Parallel NFS - pNFS

Parallel, Shared File System

NFS v4.1, Standard

Schubert Zhang

Jun, 2008

Summary

- pNFS 本身的目的是为了规范并行文件系统在 Client 端的使用接口和使用方式，从而借助标准化的 NFS 使并行文件系统的到推广。
- pNFS 用对 NFSv4 进行扩展 (NFSv4.1) 的方式定义了 Client 和 Metadata Server 之间的协议。但 Client 和 Data Server 之间的协议还需要用户选择 (目前推荐三种，可以扩展) 。 Metadata Server 和 Data Server 之间的控制协议也没有定义，需要用户根据自己选择的 Storage Layout 类型来决定和定义。即 pNFS 把内部 Componets 之间的协议留给了 pNFS Server 的实现者。
- 需用户选择合适的 Storage 方式 (File, Block, Object) 。
- Metadata Server 的可靠性和分布式方式没有定义，用户得为 Metadata Server 选择合适的 Distributed/Cluster File System 或者实现 HA 机制来保证负载均衡和可靠性。
- 支持 File Striping.
- 无 replication.

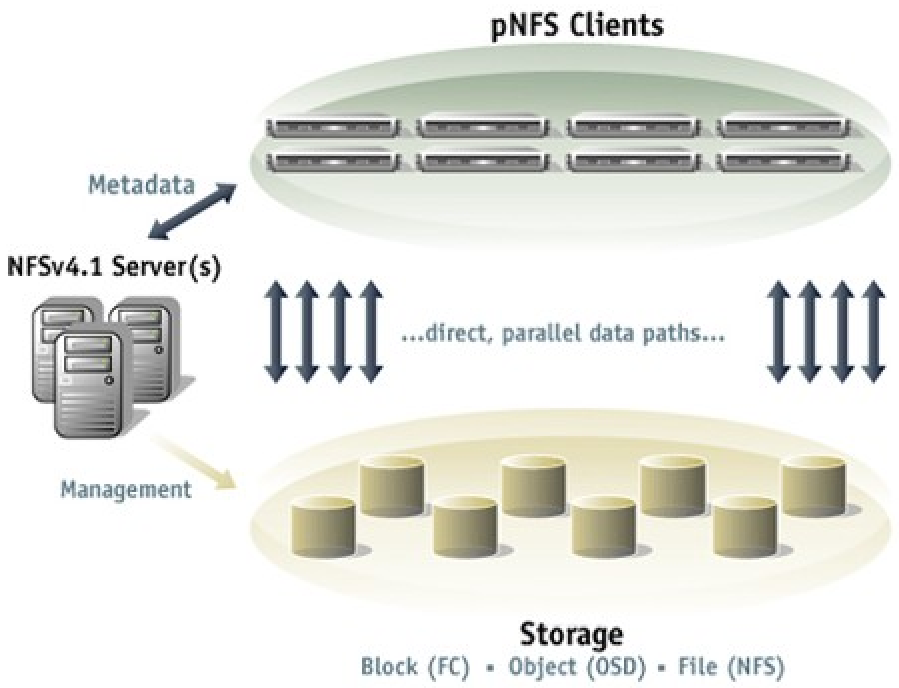
What pNFS is

- Based on NFSv4 and an extension to NFSv4.
- Separation of data and metadata.
- Moving the metadata server out of the data path.
- Benefits
  - Parallel I/O
    - Delivers Very High Application Performance
    - Allows for Massive Scalability without diminished performance
  - Use the ubiquitous standards for NFS
    - Eliminates risks of deploying proprietary technology
    - Backward compatible
    - No changes to applications.
    - pNFS => The new NFS standard, NFSv4.1

