The Parallel NFS Bugaboo

Andy Adamson
Center For Information Technology Integration
University of Michigan

Bugaboo?

Bugaboo – a source of concern.

- “the old bugaboo of inflation still bothers them”
- [n] an imaginary monster used to frighten children
- NFS: “the bugaboo of state will come back to haunt you”

The Parallel NFS Bugaboo

- Statelessness allowed NFS Versions 2 and 3 servers to export shared storage in parallel
- NFSv4 servers don't have it so easy. They have their own state to manage -- like OPEN --
- The protocol does not support distributing it among multiple servers, making it difficult to export shared storage in parallel
- The aggregate bandwidth demands of clustered clients surpass the bandwidth available with multiple parallel NFS service
- Bandwidth will be limited as long as access through the NFS protocol requires access to a single server

Outline

- pNFS
  - Won't describe pNFS, see Brent Welch's talk
  - First implementation: LAYOUTGET operation
  - Security issues
- Parallel NFS version 4 Servers on Linux
  - Which state
  - New file system interfaces?
  - Other methods

pNFS Prototype

- Work done by Dean Hildebrant, CITI
- NFSv4 on Linux 2.6.6
- Export PVFS2 0.5.1 file system
- PVFS2 suited for prototype:
  - Algorithmic file layout – layout doesn't change if number or location of disks remain constant
  - LAYOUTGET can be implemented without recall operations
- Simple file system with no meta data locking
- Used by tri-labs

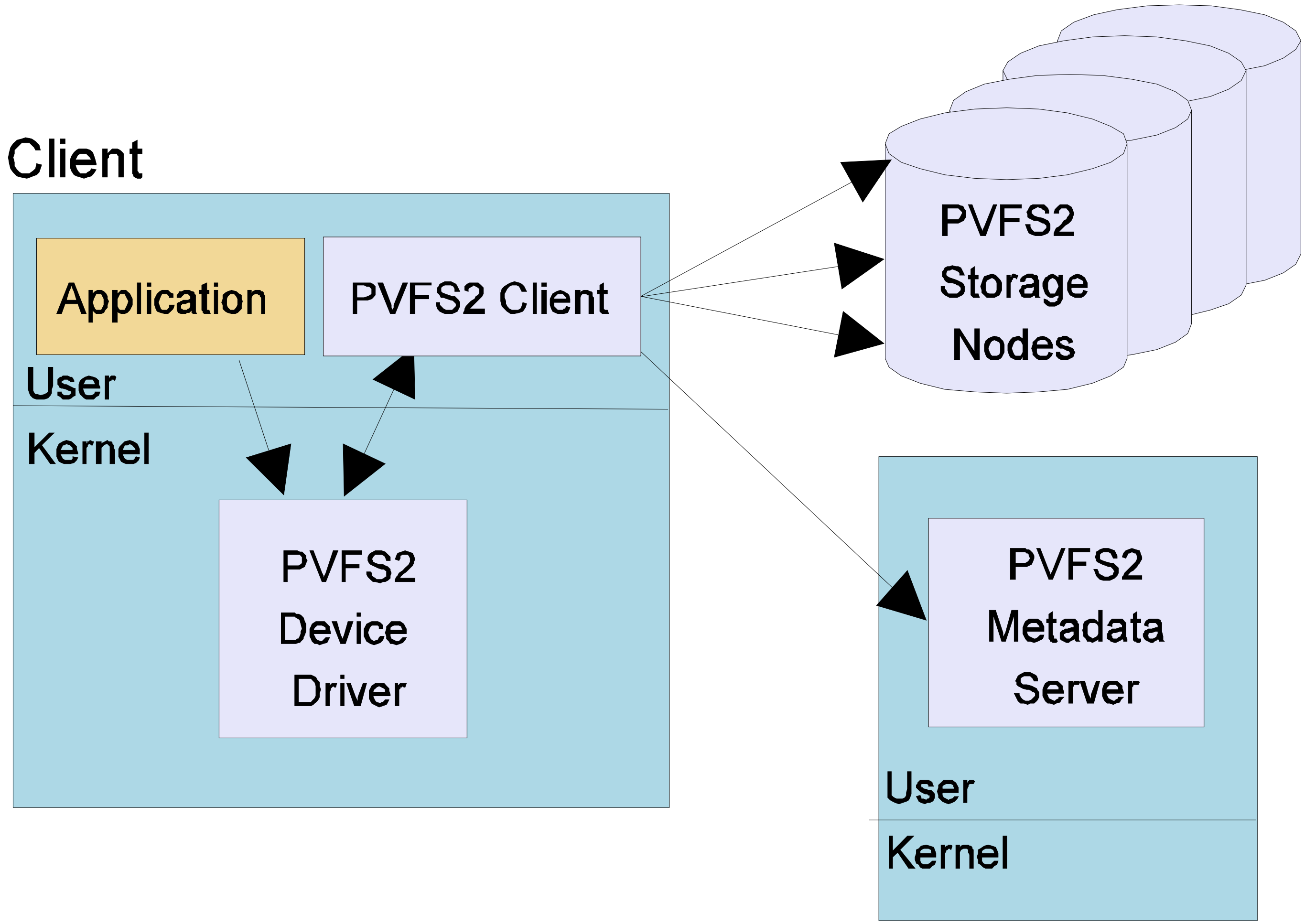
Prototype Goals

- Verify crucial portion of the pNFS protocol
  - Proposed LAYOUTGET parameters sufficient
- Validate opaque layout design
- Look at worst case: LAYOUTGET with each READ or WRITE
- Can direct access storage protocol can address the NFS 32KB limit?

