

Maximizing NFS Scalability

on Dell Servers and Storage in High-Performance Computing Environments

Popular because of its maturity and ease of use, the Network File System (NFS) can be used in high-performance computing cluster environments. This article discusses how administrators can improve performance and obtain maximum scalability from a Dell™ NFS server by adjusting parameters for the server and related systems, including I/O subsystems, back-end storage, NFS service, and network interconnect.

BY ANDREW BACHLER AND JENWEI HSIEH, PH.D.

Since the early 1980s, the Network File System (NFS) distributed file system has been deployed widely in many environments. NFS remains popular in today's enterprises because it is easy to use and simple to manage. One growing application for NFS is high-performance computing (HPC) clusters. This article describes how administrators can deploy a Dell NFS server in an HPC cluster environment and tune its system performance.

Scalability has long been considered the main disadvantage of NFS in HPC environments. NFS does not scale linearly because the protocol relies on a single server to process all client requests. As a result, in large deployments both the NFS server and the network path to the server can become bottlenecks. However, an NFS server can be highly effective in HPC environments depending on the type of traffic, back-end storage, NFS service configuration, and network interconnect used.

As shown in Figure 1, NFS architecture comprises several components, and each component affects overall performance. For example, the method by which the NFS server exports the file system to the clients and the type of back-end storage the server uses can significantly influence

performance. By adjusting the system components that most affect performance and following best practices, administrators can optimize the scalability of NFS servers.

Configuring the test environment

In June 2004, Dell engineers conducted a study to compare the effect of performance tuning on an experimental NFS HPC cluster configuration. The cluster comprised a Dell PowerEdge™ 1850 server for NFS file-sharing functions connected to a Dell PowerVault™ 220S storage system using a PowerEdge Expandable RAID Controller 4, Dual Channel (PERC 4/DC), and configured with 2 GB of RAM. The NFS server's operating system (OS) was Red Hat® Enterprise Linux® AS 3 using Intel® Extended Memory 64 Technology (EM64T). The Dell engineers configured 32 client nodes in the cluster; each client was a PowerEdge 1750 server running Red Hat Enterprise Linux WS 3. An Extreme Networks Black Diamond 6808 Gigabit Ethernet switch formed the network backbone.

To evaluate NFS performance in this study, Dell engineers used a parallel I/O program called *ior_mpiio*. (The name *ior_mpiio* stands for Interleaved or Random

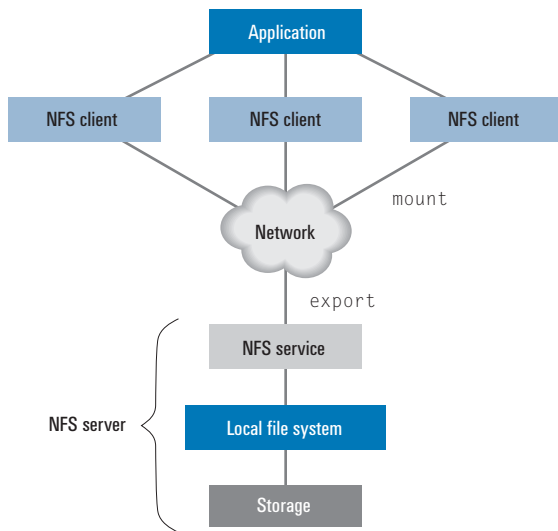


Figure 1. NFS architecture

Message Passing Interface I/O.)¹ This parallel program was developed by the Scalable I/O Project (SIOP) at Lawrence Livermore National Laboratory. It performs parallel writes and reads to and from a single file, and reports the throughput rates. The `ior_mpio` program uses Message Passing Interface (MPI) for process synchronization. The data is written and read using independent parallel transfers of equal-sized blocks of contiguous bytes that cover the file with no gaps and do not overlap each other. Unless testers request otherwise by adjusting environmental variables such as file size, record size, and file data layout, each `ior_mpio` test consists of creating a new file, writing data to that file, then reading the data back from the file. The data is written as C integers.

In conducting the tests for this article, the Dell team ran the `ior_mpio` program 10 times for each test, discarded the best and the worst results, and took the average of the remaining results. Because the NFS server was configured with 2 GB of memory, the total amount of data written to the server in all tests was 4 GB, divided across the clients in equal chunks. Each client wrote and read 1 MB of records at a time.