Design Principles

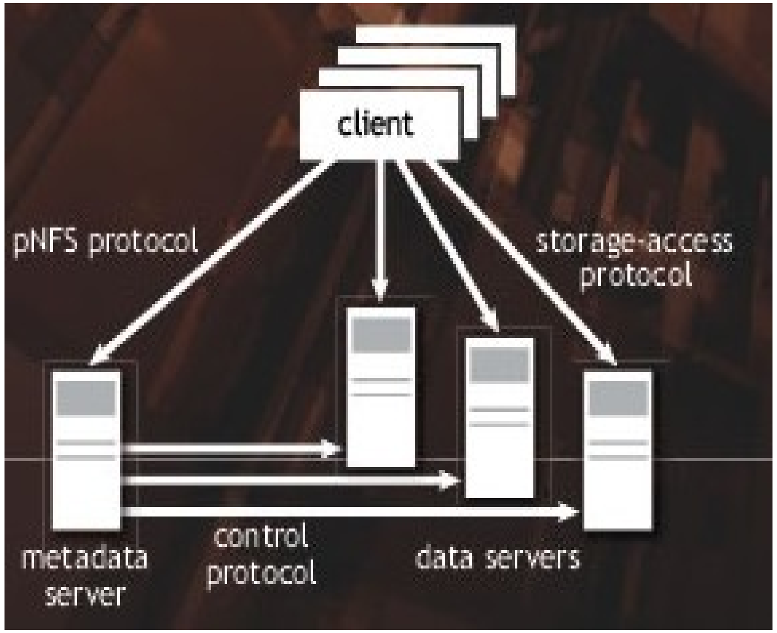
- Familiar Semantics
  - The pNFS protocol must provide consistency and security semantics similar to the base NFSv4 protocol. This simplifies the adoption of pNFS by the diverse set of existing NFS applications.
- Simplicity
  - The pNFS extension must be kept simple, yet efficient. The pNFS interface must provide the most basic primitives needed to exploit data parallelism efficiently, while allowing the layering above of additional, more complex semantics.
- Flexibility
  - The pNFS protocol must be flexible enough to accommodate a variety of storage architectures. These architectures include a federation of stand-alone file servers, a mix of a NAS file server and RAID storage devices, and a fully transparent parallel clustered file system.

Architecture


pNFS Clients
Metadata
NFSv4.1 Server(s)
...direct, parallel data paths...
Management
Storage
Block (FC) • Object (OSD) • File (NFS)

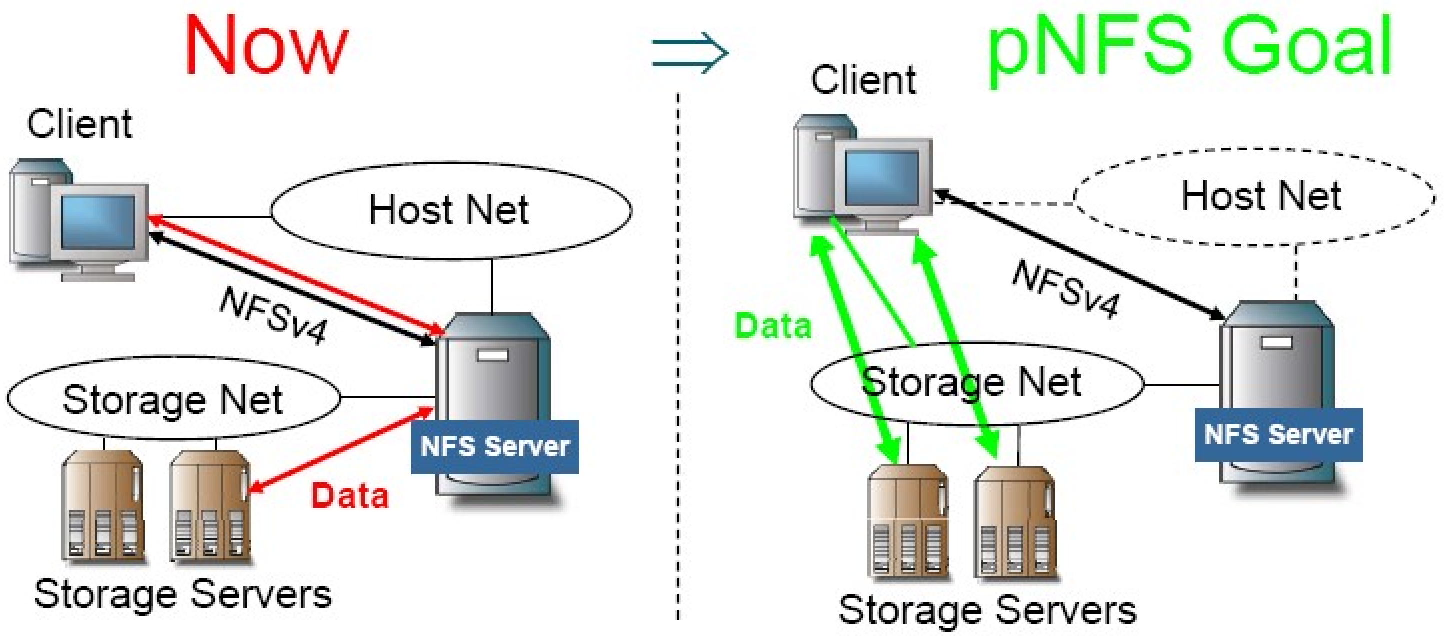
- Components
  - One metadata/control server
  - Some number of data/storage severs
  - Some number of NFS clients
- pNFS supports 3 data-access protocols (3 layout types
  - Block (SAN/iSCSI, FC)
  - Object (OSD)
  - File (NFS)