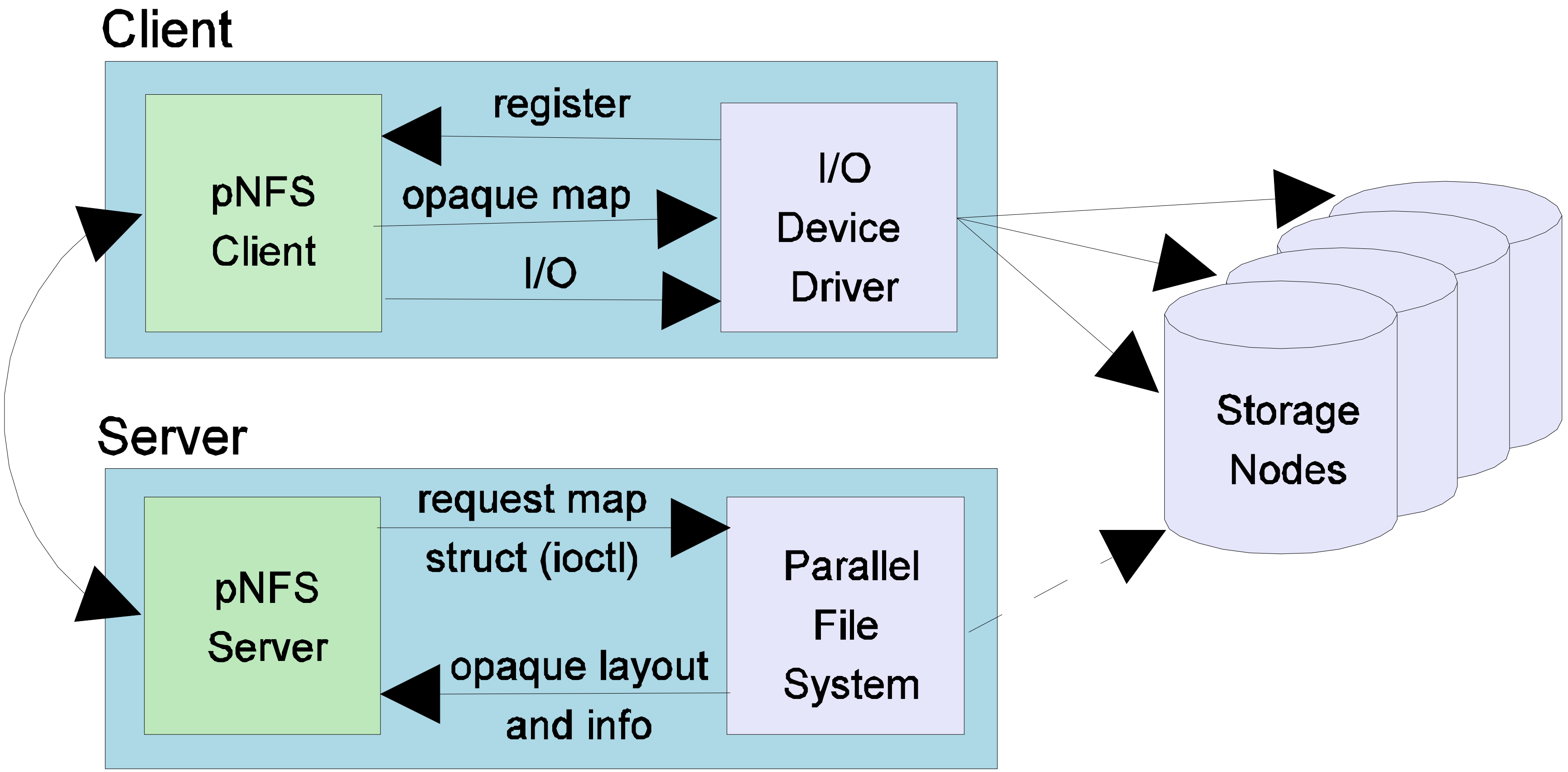
PVFS2 0.5.1 Overview

- Elements:
  - Clients (up to 10,000s)
  - I/O storage nodes (up to 100s)
  - One meta data server
- User space with OS-specific kernel module
- Algorithmic file layout
  - Currently supports round robin striping
- Simple modular design
  - No locking protocol, no caching

