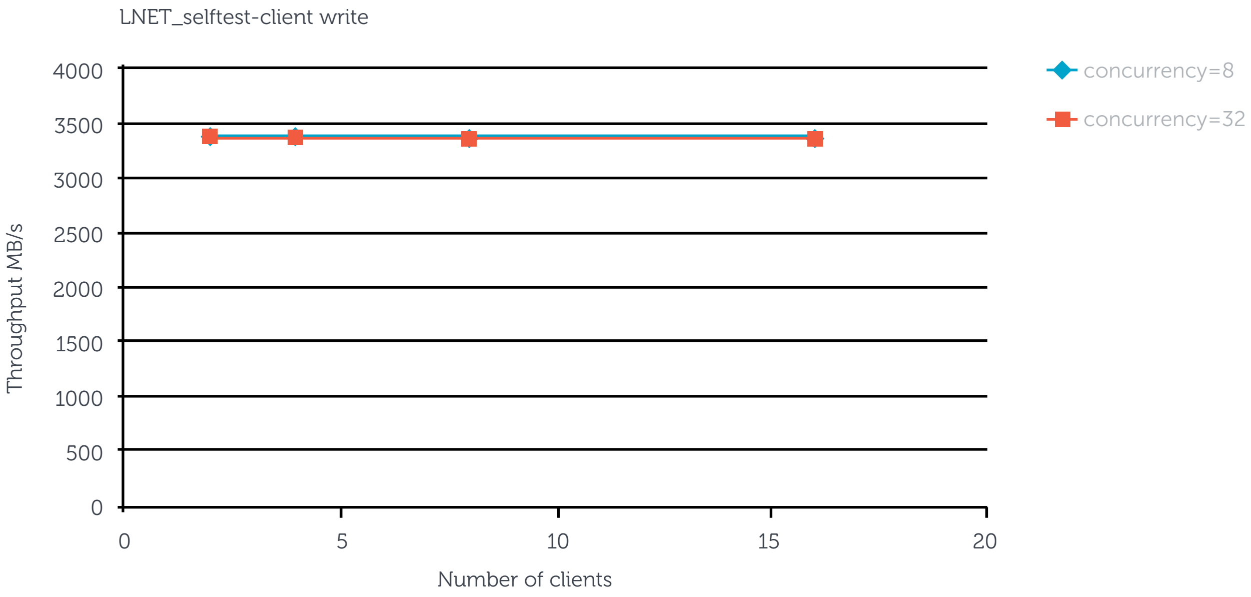
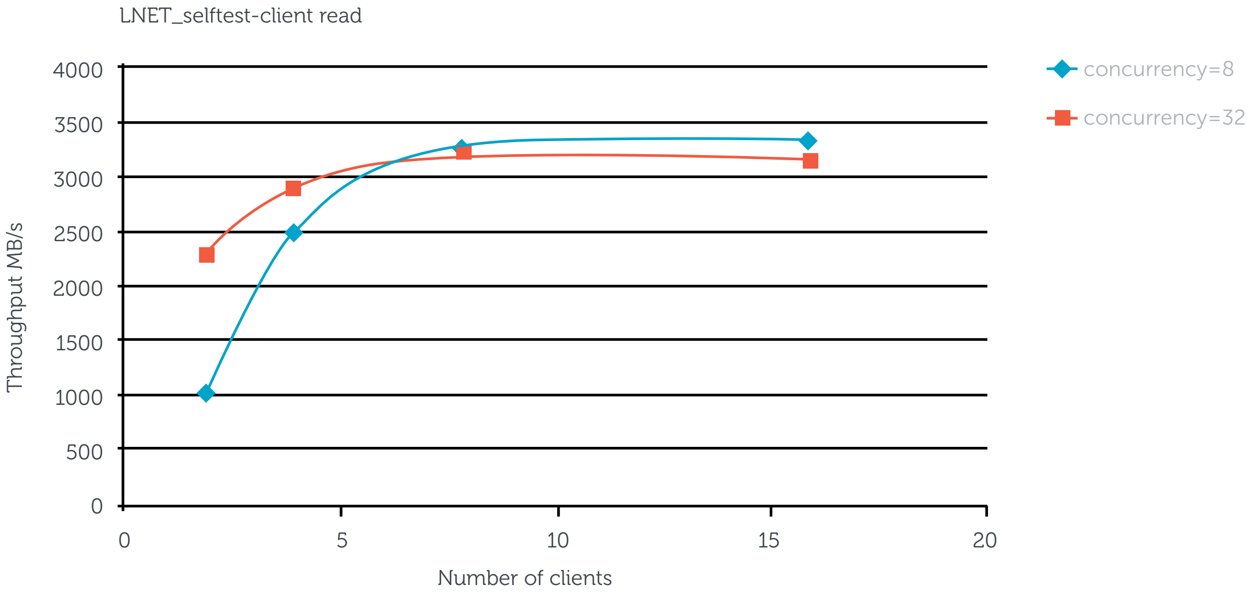


