OrangeFS Reference Architecture

for **Dell R710/MD Storage Stack**

Scaling Vertically and Horizontally

A Technical White Paper

Testing Performed at:
Clemson Next Generation Evaluation & Usability Labs
January 2012

In conjunction with:

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

© 2012 Dell and Clemson University. All rights reserved. Reproduction of this material in any manner whatsoever without the express written permission.

All trademarks and trade names used in this document belong to the respective entities claiming the marks and names or products.

January 2012

Contents

Executive Summary

The OrangeFS Reference Architecture for the integration of Dell's R710 Servers and MD Storage Arrays (R710/MD Storage Stack) can be leveraged in a wide variety of high-perfomance, data-intensive computing environments. The growing list of industries that can benefit from this level of computing validates the increasing demand for a storage solution as flexible and scalable as the Dell R710/MD Storage Stack.

This solution guide describes three different scenarios for leveraging the R710/MD Storage Stack in these environments:

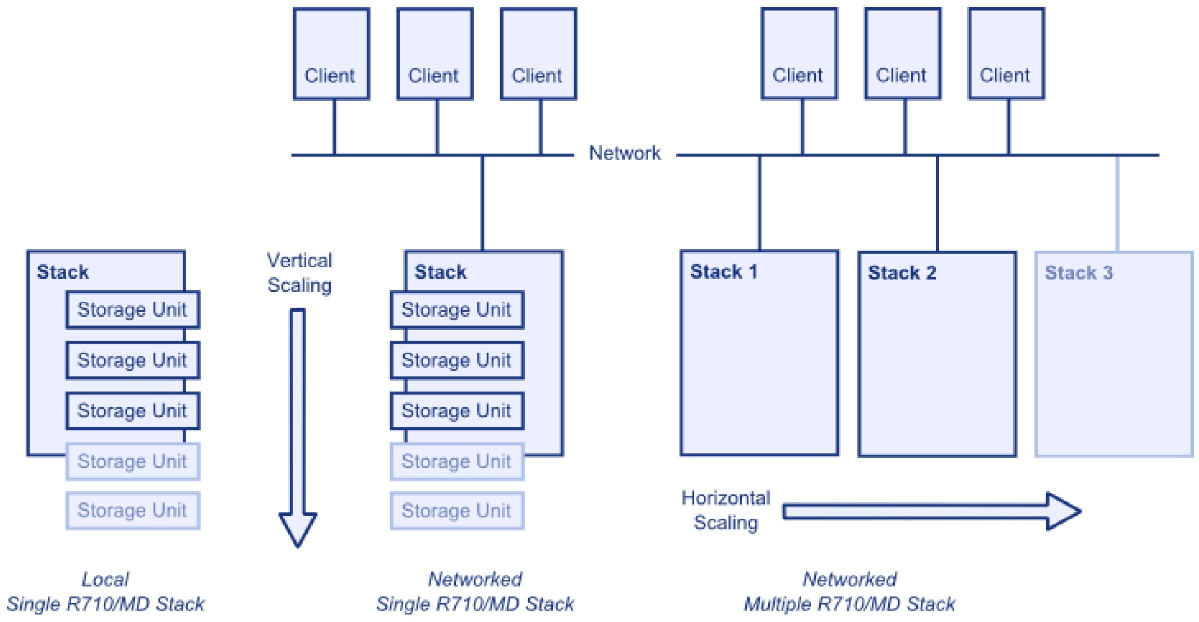
- Single stack in local (standalone) mode
- Single stack attached as a networked file system
- Dual stack attached as a networked file system

The first two scenarios illustrate vertical scaling, where increases to file system space and throughput can be achieved by adding Dell PowerVault™ MD1200s to the local or networked R710/MD Storage Stack.

The third scenario illustrates horizontal scaling, where increases in file system space and throughput can be achieved by adding more R710/MD Storage Stack s that leverage a parallel file system such as OrangeFS.

The goal is to offer more solutions to an ever-increasing number of markets that can benefit from cost-effective high-performance and data-intensive application environments.

Figure 1 - Vertical and Horizontal Scaling



1. Introduction

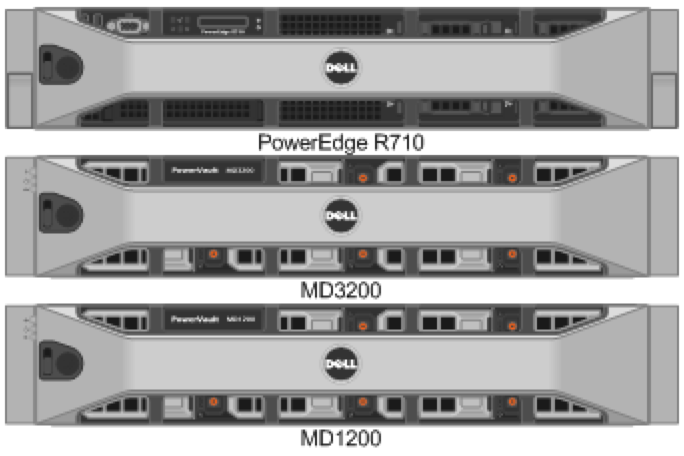
In the evolving world of high-performance, data-intensive computing, managing massive amounts of data efficiently and effectively has become a major focus. Many different kinds of workloads are pushing storage demands, including scientific applications such as the Large Hadron Collider (LHC), render farms, financial modeling, oil and gas simulation, genomics applications and many others.

In the specific case of the Large Hadron Collider, where the focus of the LHC experiments is to qualify the existence of the Higgs-Boson particle, the qualification is hidden in this massive and growing unstructured data pool. This creates an unprecedented requirement for an inexpensive, high performance, repeatable, accessible and scalable storage solution on a global scale. The data must be organized and distributed to the researchers in a tiered membership that is unique to the LHC community. Members of this community need a thorough understanding of the kind of storage required for their research. With 449 institutions across the globe working on the LHC project, many institutions need documentation on repeatedly implementing a mature storage platform specific to their functions in the LHC research community.

1.1. Dell R710/MD Storage Stack Overview

The Dell R710/MD Storage Stack base solution consists of one Dell PowerEdge™ R710 server connected to a Dell PowerVault MD3200 storage array enclosure. The MD3200 can have varying numbers of MD1200 expansion enclosures attached, up to eight, for a total of 96 hot-pluggable drives, providing vertical scalability.

Figure 2 - Basic R710/MD Storage Stack Configuration



The Dell PowerEdge R710 can have any of the following Intel® Xeon® processors: 5500 and 5600 series, six- or quad-core. It can also use up to 288GB of memory. It has 4 PCIe G2 slots and can be configured with 2 SAS-6 HBAs for MD3200 storage connectivity, Intel/QLogic QDR IB HCA or 10G Ethernet for interconnect connectivity. The R710 also has an internal H200 RAID controller that works with the Solid State Disk (SSD) for such functions as high I/O metadata storage.

The PowerVault MD3200 and one or more MD1200s can be configured with any of the following drive configurations: 15,000 RPM 6Gb/s SAS drives available in 300GB, 450GB and 600GB 7,200 RPM (Nearline) 6Gb/s SAS drives available in 500GB, 1TB, and 2TB, or SAS Solid State Drives (SSDs).

Note: For more details on the R710/MD Storage Stack test configuration, see Appendix B.

With the addition of a horizontally scalable file system, such as OrangeFS, the R710/MD Storage Stack provides a vertically and horizontally scalable solution for the data intensive storage market.

This white paper discusses three R710/MD Storage Stack configurations for vertical scaling: local single stack, networked single stack and horizontally scaled, networked dual stacks, leveraging OrangeFS.

2. Local R710/MD Storage Stack

The first goal of this evaluation was to understand the local capabilities of the R710/MD Storage Stack. To accomplish this, workloads were simulated using IOzone benchmarking software, running in varying process counts to determine the maximum throughput that could be driven to a locally attached R710/MD Storage Stack.

2.1. Configuration and Methodology

The local storage configuration for this paper leveraged a single Dell PowerVault MD3200 with four expansion MD1200 enclosures per Dell PowerEdge R710. The Dell PowerVault MD3200 and MD1200 enclosures contained twelve 3.5-inch 2TB 7200rpm NearLine SAS disks each for a total of 60 drives.