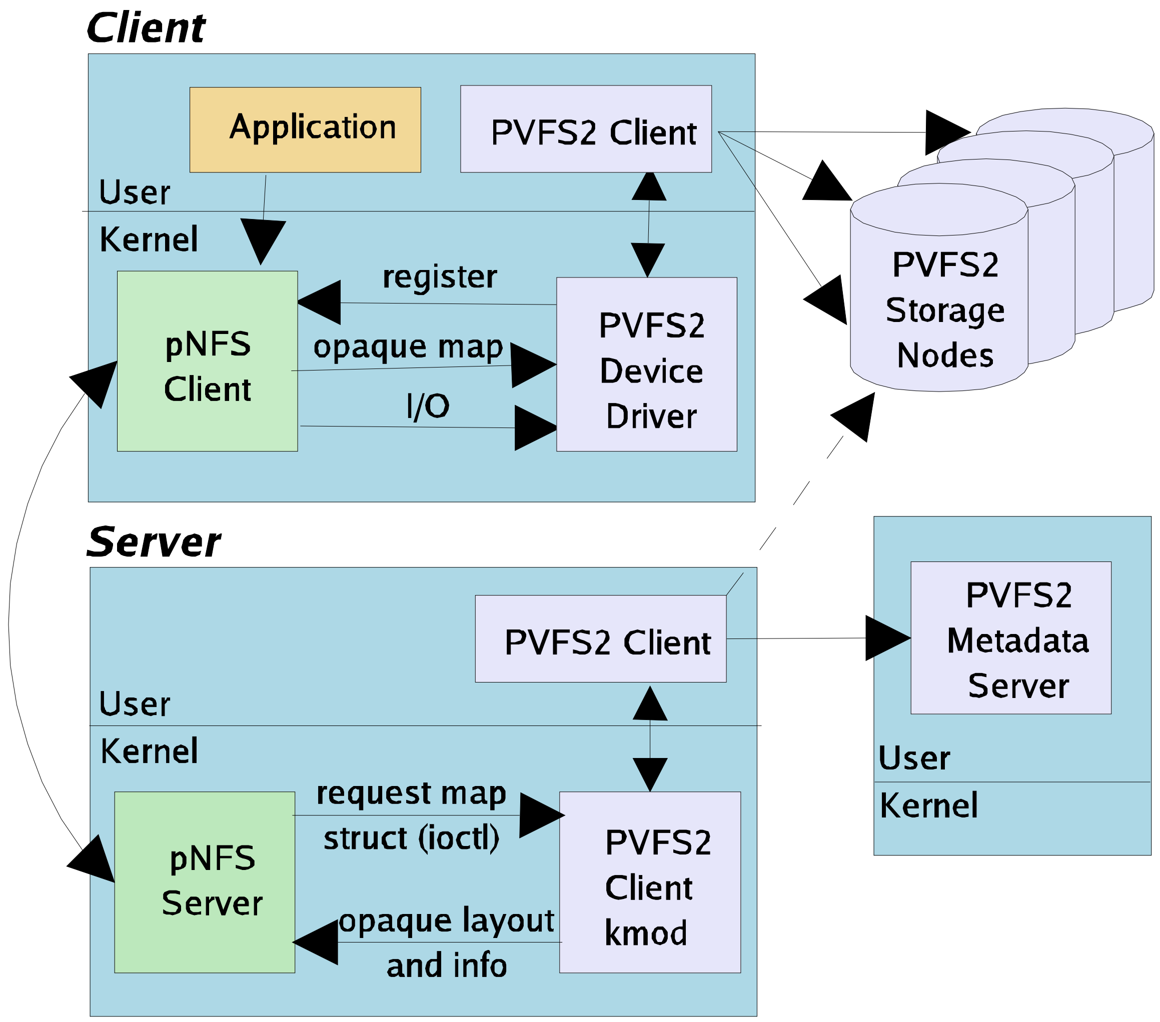
PVFS2 Architecture



pNFS Architecture



pNFS Prototype Architecture



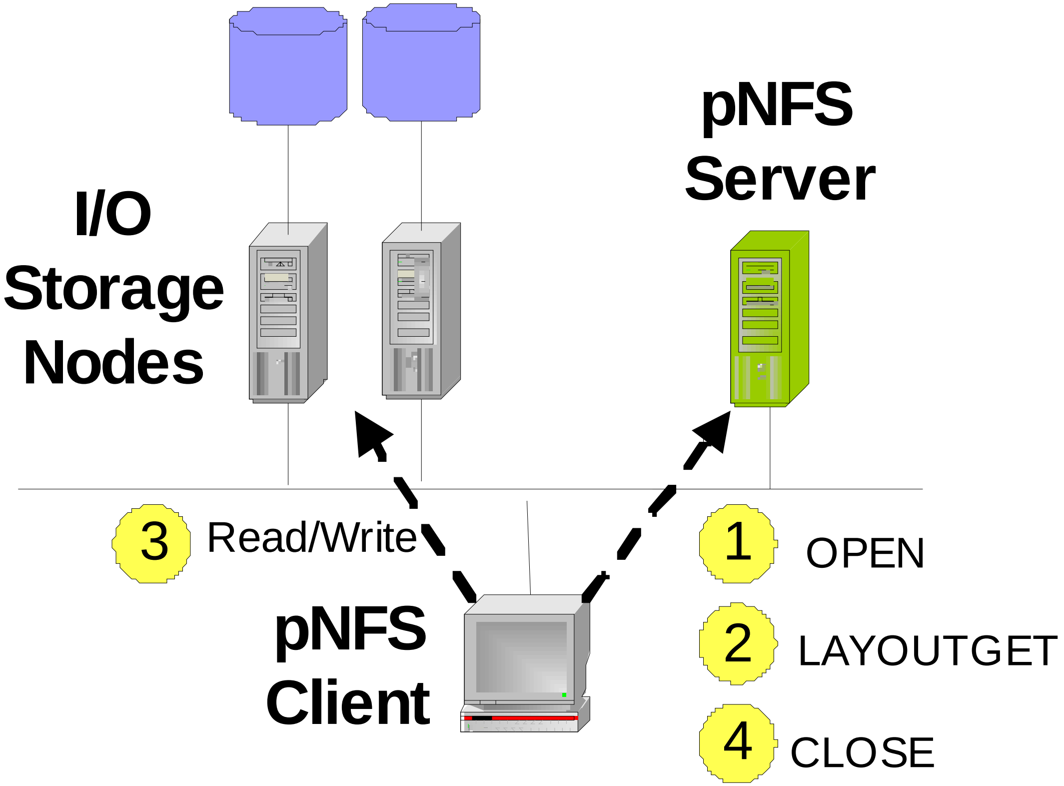
pNFS Process Flow

1
Open request

2
LAYOUTGET

3
Read/Write operations

4
Close operation



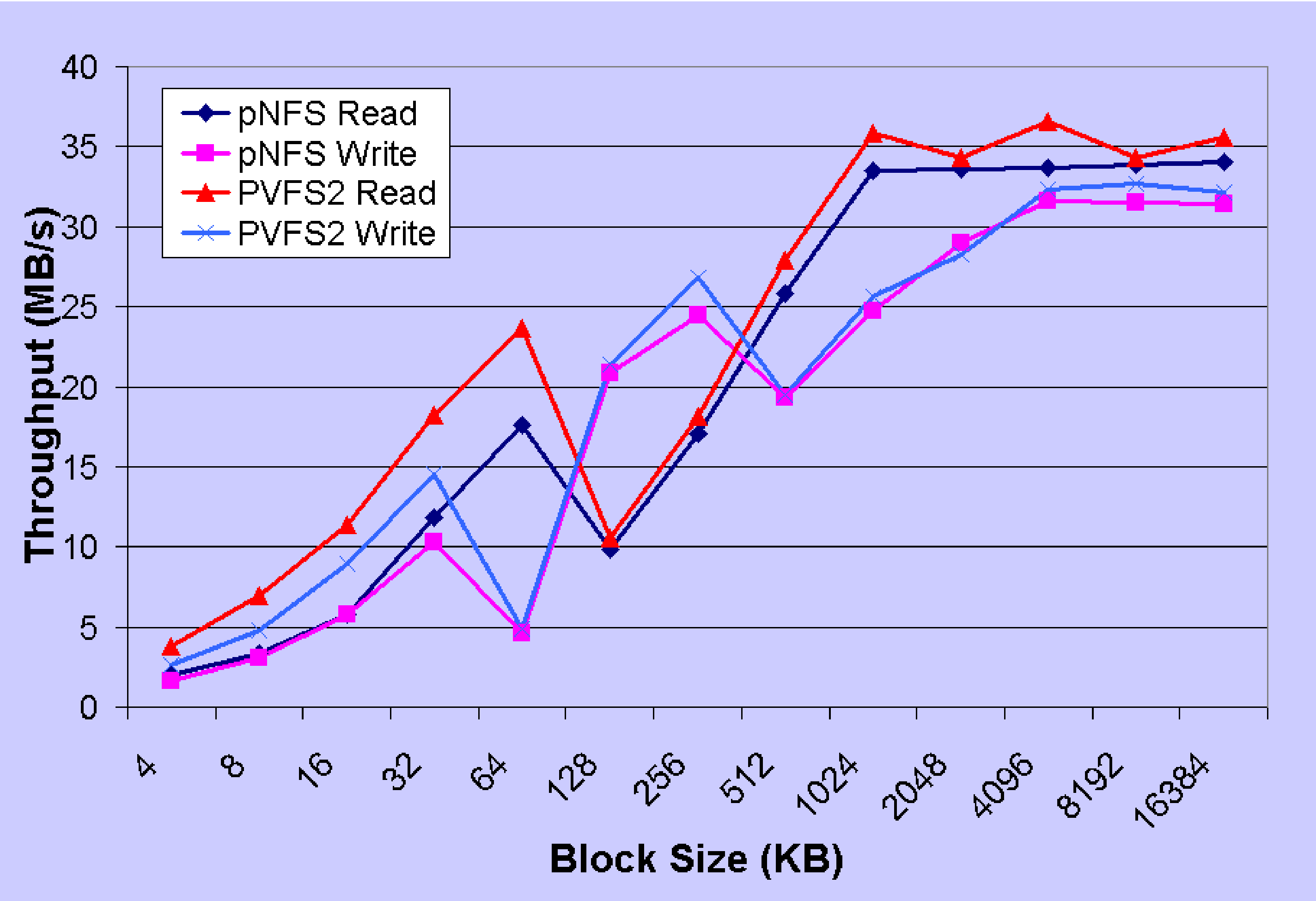
pNFS LAYOUTGET Operation

- Retrieves file layout information