Architecture



- Protocols
  - pNFS protocol
  - storage access protocol
  - control protocol
- Metadata server
  - file layout
  - all directory metadata
  - file attributes related to the striped files
- For each file, the metadata server maintains a layout:
  - the location
  - striping parameters
  - storage access protocol for each stripe
  - any other tokens required to identify that file on the remote data servers.
- File data striping
  - typically round-robin

pNFS vs. NFS


Now
⇒
pNFS Goal
Client
Host Net
NFSv4
Storage Net
NFS Server
Data
Storage Servers
Client
Host Net
NFSv4
Data
Storage Net
NFS Server
Storage Servers

How NFS works

- Standard NFS file servers with local disks keeps track of file metadata (inodes).
- Work flow
  - Client requests files to NFS server.
  - NFS server looks up metadata to locate the disk location.
  - NFS server collects data via disk I/O, and sends back to client.
- With small files most of the time is spent collecting the data.
- With big files the data transmission time becomes the limiting factor.

How pNFS works

- Together the metadata server and the data servers form a single logical NFS server with a slew of network.
- The client node use the new v4.1 NFS client.
- File-access work flow
  - Client opens a file and obtains the file’s layout from the metadata server; the layout describes where the file’s data is located.
  - The client then performs I/O in parallel to the data servers described by the layout.
  - Once the client completes its I/O, it commits modified metadata and closes the file at the metadata server.
- Scalability
  - Adding new data servers.
  - Aggregated speed/throughput increasing close to linearly.
- Backward compatible
  - Metadata server can gather data itself and send back to client.
  - Applications will never know the difference.

Protocols & Interfaces

- pNFS protocol (clients <-->metadata server)
  - NFSv4
  - A few additional operations for clients to query layouts and obtain data server locations.
- Storage-access protocol (clients <-->data servers) (user choice)
  - Blocks (SAN/iSCSI, FC)
  - Objects (OSD)
  - Files (NFS)
  - …
- Control protocol (metadata server <-->data servers)
  - Synchronize state between the metadata server and data servers
  - Deliberately left unspecified to provide flexibility to server implementations by making the back end of the storage system free to choose when and how to synchronize file metadata between the metadata and data servers.

Storage Layout Types (File, Object, Block)

- File Layouts
  - Storage-access protocol: NFSv4
  - Clients must use the NFSv4.1 read and write operations to communicate with data servers.
  - For each data server listed, the file layout contains:
    - the size of each stripe
    - the NFS file handles to be used
  - Defined in NFSv4.1 internet draft
- Block Layouts
  - Export LUNs (logical unit numbers) hosted on SAN devices, such as disks or RAID arrays.
  - Use iSCSI or FC employing the SCSI block command set to access SAN devices.
  - Refers to the use of SCSI (in its many forms)
- Object Layouts
  - Similar to file layouts.
  - Use the SCSI object command set to access data stored on OSDs.
  - Use object IDs similarly to file handles.
  - Defined by the T10 "SCSI OSD Commands" protocol draft.

