



Education

Scaling NFS Through pNFS

Joshua Konkle, NetApp, Inc.
Omer Asad, NetApp, Inc.

- The material contained in this tutorial is copyrighted by the SNIA.
 - Member companies and individuals may use this material in presentations and literature under the following conditions:
 - ◆ Any slide or slides used must be reproduced without modification
 - ◆ The SNIA must be acknowledged as source of any material used in the body of any document containing material from these presentations.
 - This presentation is a project of the SNIA Education Committee.
 - Neither the Author nor the Presenter is an attorney and nothing in this presentation is intended to be nor should be construed as legal advice or opinion. If you need legal advice or legal opinion please contact an attorney.
 - The information presented herein represents the Author's personal opinion and current understanding of the issues involved. The Author, the Presenter, and the SNIA do not assume any responsibility or liability for damages arising out of any reliance on or use of this information.
- NO WARRANTIES, EXPRESS OR IMPLIED. USE AT YOUR OWN RISK.**

➤ Scaling NFS through pNFS

- ◆ This session will appeal to Data Center Managers, Cluster/Grid/Compute Server administrators, and those that are seeking a fundamental understanding of current limitations and bottlenecks of NFS based I/O in Grid and Technical Computing. This session will explore how pNFS can scale NFS based I/O in a data center. The session will explore how heterogeneous storage servers and layout protocols file/blocks/objects can be made transparent to NFS clients through a common open systems framework. We explore how clustered file systems can be fully parallelized through pNFS without the need of host based agents or stubs.

- Background Information
- State of the union
- I/O limitations of NFS in Grid & High Performance Computing
 - ◆ Move to Out of band and Clustered NAS
- NFSv4 and NFSv4.1
- Move to Parallel I/O
- pNFS
- Current Status

➤ Network File System

- ◆ Protocol to make data stored on file servers available to any computer on a network
- ◆ NFS clients are included in all commodity Operating Systems, e.g. Linux, Solaris, AIX, Windows etc.....
- ◆ Operating System abstraction layers, hide the network complexity; Completely transparent

➤ NFS server; Inspiration to NAS appliances

- ◆ Commodity Operating Systems have NFS servers
- ◆ Vendors have specialized NFS servers with lots of disk
 - NAS appliances

- First version released in 1982: NFSv2
 - ◆ RFC 1094, published in 1989 through IETF

- NFSv3, released 1991, is the most commonly used protocol
 - ◆ RFC 1813, published 1995 through IETF
 - ◆ Layered on XDR and RPC
 - ◆ Requires support protocols: NLM (lockd), STAT (statd) and MOUNT (mountd)
 - ◆ Stateless
 - ◆ Works over TCP/UDP

- Security provided by additional protocols
 - ◆ Kerberos V4 & RPCSEC_GSS Authentication

- Read-ahead & Write-behind methods for performance

- Very prevalent in data centers
 - ◆ Home Directory environments; Storage consolidation;
 - ◆ Unix/Linux build farms; diskless boot;
 - ◆ Virtual environments over NFS

- Databases on NFS
 - ◆ About a 3rd of databases out there are NFS server backed
 - ◆ Direct I/O modifications to NFS clients/servers
 - ◆ posix based Async I/O and Direct NFS (dNFS)

- Grid Computing
 - ◆ HPC & Technical Computing Applications
 - ◆ Clusters on demand

Evolving Requirements

➤ Economic Trends

- ◆ Availability of cheap & fast computing power & commodity clusters
- ◆ Faster & cheaper hardware on both network (GigE to 10GigE) & compute side (dual core to quad core and more)