- Typically a set of <device id, data id> pairs, one for each storage node
- Layout: Opaque to the pNFS client
- Layout: Understood by the driver module
- Valid until file close

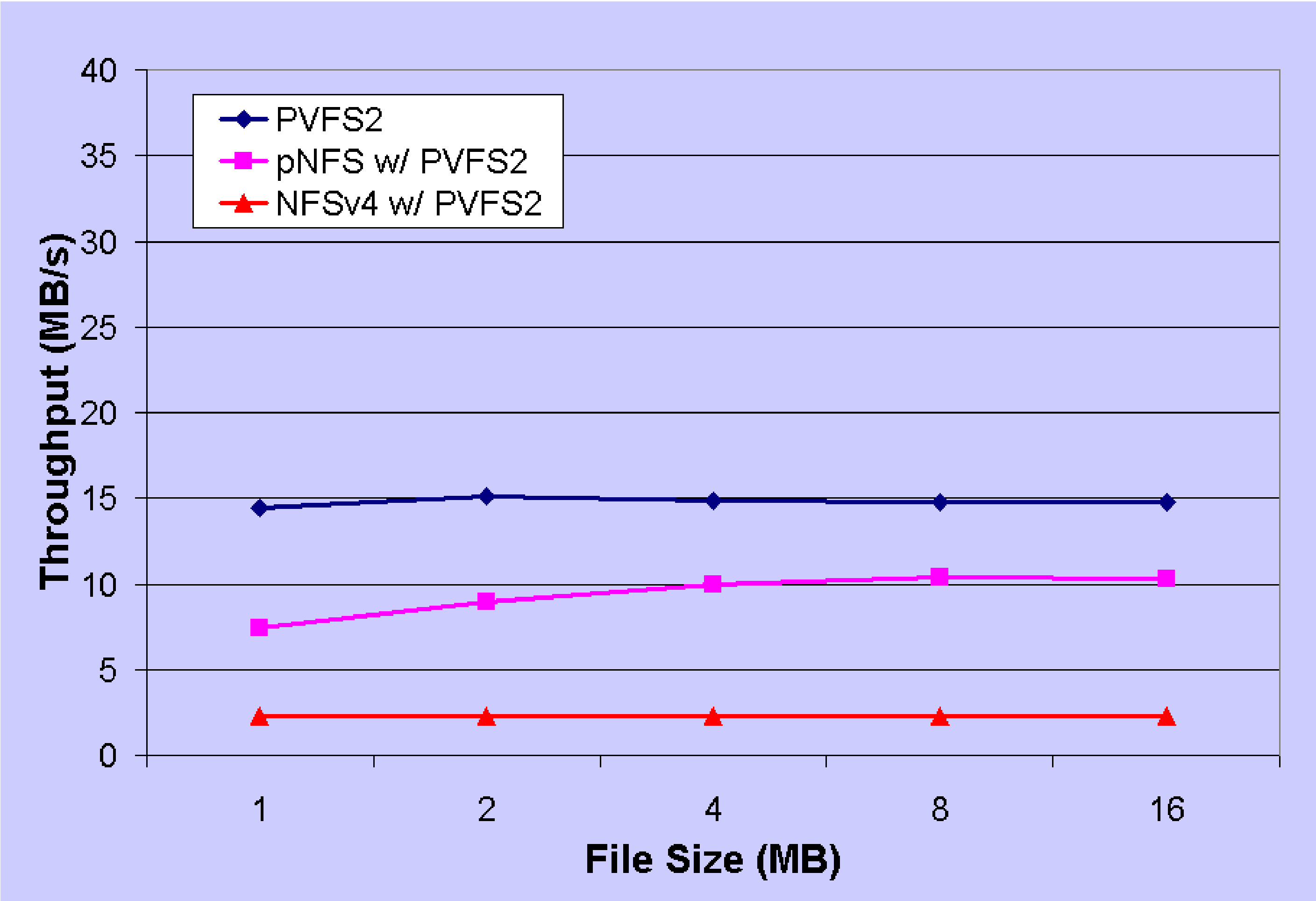
LAYOUTGET Request

	Time (ms)
Average LAYOUTGET request	1.26
Average 32KB read	2.5
Average 32KB write	2.8
Average 1MB read	28.9
Average 1MB write	29.1