The test team tuned NFS service parameters for the Dell NFS server and clients, as well as the I/O subsystems, back-end storage, and network interconnect. Based on `ior_mpio` benchmark results for the experimental NFS HPC cluster configuration described in this article, Dell engineers achieved significant performance increases by tuning these parameters instead of using their standard, default settings. The following sections describe how the Dell team obtained this performance improvement by adjusting settings for the I/O subsystems, back-end storage, NFS server and client parameters, and network interconnect.

Adjusting I/O based on application access patterns and needs

The most important considerations in designing any I/O subsystem are the needs and access patterns of the application that will use the system. Tuning choices depend on specific application characteristics such as the following: whether the application is read or write intensive, what the typical file size is, whether reads and writes are sequential or random, and whether files are accessed in large or small blocks. For instance, for small-file access—especially access that involves a significant amount of metadata operations such as deleting files—the Reiser file system is most likely the best choice of local file system for the NFS server.²

Another factor that can enhance performance is the data layout. If system administrators can control how data is stored on the disk, best practices suggest avoiding parallel writes as much as possible. Minimizing parallel writes enables the NFS server to take advantage of different caching options. Consider, for instance, an application that is distributed across multiple nodes, with each node writing its own file to the NFS server in parallel and later reading those files. These parallel writes create a noncontiguous allocation of disk blocks to each file, which later results in random data distribution, and in turn, random disk access when systems attempt to read the files. In this study, the test results indicated that when files were created contiguously, through sequential writes, client reads were significantly faster compared to files that were created in parallel and required random reads. Nonparallel writes made more effective use of caching, both in the file system and on the storage controller. Note that for this study, the Dell team ran 32 instances of `ior_mpio`—one per client—to ensure that each client had its own file to write and read. Figure 2 shows that read bandwidth was more than 2.5 times faster when data had been written sequentially compared to read bandwidth when data had been written in parallel.

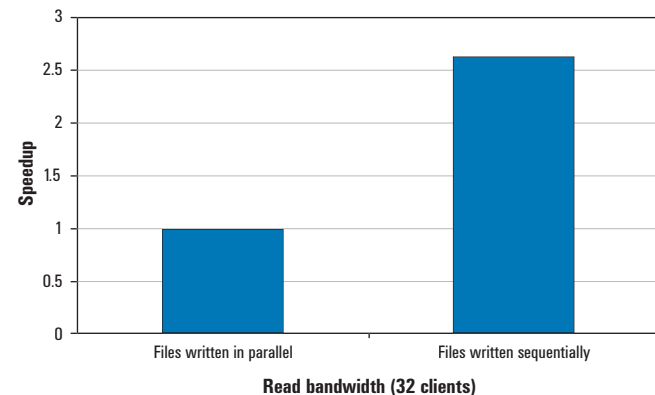


Figure 2. Effect of sequential versus parallel data layout on read bandwidth

¹For more information about `ior_mpio`, visit <http://www.llnl.gov/asci/purple/benchmarks/limited/ior/ior.mpio.readme.html>.

²For more information about the Reiser file system, visit <http://www.namesys.com>.

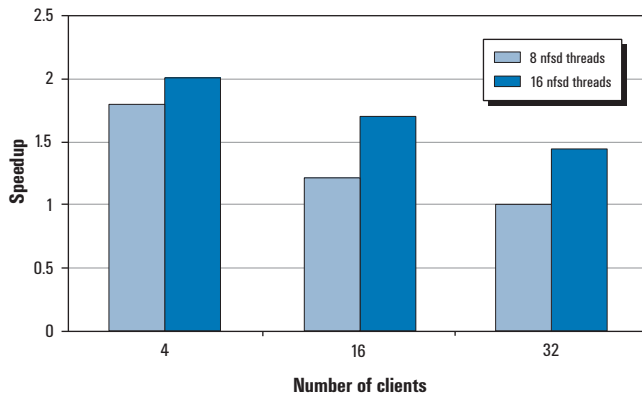


Figure 3. Effect of the number of nfsd threads on read performance

Tuning back-end storage to improve NFS server performance

Because storage is local to the NFS server, administrators can promote better overall server performance by tuning NFS storage for optimal performance. However, any adjustments to local back-end storage must take into consideration the I/O traffic patterns of the application or applications that will use the NFS server.