Prior to selecting a configuration, a performance comparison between a 4+1 Raid 5 configuration and a 4+2 Raid 6 configuration yielded slightly better read results and similar write results. Raid 5 was chosen primarily to maximize the number of virtual disks that could be used for the testing. In many production configurations, and with increasing disk sizes, a Raid 6 configuration may be better suited for a specific system.

The following configuration steps were performed:

1) Virtual disks were created using RAID 5, with 4 data disks and 1 parity disk (4+1) to provide maximum LUN capacity for the testing. The 4+1 RAID 5 configuration also provided a power-of-two configuration, allowing for cleaner alignment of data.

2) A segment size of 256KB was chosen for the virtual disks. This value should be based on the expected application I/O profile for the workload. The 256KB size was determined after testing several different segment sizes. When combined with the 4+1 RAID configuration, it yielded an optimal I/O size per LUN of 1 MB.

3) Cache block size of the array was set to 32KB to maximize performance. This value should be set based on the expected application I/O profile for the cluster. The maximum value is 32KB, and the minimum value is 4KB.

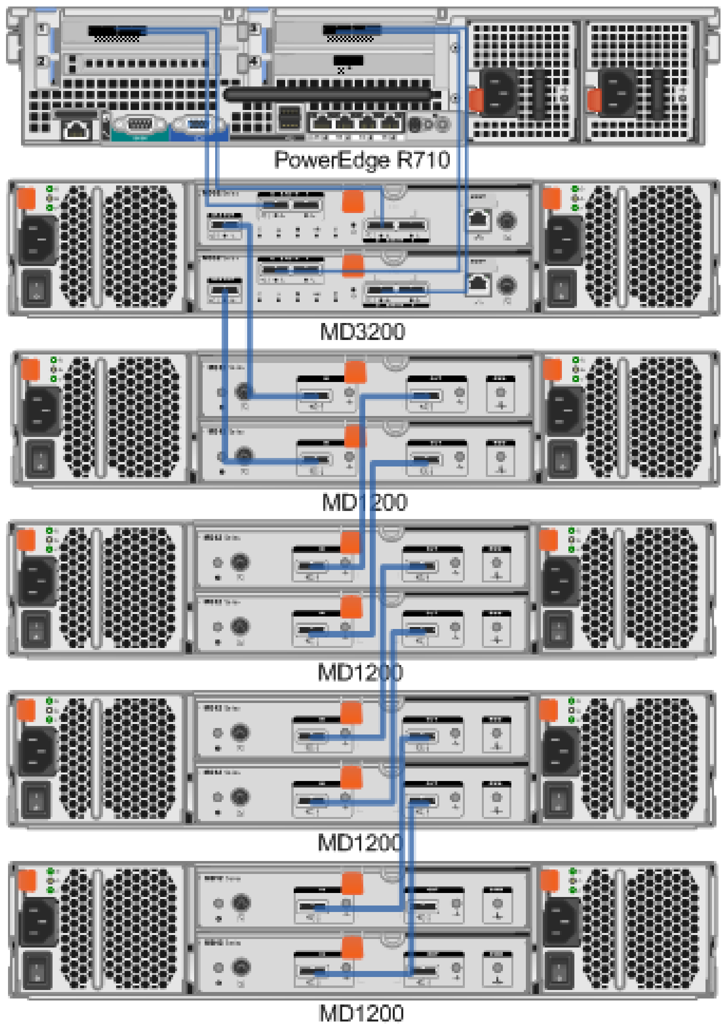
4) The RAID controllers' read and write caches were enabled to maximize performance (by allowing the controller to prefetch data into the caches based on data access patterns). Dynamic cache read prefetch was also enabled.

5) Write cache mirroring was disabled between the two PowerVault MD3200 RAID controllers. Cache mirroring between the controllers ensures that the second controller can complete the writes to disk given a failure of the owning controller. This design choice therefore eliminates a write performance penalty. Cache mirroring has little to no impact on read performance.

6) Background media scan and pre-read consistency check were disabled.

7) Initially, the R710 was connected to the MD3200 via one dual-ported PERC SAS/6 HBA. When early testing revealed a performance ceiling, a second dual-ported PERC SAS/6 HBA was added. Still, after adding a second card and cross-connecting the HBA with the MD3200 controllers, the benchmarks were not reflecting what was considered optimal performance. The units were recabled without the cross-connection between the HBA and the MD3200 per the Dell “MD3200 High Performance Tier Implementation” document. Figure 3 shows the final SAS cable configuration used for this paper.

Figure 3 - SAS Wiring Diagram for the R710/MD Storage Stack



2.2. *Software Configuration*

Scientific Linux - A Linux release put together by Fermilab, CERN, and various other labs and universities around the world. It exists primarily to reduce duplicated efforts and maintain a common install base among its scientific community of users.

LVM - A logical volume manager for the Linux kernel; it manages disk drives and similar mass-storage devices. The term “volume” refers to a disk drive or partition thereof.

XFS - A high-performance journaling file system created by Silicon Graphics, originally for their IRIX operating system and later ported to the Linux kernel. XFS is particularly proficient at handling large files and at offering smooth data transfers.

The following installation and configuration steps were performed:

1) Installed SL 6.1 with Kernel 2.6.32-131.12.1 x86_64; disabled iptables and other services not needed for a private cluster environment.

2) Installed Intel/QLogic OpenFabrics Enterprise Distribution (OFED) + 6.1.0.0.72.

3) Configured LVM across the LUNs sequentially for each individual test.

4) Configured XFS (using the Red Hat Scalable File System distribution) 3.1.1-4 across the LVM-spanned LUNs for each individual test.

   - Created the LVM physical volumes for all 12 LUNs:

     ```
     pvcreate /dev/mapper/<dev_list>
     ```

   - Created the LVM volume group:

     ```
     vgcreate --physicalextentsize 512M <vgname> <dev_list>
     ```

   - Created the LVM logical volume across the 12 LUNs:

     ```
      lvcreate --stripesize 1024 --stripes 12 --extents 100%VG
         --name <lvname> <vgname>
     ```

   - Created the XFS filesystem:

     ```
     mkfs.xfs <lvname>
     ```

5) The following Linux I/O tunables were changed to provide increased performance.

   - Modified the block devices (4+1 LUNs, MD3200 LUNs) and associated multipath mapper devices:

     ```
     echo 1024 > /sys/block/<dev>/queue/max_sectors_kb
     echo 512 > /sys/block/<dev>/queue/nr_requests
     ```

   - Modified the LVM device:

     ```
     echo 1024 > /sys/block/<dev>/queue/max_sectors_kb
     ```

- Modified all block devices to use the deadline I/O scheduler:

  ```
  echo deadline > /sys/block/<device>/queue/scheduler
  ```

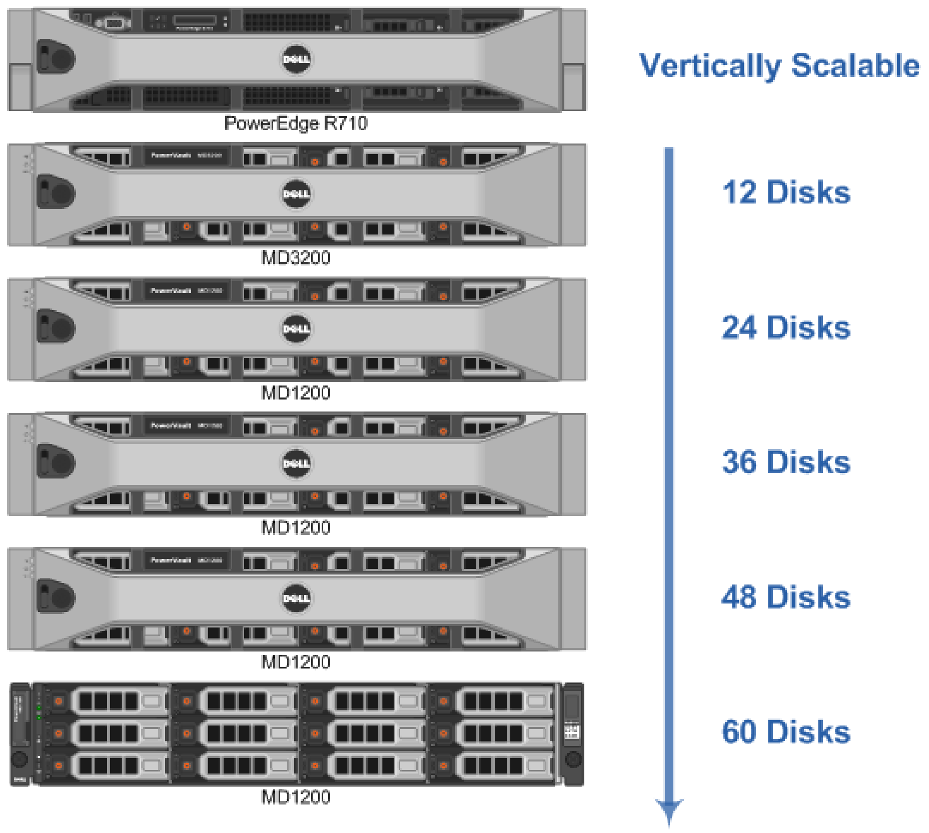
  Note: I/O scheduler attempts to minimize I/O latency by reordering requests by block number to improve performance. In an NFS environment, the algorithm that deadline uses is better suited for HPC-type workloads. Both cfq and deadline I/O schedulers were tested and the deadline scheduler performed better.