➤ Performance

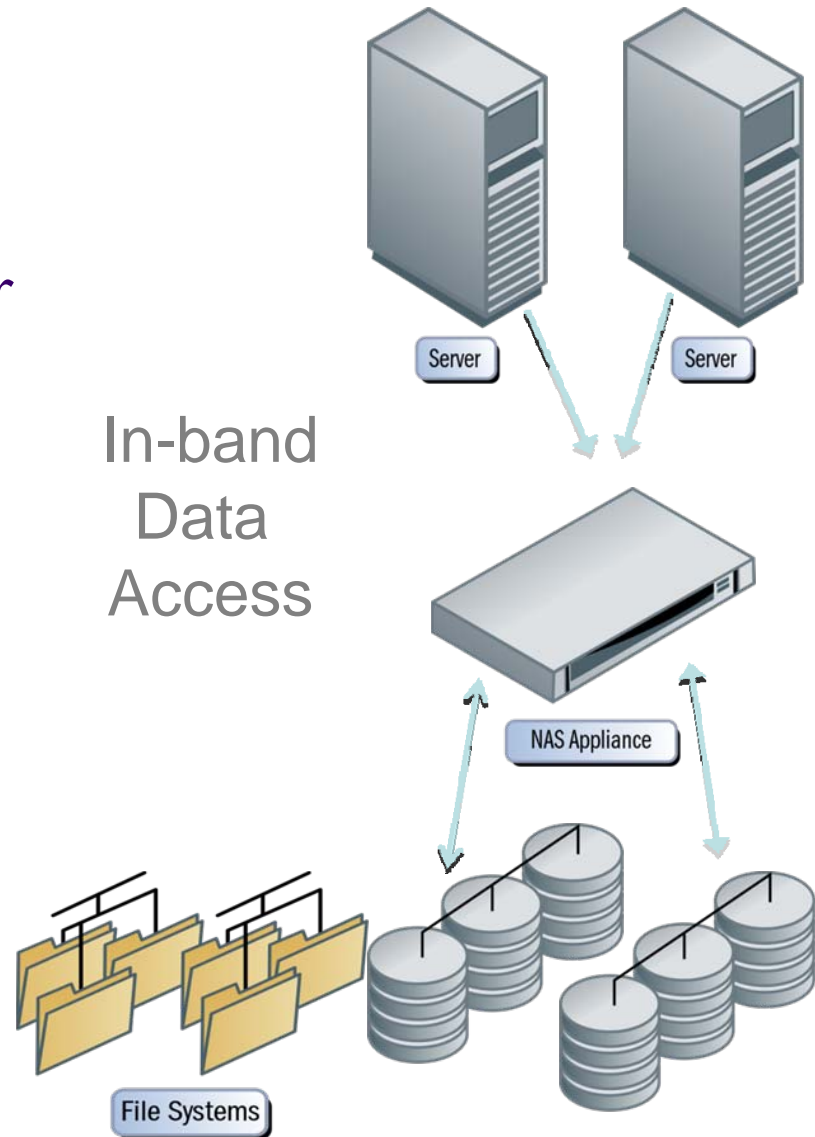
- ◆ Faster hardware exposes single threaded bottlenecks in software
 - Need for more parallelization in software to exploit compute power
- ◆ Evolution of computing models and problem complexity
 - Requires more performance from compute/network/storage layers
- ◆ Need for faster time to market and competitive edge
- ◆ Powerful compute systems = more data at explosive rates

Present Access model

- NFS at present has an in-band data access model
 - ◆ Easy to build but limited in scale
 - ◆ NFS server node is responsible for security
 - ◆ Well-defined failure modes
 - ◆ Limited load balancing options

- Results in islands of storage

- Limited by appliance hardware
 - ◆ Both Networking and I/O
 - ◆ Moderate # of clients



- Performance scalability limited by NFS server hardware
 - ◆ NFS servers can handle a few 100 clients
 - › Compute farms are growing bigger in size every day
 - ◆ NFS server hardware uses the same components, in terms of CPU, I/O, Memory, as the compute clients
 - › Imagine 100s of them performing I/O at once (drown-out)
 - ◆ Limits on the amount of spindles that can be attached to a single NFS server to distribute the I/O

- Scalability problems at controller and storage level
 - ◆ Natural inclination to apply the client clustering model to storage architectures

I/O Limitations (contd.)

- Meta data and Random I/O intensive workloads stress (NFS) server/controller memory
 - ◆ Drive up the response time
 - ◆ Introduce hotspots

