Initial Characterization of Parallel NFS Implementations

Weikuan Yu[†‡] Jeffrey S. Vetter[‡]

Auburn University[†] Oak Ridge National Lab[‡]
wkyu@auburn.edu *vetter@ornl.gov*

Abstract

Parallel NFS (pNFS) is touted as an emergent standard protocol for parallel I/O access in various storage environments. Several pNFS prototypes have been implemented for initial validation and protocol examination. Previous efforts have focused on realizing the pNFS protocol to expose the best bandwidth potential from underlying file and storage systems. In this presentation, we provide an initial characterization of two pNFS prototype implementations, lpNFS (a Lustre-based parallel NFS implementation) and spNFS (another reference implementation from Network Appliance, Inc.). We show that both lpNFS and spNFS can faithfully achieve the primary goal of pNFS, i.e., aggregating I/O bandwidth from many storage servers. However, they both face the challenge of scalable metadata management. Particularly, the throughput of spNFS metadata operations degrades significanlty with an increasing number of data servers. Even for the better-performing lpNFS, we discuss its architecture and propose a direct I/O request flow protocol to improve its performance.

1 Introduction

Since its inception in the mid-80's, the Network File System (NFS) [10] protocol has been ubiquitously accepted as a means for file sharing across a network of compute nodes and across many operating systems simultaneously. Parallel NFS (pNFS) was further introduced as an extension of NFS [10] to meet the bandwidth demands of these applications. It aims for this goal by decoupling the data path of NFS from its control path, thereby enabling concurrent I/O streams from many client processes to storage servers.

Compared to existing parallel file systems such as GPFS [11], Panasas [4], Lustre [15], and PVFS [2], pNFS stands out as the only open standard that is currently being developed and endorsed by many commercial and non-profit organizations. pNFS offers a number of attractive features. First, its versatile layout drivers make it feasible to be built on any existing I/O and storage protocols as has been exemplified for PVFS [7], Panasas, and others. Second, it provides a smooth, effortless path to migrate and upgrade existing NFS deployments in order to improve scalability and performance. Third, recent efforts have shown that pNFS is a compelling technology for wide-area storage access [6]. As a result, pNFS offers a convenient approach to datacenter-wide file system, or a geographically-distributed file system.

Several prototypes have been implemented for initial validation and protocol examination of pNFS. In particular, a prototype lpNFS [13] (Lustre-based pNFS) is implemented to export the storage hosted by Lustre, a common file system at various computing centers. By nature, when it acquires and exports data blocks from the underlying file system, pNFS implementations add processing overhead. Thus, it is important to examine and analyze the performance behavior of representative pNFS implementations to pinpoint possible design deficiencies and shed light on future research and developmental directions.

In this presentation, we characterize the performance of two pNFS prototypes: lpNFS and spNFS (a reference parallel NFS implementation from Network Appliance, Inc.). Our experiments were conducted with both sequential and parallel I/O benchmarks, including data-intensive and metadata-intensive tests. We show that lpNFS

and spNFS can realize the original goal of pNFS and aggregate bandwidth from multiple storage devices. Both of them can achieve peak read bandwidth as allowed by the storage devices, while lpNFS provides better write performance. We also show that lpNFS and spNFS can support data-intensive scientific applications, such as NAS BT-IO. However, metadata management is an important scalability issue. The metadata performance of spNFS degrades with an increasing number of data servers, while lpNFS scales gracefully with more data servers. Nonetheless, the internal Lustre metadata server in lpNFS still faces a known scalability challenge in handling metadata requests from many concurrent clients. Since Lustre is a popular file system in large-scale computing centers, this can prevent lpNFS from efficiently exporting the storage hosted by Lustre to many concurrent clients. To address this problem, we exam the internal protocol of lpNFS and propose an alternative design that could improve the performance of lpNFS.

