White Paper

pNFS: A New Standard for Shared, High-Performance Parallel I/O

Mike Eisler, Sachin Chheda, NetApp
January 2012 | WP-7153

EXECUTIVE SUMMARY

Parallel NFS (pNFS) is a new standard documented in RFC 5661 by the Internet Engineering Task Force (IETF) in conjunction with NFS Version 4.1 (NFSv4.1). pNFS offers an industry-standard framework for shared, high-performance parallel I/O suitable for use with data-intensive scientific, engineering, and other applications running on large compute clusters. pNFS overcomes the single-file server design of standard NFS by utilizing a metadata server in combination with multiple data servers. pNFS client systems are able to do I/O in parallel to file data striped across multiple data servers. File-, block-, and object-based data servers are currently supported by the protocol.

This white paper discusses the basic operation of pNFS and describes the pros and cons of the available data layouts. Guidance is also provided for implementation planning.

TABLE OF CONTENTS

LIST OF FIGURES

1 INTRODUCTION

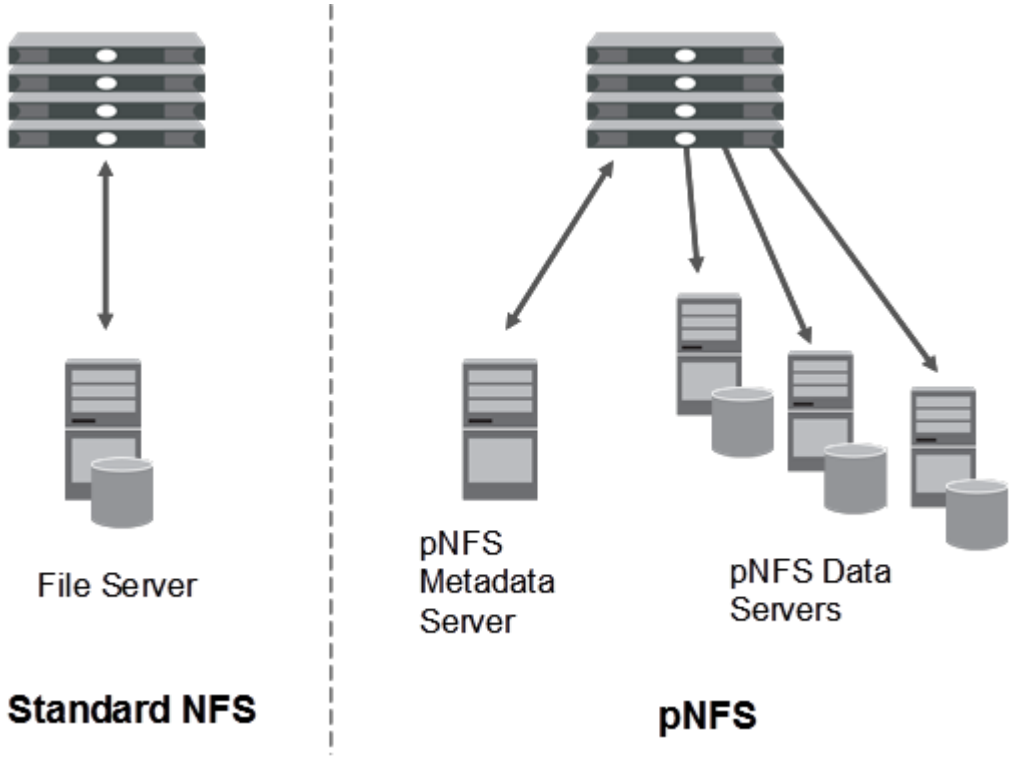
Since its introduction in 1984, the Network File System (NFS) has become a standard for network file sharing, particularly in UNIX® and Linux® environments. The NFS protocol has steadily evolved to adapt to new requirements and market changes. Today, NFS Version 3 is still the most widely used version of the protocol. However, NetApp is finding that NFSv4 attach rates are far outpacing NFSv3 attach rates. NFS plays a critical role in a variety of science and engineering applications as well as in more general business applications.

Compute clusters running computer-aided engineering (CAE), seismic data processing, bioinformatics, and other science and engineering applications often rely on NFS to access shared data. However, because all files in a file system must be accessed through a single file server, NFS can result in significant bottlenecks for applications such as these. One common way to scale performance is to "scale up" the performance of that single file server. Another way to scale performance is to "scale out" storage—clients connect to a single file server, but data is distributed across multiple servers using a clustered file system. For some applications this can increase performance, but it might not accelerate access to single, large files or eliminate all bottlenecks. These limitations have created a strong need for an industry-standard method to "scale out" shared storage, analogous to the way servers are scaled out in a compute cluster.