6) Mounted the file system on the client host using the `nobarrier,noatime` mount parameters.

2.3. Vertical Scalability Evaluation

Figure 4 illustrates the local testing methodology that was used to determine the base level vertical scalability of the R710/MD Storage Stack.

Figure 4 - Vertically Scaled Local R710/MD Storage Stack



2.4. Process Number Evaluation

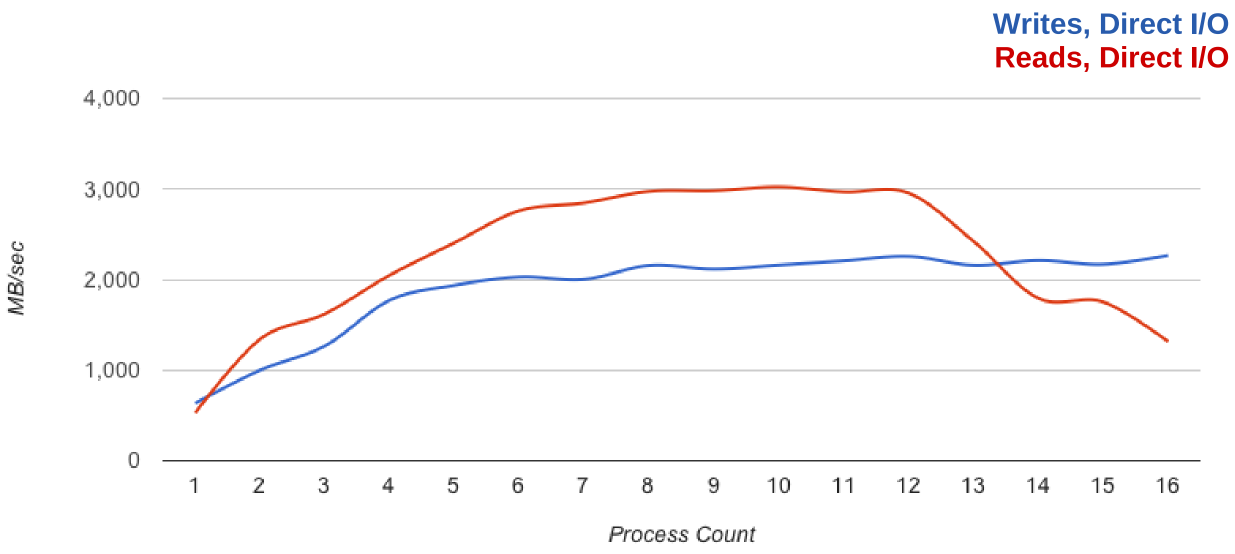
In preparation for the Disk and LUN scaling benchmarks, the optimal process count for the evaluation had to be determined. IOzone, with the Direct I/O flag enabled, was run against the local XFS file system built across all 12 LUNs (60 disks). The tests were performed against increasing process counts from 1 to 16. Figure 5 shows how writes continued to scale beyond 16 processes, reaching 2,259 MB/s. Adding disks achieves higher throughput. Reads reached saturation at 3,014 MB/s with 10 processes.

The following commands were used to generate these results:

```
iozone -i 0 -c -e -w -I -r 1024k -s 24g -t 1
iozone -i 1 -c -e -w -I -r 1024k -s 24g -t 1
```

Note: The `-t` parameter was incremented to increase the number of process.

Figure 5 - Local Performance, Increasing Processes



2.5. Disk and LUN Scaling Evaluation

Based on the results shown in Figure 6, the optimal number of processes to run against the local XFS file system would be six. This number was less than either saturation point and would be effective in demonstrating how adding disks and LUNs affects scaling. For the testing, the direct I/O flag was enabled and the I/O (chunk size) was set to 1MB.

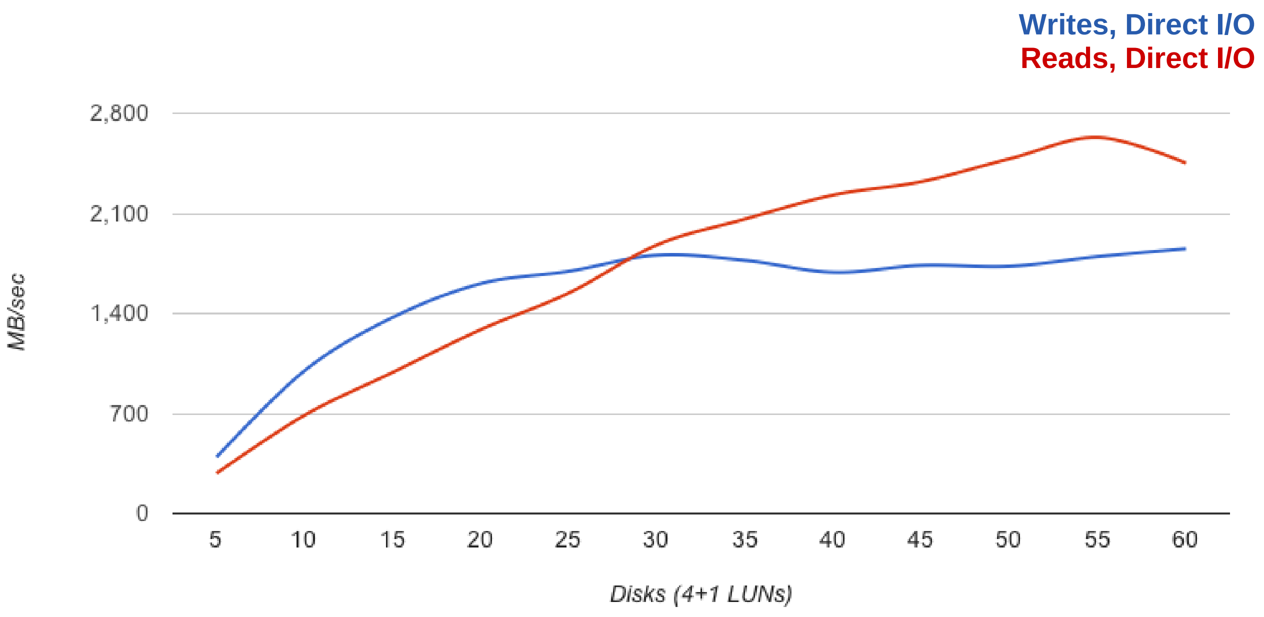
The benchmark began with one RAID 5 (4+1) LUN (5 disks), adding one LUN at a time until 12 LUNs (60 disks) were reached. Each LUN was being added to the XFS/LVM configuration.

The resulting R710/MD Storage Stack scalability is shown in Figure 6.