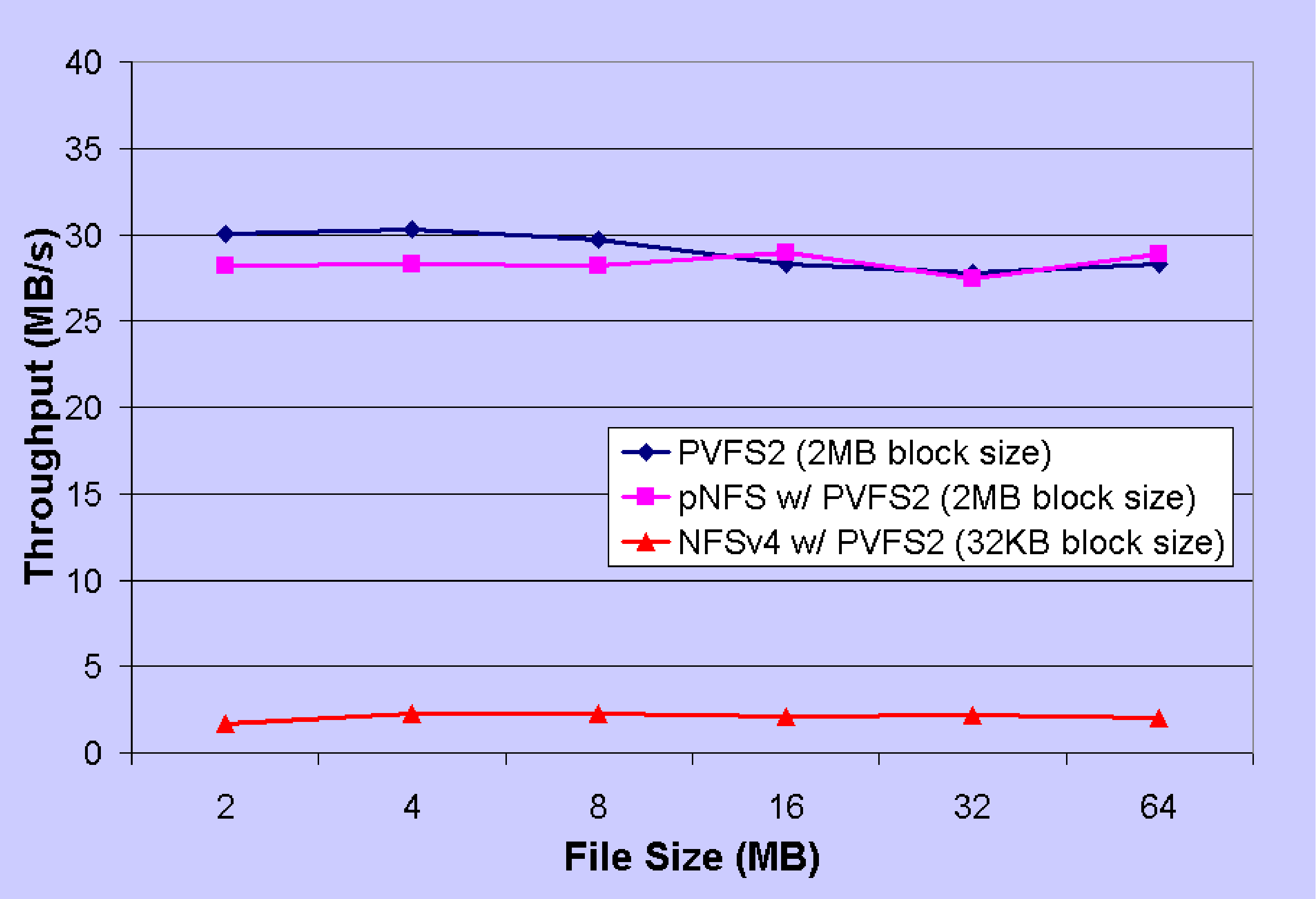
Experiment – Block Size



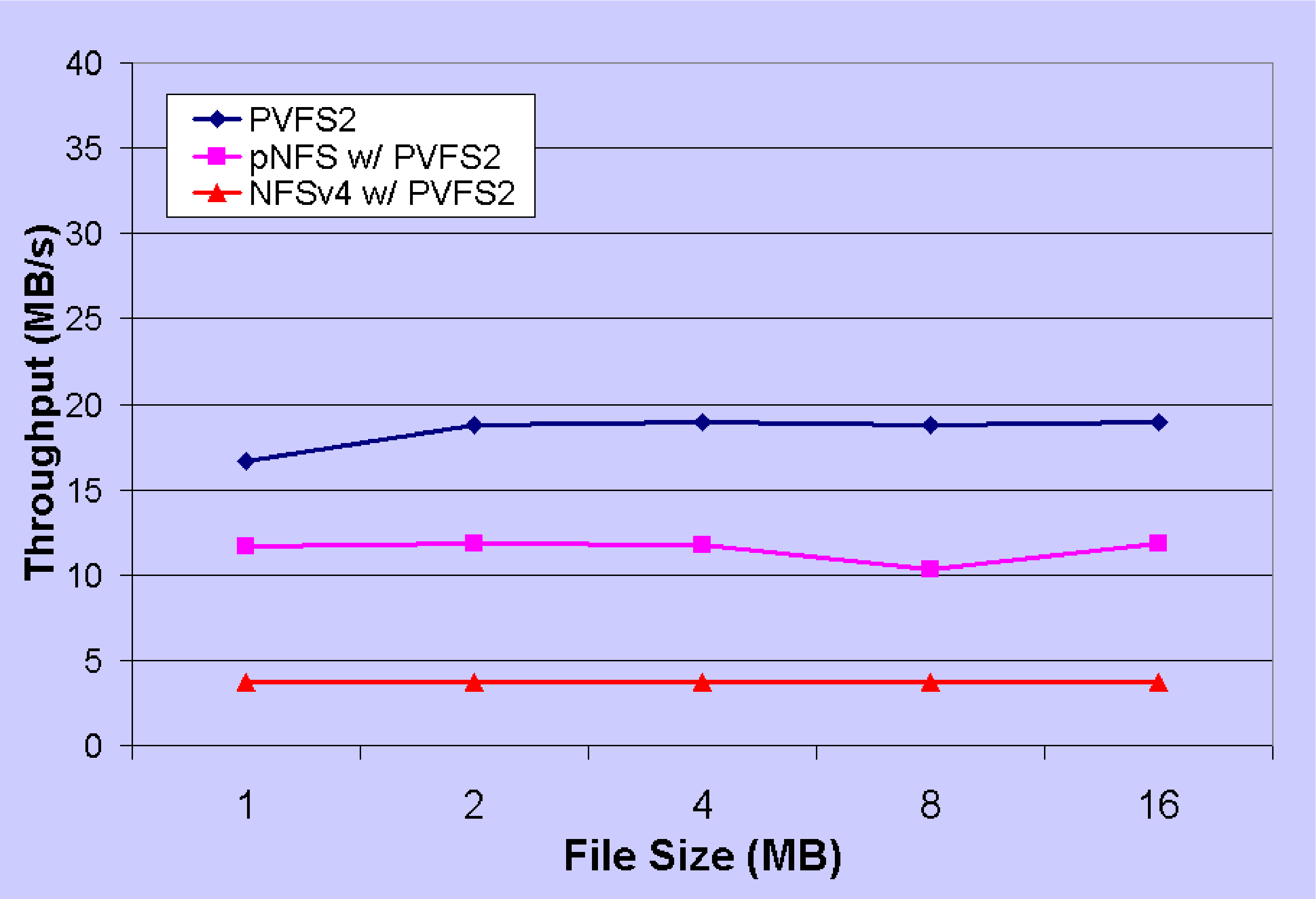
Write - 32K Block Size



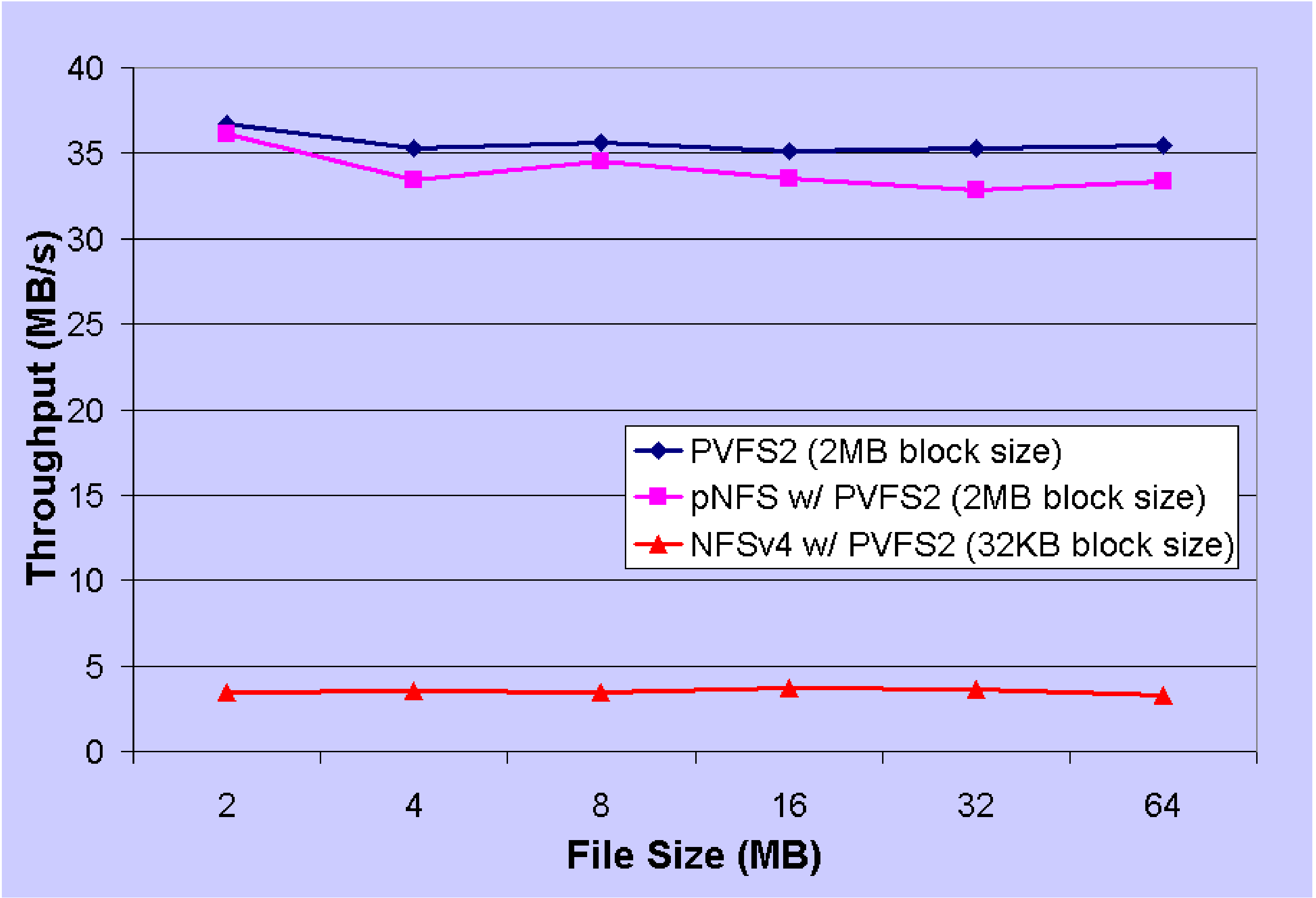
Write – 2MB Block Size



Read – 32K Block Size



Read – 2MB Block Size



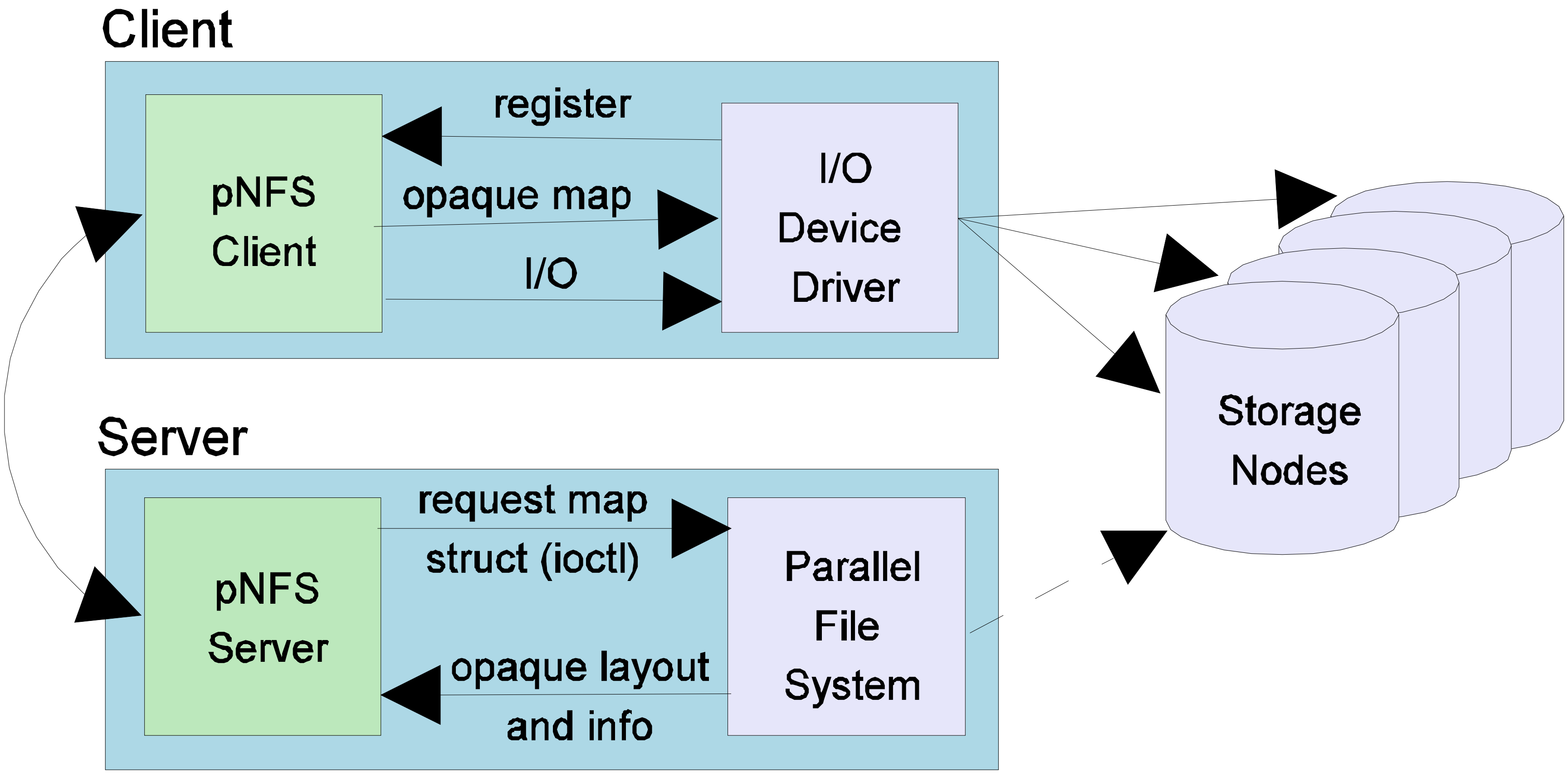
pNFS Prototype: Whats next?

- Cache layouts
- Implement CB_LAYOUTRETURN
- Implement LAYOUTRELEASE
- Continue performance testing

pNFS What about Security?

- Goal: keep all the strong NFSv4.0 security properties
- ACL and authentication checks remain
- Still able to get a GSS context between pNFS user and pNFSD
- pNFSD on the meta data service still does access checks at OPEN

pNFS Architecture



pNFS: Storage Channel Security

- No issues with AUTH_NONE, AUTH_UNIX
- RPCSEC GSS: three issues
  - RPCSEC GSS header checksum on all packets
  - Mutual authentication: if GSS context is revoked, need to stop storage access for the user
  - Data integrity and data privacy enforcement
- Each layout type will have a different security story
- pNFS working group just beginning to detail the possible solutions

OSD Channel Security

- OSD capabilities:
  - Meta data service and storage nodes share keys (setup out of band)
  - Keys sign a per object capability
- Capability signs each OSD command to storage
  - similar to RPCSEC GSS header checksum
- Associate capability set to a GSS context
  - If GSS context disappears, capabilities are revoked
- Data integrity, privacy: rely on underlying transport, perhaps IPSEC

Block Storage Channel Security

- Relies on SAN-based security: trusts that clients will only access the blocks they have been directed to use.
- Fencing techniques:
  - Heavy weight per client operations – not per user.
  - Not expected to be a part of normal pNFS execution path
- Need to rely on underlying transport for all security features (IPSEC)

