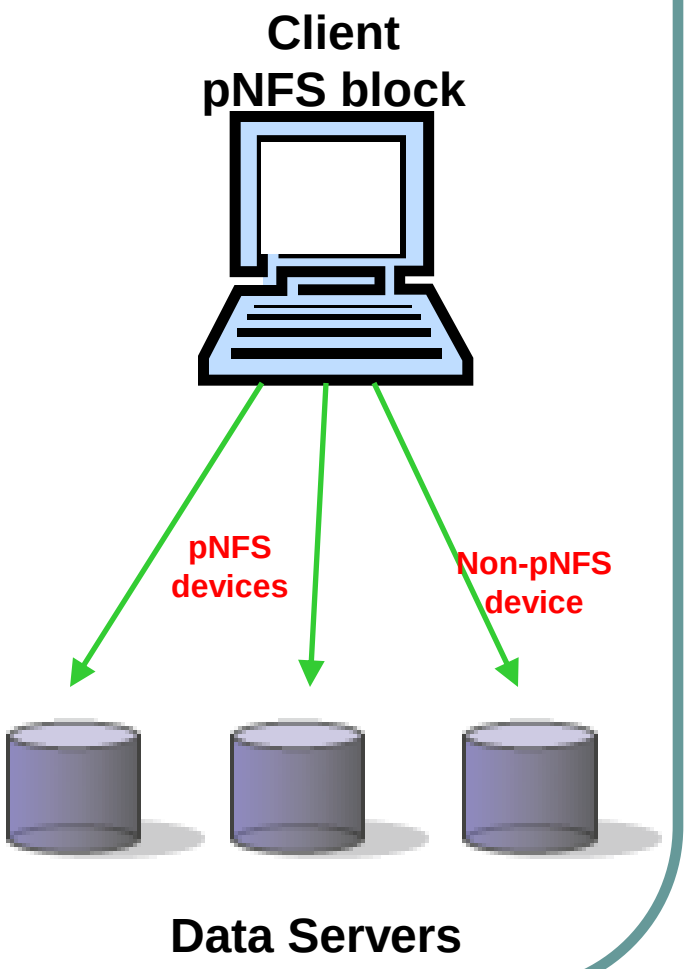
pNFS block signature/disk protection

WG Interim Meeting

Problem Statement

- Client cannot differentiate pNFS devices from non-pNFS devices
- pNFS devices are discovered after mount time via GETDEVICEINFO
- Before pNFS mount kernel/apps may write to pNFS SCSI devices and destroy pNFS FS
- No way to protect pNFS devices when there are multiple paths to same device
- There is no protocol way (5663) to identify devices used by pNFS
- Problem observed when complex volumes support was implemented



Additional Issues

- Current block protocol signature defines deviceid but no pNFS specific name
- Location of signature only communicated to client on GETDEVICEINFO call at mount time
- Even if there was a pNFS specific signature it cannot be found before mount; client doesn't know the MDS IP
- Signature can be vendor specific but client doesn't know it relates to pNFS
- Even if location is known, there is no write protection mechanism for pNFS devices

Possible Solution

- Need to extended RFC5663
  - Define fix location/s (configuration)
  - Define pNFS signature: “pnfs block device”
  - Recommend protection mechanism
- Will have to be vendor independent (generic name)
- Extend signature to indicate that a disk is used by pNFS
- Use offset (from beginning or end of disk), a length, and an array of bytes same as deviceid
- Possible use multiple matches to uniquely identify a pNFS device

Different Approaches

1. Clients can have configuration file that specifies the signatures to protect.
2. Clients could come pre-configured with a protection file that recognizes "most signatures"
3. pNFS servers could define a function that will return such signatures. Every time a client does a mount, the client could retrieve the signatures (will be maybe too late) , and match the configuration file.

WG Asks and Q&A

- WG Asks:
  - Extension to block or 4.2 feature
  - Can use a configuration file
  - Can use fix location
  - Can add pre-mount call

Servers implementation strategy

- Volatile file handle
- Named Attributes
- Session trunking
- Clientid trunking
- Multi-segment layouts
- Directory delegation
- CB's NOTIFY, DEVICEID
- FS location

Servers implementation strategy(2)

- current_stateid
- MACH_CRED
- DESTROY_CLIENTID
- FREE_STATEID
- SECINFO_NO_NAME
- TEST_STATEID
- Persistent stateid (small files perf.)
- ACL retention bits

pNFS over IPv6 (problem)

- MDS and DS's support both IPv6 and IPv4
- The mount and MD operations over the IPv6
- I/O operations were done over IPv4
- When configured DS to support only IPv6 the clients hang
- Current Linux client hang if DS supports only IPv6
- Problem: the layout includes only IPv6 but client only access via IPv4.

pNFS over IPv6 (2)