The folllowing commands were used to generate these results:

```
iozone -i 0 -c -e -w -I -r 1024k -s 4g -t 6
iozone -i 1 -c -e -w -I -r 1024k -s 4g -t 6
```

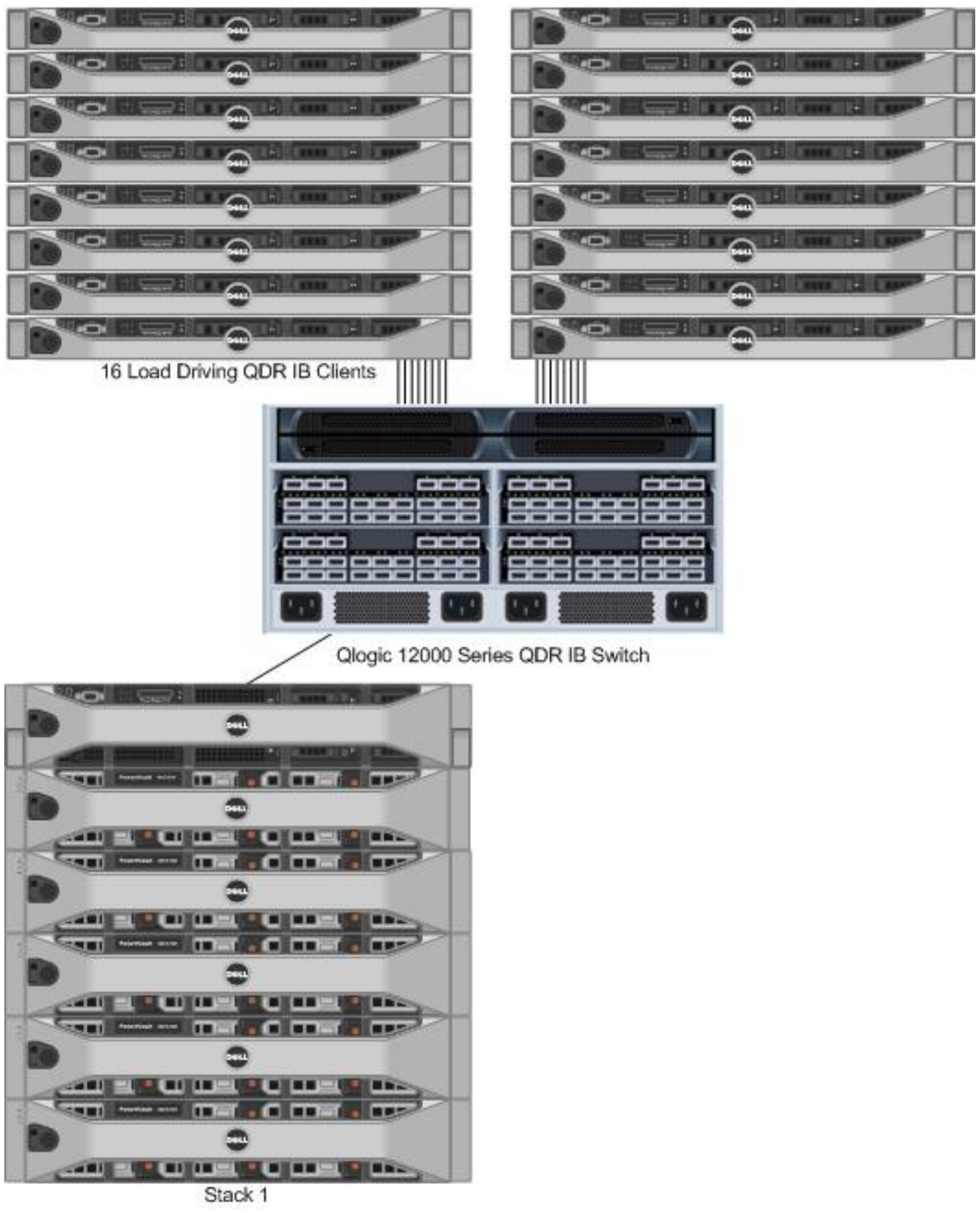
Figure 6 - Local Performance, Increasing LUNs, 6 Processes



3. Networked R710/MD Storage Stack

For the next testing scenario, essentially the R710/MD Storage Stack that had been prepared for local testing was attached to an InfiniBand network as a single node. Two network storage systems, NFS and OrangeFS, were applied to further evaluate vertical scalability. As shown in Figure 7, both cases were tested with 16 client nodes running IOzone to simulate workloads.

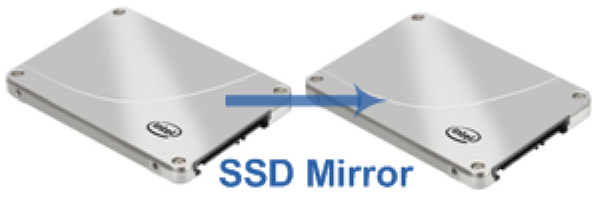
Figure 7 - Single R710/MD Storage Stack, Networked to 16 Client Nodes


16 Load Driving QDR IB Clients
Qlogic 12000 Series QDR IB Switch
Stack 1

3.1. Additional Configuration and Methodology

Mirrored SSDs - In addition to the local R710/MD Storage Stack configuration, mirrored Solid State Disks (SSDs) were added to accommodate the use of metadata by OrangeFS (see Figure 8).

Figure 8 - Mirrored SSDs for OrangeFS Metadata



InfiniBand Network – To facilitate the network, the Intel/QLogic® 12000 Series InfiniBand QDR Switch was used to connect the Intel/QLogic PCI Express Adapters installed in the R710/MD Storage Stack and 16 load driving clients.

The following configuration steps were performed:

1) The following Linux I/O tunables were changed to provide increased performance. Modified the block devices (SSD) and associated multipath mapper devices:

```
echo 8 > /sys/block/<dev>/queue/max_sectors_kb
echo 128 > /sys/block/<dev>/queue/nr_requests
```

2) The Solid State Drives (SSDs), installed in the R710, were configured with ext4:

```
mkfs.ext4 -m 0 -E stride=2,stripe-width=2 <device>
mount options nobarrier, noatime
```

To produce a simulated workload, IOzone was run concurrently on up to 16 client nodes over both NFS and OrangeFS. NFS leveraged IP over IB and OrangeFS leveraged native IB.

3.2. Additional Software Configuration

This phase of the evaluation called for a network storage system to be added to the base local R710/MD Storage Stack configuration with XFS over LVM volumes. The two network storage systems tested were NFS and OrangeFS.

NFS Overview
Network File System (NFS) is a network file system protocol originally developed by Sun Microsystems in 1984, allowing a user on a client computer to access files over a network in a manner similar to how local storage is accessed. NFS, like many other protocols, builds on the Open Network Computing Remote Procedure Call (ONC RPC) system. The Network File System is an open standard defined in RFCs, allowing anyone to implement the protocol. Following is a discussion of the modifications made to the standard NFS configuration:

Server Side
In `/etc/sysconfig/nfs`, the following setting was entered:

```
RPCNFSDCOUNT=64
```

This increase in the number of NFSD processes (default setting is 8) was made to accommodate the larger numbers of client requests.

In `/etc/exportfs`, the `async` option was enabled for the exported file system:

```
/media/lun0 *(rw,async,no_root_squash)
```

This setting exports the storage for use by NFS clients. While using the `async` option increases the risk of data loss in the event of a system failure, the overall performance is increased because the server tells the client that the data is in the storage even if it is still in cache. With customers for whom performance is not as important as data integrity, the default mount option of `sync` is recommended.

Client Side
In `/etc/fstab`, the following statement was entered:

```
server:/path /local_mount nfs defaults, noac, rw, noatime,
  rsize=1048576, hard, intr 0 0
```

This setting increases the read and write buffer sizes to the maximum of 1MB, ensuring optimal read/write performance.

OrangeFS Overview
OrangeFS is a next-generation parallel file system based on PVFS for compute and storage clusters of the future. Its original charter—to complement high-performance computing for cutting edge research in academic and government initiatives--is fast expanding into a versatile array of real-world applications.

The general class of distributed parallel file systems ensures scalable, virtually unlimited growth. Simply take the highest performing single storage system at your site and add more of them. Clients now seamlessly write data across all of them.

In the case of OrangeFS, thanks to its unique architecture, the performance of additional systems translates to proportional increases in capacity and throughput. OrangeFS clients also leverage the existence of a single to multiple storage servers in the file system. In fact, the client can decide the order and number of storage servers to use when accessing a file. This allows the client to stripe a single file across multiple storage servers. The metadata of OrangeFS can also be stored on a single storage server in conjunction with the file data or spread across multiple storage servers (configurable on system, directory or file basis).

When compared to similar parallel files systems, OrangeFS offers two major benefits:

- Based on the powerful and modular PVFS architecture, OrangeFS potential for performance enhancement has always been a fluid, ongoing process. This has allowed an ever-evolving design to incorporate distributed directories, optimized requests, a wide variety of interfaces and features. *It is well designed*.