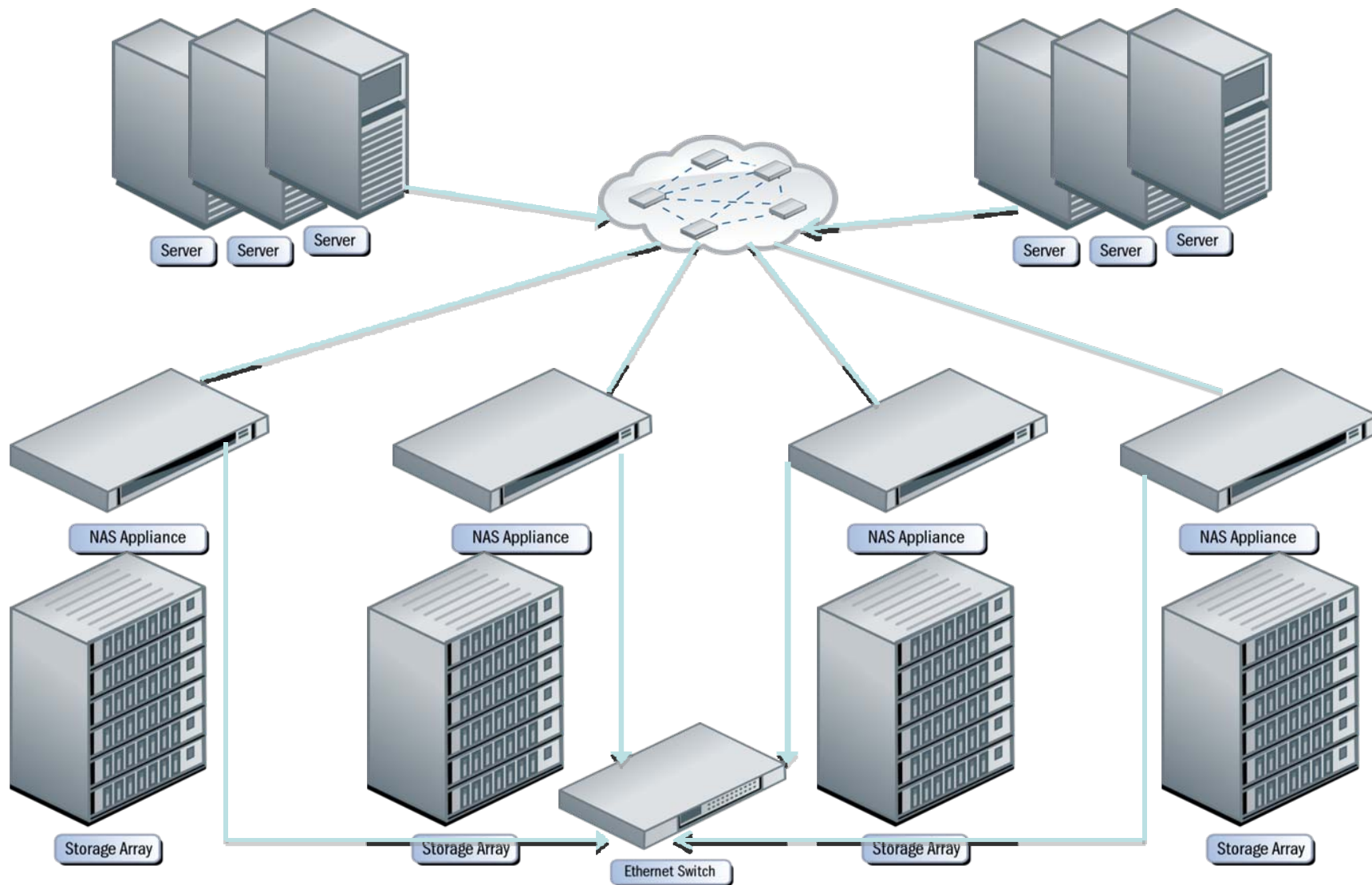
- Load balancing options are limited across single NFS controller or a controller pair

- Unless configured properly, upgrades to higher performing hardware require down time
 - ◆ Resource re-provisioning is disruptive