NFS Version 4 (NFSv4) addresses some of the limitations found in NFSv3 through compound operations and the use of delegations for improved client-side caching.[1] However, the single-server bottleneck remains. The latest updated to the NFS protocol, NFS Version 4.1, includes a specification for parallel NFS (pNFS), which has been designed to eliminate the single-server bottleneck using a standard protocol capable of supporting heterogeneous storage systems. Other attempts to create parallel file systems such as Red Hat GFS, SGI CXFS, and Lustre were not based on the same type of broad-based standards effort, and none has been broadly adopted on anything approaching the scale of NFS.

[1] www.netapp.com/us/communities/tech-ontap/nfsv4-0408.html

Figure 1) With standard NFS, all data must be accessed through a single server, which might become a bottleneck for certain applications. In pNFS, data is striped across multiple data servers. Clients access data from data servers directly based on information received from a metadata server.



pNFS will offer a number of advantages over existing shared file system options. In addition to parallel I/O and support for a broad range of hardware, pNFS provides:

- **Application transparency.** Applications will be able to take advantage of pNFS with no code changes.
- **Integration.** Once pNFS becomes widely available (see section 3), it will be integrated into common operating systems, including standard releases of Linux, so it won't require any installation, recompiling, debugging, and so on.
- **No client-side software**. This is a significant consideration for large compute clusters. It can take hours to install software and drivers on large numbers of clients, not to mention ongoing maintenance, patches, and so on.

This white paper explores the pNFS architecture and discusses current design and implementation challenges.

2 UNDERSTANDING PNFS

The pNFS protocol gives clients direct access to files distributed across two or more data servers. By accessing multiple data servers in parallel, clients achieve significant I/O acceleration. The pNFS protocol has been designed to deliver graceful performance scaling on both a per-client and per-file basis, without sacrificing backward compatibility with the standard NFS protocol; clients without the pNFS extension are still able to access data.

Guiding principles of the pNFS design include:

- **Flexibility**. The protocol is designed to be implemented across a variety of clustered storage architectures.
- **Simplicity.** The pNFS interface provides the most basic primitives needed to exploit data parallelism, while leaving room for additional semantics to be supported on top of those.

- **Familiar semantics**. pNFS provides consistency and security semantics similar to those of NFSv4, simplifying the adoption of pNFS for diverse applications.

PNFS ARCHITECTURE AND CORE PROTOCOLS

The pNFS architecture consists of three main components:

- The **metadata server** (MDS), which handles all nondata traffic. The metadata server is responsible for maintaining metadata that describes where and how each data file is stored.
- **Data servers,** which store file data and respond directly to client read and write requests. File data can be striped across a number of data servers.
- One or more **clients,** which are able to access data servers directly based on information in the metadata received from the metadata server.

There are three types of protocols used between the clients, metadata server, and data servers:

- A **control protocol** is used between the metadata server and data servers to provide synchronization.
- **pNFS protocol** is used between clients and the metadata server. This is essentially NFSv4 with a few pNFS-specific extensions. It is used to retrieve and manipulate **layouts**, which contain the metadata that describes the location and storage access protocol required to access files stored on multiple data servers.
- A set of **storage access protocols** used by clients to access data servers. The pNFS specification currently has three categories of storage protocols: file based, block based, and object based. These allow pNFS to accommodate various layout types to support different kinds of storage infrastructure.

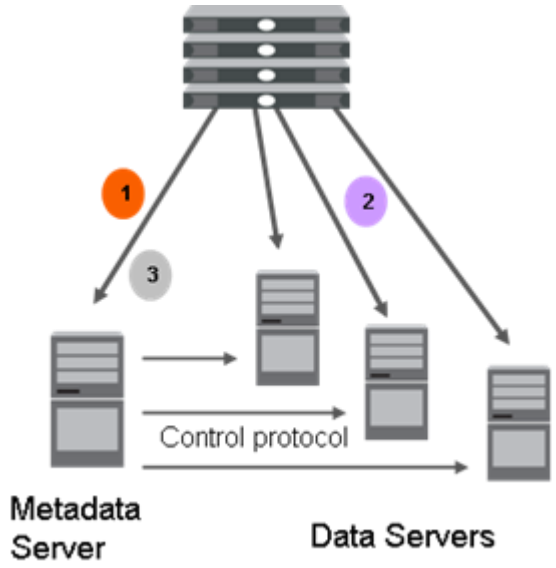
LAYOUT TYPES AND FILE ACCESS

The storage access protocol that is employed depends on the type of storage on the underlying data servers. The **layout** for a file that the metadata server sends to a client provides the client with the information to determine where each stripe of a file is stored, how to access it, and with what protocol.