- It is an extremely easy file system to get, build, install and keep running. PVFS has been used in numerous educational, experimental and research settings and has formed the basis of many graduate theses. *It is very usable*.

Primary targeted environments for OrangeFS include:

- High performance computing in all scales
- Rendering farms
- Financial analytics firms
- Oil and gas industry applications

OrangeFS builds using autoconf, make, and gcc from GNU. Most of the code will build and run on any Linux-based system. An experimental FUSE module is included. OrangeFS can be built for a regular user in virtually any location and tested on one or more machines. Access to the “root” account is required to use the VFS interface. The file system can be operated without the VFS module, but most users will want to install it.

For many years PVFS development focused primarily on a few large scientific workloads. At the same time members of the community used PVFS as a research tool to experiment with different aspects of parallel file system design and implementation. OrangeFS is broadening that scope to include production quality service for a wider range of data intensive application areas. This has led to re-evaluating a number of assumptions that were valid for PVFS but may, or may not, be appropriate for these other workloads. Thus a new generation of development is under way to address these new scenarios.

OrangeFS design features include:

- Object-based file data transfer, allowing clients to work on objects without the need to handle underlying storage details, such as data blocks
- Unified data/metadata servers
- Distribution of metadata across storage servers
- Distribution of directory entries
- Posix, MPI, Linux VFS, FUSE, Windows, WebDAV, S3 and REST interfaces
- Ability to configure storage parameters by directory or file, including stripe size, number of servers, replication, security
- Virtualized storage over any Linux file system as underlying local storage on each connected server

OrangeFS has been hardened through several years of development, testing, and support by a professional development team. Now it is being deployed for a range of applications with commercial support, while still maintaining complete integrity as an open source project. There are no commercial or “pro” versions, every new development is returned to the community, which is still encouraged to participate in development and supported at orangefs.org and in beowulf mailing lists.

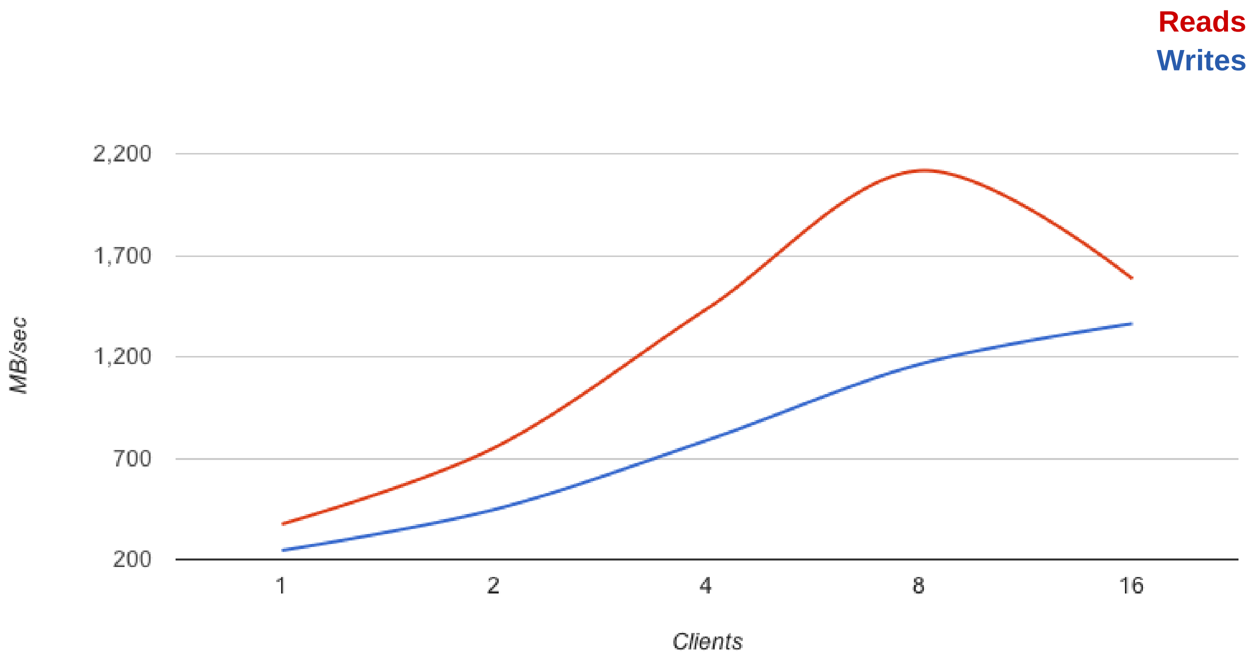
Cache Effects
To reduce the effects of cache during the testing, the following options were enabled:

- Where applicable, the –l flag was used with IOzone, which causes local cache to be bypassed.
- For OrangeFS, which currently has no client-side cache, each process writes 24GB of data, so as the number of processes and clients increases the impact of cache becomes minimal.
- For all tests, the client and server file systems were unmounted and remounted between write/read passes. This eliminated potential for cache hits during read tests.

3.3. Networked R710/MD Storage Stack Evaluation

In Figure 9, the NFS configuration of the R710/MD Storage Stack hardware (described in section 3.2) produced NFS read/write performance of nearly 2,200 MB/s and 1,400 MB/s, respectively.

Figure 9 - NFS Performance, Increasing Clients, 1 Process



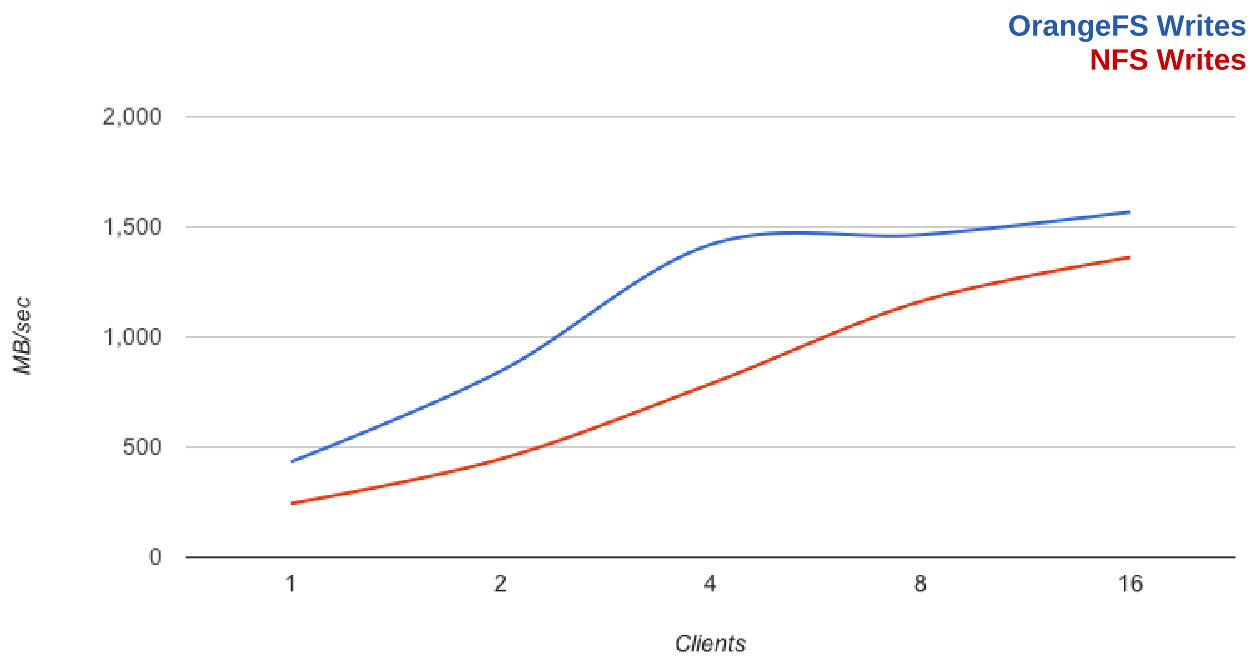
The following commands were used to generate results shown in figures 9, 10, 11, 13 and 14:

```
iozone -i 0 -c -e -w -r 1024k -s 24g -t client_count -+n
  -+m client_list
iozone -i 1 -c -e -w -r 1024k -s 24g -t client_count -+n
  -+m client_list
```

Note: The `client_count` parameter value corresponds to the x axis in each of the figures.