Figure 18 LNET selftest

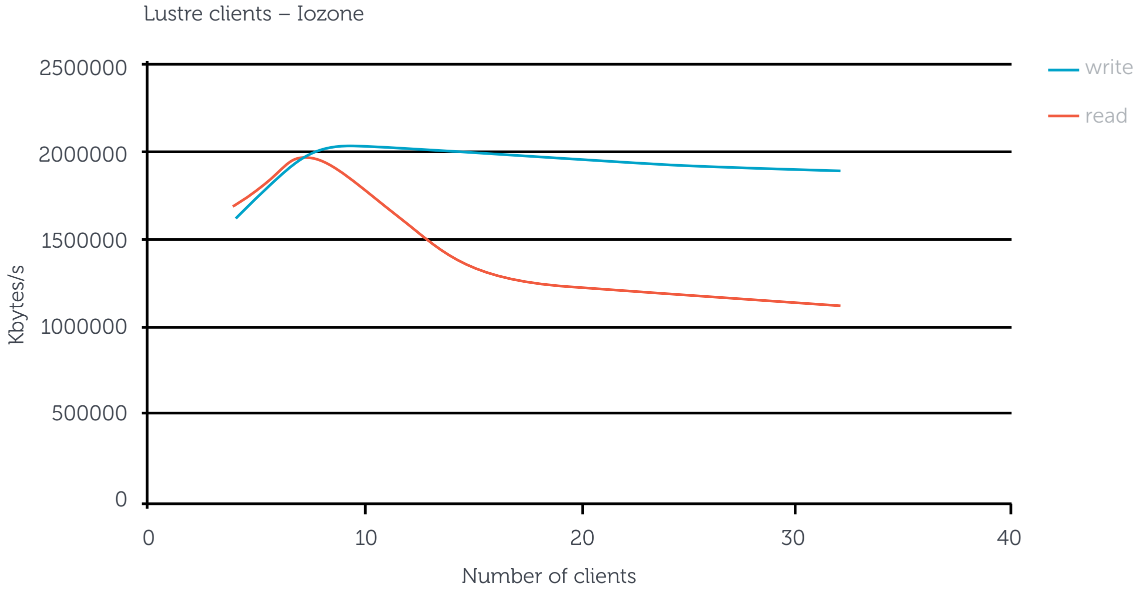


Concurrency – determines how many requests each client node in a test keeps on the wire.

10.5 Measuring Lustre™ Sequential Throughput Performance with Iozone

Iozone is a benchmark that measures read/write performance across multiple Lustre clients. In cluster mode it generates a number of sequential writes and then reads per client and it can run on many clients simultaneously, providing aggregate performance results. Below are the results of a run of Iozone on number of clients from 6 to 32 where each client was writing/reading a single file and Lustre uses the default striping policy. The chart below shows that on larger node counts write performance is better than read performance. This behaviour is to be expected with a Lustre filesystem. When doing writes, clients send RPCs asynchronously and those RPCs are allocated and written to disks in the order they arrive, and the back-end storage can aggregate those writes efficiently. In the case of reads, the read requests from clients may come in a different order, which requires a lot of seeking and which throttles overall read throughput.

Figure 19 Iozone



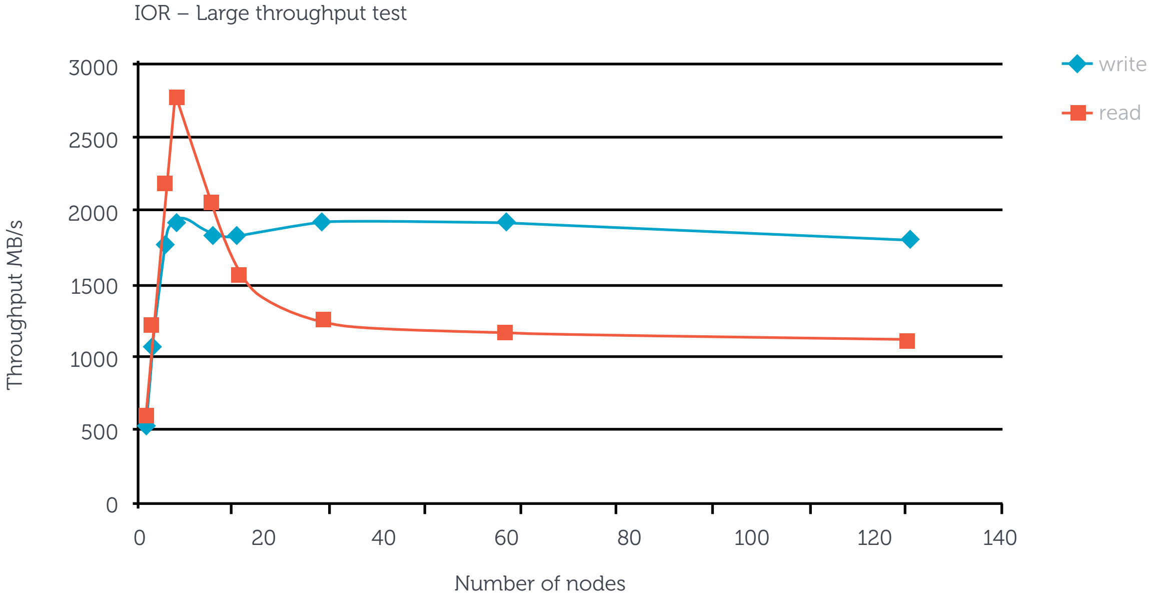
10.6 Measuring Lustre Throughput and IOPS Performance with IOR

IOR is another very good I/O benchmark providing a wide range of useful functions and features. With IOR, clients first write data and then when read is performed clients read data written by another client. This completely eliminates the client read cache effect, so avoiding the problem of having to flush the client cache after write. IOR uses MPI for multimode communication and thread synchronization, which helps to provide very accurate results from large multinode tests.