The rest of the paper is organized as follows. An overview is provided on the software architecture of lpNFS and spNFS in the next section. The experiment results are provided in Sections 3 and 4, for sequential and parallel I/O tests, respectively. Related work is discussed in Section 5. We then discuss the need of a direct flow path for I/O requests in lpNFS in Section 6. Finally, we conclude the paper in Section 7.

2 Prototype pNFS Implementations: lpNFS and spNFS

In parallel NFS, a metadata server uses a control protocol to manage data servers, while clients perform I/O to data servers according to the storage protocol. Figures 1 and 2 show the software architectures of lpNFS and spNFS, respectively. The lpNFS prototype [13] leverages the existing Lustre control protocol for managing data servers. As shown in Figure 1, lpNFS Metadata server (MDS) includes co-located Lustre client and MDS. When receiving a client request, the lpNFS MDS translates the intended control and/or management operation to Lustre client and Lustre MDS for execution. For the storage protocol, lpNFS avoids the use of a Lustre-specific layout driver at its clients. This is because such a layout driver requires a co-located Lustre client to translate I/O requests from a pNFS client, and limits the availability of Lustre storage to only Lustre-capable clients. Instead, the storage protocol of lpNFS uses the ubiquitous NFS file protocol. An lpNFS data server is co-located with a Lustre client and Lustre Object Storage Server (OSS). They work together to complete the I/O operations.

The spNFS prototype leverages as much as possible from the existing NFS code base and the local disk file system for a reference pNFS implementation. As shown in Figure 2, the data servers are NFS data servers with locally attached ext3 disk file system. The main component of spNFS is its control protocol (marked as "spnfs") that integrates all ext3 local file systems into a parallel NFS file system namespace.

3 Sequential I/O Benchmarks

The implementation of lpNFS is developed on top of a pNFS source tree, Linux-2.6.25-pNFS from linux-nfs.org, and a base Lustre release, version 1.6.5. The spNFS (simple parallel NFS) is available from linux-nfs.org. Our performance characterization includes both data and metadata operations. The experiments were conducted on a cluster of Clara VXB-7520J blades: each with dual Intel Xeon 3.4 GHz processors, 2MB cache, 800MHz Front Side Bus (FSB), 4GB physical memory, and a Gigabit Ethernet card connected to PCI-X 100 Mhz bus. On each Lustre OSS, a 80GB, 7200 RPM, ATA/100 Western Digital hard disk WD800JB is used for disk storage. All tests were conducted with one MDS, up to 16 clients, and a varying number of data servers.

3.1 IOzone

IOzone [1] is a popular sequential I/O benchmark that measures a broad range of performance characteristics of a file system. A varying number of threads were used for the sequential I/O tests. IOzone record size was set at 16 MB. The total size of data from all threads was chosen to be 5120 MB, bigger than the memory size to eliminate client-side caching effects.

Figure 3 shows the performance of IOzone with multiple threads on lpNFS and spNFS, using 2 or 4 data servers. For reads, both lpNFS and spNFS can reach the peak network performance of Gigabit Ethernet. This is because the spNFS and lpNFS servers can keep all the data in their buffer caches.