- If DS IP in layout 1st item is IPv4 then client perform the I/Os to DS
- MDS sends only one IP in the layout for each DS using IP addresses from list
- If MDS selects the IPv4 for DS from list: OK
- If MDS selects IPv6 for DS: clients hang
- Client doesn't retry the I/O to the MDS: it should
- IPv6 with pNFS block server works correctly as the I/O's are done to iSCSI on either IPv4 or IPv6 as configured

pNFS over IPv6: solution

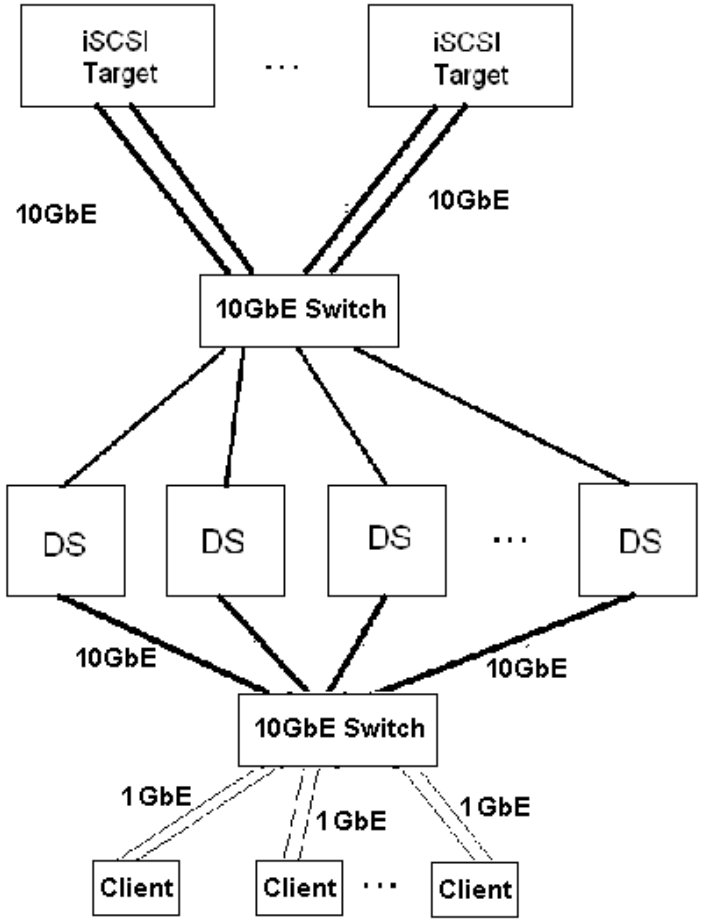
- Correct behavior according to RFC:
  - if the client cannot access DS via IPv6 should fall back to use IPv6 on MDS
  - If DS supports both IPv4 and IPv6 and client can mount the DS using IPv6 it should use IPv6 for I/O to DS's
  - If client mount using IPv6 the MDS there is no guarantee about what IP will use to access the DS for I/O

pNFS over IPv6: alternatives

- Server send a list of IPs supported by the DS in the layout; client select the one of choice (concern on size of layout)
- Client to do a layoutreturn with new connectivity error code (concern on additional operation)
- What is the preferred solution(?)

Performance testing of pNFS

- Need to have testers
- Benchmark to use:
  - IOzone
  - Netmist (like sfs2008)
- What configuration
- INcast effect?
- What trunking
- What security



pNFS Performance Testbed Diagram

What we need to test

- Shared file access in throughput mode with clients that are accessing by region, or by interleave (test stripe).
- Benchmark have different impact on server behavior:
  - striping
  - allocation policies for small files
  - sparse files behavior
- Define specifications for benchmark
  - Performance
  - Scalability
  - ?
- Netmist (Don Capps) will implement what we define.
- Will create new SPEC benchmark for pNFS

General testing strategy

- What functionality to be tested by extended “cthon” or similar tools
- Discussed the possibility to create a new/different tool than cthon (BATon)
  - New license issue
  - Will be open source
  - GPL?

General testing strategy (2)

- Can include the modified/enhanced IOzone as the base for the new BAThon:
  - Performance
  - Functionality interop
- Gradually replace the cthon with BAThon
- Extend to include new functionality:
  - CB
  - New/addl lock tests

Multi-layout pNFS server support

- RFC5661 allows pNFS servers to implement support for multiple types of layouts by same server
- Can/should support access to same FS?
- Can/should support access to same file?
- Can/should allow access by same client to same file?
- Can consistency be preserved across different layouts in a shared model?

Multi-layout pNFS issues

- Several servers that support 2 types of layout but in 2 different MDS's
- There are possible lock issues for data access and range locking
- Multiple clients shared access to same fsid using 2 MDS's with different pNFS layout types (RFC allows)
- The protocol is well defined for this but nobody really thought to which extent the shared implementations can go
- Do we really expect servers to support multiple layout in shared mode?

Multi-layout pNFS actions

- Clients want access same file on an MDS that support multiple layouts on same fsid to access same file
- Should we add some implementation recommendations
- Define usecases
- Possible extension to all the 3 RFC5661, 5663 and 5664
- Should we prototype implementation on Linux server/client to validate shared access?