The IOR large throughput test shown in Figure 20 was run on a range of Lustre clients from 1 to 126. Each client writes its own file, where the files' distribution is balanced equally across all six OSTs. For small node counts, read outperforms write throughput. However with increase of node count, therefore writing more files, I/O concurrency on the OSS grows so read performance drops.

```
Command line used: ./IOR/IOR -a POSIX -N 30 -b 8g -d 5 -t 1024k —o     /
testfs/wjt27/test_file -e -g -w -r -s 1 -i 2 -vv -F -C
```

Figure 20 IOR throughput



The IOPS tests were run on a range of Lustre™ clients from 1 to 16. A set of block sizes were tested from a small block size of 4KB to a large block size of 1MB. Each node writes and reads from its own file. As shown in figures 21 and 22 I/O rates scale almost linearly with the number of OSTs added to the test until the total number of OSTs is reached. At that point I/O rates stop scaling and for large file counts they may slightly drop. The Dell Lustre Storage system shows very good IOPS rates for both sequential reads and writes. This is mainly due to internal Lustre client caches which aggregate small I/O transactions in a dirty cache. Data from the client's cache can then be efficiently transferred in 1MB RPCs to a Lustre server.

Figure 21 IOR IOPS

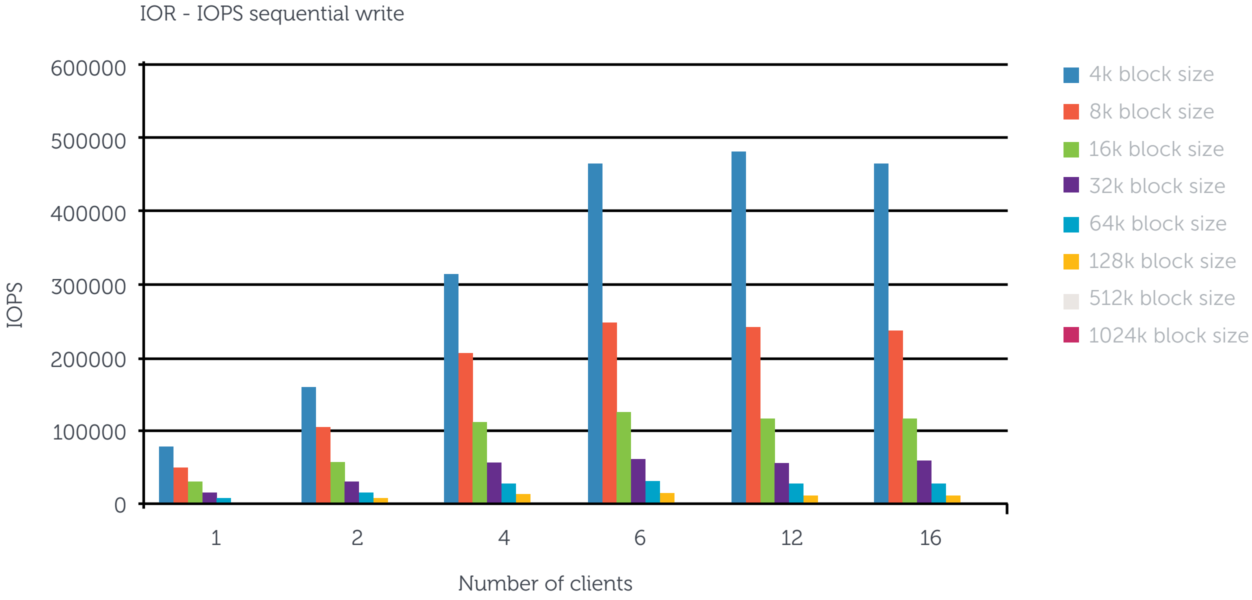
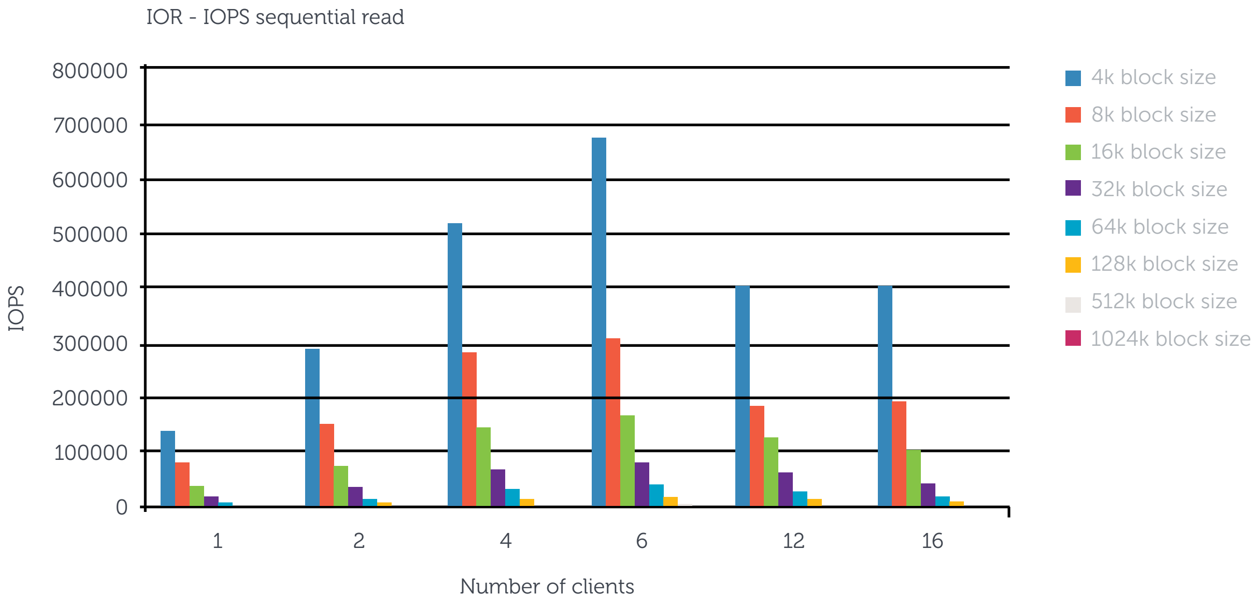