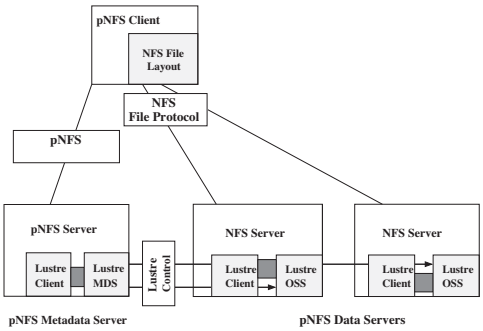


Figure 1. lpNFS Software Architecture

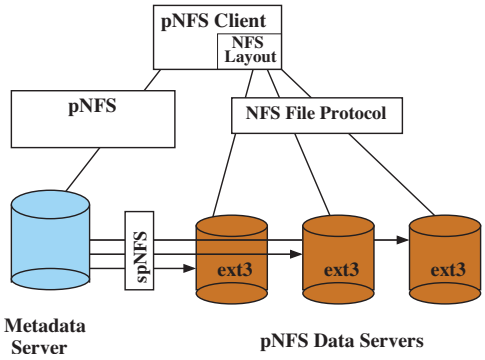
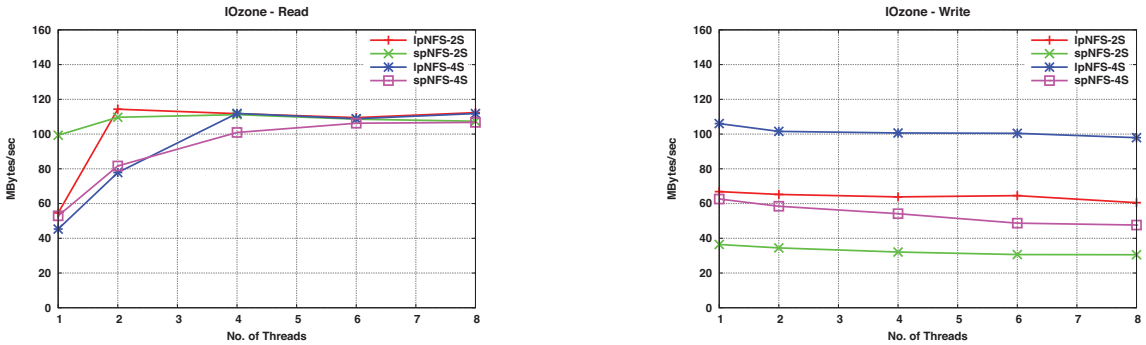


Figure 2. spNFS Software Architecture



(a) Read (b) Write

Figure 3. Performance of IOzone

With an increasing number of threads, lpNFS performs slightly better than spNFS. For writes, both lpNFS and spNFS achieve better aggregated I/O bandwidth with more data servers, while lpNFS performs better than spNFS.

3.2 Fileop

Fileop is a benchmark distributed along with IOzone. It tests the performance of a variety metadata operations including file creation (*create*) and removal (*unlink*). We measured the performance of Fileop from a single client to lpNFS and spNFS with a varying number of data servers. Figure 4 shows the performance of create and unlink operations. On spNFS, the throughput for both operations decreases with an increasing number of data servers. But the metadata throughput on lpNFS remains mostly constant. This is because spNFS and lpNFS have different control protocols for metadata management. spNFS creates metadata (i.e.

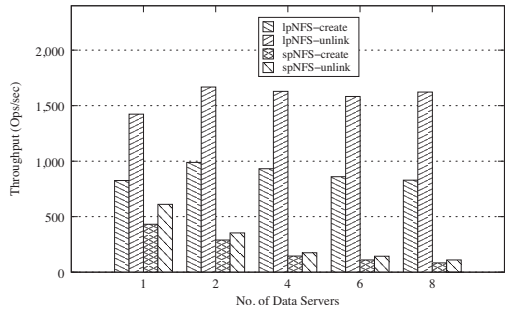


Figure 4. Throughput of Create and Unlink

inodes) on each data server for every file. In contrast, lpNFS creates only one inode for each file in the encapsulated Lustre metadata server. Therefore, it takes longer for spNFS to create or unlink files when there are more data servers. However, the single meta-data server in lpNFS is still a performance and scalability bottleneck. This is a known scalability issue [14] for Lustre when handling re-

quests from many concurrent clients on large-scale platforms.

4 Parallel I/O Benchmarks

4.1 IOR

We used IOR [9] to measure the performance of parallel I/O, including concurrent I/O to separate files or to a shared file. IOR supports a number of different I/O interfaces, including POSIX, MPI-IO, and HDF5 [5]. The POSIX interface of IOR is used to measure the aggregate bandwidth from concurrent reads or writes, using separated files. All data were collected through 10 runs and the average of the best five was taken as the final number. Small deviations were observed, but not included in these figures.