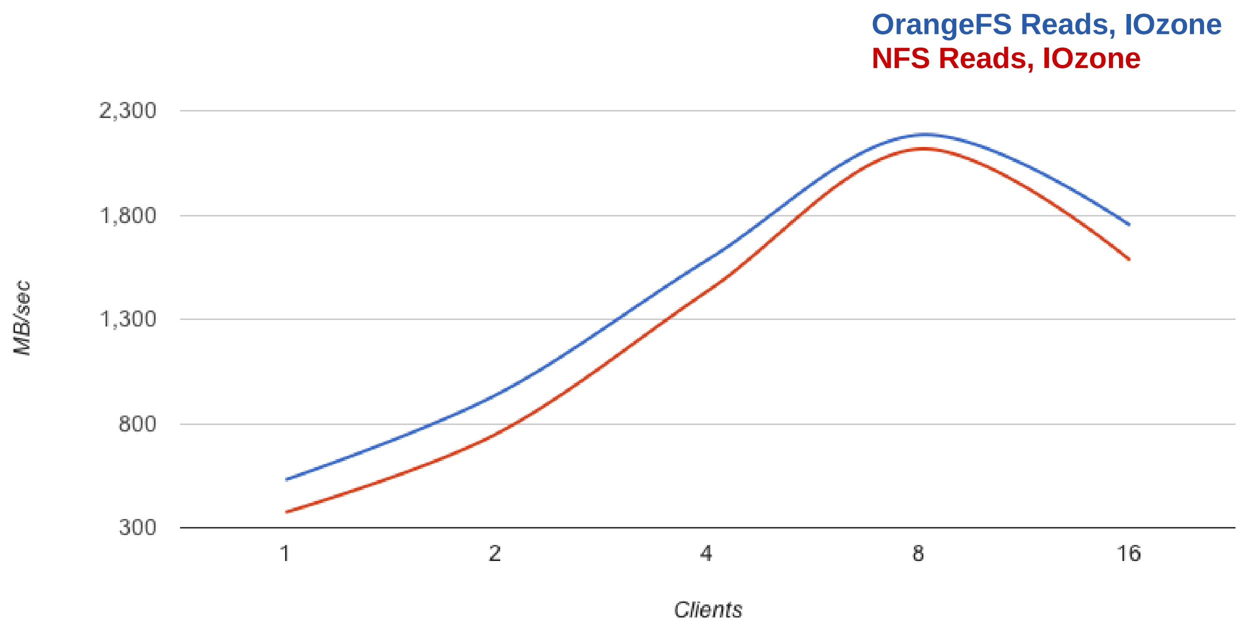
In Figure 10, the OrangeFS results are compared with the NFS results. In a single R710/MD Storage Stack configuration, OrangeFS demonstrates the ability to write 1MB I/O chunks at 1,565 MB/s compared to NFS at 1,359 MB/s driven by 16 clients. These results demonstrate that both NFS and OrangeFS are solid R710/MD Storage Stack software configuration options.

Figure 10 - NFS Performance, Compared to OrangeFS Performance



In Figure 11, OrangeFS results are compared with NFS results. In a single R710/MD Storage Stack configuration, OrangeFS demonstrates the ability to read 1MB I/O chunks at 2,181 MB/s compared to NFS at 2,113 MB/s, peaking at 8 clients.

Figure 11 - NFS Performance, Compared to OrangeFS Peformance



NFS and OrangeFS are good software alternatives for the networked R710/MD Storage Stack configuration, both providing excellent read and write throughput performance.

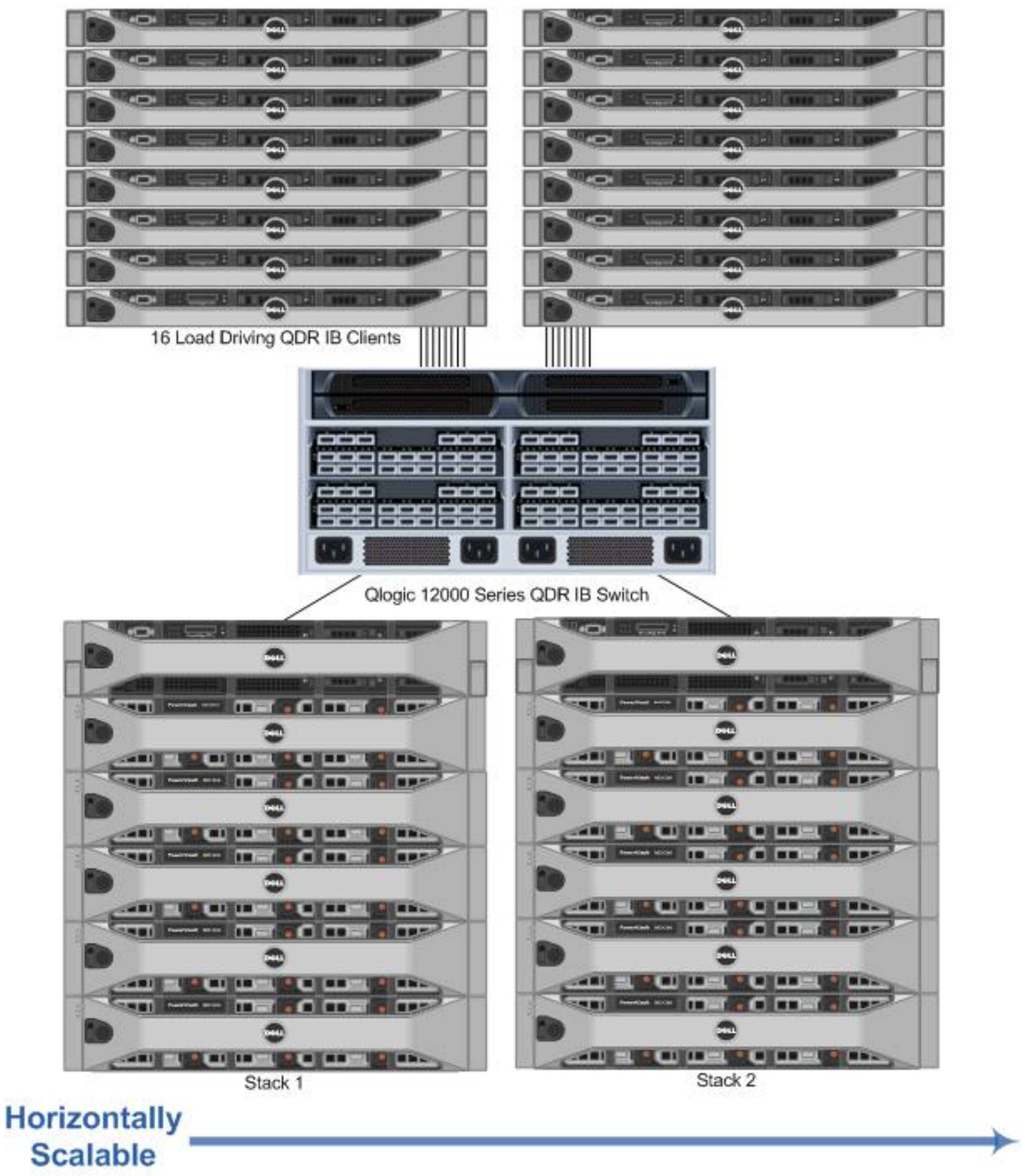
Running the reference IOzone benchmarks and comparing OrangeFS to NFS shows that OrangeFS has slightly better performance than NFS for sequential workloads.

In addition to good single R710/MD Storage Stack performance, OrangeFS brings the capability to scale across multiple stacks, providing horizontal scalability across a single name space and distributed metadata.