SCSI or Fibre Channel storage. Dell offers several SCSI and Fibre Channel storage enclosures. The PowerVault 220S is the most commonly used SCSI enclosure in combination with a PERC 4/DC. Fibre Channel products, however, are designed to be more efficient than SCSI at handling larger data blocks.

RAID level. The RAID level also influences system performance and reliability. Although RAID-0 offers outstanding throughput, RAID-5 is a common choice because it is designed to protect data if a hard drive fails by storing parity information on each disk in the logical storage unit. When feasible, administrators can create the RAID device across multiple channels of the controller to utilize available bandwidth. This approach can be particularly helpful for RAID-0 configurations because adding more spindles can help improve RAID-0 performance. Conversely, adding more spindles to a RAID-5 configuration eventually reduces performance because RAID-5 overhead is directly proportional to the number of drives (parity is read from each drive).

File system characteristics. The local file system on the NFS server can be another critical factor affecting storage performance. Every file system exhibits different characteristics and targets distinct access patterns. For example, the Reiser file system can be an excellent choice for access patterns that consist mainly of metadata operations or small-file operations. Both the Reiser and the ext3 file systems are journaling file systems that are available in standard Red Hat distributions. Journaling file systems are designed to offer an advantage over non-journaling file systems by requiring less time than non-journaling file systems to bring the server to a stable state after a crash, but the overhead incurred by journaling can reduce system performance. However, Reiser and ext3 file systems

allow administrators to store the journal information in a separate device such as a flash card—which can help improve file system performance if a very fast device is used. Despite their performance penalty, journaling file systems have become standard thanks to advances in file system design. Nonetheless, when administrators require top performance from an NFS configuration, ext2—a non-journaling file system—may be a better option.

Adjusting the NFS service to achieve greater scalability

After administrators tune the back-end storage on the server, the next step is to adjust the NFS service itself. Several important factors influence the capability of the NFS server to handle client requests such as the number of nfsd threads and the parameters assigned to the `export` command. The choice of an appropriate server is crucial as well.

Choice of server. The Dell tests in this study indicate that the PowerEdge 1850 server can be an excellent platform for NFS file serving. The PowerEdge 1850 server is an EM64T-capable platform whose 64-bit extension OS is designed to allow for better memory utilization than a 32-bit OS. Furthermore, the enterprise kernel in Red Hat Enterprise Linux AS 3—the OS installed on the PowerEdge 1850 server used in this study—is designed to enable administrators to address all the physical memory of the server. The PowerEdge 1850 server also includes a choice of riser cards based on Peripheral Component Interconnect Extended (PCI-X) or PCI Express technology.

Number of nfsd threads. Because the NFS service is a multi-threaded application on the server side, the number of nfsd threads that the server launches can affect the scalability of the NFS server. By default, the PowerEdge 1850 server launches eight nfsd threads. However, the results of the Dell study described in this article indicate that performance can be improved by increasing the number of nfsd threads as the number of clients exercising the server increases. One way to check nfsd thread usage is to observe the last few entries on the `th` line in the `/proc/net/rpc/nfsd` file. The last 10 numbers in the `th` line represent the number of seconds that the thread usage was at the maximum allowable percentage. For instance, if the ninth number is 600, the thread usage was at 90 percent for 600 seconds. High numbers in the last three or four places indicate that the system might benefit from an increased thread count. To change the thread count, administrators can modify the value of the `RPCNFSDCOUNT` variable in `/etc/init.d/nfs` and restart the NFS service. By increasing the nfsd thread count, the Dell team observed performance improvements in this study, particularly on reads. The baseline for the testing in this study was the performance of 32 clients using eight nfsd threads. Figure 3 shows that the performance improvements were more significant as the number of nodes increased, which is typical in environments such as HPC clusters. The improved bandwidth can be attributed to an increase in the number of threads. In this study, doubling the number of nfsd threads from 8 to 16 helped speed up

the read performance of incoming NFS requests from many clients.

Export command parameters. Another component of the NFS service that administrators should consider is how the file system is made available, or *exported*, to the clients. The `export` command has several parameters. The one that most affects performance is the `sync` or `async` option.