We measured the parallel I/O performance using four data servers. The data size was chosen as 512 MB per process, and the transfer size as 16 MB. Figure 5 shows the parallel I/O performance of lpNFS, compared to spNFS. The aggregate read bandwidth of lpNFS and spNFS both reached 450 MB/sec, close to the aggregate link bandwidth of four data servers. The write performance of lpNFS is significantly better than that of spNFS. This suggests that, compared to the storage hosted by regular local disk file systems, it is beneficial to export storage from the high performance Lustre file system to parallel clients.

4.2 NAS BT-IO

Scientific applications can have very diverse I/O access patterns. We used the NAS BT-IO benchmark [12] as a representative application. We measured the performance of NAS BT-IO on lpNFS and spNFS with four data servers. Figure 6 shows the performance of BT-IO, classes A and B, using a varying number of client processes. lpNFS consistently performs better than spNFS for Class A and Class B. With an increasing number of processes, the aggregated BT-IO bandwidth decreases. This is because, when the file size of NAS BT-IO remains the same, more processes lead to finer-grained data patterns. Therefore, more time is spent at the MPI-IO layer to perform data aggregation in collective I/O.

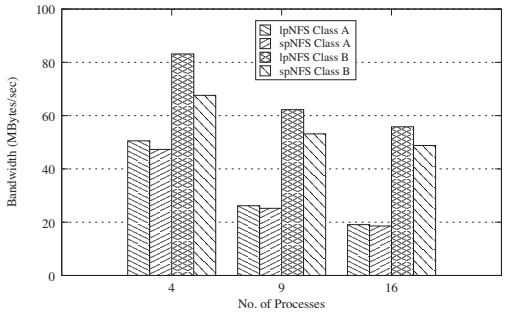


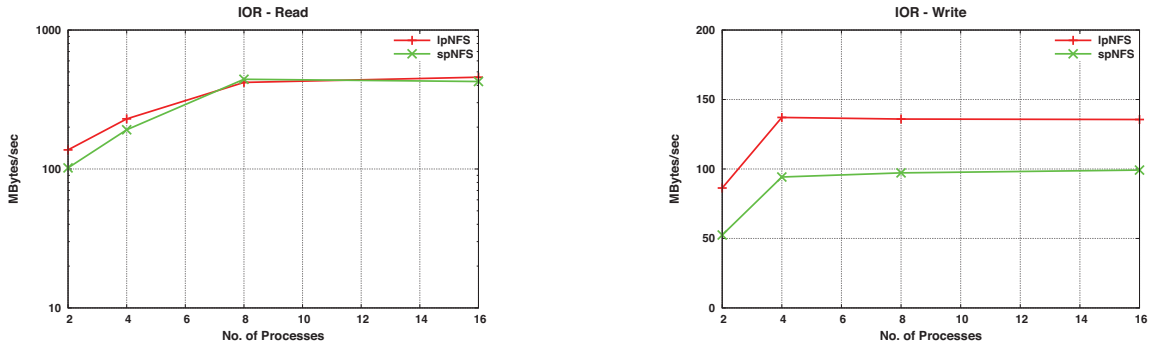
Figure 6. Performance of NAS BT-IO

5 Related Work

Multiple research efforts have investigated pNFS over the recent years. Hildebrand et al. [3] first demonstrated a prototype of pNFS over PVFS in 2005. Using this prototype, Hildebrand et al. [8] also investigated how to improve the overall performance of pNFS by enabling different I/O paths for small and large write requests. Further efforts from the same team demonstrated the direct-pNFS architecture using PVFS [7]. There has been continuous interest in pNFS from a number of storage vendors: Panasas has undertaken the task of developing an object-based layout driver; EMC, the task of block-based layout driver; and, Network Appliance, the task of file-based layout driver. Some early prototypes are already available for adventurous users. In particular, the simple parallel NFS (spNFS) used in this paper is developed by Network Appliance, Inc. Our earlier work [13] provided an efficient pNFS implementation on top of Lustre and supported transparent pNFS clients to storage hosted by Lustre file system. This work focuses on the characterization of two prototype pNFS implementations and lpNFS and spNFS. We examine the performance of both metadata and data, and then describe an alternative design that can improve metadata access in lpNFS.