- When the **file layout** is employed, pNFS uses multiple NFSv4.1 file servers as its data servers. NFSv4 itself serves as the file access protocol.
- When the **block layout** is used, disk LUNs are hosted on a SAN. Either the iSCSI or Fibre Channel protocol is used to access SAN devices using the SCSI block command set.
- **Object layouts** allow data to be stored on object-based storage devices (OSDs) and accessed using the T-10 object-based storage device protocol currently being standardized.

Regardless of the layout type, to access a file, a client contacts the metadata server to open the file and request the file's layout. Once the client receives the file layout, it uses that information to perform I/O directly to and from the data servers in parallel, using the appropriate storage access protocol without further involving the metadata server. When the client completes its I/O, it sends modified metadata to the metadata server and closes the file.

Figure 2) Typical pNFS process flow. (1) Client requests layout from metadata server (using pNFS protocol). (2) Client accesses data servers directly (using a storage access protocol). (3) Client updates metadata and closes file



CHOOSING A LAYOUT TYPE

The following are a few criteria you might want to consider when deciding which type of layout to use with pNFS.

STARTING POINT

- If you are starting from a NAS framework to begin with, a file layout obviously makes the most sense. You can use your existing NAS systems and networks.
- If you have an existing Fibre Channel SAN, you will probably want to choose a block-based layout.
- Continue reading if you are implementing from scratch.

NETWORK INFRASTRUCTURE

- With a file-based layout, you can use your existing NAS storage and your existing Ethernet infrastructure. You don't necessarily have to add more bandwidth.
- With a block- or object-based layout, you most likely will have a Fibre Channel storage area network (FC SAN). All clients will have to connect to the FC SAN to access data servers directly. iSCSI might be a less expensive alternative.

SECURITY AND DATA INTEGRITY

- With a file-based back end, security and data integrity are enforced by each data server, which uses the same methods as a standard NFS server, including Kerberos authentication, access control lists (ACLs), and so on. Security is well understood and familiar.
- Using pNFS with a block- or object-based back end puts a large part of the burden for security and data integrity on the pNFS client implementation. Since the client is part of the client operating system, you might not have much control over the security it provides (or does not provide). In other words, you might not have much choice in choosing client implementations, so you might prefer to choose a layout where the data servers enforce security.

MULTIPLE-CLIENT ACCESS AND MANAGEMENT

- With a file-based layout, two different pNFS clients can access the same logical region of a file for reading or writing.

- With the block- or object-based layout, multiple clients can read from the same region of a file, but if one pNFS client holds a write layout, no other client can hold a layout for that byte range even for reading. This means the server has to recall portions of layout to let the pNFS client write.

BUILDING BLOCKS FOR STANDARDIZED STORAGE

- Available sources for object-based back-end storage might be limited.
- Standardizing access using the pNFS standard for a block-based SAN is a positive step, but it still requires an additional, often proprietary, SAN file system.

3 STATUS OF PNFS DEVELOPMENT

NFSV4.1 AND PNFS STANDARDS EFFORT

The NFSv4.1 standards effort, of which pNFS is a part, is a broad-based effort within the Internet Engineering Task Force (IETF). The working group includes members from a broad cross section of leading storage and system vendors and researchers such as NetApp, EMC, IBM, University of Michigan, Panasas, and Sun Microsystems. The NFSv4.1 and pNFS standard was ratified in December of 2008 and was finalized by the IETF in January of 2010. NetApp has been a major driver of both NFSv4.1 and pNFS, cochairing the efforts of the working group. In addition, NetApp has authored and edited most of the NFSv4.1 specification. This is consistent with our commitment to tackling the problems of storage using industry standards.

For more information on the IETF specification for NFSv4.1 and pNFS, visit the IETF NFS v4 working group Web site.[2]

NFSV4.1/PNFS TESTING

Interoperability is essential to the adoption of pNFS, as it was for NFS. Making sure that all client and server implementations work seamlessly together will accelerate adoption. Interoperability testing of various pNFS implementations has been under way since March 2005.

NFSv4.1 and pNFS have been tested at the annual Connectathon,[3] a vendor-neutral forum for testing hardware and software interoperability. (NetApp is a Connectathon sponsor.) In addition, less formal Bake-a-thons are held several times a year under the auspices of Sun Microsystems and the University of Michigan.