Figure 22 IOR IOPS



10.7 Benchmarking Metadata Performance of the Dell HA-MDS Module

The IOPS tests were run on a range of Lustre^{TM} clients from 1 to 32. In both tests the most common metadata operations were tested, such as creation, stat, and removal. The first test was conducted using 66000 files per client with sync after each write. The second test also used 66000 files, however sync after write was not enabled, so benchmark could benefit from Lustre client caches. The results below show that the Dell HA-MDS module can provide good and sustainable metadata performance. In case of 'stat', adding more clients can increase the number of operations. Performance of directory and file 'create' and 'remove' operations remains roughly unchanged across the tests. The Lustre filesystem has been designed to work with massive streaming parallel I/O where the typical file size is in the range of megabytes or more. However, as the above results show, Lustre not only excels in a typical HPC environment but also can provide good performance for general purpose and home filesystems with mixed pattern filesystem access.

```
Command line used: mdtest-1.8.3/mdtest -u -d /testfs/wjt27 -i 3 -b 32 -z 1
-L -I 2000 -y -u -t -r
```

Figure 23 Metadata – mdtest

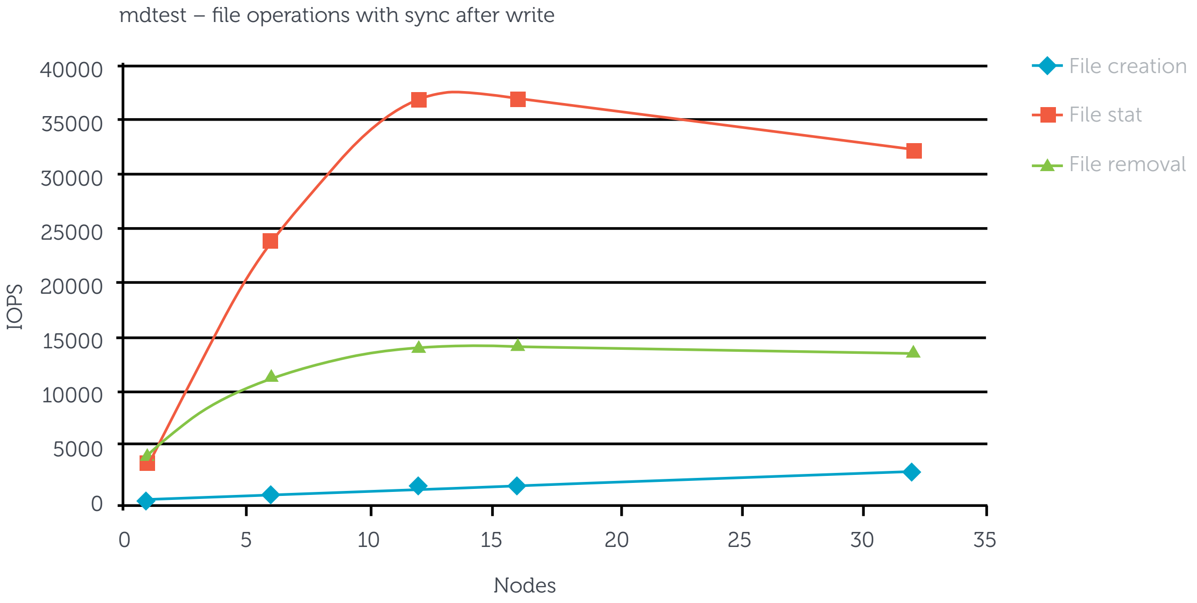
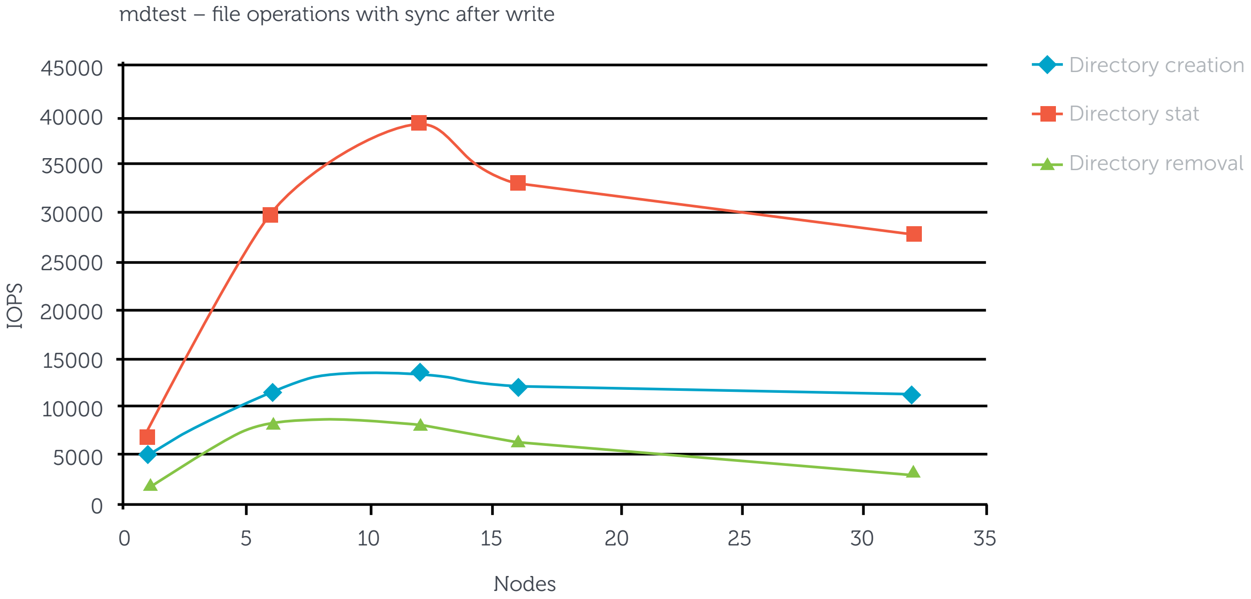


Figure 24 Metadata – mdtest



```
Command line used: mdtest-1.8.3/mdtest -u -d /testfs/wjt27 -i 3 -b 32 -z 1
-L -I 2000 -u -t -r
```