4. R710/MD Storage Stack Horizontal Scaling with OrangeFS

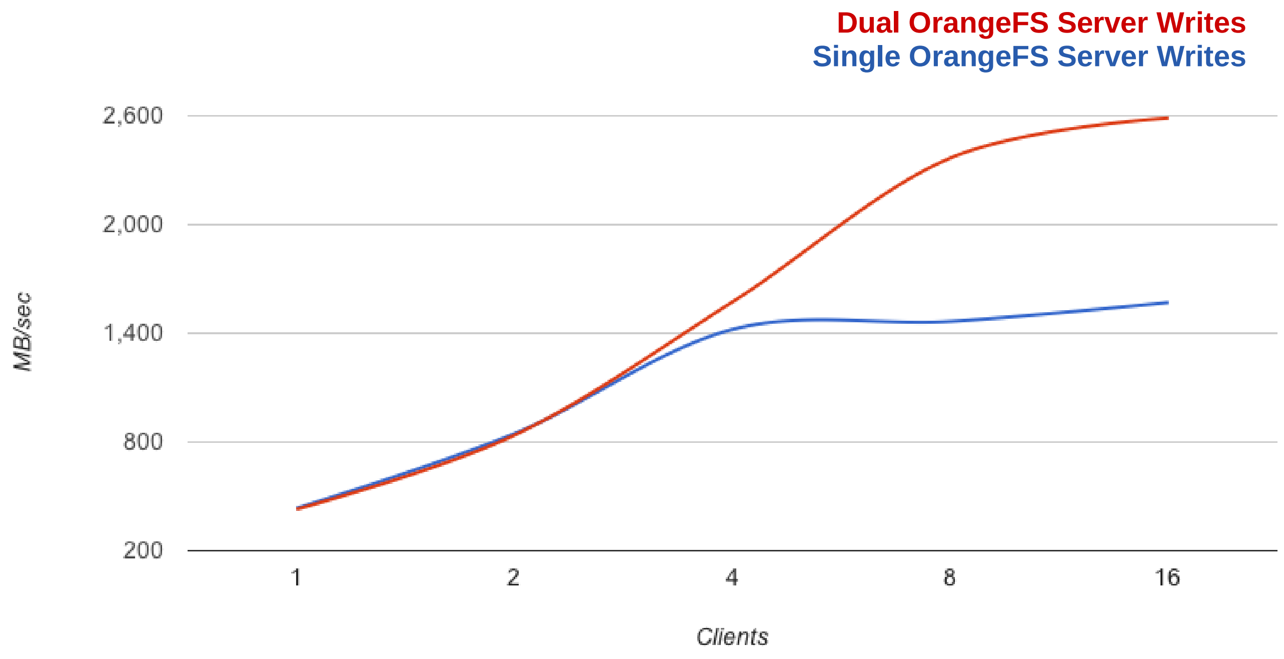
For the OrangeFS horizontal scaling evaluation, a second R710/MD Storage Stack was added, as illustrated in Figure 12. The OrangeFS file system was distributed across both stacks, providing for a unified name space. Each stack also was configured to be a metadata server, supporting the distributed metadata capability of OrangeFS.

Figure 12 - Horizontal Configuration


16 Load Driving QDR IB Clients
Qlogic 12000 Series QDR IB Switch
Stack 1
Stack 2
Horizontally Scalable

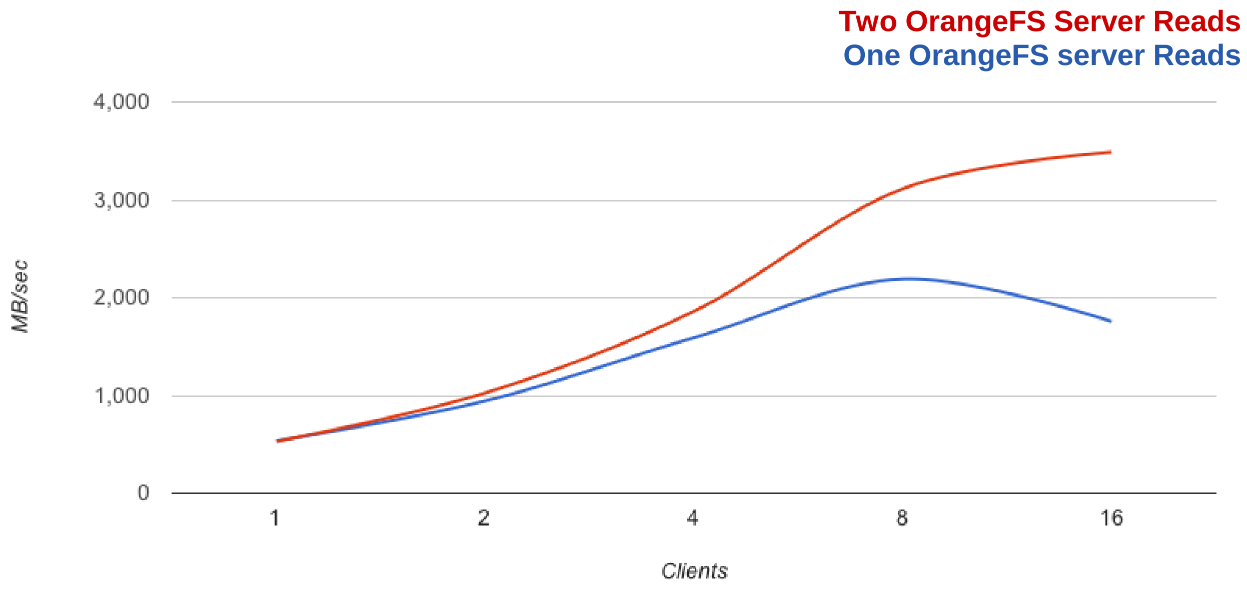
As Figure 13 shows, the dual R710/MD Storage Stack configuration scales well in comparison to a single stack configuration. The single OrangeFS stack configuration with 16 processes writes at 1,565 MB/s as compared to the dual OrangeFS configuration at 2,583 MB/s.

Figure 13 - OrangeFS Write Performance, Single vs. Dual Stacks



In Figure 14, the single OrangeFS R710/MD Storage Stack configuration with 16 processes demonstrates reads at 1,747 MB/s as compared to the dual OrangeFS configuration, which produced 3,480 MB/s.

Figure 14 - OrangeFS Read Performance, Single vs. Dual Stacks



When a second OrangeFS R710/MD Storage Stack was added, streaming throughput continued to scale beyond the single stack configuration. Limited by the objectives of this white paper, the client configuration and performance tests do not reflect the maximum throughput that could have been achieved. OrangeFS can also facilitate horizontal scaling to hundreds of R710/MD Storage Stacks to meet your storage requirements. This scaling includes the ability to distribute metadata across some or all stacks, eliminating many of the server hot spots associated with centralized metadata in other parallel file systems.

5. Conclusion

In the evolving world of high performance computing and data-intensive applications, managing massive amounts of data efficiently and effectively is becoming a major focus. Many different kinds of workloads are pushing storage demands, including scientific applications such as the Large Hadron Collider (LHC), render farms, financial modeling, oil and gas simulation and genomics applications.

The Dell R710/MD Storage Stack is an excellent choice to meet these demanding applications for both vertical and horizontal scaling. Ease of use and the ability to start with a base unit and grow as needed make this an excellent choice for the most demanding storage needs.

The vertical scalability of the Dell R710/MD Storage Stack can additionally be scaled horizontally with the capabilities of OrangeFS. OrangeFS simplifies the issues with horizontal scalability and allows customer to use a free and open source software stack to take advantage of multiple R710/MD Storage Stacks. These capabilities make it possible for customers to start small to meet their current needs, then simply scale vertically and/or horizontally as space or I/O requirements dictate. OrangeFS is already able to exceed streaming throughput results as compared to NFS on a single R710/MD Storage Stack, so it can be leveraged as an alternative to NFS-based starter storage solutions.

Appendix A: References

High Performance Tier Implementation Guideline , PowerVaultTM MD3200 and MD3200i Storage Arrays, (page 7), http://www.dell.com/downloads/global/products/pvaul/en/powervault-md3200-high-performance-tier-implementation.pdf .

IOzone Filesystem Benchmark, http://www.iozone.org .

OrangeFS Open Source Project, http://www.orangefs.org.

Omnibond Commercial Services for OrangeFS, http://www.orangefs.com.