6 Discussion

As described earlier, there is a metadata scalability challenge faced by the pNFS implementations. It is very important to exam these prototype pNFS implementations for scalable pNFS on large-scale platforms. Particularly, we focus on lpNFS due to its better performance and relevance to Lustre, a widely deployed file system in leadership comput-



(a) Read (b) Write

Figure 5. Performance of IOR

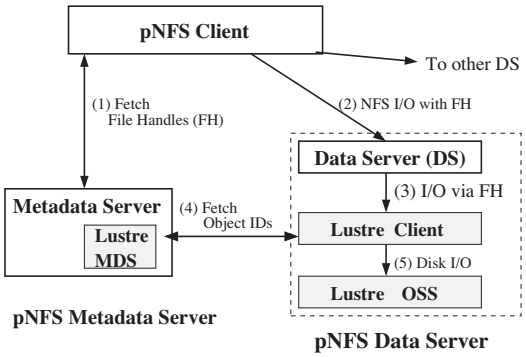


Figure 7. The Flow for I/O Requests in lpNFS

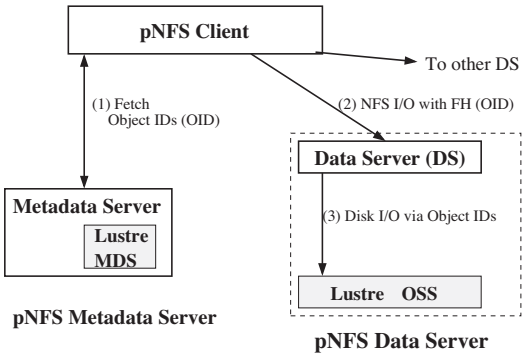


Figure 8. Proposed Flow for I/O Requests in lpNFS

ing centers.

The current design of lpNFS has an inherent deficiency in handling large amount of metadata. As shown in Figure 7, I/O requests (data and metadata) have to go through up to five steps before finally completed at the storage devices. First, a client fetches NFS file handles (FH) from the MDS. Second, it presents the FH to an individual data server for I/O. Third, the NFS data server uses the FH for data access on the locally mounted Lustre file system. Fourth, the client has to fetch and translate the FH to a Lustre Object ID by interacting with the Lustre MDS (located within lpNFS MDS). Finally, disk I/O can take place at the data server. Among the five, Steps 3 and 4 are purely artifacts due to the export of an entire Lustre file system stack from the lpNFS data server.

In view of the prolonged path for I/O handling, we propose a direct flow path for I/O requests in lpNFS. As shown in Figure 8, the new design completely avoids the Lustre client and the Lustre MDS for I/O request handling at the data server. To make that possible, the lpNFS layout operations shall be capable of returning a file layout that contains the native Lustre Object IDs (OID). With these OIDs , an lpNFS client then presents an I/O request (with a corresponding OID wrapped inside) to the corresponding lpNFS data server. Using this OID, the lpNFS data server can directly access data blocks from Lustre OSS. This design greatly simplifies the processing of lpNFS I/O operations. We are currently in the process of implementing the proposed optimization.

7 Conclusion

In this paper, we presented an initial performance characterization of two pNFS prototype implementations, lpNFS (Lustre-based parallel NFS) and spNFS (simple parallel NFS from Network Appliance,

Inc.). Our experiments were conducted with both sequential and parallel I/O benchmarks, including data-intensive and metadata-intensive tests. We show that both lpNFS and spNFS can achieve peak read bandwidth as allowed by the storage devices, while lpNFS provides better write performance. We also show that lpNFS and spNFS can support data-intensive scientific applications, such as NAS BT-IO.