Figure 25 Metadata – mdtest

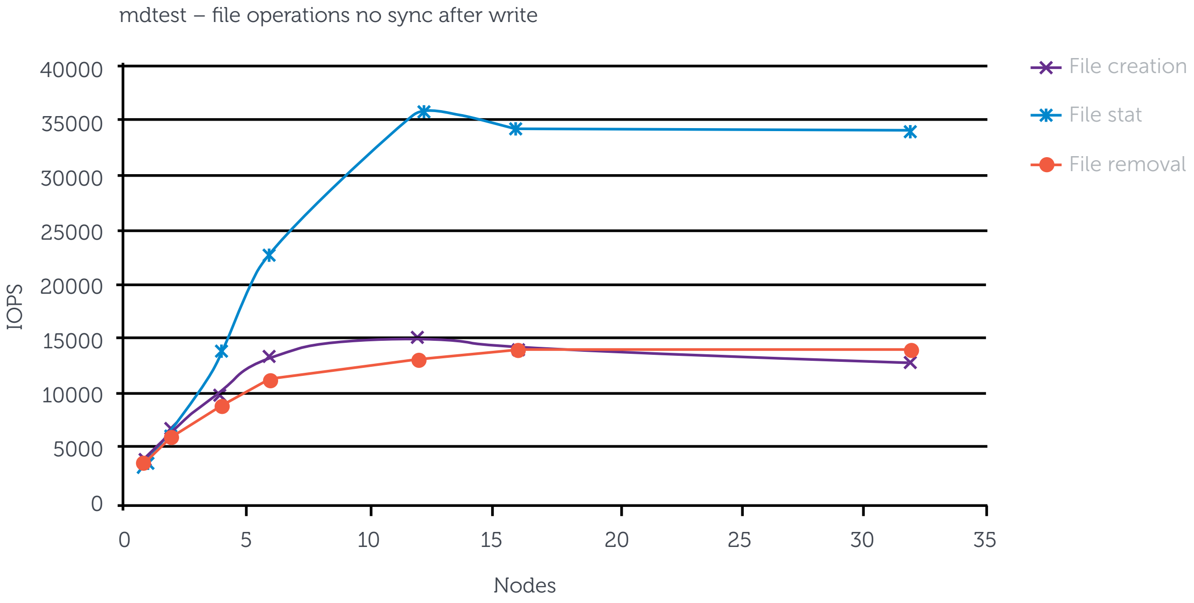
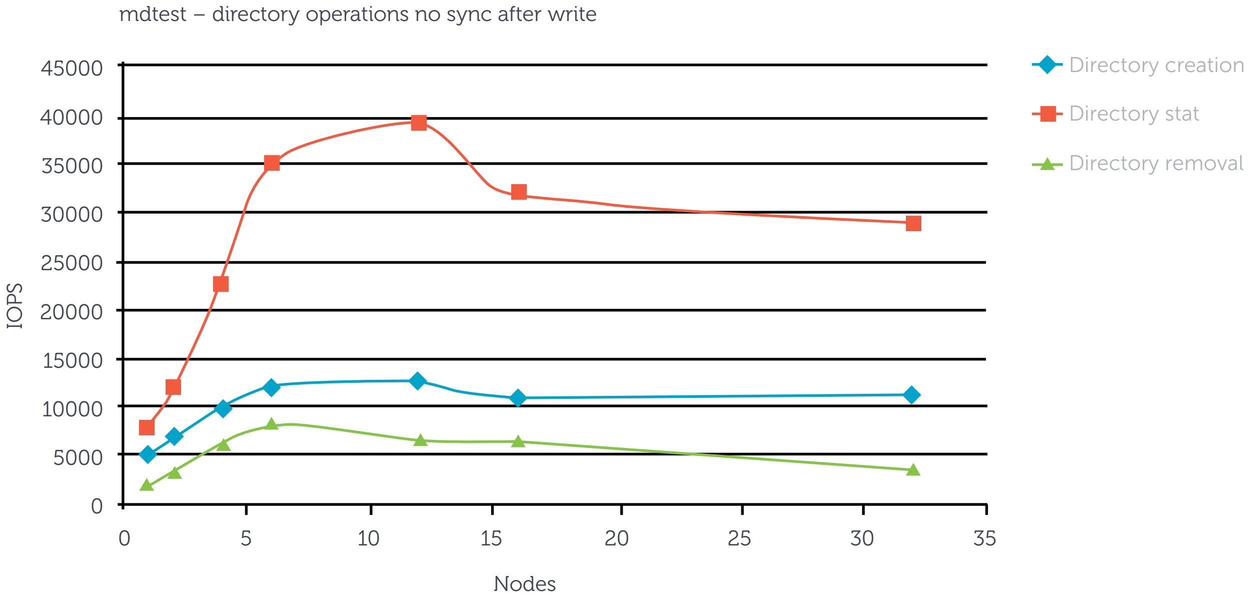


Figure 26 Metadata – mdtest



11.0 Summary

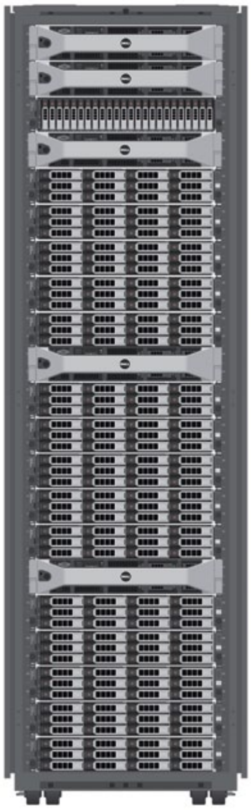
As commodity clusters continue to grow in performance and more applications adopt parallel I/O traditional filesystems such as NFS, NAS, SAN and DAS are failing to keep up with HPC I/O demand. The Lustre^{TM} parallel filesystem is gaining ground within the departmental and workgroup HPC space since it has proved capable of meeting the high I/O workload within HPC environments, as well as presenting a range of additional operational benefits as described in the introduction.

This paper presents a detailed description of how to build a Dell PowerVault^{TM} MD3200 Lustre storage brick, presenting a current best-in-class commodity Lustre. The storage brick as described shows good sequential I/O throughput, good IOPS/metadata performance, high redundancy, high availability with good data safety features, all of which are important within the HPC environment.

The storage brick shows very good Lustre filesystem performance. This solution has demonstrated scalable performance across the storage array, yielding over 2GB/s file I/O bandwidth with high levels of availability and data security. Operational experience has demonstrated less than 0.5% un-planned downtime. We have found the Dell Lustre storage brick is a good match to both HPC performance and the operational requirements seen within a large University HPC data centre.

The Cambridge HPC Service has constructed a 3-brick solution housed in one rack cabinet and providing 280 TB of usable storage with 6GB/s back end I/O bandwidth. With new 3TB disks instead of the 2TB used here, the single rack capacity would rise to 420 TB. The solution described here has been used in the Cambridge HPC production environment for the last six months, serving over 500 users. Availability and data security demonstrates less than 0.5% unscheduled downtime in our 24/7 operational service.