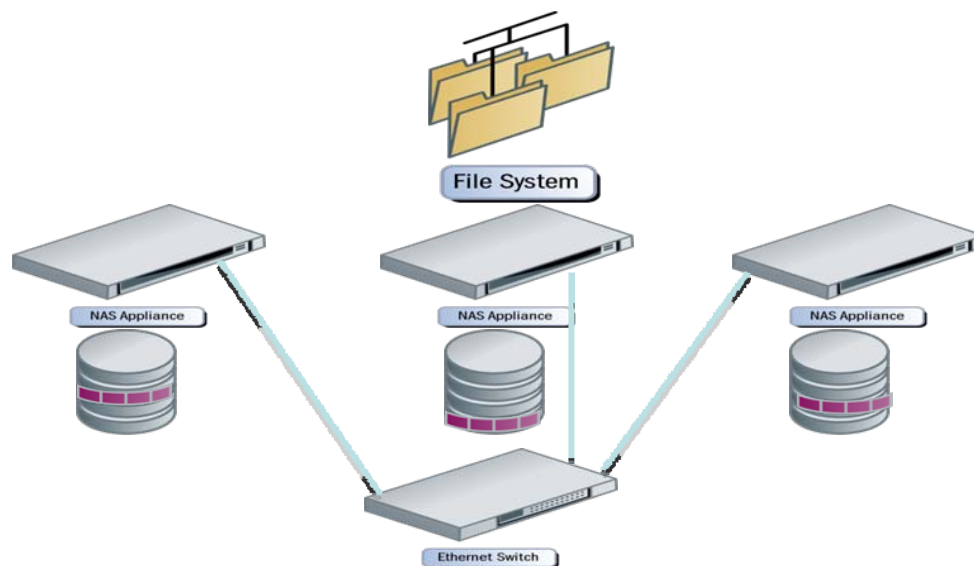
- Limited failure domains

- Apply the client side model to the NFS server side
- Cluster NFS servers together to form storage clusters
 - ◆ Requires a move to clustered/parallel storage file systems
- Immediate effect:
 - ◆ Failure domain is enhanced
 - ◆ Both controller (compute) and storage (spindle) scale independently and without limitations
 - ◆ Sophisticated load balancing techniques can be applied
 - ◆ Introduces global namespace
- Resolves hotspots and introduces simplicity
 - ◆ Single management point & non-disruptive scaling

Clustered NAS (Contd.)



- Symmetric model
 - ◆ All nodes participate in all file systems activity
 - ◆ Requires a clustered file system
- File & Metadata striping
 - ◆ Single view of the file system from every NAS(NFS) appliance
- Request forwarding and switching



Performance advantages

- Performance not limited to what a single NAS (NFS) head or a controller can handle
 - ◆ Clients can be effectively balanced on the front end

- Large file performance improvement
 - ◆ Both data and metadata are distributed
 - ◆ Each controller contains part (stripe) of the file

- Large file count performance improvement
 - ◆ Better data and metadata distribution results in even placement of files across the storage cluster

- Data and metadata are cached at more locations

Performance Advantages

- Faster cluster inter-connect helps with the latency & response times
 - ◆ Increasing adoption of Infiniband and 10GigE as the storage cluster backend inter-connect

- Clustered NAS solution on the market have demonstrated
 - ◆ 10GB/sec of aggregate I/O performance
 - ◆ Improved SpecSFS numbers
 - ◆ Reduce computation times

- Reduction in storage management complexity has improved storage admin efficiency

Road to NFSv4

- NFSv4 specification released in 2003 (RFC 3530)
- Design features of version 4
 - ◆ Improved access and good performance over Internet
 - ◆ Strong security and negotiation built into the protocol
 - ◆ Cross platform interoperability
 - ◆ Designed for future protocol extensions
- All auxiliary protocols and ports go away
 - ◆ No mountd/statd/lockd/; just port 2049
 - ◆ Locking integrated into the protocol; OPEN and CLOSE operations introduced

Road to NFSv4 (Contd.)

NFSv3

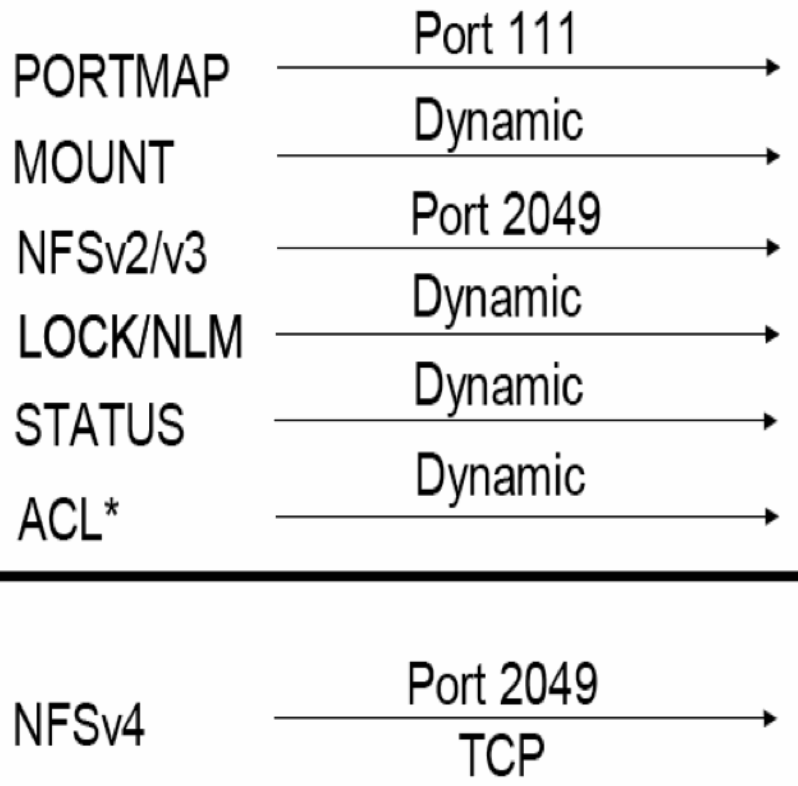
- Collection of protocols (file access, mount, lock, status)
- Stateless
- UNIX-centric (some windows)
- 32 bit numeric uids/gids
- UNIX permissions
- Works over TCP & UDP
- Ad-hoc caching