LINUX CLIENT AND SERVER DEVELOPMENT

Because of the prevalence of Linux in the compute clusters used by many of the scientific, engineering, and other applications that stand to benefit most from pNFS, it is important to have a well-designed and well-tested pNFS client for Linux. NetApp and others have recognized this need and are investing in the creation of a robust Linux pNFS client capable of meeting the performance needs of data-intensive applications. A pNFS client for Linux is available in Fedora version 15 and in the Linux Kernel version 2.6.40. Several Linux distributions are in the process of qualifying for production, with Red Hat Linux version 6.2 being first to hit market. Additional OS vendors are also in development of a pNFS client.

NetApp is contributing heavily to the pNFS client and the file layout driver. In September of 2011, NetApp released a production-ready pNFS server with the release of NetApp® Data ONTAP® 8.1 operating in Cluster-Mode. This pNFS server is being tested with Linux distributions and will be used

[2] www.ietf.org/html.charters/nfsv4-charter.html
[3] www.connectathon.org/

in development for other OS clients. In December 2011, Red Hat released Red Hat Enterprise Linux version 6.2 (RHEL 6.2), which includes the aforementioned pNFS client and file layout driver. This release complements NetApp's release in November 2011 of its pNFS server for Data ONTAP 8.1 operating in Cluster-Mode, which supports file layouts.

4 CONCLUSION: THE PATH TO PNFS

NetApp delivered a path to offer scalable storage based on the IETF pNFS standard that builds on the capabilities we have developed for the Data ONTAP operating systems.[4] With Data ONTAP 8.1 operating in Cluster-Mode, standard NFS clients mount file systems from a cluster of storage systems with a global namespace without regard for which individual storage system controls the data. Data ONTAP 8.1 operating in Cluster-Mode offers a massively scalable unified storage platform for both SAN and NAS workloads. Large storage environments can be managed more effectively with fewer IT resources. The storage environment can also be updated and upgraded without disruption to applications.

The NetApp pNFS server of Data ONTAP 8.1 offers immediate benefits. For one, the metadata server is included in each FAS or V-Series controller/node. This design eliminates the need for a dedicated metadata server. Additionally, if there is any hardware failure, surviving nodes resume the metadata server duties of the failed node until it comes back online. This design simplifies the configuration and management of pNFS deployments and offers increased resilience in the event of a hardware failure. Other storage implementations might be able to achieve a similar result, but will require a lot of additional logic within the storage system itself. For NetApp, this capability comes for free because of the work already done on the back-end file system and clustered architecture. No additional logic is required within the pNFS server.

PREPARING FOR PNFS

With emergent availability of an end-to-end pNFS solution, you might want to consider including pNFS in your storage plans for scientific, engineering, business, and enterprise workloads. To begin to prepare for the transition:

- Review how your file data is stored and served now and how it will need to be structured in the future.
- Talk to your operating system and storage vendors to find out their plans for pNFS. Your application vendors might also provide guidance about storage needs going forward.
- If you will be implementing a NetApp solution, talk to your representative to understand how you can smoothly transition to scale-out storage.
- Understand NFSv4 and NFSv4.1. Many NetApp customers are already using NFSv4 with production NetApp storage systems. If you begin the transition now, not only will you gain the advantages of NFSv4, but also it will simplify the process of transitioning client systems to pNFS.

By taking a few appropriate steps now, you'll be able to make a smooth transition to pNFS, with less disruption and better results.

AUTHORS

Mike Eisler, Senior Technical Director

Mike is the leader of NetApp's NFS-related development efforts. He is the author of the NFSv4.0 and NFSv4.1 specifications and several other specifications relating to NFS and security. Mike's first exposure to NFS and NIS came while working for Lachman Associates, Inc., where he was

[4] www.netapp.com/us/library/technical-reports/tr-3468.html

responsible for porting NFS and NIS to System V platforms. He joined NetApp from Sun, where he was responsible for several NFS and security-related projects.

Sachin Chheda, Solution Marketing Manager for File Services and NAS

Sachin drives the marketing effort for NetApp's NAS products and file services solution. He has long been involved in the enterprise IT space as an engineer, a platform architect, and a product manager. He holds several patents in the area of enterprise computing and regularly mentors students interested in pursuing careers in technology.

CONTRIBUTORS

Joshua Konkle is vice president, Product Management at DCIG and was a contributing author on a previous draft of this paper.

Jason Blosil is a product marketing manager for core software products at NetApp.

Go further, faster®

www.netapp.com