Application performance can benefit greatly from the asynchronous behavior of NFS. In this test, write performance increased by more than 50 percent when Dell engineers used the `async` option instead of the `sync` option. (See Figure 4, which shows relative performance improvement for writes using asynchronous I/O.) However, it is very important for system administrators to weigh the risks and benefits of using the `async` option, and understand that with greatly improved write throughput from `async` I/O comes a higher risk of data loss if the server crashes before committing data to the disk. Also note that the default export option differs with different versions of the NFS server's OS.³

Using client parameters to specify the size of data blocks

The most commonly tuned parameter for NFS clients is the `rsize` or `wsize` option to the `mount` command. The values of `rsize` and `wsize` specify the amount of read and write data, respectively, that is transferred in each NFS request. The limit on the values of `rsize` and `wsize` in NFS version 2 is up to 8 KB each. The values of `rsize` and `wsize` need not be the same. In version 3 of the NFS protocol, the maximum values of `rsize` and `wsize` are defined by the `NFSSVC_MAXBLKSIZE` variable in the `include/linux/nfsd/const.h` file. Increasing these values and recompiling the kernel may benefit IT departments that seek to achieve high performance over a non-lossy network.

Note: Although User Datagram Protocol (UDP) is the default protocol for NFS, it can be unreliable. For example, when large chunks of data are transferred using UDP, a single lost packet results in the retransmission of the entire request. This can make UDP an untenable option for lossy networks.

Choosing optimal system components

The choice of individual system components that affect the baseline storage performance can be very important in tuning the NFS server. For example, the PowerEdge 1850 server used in this study

The most important considerations in designing any I/O subsystem are the needs and access patterns of the application that will use the system.

provides a choice of two different riser cards based on PCI-X and PCI Express technology.

PCI Express is the new-generation I/O architecture that Intel plans to feature across all Intel platforms for workstations, desktops, and servers.⁴ PCI Express is based on a type of serial communications technology equivalent to that in the Universal Serial Bus (USB) and Serial ATA (SATA) hard drives. PCI Express accommodates board connectors that come in $\times 1$, $\times 4$, $\times 8$, $\times 16$, and $\times 32$ widths. A single lane, referred to as a $\times 1$ link (pronounced "by one link"), can help support up to 500 MB/sec in both directions (dual simplex). Lanes can be aggregated to create higher-bandwidth lanes. For example, as lane widths are increased to a $\times 32$ width, the PCI Express specification is designed to offer a potential peak bandwidth capability of up to 16 GB/sec dual simplex while remaining compatible with PCI software.

Additional PCI Express features are designed to provide the following advantages:

- Compatibility with existing PCI drivers and software as well as many operating systems
- High bandwidth per pin, low overhead, and low latency
- A point-to-point connection, which enables each device to have a dedicated connection without sharing bandwidth
- Power consumption and power management
- Hot swappability and hot pluggability for devices
- Coexistence with current PCI hardware

Dell engineers installed a PCI Express PERC 4 Express/DC (PERC 4 E/DC) card in a PowerEdge 1850 NFS server that was connected to an external storage system containing fourteen

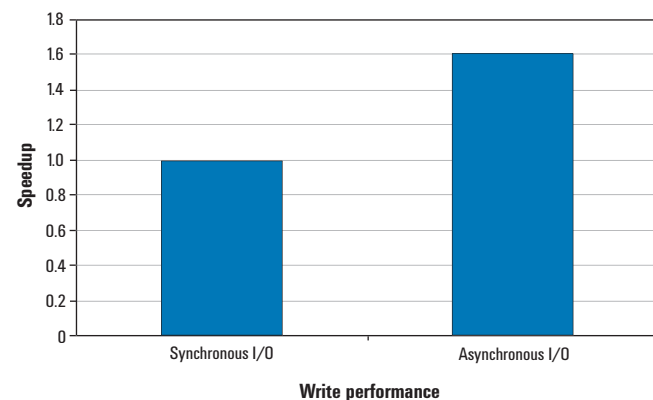


Figure 4. Write performance improvements using asynchronous I/O

³The `exportfs` program, which is responsible for exporting the file systems, is installed from the `nfs-utils` RPM™ (Red Hat Package Manager). Version 1.0.1 of the `nfs-utils` utility, installed with Red Hat Linux 9.0, exports file systems with the `sync` option by default. Earlier versions of the utility, including the one installed with Red Hat Enterprise Linux AS 2.1, default to `async` export mode.

⁴For more information about PCI Express, see "Exploring the Advantages of PCI Express" by Chris Croteau and Paul Luse in *Dell Power Solutions*, June 2004.