NFSv4

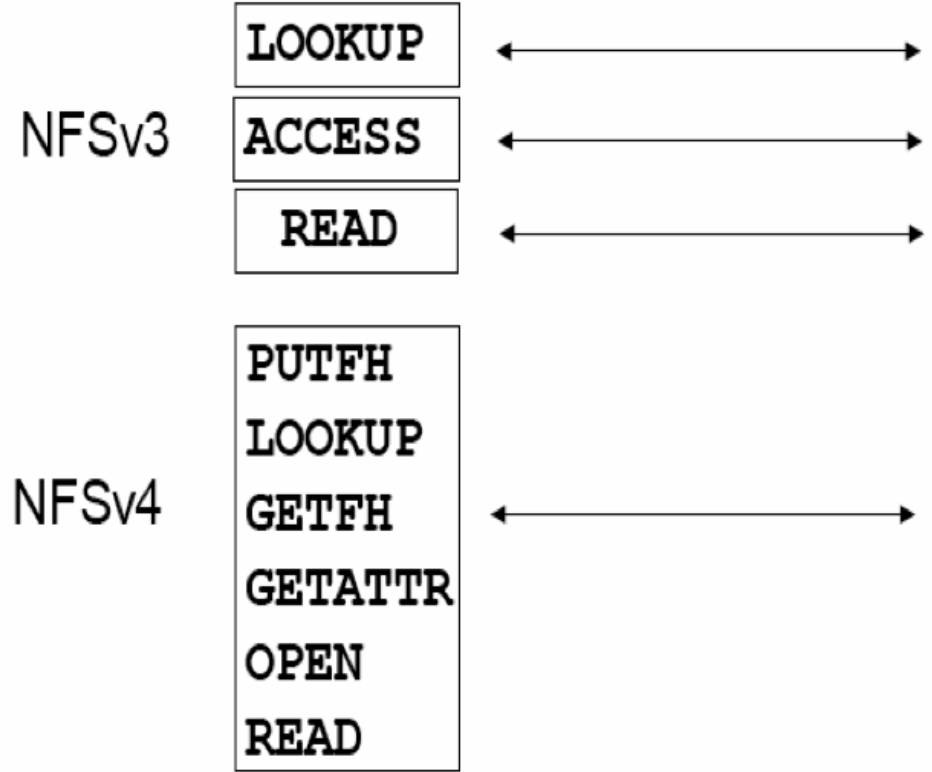
- One protocol to a single port (2049)
- Lease based state
- Support UNIX and Windows file semantics
- Mandates strong authentication
- Windows-like access
- TCP only
- Universal character set for file names

Road to NFSv4 (Contd.)