pNFS Security

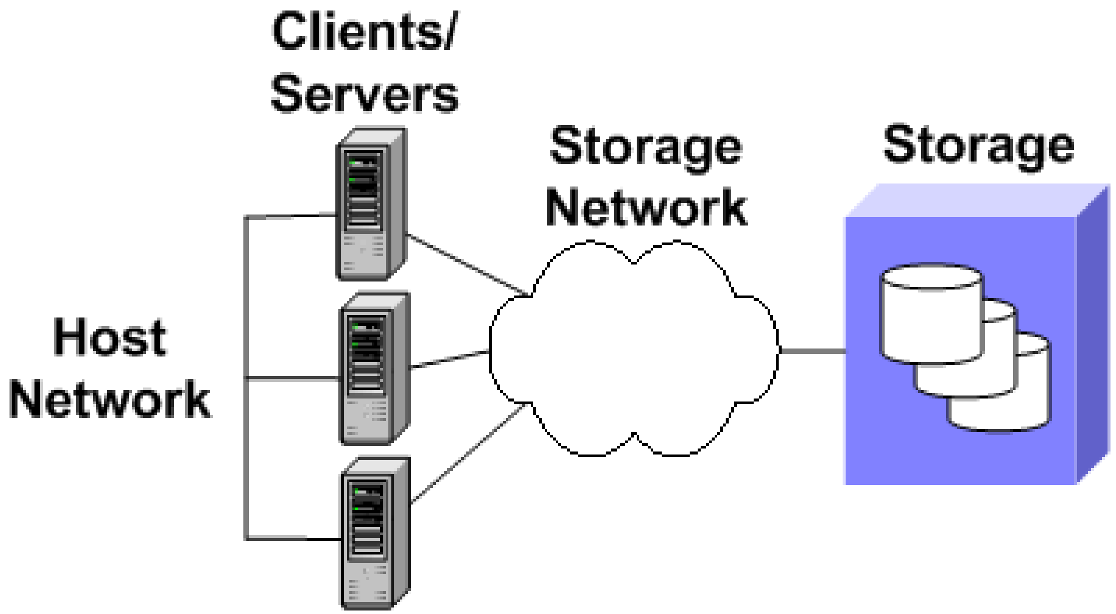
- Many issues related to security
- pNFS working group is moving toward in depth security discussions

Parallel NFS version 4 Servers on Linux

- Problem: How to share NFSv4 state between NFSv4 servers
  - Not part of the pNFS protocol
- Problem: Exporting NFSv4 on cluster file systems
- Problem: Supporting multiple meta data servers on parallel file systems

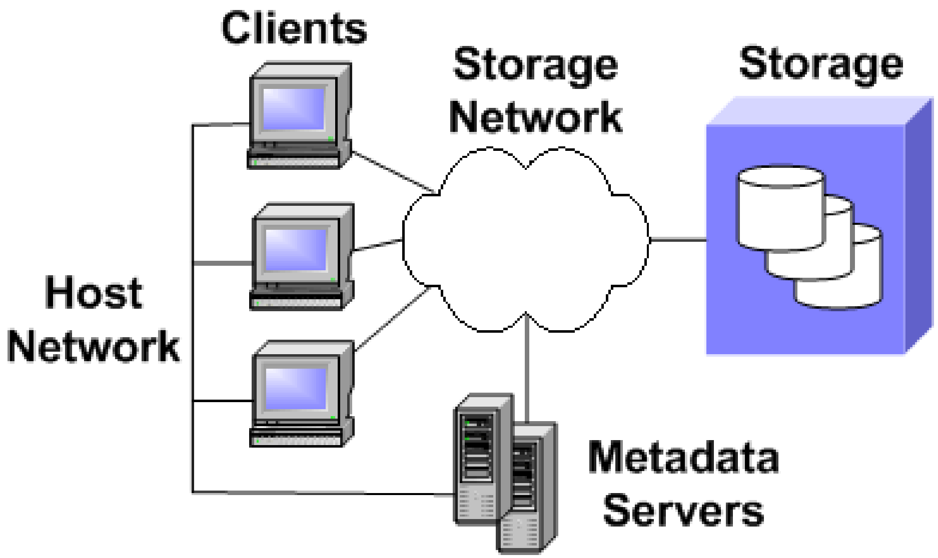
Cluster File Systems

- "Symmetric Out-Of-Band"
- Every node is a fully capable client, data server and meta data server
- Examples: IBM GPFS, Redhat GFS, Polyserve Matrix Server



Parallel File Systems

- "Asymmetric Out-Of-Band"
- Clients access storage directly, Separate meta data server(s)
- Object Based: Lustre, Panasas ActiveScale
- Block Based: EMC's High Road, IBM SAN FS
- File Based: PVFS2



NFSv4 Server State Sharing Methods

- Via new server to server protocol
  - Could conflict with existing meta data sharing architecture
- Redirect all OPENS for a file to a single NFSv4 server
  - Need to track which server is handling OPEN
  - Large number of clients opening a file is problematic

NFSv4 Server State Sharing Methods

- Via underlying file system
  - Need additional file system interfaces and functionality
  - Uses existing file system meta data sharing architecture
  - Coordinates with local access

Potential NFSv4 Server State to Share

- ClientID
- Open owner, Open stateID (Share/deny locks)
- Lock Owner, Lock stateID (Byte-range locks)
- Delegation stateID and call back info

Simplifying Assumption

- Assumption: An NFS client mounts only one server per parallel or cluster file system at a time
- Allows for clientIDs, open owners, and lock owners to be kept in memory on the single server

File System Queries for Share Locks

- NFSD: Upon each OPEN ask the file system if any other NFSD has a conflicting share lock
- File system will need to add book keeping for deny access (for Windows clients)
- NFSD: OPEN stateID can be created and used as usual.

File System Queries for Byte-Range Locks

- NFSD: Upon each LOCK/UNLOCK ask the file system to manage the locks.
- There is an effort underway led by Sridhar Samudrala (sri@us.ibm.com) to expand the existing file_operations lock call to include enable NFS locking over clustered file systems.
- Useful for both LOCKD and NFSv4 server
- LOCK stateID can be created and used as usual

File System Queries for Delegation Support

- Hand out a delegation
  - count readers/writers
  - check if in recall state
- Recall a delegation
  - register a recall callback
  - receive a recall request
- Linux file_lock FL_LEASE has this functionality

Discussion

- The combination of pNFS and parallel NFSv4 server state sharing can solve the 'parallel NFS bugaboo'
- pNFS effort is well underway
- Parallel NFSv4 server state sharing is left as a per OS solution
- CITI will prototype file system extensions to enable parallel NFSv4 server state sharing on Linux

Any Questions?

http://www.citi.umich.edu/projects