However, metadata management is an important scalability issue, for which the performance of spNFS degrades with an increasing number of data servers, while lpNFS scales gracefully with more data servers. The internal Lustre metadata server in lpNFS still faces a known scalability challenge in handling metadata requests from many concurrent clients. Based on our initial characterization and examination of parallel NFS implementations, we discuss how to optimize the design of lpNFS to efficiently export Lustre storage to many parallel NFS clients. We propose a direct flow path in lpNFS to expedite the delivery of I/O requests to Lustre storage servers. We are in the process of implementing this direct flow protocol.

In the future, we plan to extend lpNFS further with more features that are defined for pNFS, such as client-side caching, session management, and file delegation.

8 Acknowledgments

This work was funded in part by a UT-Battelle grant (UT-B-4000087150) to Auburn University, and in part by a National Science Foundation award CNS-0917137. This research was sponsored in part by the Office of Advanced Scientific Computing Research; U.S. Department of Energy. This research used resources of the National Center for Computational Sciences at Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

References

[1] IOzone Filesystem Benchmark. In *http://www.iozone.org*.

[2] The Parallel Virtual File System, version 2. http://www.pvfs.org/pvfs2.

[3] D. L. Black and S. Fridella. pNFS Block/Volume Layout. http://www.ietf.org/internet-drafts/draft-ietf-nfsv4-pnfs-block-00.txt.

[4] A. M. David Nagle, Denis Serenyi. The Panasas ActiveScale Storage Cluster – Delivering Scalable High Bandwidth Storage. In *Proceedings of Supercomputing '04*, November 2004.

[5] T. N. C. for Supercomputing Applications. HDF Group - HDF5. http://www.hdfgroup.org/HDF5/.

[6] D. Hildebrand, M. Eshel, R. Haskin, P. Andrews, P. Kovatch, and J. White. Deploying pnfs across the wan: First steps in hpc grid computing. In *Proceedings the 9th LCI International Conference on High-Performance Clustered Computing*, Urbana, IL, April 2008.

[7] D. Hildebrand and P. Honeyman. Direct-pNFS: Scalable, Transparent, and Versatile Access to Parallel File Systems. In *Proceedings of the 16th IEEE International Symposium on High Performance Distributed Computing*, Monterey, CA, June 2007.

[8] D. Hildebrand, L. Ward, and P. Honeyman. Large files, small writes, and pnfs. In *Proceedings of the 20th ACM International Conference on Supercomputing (ICS06)*, Cairns, Australia, June 2006.

[9] LLNL. IOR Benchmark. http://www.llnl.gov/asci/purple/benchmarks/limited/ior.

[10] R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, and B. Lyon. Design and implementation of the Sun Network Filesystem. In *Proc. Summer 1985 USENIX Conf.*, pages 119–130, Portland OR (USA), 1985.

[11] F. Schmuck and R. Haskin. GPFS: A Shared-Disk File System for Large Computing Clusters. In *FAST '02*, pages 231–244. USENIX, Jan. 2002.

[12] P. Wong and R. F. Van der Wijngaart. NAS Parallel Benchmarks I/O Version 2.4. Technical Report NAS-03-002, Computer Sciences Corporation, NASA Advanced Supercomputing (NAS) Division.

[13] W. Yu, J. Vetter, and O. Drokin. Design, implementation, and evaluation of transparent pnfs on lustre. In *Proceedings of the 23rd International Parallel and Distributed Processing Symposium (IPDPS)*, April 2009.

[14] W. Yu, J. Vetter, and S. Oral. Performance characterization and optimization of parallel i/o on the cray xt. In *Proceedings of the 22rd International Parallel and Distributed Processing Symposium (IPDPS)*, April 2008.

[15] R. Zahir. Lustre Storage Networking Transport Layer. http://www.lustre.org/docs.html.