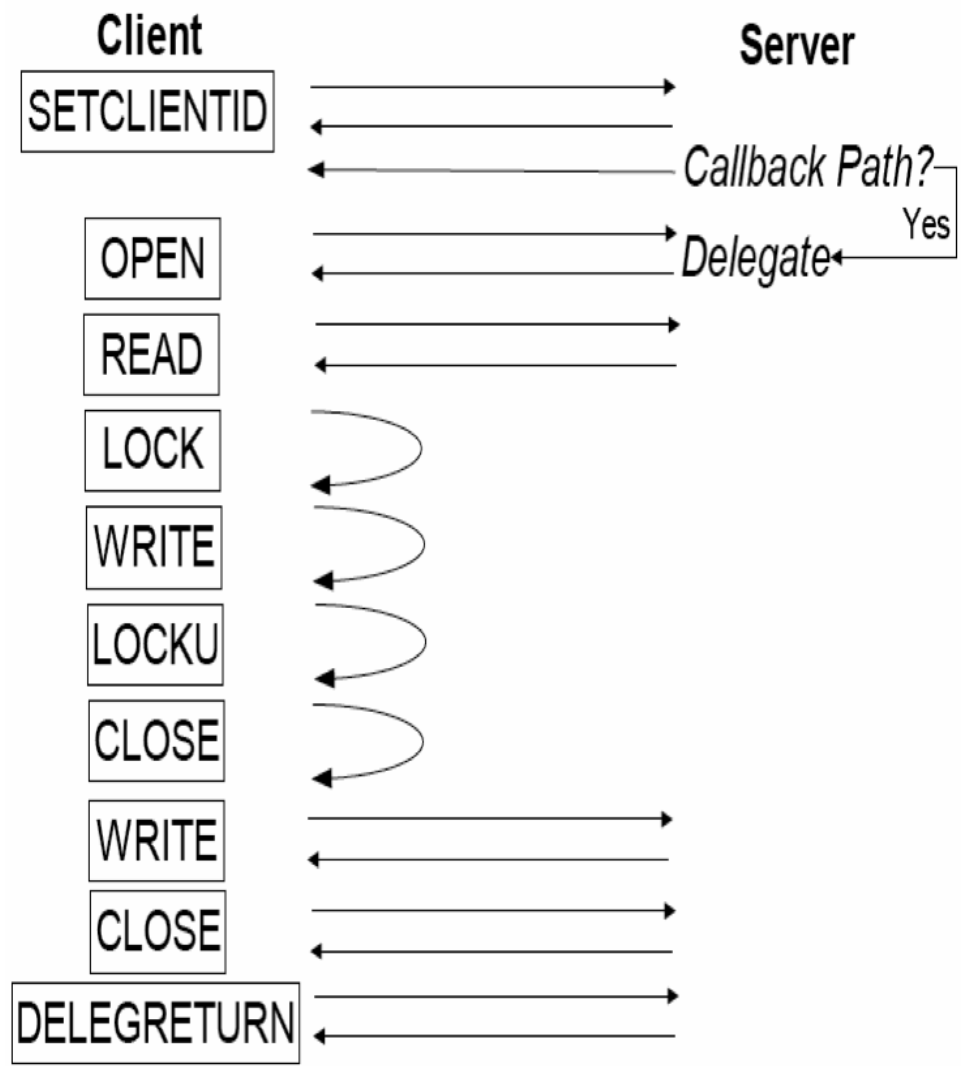
➤ Single protocol & Stateful



➤ Compound procedure



Road to NFSv4 (Contd.)



➤ Delegations

- ◆ Intended form minimal sharing environments
- ◆ Server decides when to provide
- ◆ Can be revoked via callbacks

➤ Integrated locking and lease management

- ◆ Byte-range locking/non-blocking/Mandatory
- ◆ Locks recovered at lease expiration
- ◆ Client does lock recovery at server restart

➤ NFSv4.1

- ◆ Minor version of NFSv4; does not modify NFSv4
- ◆ Delegation on directories & Session Model
- ◆ pNFS; also known as parallel NFS

➤ pNFS

- ◆ Enables parallel data access and allows NFS servers to stripe data across multiple storage devices
- ◆ Enables NFS to transparently support various diverse bank-end storage architectures
- ◆ Enables the NFS client movement of data outside of NFS-server path
 - Directly between the client and storage
- ◆ pNFS client finds out storage devices from the pNFS server through a new GETLAYOUT operation

- Defines the NFSv4.1 protocol extensions between NFS server to servers and server to client