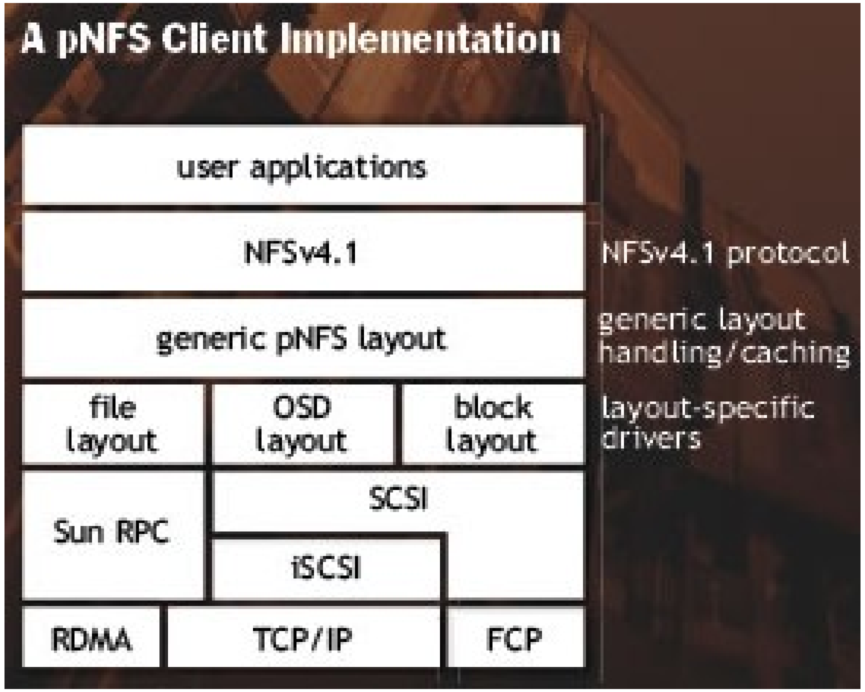
File Operations

- The pNFS protocol adds a total of six operations to NFSv4.
  - 4: get/return/recall/commit change to file metadata.
  - 2: data server name mapping
- The steps to access a file distributed by pNFS
  - The client must obtain a layout that describes to the client where to find the file's data and how to access it.
  - The client must obtain the proper access rights by opening the file at the metadata server.
  - The client can now perform I/O directly to the data servers using the storage-access protocol associated with the file's layout type.
  - If the client writes data to the data servers, it must commit changes to the file's metadata (including modification times and size) to the metadata server. For example, if the client writes to a block device, the metadata server must track how the written disk blocks map to data blocks within the file.
  - The client may close the file and return the layout once it has completed its I/O and committed any metadata changes.

File Operations

- File-sharing semantics
  - Close-to-open consistency (clients observe changes made to a file by other clients only after they close the file) ---- for no write-sharing cases
  - Byte-range **locks** through the metadata server ---- for file operations lock.
  - Special requirements to clients
    - notify the metadata server after new data updates have been made before close.
    - responsible for ensuring the atomicity of cross-data server writes
- Access control and security

Client/Server Implementation



- Client Implementation Stack

Other Consideration

- The ability to fall back to performing I/O through the metadata server using regular NFS accesses is useful in a number of error cases.
- Metadata sever HA or distribution
  - Metadata serve usually use another cluster file system to distribute among multiple servers.
  - The metadata servers are free to use any internal protocol to locate file layouts and keep them consistent among themselves and with data servers, as long as they provide coherent layouts.
- Dynamically load-balance client accesses across the set of data servers.

Organizations

- Panasas
- IBM
- EMC
- Network Appliance
- Sun (Solaris)
- CITI – Center for Information Technology Integration (University of Michigan)

Examples - Solaris pNFS

- Use the files layout type
- Transports
  - RPC, same as NFS
- Client
  - Same as NFS (v4.1)
- Metadata Server
  - With HA (active/passive)
  - Looks and feels like a regular NFS server.
  - Extensions to manage data servers.
  - Use ZFS underlying for storage of the filesystem namespace and attributes.
- Data Server
  - Direct use of ZFS pools for file data storage.
- pNFS Control Protocol (Metadata<-->Datd) Implementation depended
  - Meta-data and data server reboot / network partition indication
  - Filehandle, file state, and layout validation
  - Reporting of data server resources
  - Inter data server data movement
  - Meta-data server proxy I/O
  - Data server state invalidation

Thank You!