Figure 27 – one rack storage



- Provides 280 TB of usable storage space
- 6GB/s back end I/O bandwidth
- Could be extended with 3TB disk drives
- Time proven reliability
- Less than 0.5 % unscheduled downtime

Looking forward, it is likely that Lustre™ will continue to have strong open source development and support, with a strong feature roadmap. Also the commodity storage hardware platforms will continue to improve in terms of disk density, raw performance and cost per TB. These developments, combined with the use of Infiniband storage networks and solid state disks for metadata, will combine to offer large sequential I/O and IOPS performance increases. This will help ensure that commodity Lustre storage arrays will continue to grow their presence within the mid-range HPC segments and beyond.

Appendix A

Following script installs software necessary to access, configure and manage the Dell PowerVault™ MD3200 disk arrays on RHEL-5.5

```
mdsm_install.sh
```

```
#!/bin/bash
INSTALL_TYPE=$1
RESOURCE_CD=$2
#======================================================================
install_rpms() # Installs all required RPMS from the Dell resource CD
#======================================================================
{
        cd ${RESOURCE_CD}/dkms
        echo Installing prerequisites:
        echo Installing DKMS
        DRIVER_VERSION=`ls dkms* | cut -d - -f 2,3`
        rpm -Uvh dkms-${DRIVER_VERSION}
    cd ${RESOURCE_CD}/ linux/dm/rhel55
        echo Installing scsi_dh_rdac
        DRIVER_VERSION=`ls scsi_dh_rdac* | cut -d - -f 2,3`
        rpm -Uvh scsi_dh_rdac-${DRIVER_VERSION}
}
case $1 in
storage_server)
        install_rpms
        cat >> /tmp/mdsm_installer.option <<-EOF
          INSTALLER_UI=silent
          CHOSEN_INSTALL_FEATURE_LIST=SMruntime,SMutil,SMagent
          AUTO_START_CHOICE=0
          USER_REQUESTED_RESTART=YES
        EOF
        echo Installing MD Storage Software:
 ${RESOURCE_CD}/linux/mdsm/SMIA-LINUX-*.bin -i silent -f /tmp/mdsm_
installer.option
        ;;
mgmnt_server)
        echo Creating mdsm_installer.option install file:
        cat >> /tmp/mdsm_installer.option <<-EOF
          INSTALLER_UI=silent
          CHOSEN_INSTALL_FEATURE_LIST=SMclient,SMFirmware,SMfwupgrade,SMrunt
ime
          AUTO_START_CHOICE=0
          USER_REQUESTED_RESTART=YES
        EOF
        echo Installing MD Storage Software:
        ${RESOURCE_CD}/linux/mdsm/SMIA-LINUX-*.bin -i silent -f /tmp/mdsm_
installer.option
        ;;
full)
#       install_rpms()
        echo Creating mdsm_installer.option install file:
        cat >> /tmp/mdsm_installer.option <<-EOF
          INSTALLER_UI=silent
          CHOSEN_INSTALL_FEATURE_LIST=SMclient,SMFirmware,SMfwupgrade,SMrunt
ime,SMutil,SMagent
          AUTO_START_CHOICE=0
          USER_REQUESTED_RESTART=YES
        EOF
        echo Installing MD Storage Software:
        ${RESOURCE_CD}/linux/mdsm/SMIA-LINUX-*.bin -i silent -f /tmp/mdsm_
installer.option
        ;;
*)
        echo Please specify installation type and dell resource CD mount
point location
        echo Installation type can be on of: storage_server, mgmnt_server or
full
        echo storage_server installation is suitable for servers directly
connected to disk array
        echo mgmnt_server installation installs MDSM monitoring software
only
        echo full installation is suitable for storage server working also
as management server

        ;;

esac
```

Appendix B

Dell PowerVault™ MD3200 configuration script for Dell Lustre™ HA-OSS module:

Host topology configuration:

```
// Storage Array global logical configuration script commands
show “Setting the Storage Array user label to scratch_brick.”;
set storageArray userLabel=”scratch_brick”;

show “Setting the Storage Array media scan rate to 15.”;
set storageArray mediaScanRate=15;

show “Setting the Storage Array cache block size to 32.”;
set storageArray cacheBlockSize=32;

show “Setting the Storage Array to begin cache flush at 80% full.”;
set storageArray cacheFlushStart=80;

show “Setting the Storage Array to end cache flush at 80% full.”;
set storageArray cacheFlushStop=80;

// Creating Host Topology
show “Creating Host Group oss01_oss02.”;
create hostGroup userLabel=”oss01_oss02”;

show “Creating Host oss02 with Host Type Index 1 on Host Group oss01_
oss02.”;
// This Host Type Index corresponds to Type Linux
create host userLabel=”oss02” hostType=1 hostGroup=”oss01_oss02”;

show “Creating Host oss01 with Host Type Index 1 on Host Group oss01_
oss02.”;
// This Host Type Index corresponds to Type Linux
create host userLabel=”oss01” hostType=1 hostGroup=”oss01_oss02”;

show “Creating Host Port oss02P0 on Host oss02 with WWID 5782bcb01b754001
and with interfaceType SAS.”;
create hostPort host=”oss02” userLabel=”oss02P0”
identifier=”5782bcb01b754001” interfaceType=SAS;

show “Creating Host Port oss01P0 on Host oss01 with WWID 5782bcb01b753701
and with interfaceType SAS.”;
create hostPort host=”oss01” userLabel=”oss01P0”
identifier=”5782bcb01b753701” interfaceType=SAS;
```

```
show “Creating Host Port oss02P1 on Host oss02 with WWID 5782bcb01b753f01
and with interfaceType SAS.”;
create hostPort host=”oss02” userLabel=”oss02P1”
identifier=”5782bcb01b753f01” interfaceType=SAS;

show “Creating Host Port oss01P1 on Host oss01 with WWID 5782bcb01b753700
and with interfaceType SAS.”;
create hostPort host=”oss01” userLabel=”oss01P1”
identifier=”5782bcb01b753700” interfaceType=SAS;

show “Creating Host Port oss02P2 on Host oss02 with WWID 5782bcb01b754000
and with interfaceType SAS.”;
create hostPort host=”oss02” userLabel=”oss02P2”
identifier=”5782bcb01b754000” interfaceType=SAS;

show “Creating Host Port oss02P3 on Host oss02 with WWID 5782bcb01b753f00
and with interfaceType SAS.”;
create hostPort host=”oss02” userLabel=”oss02P3”
identifier=”5782bcb01b753f00” interfaceType=SAS;
```

Example virtual disk configuration:

```
show “Creating RAID 6 Virtual Disk vd_ost00 on new Disk Group dg_ost00.”;
//This command creates disk group <dg_ost00> and the initial virtual disk
<vd_ost00> with offset 0 on the disk group.
// NOTE: For Disk Groups that use all available capacity, the last Virtual
Disk on this group is
// created using all remaining capacity by omitting the capacity= virtual
disk creation parameter.
create virtualDisk physicalDisks=(0,0 1,0 2,0 3,0 4,0 0,1 2,1 2,1 3,1 4,1)
raidLevel=6 userLabel=”vd_ost00” volumeGroupUserLabel=”dg_ost00” owner=0
segmentSize=128 dssPreAllocate=true securityType=none;
show “Setting additional attributes for Virtual Disk vd_ost00.”;
// Configuration settings that can not be set during Virtual Disk creation.
set virtualDisk[“vd_ost00”] cacheFlushModifier=10;
set virtualDisk[“vd_ost00”] cacheWithoutBatteryEnabled=false;
set virtualDisk[“vd_ost00”] mirrorEnabled=false;
set virtualDisk[“vd_ost00”] readCacheEnabled=true;
set virtualDisk[“vd_ost00”] writeCacheEnabled=true;
set virtualDisk[“vd_ost00”] mediaScanEnabled=true;
set virtualDisk[“vd_ost00”] consistencyCheckEnabled=true;
set virtualDisk[“vd_ost00”] readAheadMultiplier=1;
set virtualDisk[“vd_ost00”] modificationPriority=lowest;
set virtualDisk[“vd_ost00”] preReadRedundancyCheck=false;
show “Creating Virtual Disk-to-LUN Mapping for Virtual Disk vd_ost00 to LUN
0 under Host oss01.”;
set virtualDisk [“vd_ost00”] logicalUnitNumber=0 host=”oss01”;
```

Appendix C

HA-Linux Heartbeat configuration (V1)

/etc/ha.d/authkeys

```
auth 2
#1 crc
2 sha1 puthereyourownkey
#3 md5 Hello!
```

/etc/ha.d/ha.cf

```
# File to write debug messages to
debugfile /var/log/ha-debug
# File to write other messages to
logfile /var/log/ha-log
# Facility to use for syslog()/logger
logfacility     local0
# keepalive: how long between Heartbeats?
keepalive 2
# deadtime: how long-to-declare-host-dead?
deadtime 60
# warntime: how long before issuing “late Heartbeat” warning?
warntime 10
initdead 180
#       What UDP port to use for bcast/ucast communication?
udpport 697
#       What interfaces to broadcast Heartbeats over?
bcast eth0 mds01
bcast eth0 mds02
auto_failback off
stonith_host mds01 external/ipmi mds02 mds02_ipmi root /etc/ha.d/ipmitool.
passwd
stonith_host mds02 external/ipmi mds01 mds01_ipmi root /etc/ha.d/ipmitool.
passwd
# node    nodename ...    -- must match uname -n
node    mds01
node    mds02
```

/etc/ha.d/ha.cf

```
# HA-Linux resource file for Dell Lustre HA-MDS module
         \ Filesystem::/dev/mpath/mgs::/lustre/mgs::lustre
\ Filesystem::/dev/mpath/testfs_mdt00::/lustre/testfs_mdt00::lustre
\
```

THIS Whitepaper IS FOR INFORMATIONAL PURPOSES ONLY, AND MAY CONTAIN TYPOGRAPHICAL ERRORS AND TECHNICAL INACCURACIES. THE CONTENT IS PROVIDED AS IS, WITHOUT EXPRESS OR IMPLIED WARRANTIES OF ANY KIND.