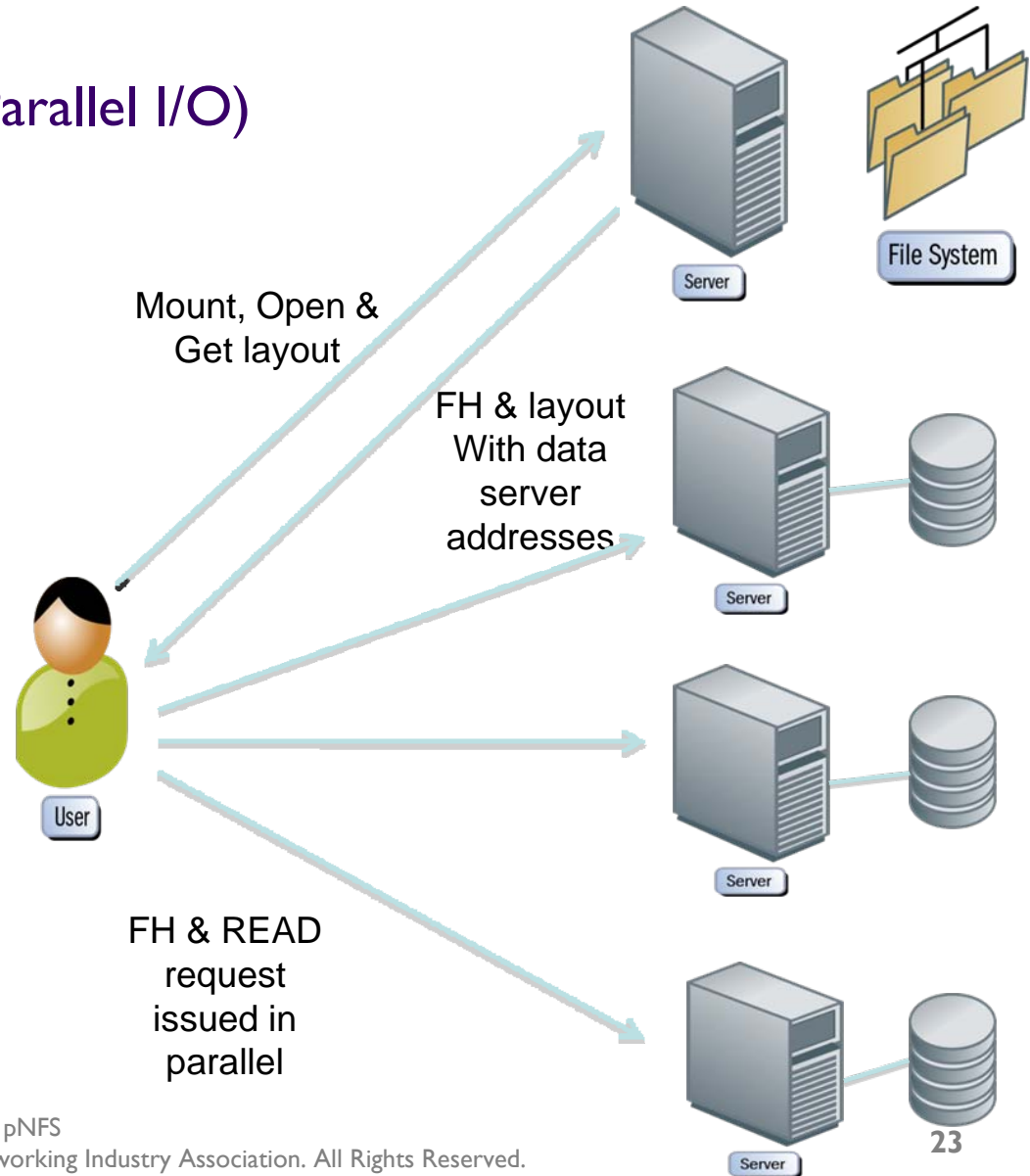
- I/O protocols between client and storage are being specified separately
 - ◆ NFS (Files access method); Proposed by NetApp
 - › Storage device is an NFS4.1 server
 - ◆ SCSI over FC (Blocks access method); Proposed by EMC
 - › Storage device is a iSCSI or Fiber Channel target
 - ◆ OSD (Object access method); Proposed by Panasas
 - › Storage target is an Object Storage Device (OSD)

- Control protocol between the server/servers and storage may be vendor private

pNFS data access model & layouts

➤ Typical client pNFS access (Parallel I/O)

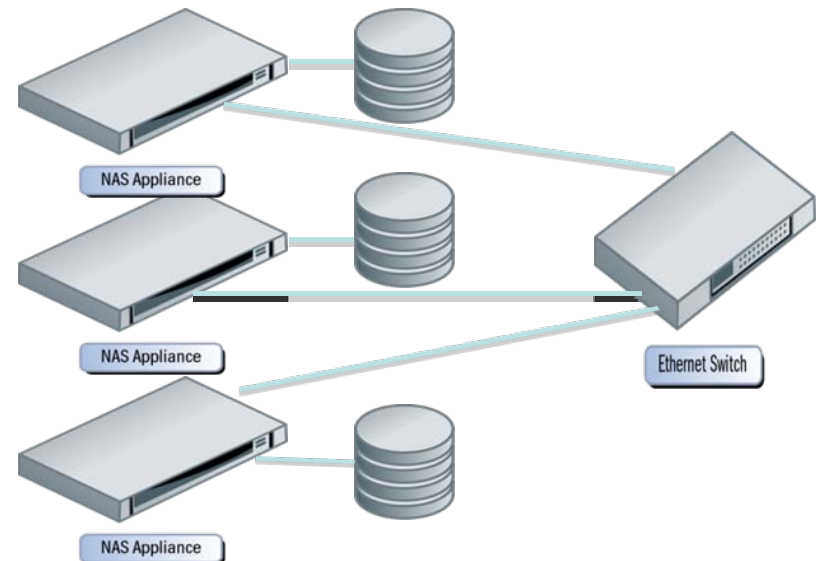
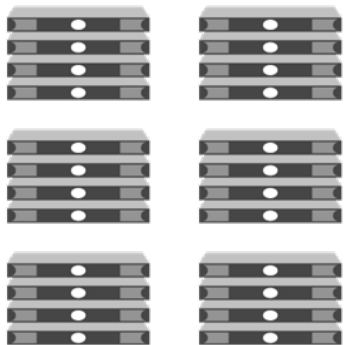
- ◆ Client mounts and opens a file on the server
- ◆ Servers grants the open and a file stripe map (layout) to the client
- ◆ Based on the layout obtained (read or write); the client can read/write in parallel directly to the data servers or what ever the storage device is