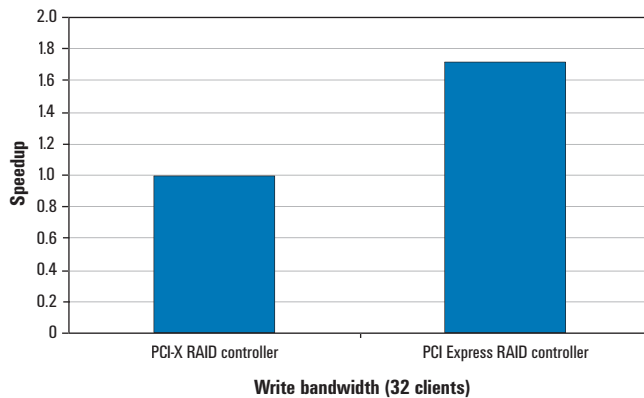


Figure 5. Effect of server's peripheral bus and RAID controller on NFS performance

146 GB SCSI drives in a RAID-0 configuration. Compared to the PERC 4/DC option, which was tested in the same RAID-0 configuration for this study, the PERC 4 E/DC configuration provided a 60 percent to 70 percent increase in NFS performance when a server utilized the PCI Express bus and PERC 4 E/DC compared to the PCI-X equivalents (see Figure 5).^{5, 6, 7}


Using NIC teaming to alleviate network bottlenecks

If the network becomes the bottleneck in an NFS setup, then network interface card (NIC) teaming in the NFS server can help increase the bandwidth available to the server. The NIC teaming method uses a driver to combine two or more NICs into a single logical network link. NIC teaming can work well with UDP traffic,⁸ making it appropriate for use in an NFS server. However, NIC teaming benefits NFS environments only when the bandwidth to the back-end storage exceeds the bandwidth available through a single link.

Boosting NFS application and cluster performance

The NFS distributed file system can operate effectively in many HPC installations. Tests performed for the study described in this article indicate that tuning the NFS service to perform optimally in a specific environment can help increase its performance—and consequently help increase the performance of an application using NFS services. No single method exists to optimize NFS performance. Tuning choices are not universal and should be determined by the application needs of each cluster.

However, administrators can adjust the many components of an NFS server—and the systems that affect the server—to help improve speed and throughput of both the application and the cluster as a whole. Administrators can choose the local file system for the NFS server based on the application that will use the server, and optimize the data layout accordingly. In addition,

adjusting local back-end storage, NFS client and server parameters, and the network configuration can contribute to performance gains. For example, using the parameters and options described in this article, Dell engineers achieved significant improvements in the NFS performance for different application patterns on a cluster of 32 Dell servers, compared to the same configuration using default NFS service parameters and standard settings. 

Acknowledgments

The authors would like to thank Rizwan Ali, Baris Guler, and Amina Saify for their assistance in conducting the tests and preparing this article.

Andrew Bachler is a systems engineer in the Scalable Systems Group at Dell. He has an associate's degree in Electronic Engineering and 12 years of UNIX® and Linux experience.

Jenwei Hsieh, Ph.D., is an engineering manager in the Scalable Systems Group at Dell, where he is responsible for developing high-performance clusters. His work in the areas of multimedia computing and communications, high-speed networking, serial storage interfaces, and distributed network computing has been published extensively. Jenwei has a Ph.D. in Computer Science from the University of Minnesota and a B.E. from Tamkang University in Taiwan.

FOR MORE INFORMATION

Dell storage products:
<http://www.dell.com/storage>

Dell HPC cluster products:
<http://www.dell.com/hpcc>

⁵This test was run locally on the NFS server, without utilizing the network.

⁶Aside from running at a faster clock rate, the PERC 4 E/DC was designed with other advantages over the PERC 4/DC. For example, the PERC 4 E/DC uses a 200 MHz cache that has 256 MB of double date rate (DDR) memory versus a 100 MHz cache with 128 MB of synchronous dynamic RAM (SDRAM) on the PERC 4/DC. This study tested each controller's standard, default feature set.

⁷Unless otherwise indicated, all tests in this study were performed using PCI-X technology.

⁸NIC teaming works well with UDP traffic because UDP packets are not sequenced; therefore, the order in which they arrive is irrelevant. (When using TCP, the CPU must perform additional work to properly sequence the packets arriving on two separate interfaces.)

Because storage is local to the NFS server, administrators can promote better overall server performance by tuning NFS storage for optimal performance.