Appendix B: R710/MD Storage Stack Test Configuration Hardware Details

Table 1 – R710/MD Storage Stack Hardware Configuration Details

HARDWARE	
I/O Server Model	PowerEdge R710
Processor	Dual Intel Xeon E5620 2.4Ghz, 12M Cache,Turbo, HT, 1066MHz Max Mem
Memory	24GB Memory (6x4GB), 1333MHz Dual Ranked RDIMMs for 2 Processors
Local disks and RAID controller	PERC H700 Integrated RAID Controller, 512MB Cache, x6
InfiniBand slot	4
Internal Networking	one 1GbE Connector
External storage controller (slot 1 & 3)	Two dual ported 6Gbps SAS HBAs
Systems Management	iDRAC Enterprise
Power Supply	Dual PSUs
STORAGE CONFIGURATION	
Storage Enclosure	PowerVault MD3200, 4 MD1200 expansion arrays
RAID controllers	Duplex RAID controllers in the MD3200
Hard Disk Drives	Segate 2TB 7200 rpm NL SAS drives
InfiniBand Switch	Intel/QLogic 12300 QDR 36 port

Table 2 – Software Configuration Details

SOFTWARE	
Operating System	Scientific Linux 6.1
Kernel version	2.6.32-131.12.1 x86_64
File system	XFS (Red Hat Scalable File System) 3.1.1-4
Systems Management	Dell OpenManage Server Administrator 6.5.0
Storage Management	Dell Modular Disk Storage Manager 10.80.G6.47

Table 3 – Firmware and Driver Configuration Details

FIRMWARE AND DRIVERS	
PowerEdge R710 BIOS	3.0.0
PowerEdge R710 iDRAC	1.7
InfiniBand switch firmware	6.1.0.0.72, Apr 5 2011
InfiniBand driver	QLogic OFED+ 6.1.0.0.72
PERC H700 firmware	12.10.0-0025
PERC H700 driver	megaraid_sas 00.00.05.34-rc1
6Gbps SAS firmware	7.02.42.00
6Gbps SAS driver	mpt2sas 08.101.00.00
MD3200	07.80.41.60
NVSRAM	N26X0-780890-001
MD1200	1.01
Segate 2TB 7200 rpm NL SAS drives	KS68 (as recognized by Dell PowerVault Modular Disk Storage Manager)

Table 4 – Client Configuration Details

CLIENTS / HPC COMPUTE CLUSTER	
Clients	16 Scientific Linux based compute nodes
InfiniBand	Intel/QLogic QLE7340 QDR HCAs Intel/QLogic OFED+ 6.0.2.1.11
InfiniBand fabric	All clients connected to a single large port count InfiniBand switch. Both R710 NFS servers also connected to the InfiniBand switch.

Appendix C: OrangeFS Server Configuration Files

OrangeFS includes a common configuration file that is referenced by each server in a given installation. Following are some of the key settings captured in this file, which is created during the build process and copied to each server in the file system deployment:

- The protocol used by your network.
- The number of the port through which all servers will communicate.
- The directories where each OrangeFS server will store its data, metadata and log messages.
- The hostname of each OrangeFS server.

Table 5 compares the two OrangeFS server configuration files used in this study. The one on the left reflects the single networked R710/MD Storage Stack solution and the one on the right is for the dual networked stack solution.

Table 5 – OrangeFS Server Configuraiton Files

ORANGEFS SERVER CONFIGURATION FILES

SINGLE R710/MD Storage Stack

```
<Defaults>
        UnexpectedRequests 256
        EventLogging none
        EnableTracing no
        LogStamp datetime
        BMIModules bmi_ib
        FlowModules flowproto_multiqueue
        PerfUpdateInterval 10000000
        ServerJobBMITimeoutSecs 30
        ServerJobFlowTimeoutSecs 60
        ClientJobBMITimeoutSecs 30
        ClientJobFlowTimeoutSecs 60
        ClientRetryLimit 5
        ClientRetryDelayMilliSecs 100
        TroveMaxConcurrentIO 16
</Defaults>

<Aliases>
        Alias r710-3 ib://r710-3:3334
</Aliases>

<ServerOptions>
        Server r710-3
        DataStorageSpace /media/lun0
        MetadataStorageSpace /media/lun1
        LogFile /root/tests/ofs-server.log
</ServerOptions>

<Filesystem>
        Name test-fs
        ID 180520100
        RootHandle 1048588
        FileStuffing yes
          DefaultNumDFiles 1
        FlowBufferSizeBytes 1048576

          <StorageHints>
            TroveSyncMeta yes
            TroveSyncData no
```

DUAL R710/MD Storage Stack

```
<Defaults>
        UnexpectedRequests 256
        EventLogging none
        EnableTracing no
        LogStamp datetime
        BMIModules bmi_ib
        FlowModules flowproto_multiqueue
        PerfUpdateInterval 10000000
        ServerJobBMITimeoutSecs 30
        ServerJobFlowTimeoutSecs 60
        ClientJobBMITimeoutSecs 30
        ClientJobFlowTimeoutSecs 60
        ClientRetryLimit 5
        ClientRetryDelayMilliSecs 100
        TroveMaxConcurrentIO 16
</Defaults>

<Aliases>
        Alias r710-2 ib://r710-2:3334
        Alias r710-3 ib://r710-3:3334
</Aliases>

<ServerOptions>
        Server r710-2
        DataStorageSpace /media/lun0
        MetadataStorageSpace /media/lun1
        LogFile /root/tests/ofs-server.log
</ServerOptions>

<ServerOptions>
        Server r710-3
        DataStorageSpace /media/lun0
        MetadataStorageSpace /media/lun1
        LogFile /root/tests/ofs-server.log
</ServerOptions>

<Filesystem>
        Name test-fs
        ID 180520100
```

```
            TroveMethod alt-aio
              DBCacheSizeBytes 4294967296
        </StorageHints>

        <Distribution>
              Name simple_stripe
            Param strip_size
            Value 1048576
        </Distribution>

        <MetaHandleRanges>
            Range r710-3 3-1152921504606846986
        </MetaHandleRanges>

        <DataHandleRanges>
              Range r710-3 2305843009213693971-
                              3458764513820540954
        </DataHandleRanges>

</Filesystem
```

```
        RootHandle 1048588
        FileStuffing yes
          DefaultNumDFiles 2
        FlowBufferSizeBytes 1048576

        <StorageHints>
              TroveSyncMeta yes
              TroveSyncData no
              TroveMethod alt-aio
                DBCacheSizeBytes 4294967296
        </StorageHints>

        <Distribution>
              Name simple_stripe
              Param strip_size
              Value 1048576
        </Distribution>

        <MetaHandleRanges>
              Range r710-2 3-1152921504606846986
              Range r710-3 1152921504606846987-
                                   2305843009213693970
        </MetaHandleRanges>

        <DataHandleRanges>
                Range r710-2 2305843009213693971-
                                       3458764513820540954
              Range r710-3 3458764513820540955-
                                       4611686018427387938
        </DataHandleRanges>

</Filesystem>
```

Appendix C: OrangeFS Build and Installation

OrangeFS was installed on the systems discussed in this white paper as follows:

1) Retrieve the latest OrangeFS release:

```
wget "http://www.orangefs.org/download/orangefs-2.8.5.tar.gz"
```

2) Expand the archive:

```
tar -xzf orangefs-2.8.5.tar.gz
```

3) Change into the working directory:

```
cd orangefs
```

4) Run the configuration scripts:

```
./configure --prefix=/opt/orangefs
        --with-kernel=/usr/src/kernels/2.6.32-131.12.1.el6.x86_64/
        --with-db=/opt/db4 --enable-threaded-kmod-helper
        --with-openib=/usr --without-bmi-tcp
```

5) Build the software:

```
make
```

6) Build the kernel module and helper programs:

```
make kmod
```

7) Install the software to the prefix location:

```
make install
```

8) Install the kernel module and helper programs to the prefix location:

```
make kmod_prefix=/opt/orangefs kmod_install
```

9) Copy installation directory (/opt/orangefs) to all server and client hosts.

10) On servers, place the configuration file on each host in

```
/opt/orangefs/etc/orangefs-server.conf
```

Note: Steps needed to configure and create file systems, as detailed earlier in this paper, must be completed before proceeding.

11) Initialize storage space on each server:

```
/opt/orangefs/sbin/pvfs2-server –f
/opt/orangefs/etc/orangefs-server.conf
```

12) Start process on each server

```
/opt/orangefs/sbin/pvfs2-server
/opt/orangefs/etc/orangefs-server.conf
```

13) On each client insert the kernel module:

```
/sbin/insmod
/opt/orangefs/lib/modules/2.6.32-131.12.1.el6.x86_64
     /kernel/fs/pvfs2/pvfs2.ko
```

14) On each client start the user space daemon:

```
/opt/orangefs/sbin/pvfs2-client
```

15) Create a directory for the OrangeFS mount point:

```
mkdir /mnt/orangefs
```

16) Mount the OrangeFS file system on each client:

```
mount -t pvfs2 ib://<hostname>:3334/test-fs /mnt/orangefs
```