- LAYOUTGET
 - ◆ Obtains the files map from the meta-data server
- LAYOUTRETURN
 - ◆ Returns the layout; Or the new layout, if the file is modified
- LAYOUTCOMMIT
 - ◆ Servers commits the layout and updates the meta-data maps
- GETDEVICEINFO
 - ◆ Client gets updated information on a data server in the storage cluster
- GETDEVICELIST
 - ◆ Clients requests the list of all data servers participating in the storage cluster
- CB_LAYOUT
 - ◆ Server recalls the file layout from a client; if conflicts are detected

Parallel I/O with pNFS

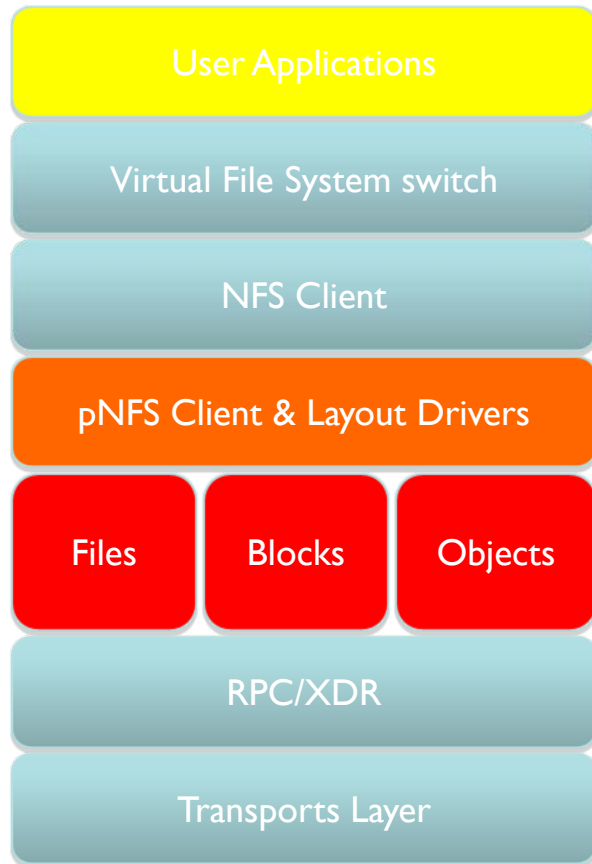
- Client obtains layout from any NAS (NFS) appliance in the cluster
- Layout is map of how the file is striped on the participating cluster nodes or the storage backend
 - ◆ Remove inter-cluster forwarding/switching
- Server can recall the layout



Load balanced compute grid accesses the storage without being switched around in the storage cluster

Every NAS controller can(cannot) be a meta-data server depending on vendor implementation and communicate file layouts

pNFS clients and Layout Driver



- Client abstraction under NFS client layer
- Layout driver for each of the previously mentioned storage backend types, e.g. files, blocks & Objects

Performance Aspects

- Client and storage only define the protocol but no transports
 - ◆ Can use IB/RDMA, iWARP/RDMA
 - ◆ ISCSI Blocks over FC
 - ◆ Any low latency protocol

- Real Advantage is flexibility and parallel I/O
 - ◆ Frame work to apply any storage backend standard based on the users needs
 - ◆ All transparent to users and clients through pNFS clients
 - ◆ Parallel access to storage; no storage cluster lookups and switching latency involvement

- At the sometime allows parallel data access for added performance

- Access to files uses standard OPEN, LOCK, ACCESS operations
- Client and server must check access rights before issuing or using layout
- If layouts/access rights are modified, pNFS NAS consistency model to recall layouts
- Credential based security models can also work
 - ◆ Cluster keys are passed to the client on layoutget from metadata server
 - ◆ On subsequent reads servers want to see it again

- Part of IETF NFSv4 minor version 1 draft
 - ◆ Currently draft waiting approval towards RFC standard

- Participants
 - ◆ NetApp (Files over NFSv4.1)
 - ◆ EMC (Blocks over SAN based architectures)
 - ◆ Panasas (Objects over OSDs based clustered file systems)
 - ◆ IBM (Files over GPFS)
 - ◆ Sun (Files and objects over clustered NAS)
 - ◆ CITI (Files over PVFS & Files over NFSv4.1)

- Reference open source clients done at CITI

- Please send any questions or comments on this presentation to SNIA: tracknetworking@snia.org

**Many thanks to the following individuals
for their contributions to this tutorial.**

- SNIA Education Committee

**Mike Eisler
Joe White**

Rob Peglar