Parallel NFS (pNFS)

Feb 28th, 2006

Garth Goodson

Introduction

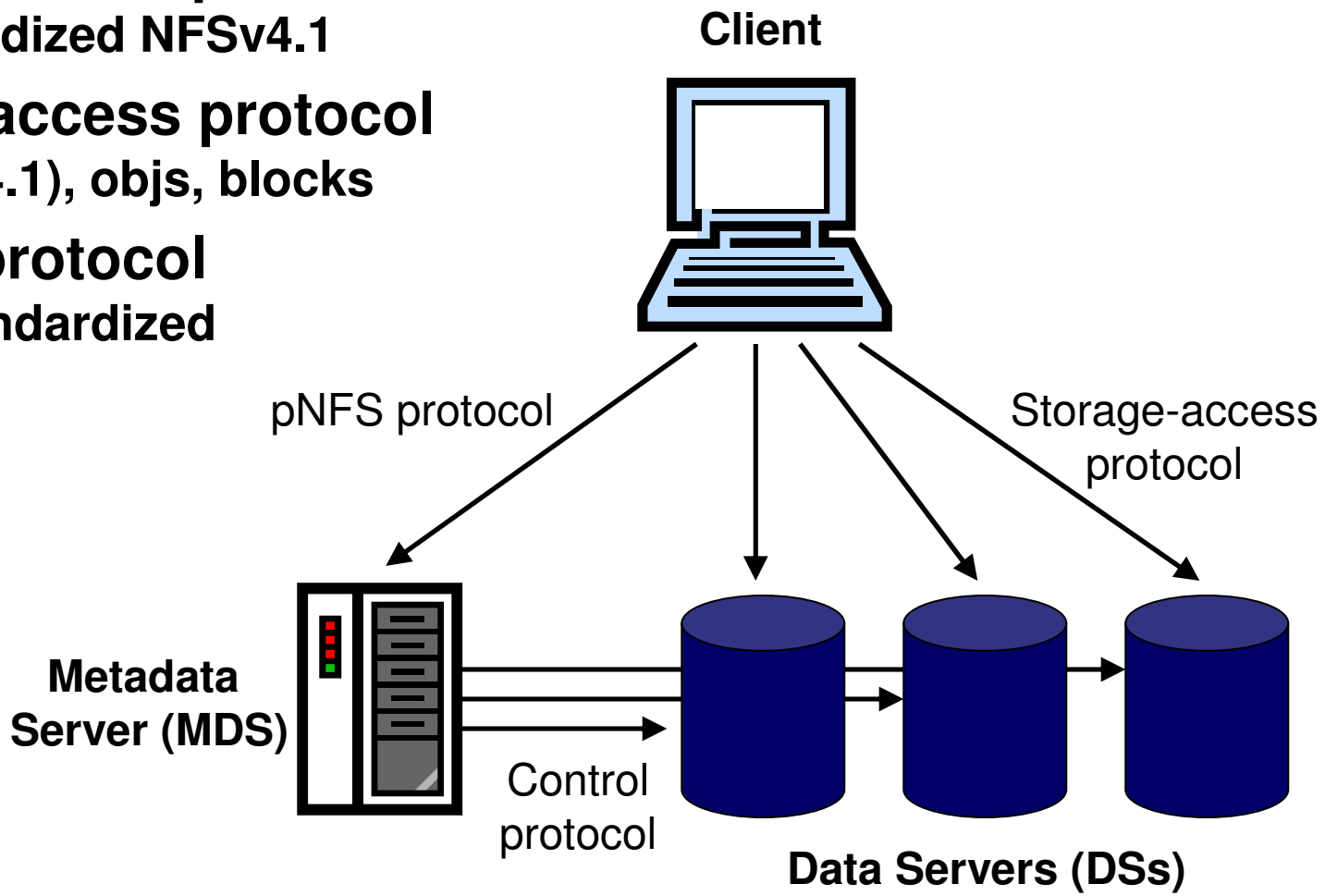
- **Proposition:**
  - Extend NFSv4 to support data parallelism
- **Goals:**
  - Remove single server bottleneck
  - Standardize client interface for parallel data access
  - Meet needs of HPC, and Linux cluster communities
- **Leverage NFSv4 implementations**
  - Add small set of protocol extension (pNFS)
  - Separation of metadata and data
- **Current draft close to finalization**
  - Being folded into NFSv4.1 specification

Outline

- pNFS architecture overview
- What's in current spec/what's not
- Layouts and operations to manipulate them
  - Obtaining a layout
  - Committing/updating a layout
  - Recalling/returning a layout
  - Reclaiming a layout
- NFSv4 file-layout overview
- Status and conclusions

pNFS Architecture

- **pNFS metadata protocol**
  - **standardized NFSv4.1**
- **Storage-access protocol**
  - **files (v4.1), objs, blocks**
- **Control protocol**
  - **not standardized**



What's in V4.1 pNFS specification?

- **Description of *Client-Metadata Server* protocol**
  - **This is the core pNFS protocol**
- **Includes:**
  - **Semantics of layouts**
  - **Crash recovery**
  - **Security considerations**
  - **Definition of core data structures and operations**
- **Description of the *Client-Data Server* file-layout**
  - **This is the NFSv4 file-layout protocol**
- **Includes:**
  - **Definition of file-layout types and semantics**

What's not in the v4.1 pNFS spec.

- Object and Block-layout protocol definitions
- These are addressed in separate documents
  - Currently as IETF WG drafts

Outline

- pNFS architecture overview
- What's in current spec/what's not
- **Layouts and operations to manipulate them**
  - **Obtaining a layout**
  - **Committing/updating a layout**
  - **Recalling/returning a layout**
  - **Reclaiming a layout**
- NFSv4 file-layout overview
- Status and conclusions

pNFS metadata protocol extensions

- A handful of new ops for manipulating layouts
  - `LAYOUTGET, LAYOUTCOMMIT, LAYOUTRETURN`
- Some ops for mapping devices to IDs
  - `GETDEVICEINFO, GETDEVICELIST`
- A few attributes for determining pNFS support
  - `FS_LAYOUT_TYPES, FILE_LAYOUT_TYPES, FILE_LAYOUT_HINT, FS_LAYOUT_PREFERRED_BLOCKSIZE, FS_LAYOUT_PREFERRED_ALIGNMENT`
- And two callbacks
  - `CB_LAYOUTRECALL, CB_SIZECHANGED`

Layouts: How data are located?

- Layouts define the data mapping to the client
  - E.g., enumerates servers data is mapped across
- Layout unique to storage access protocol used
  - May be small/static for file/obj-based protocols
  - May be large/dynamic for block-based protocols
- Layouts may be delegated to clients
  - Not for data caching; do not grant data-access rights
  - Delegates ability to access data out-of-band
- Layouts may be recalled
  - When state encapsulated by layout becomes invalid
- Layout updated on `LAYOUTCOMMIT`

Obtaining a layout

- A client gets a layout through `LAYOUTGET`
  - *<ClientID, FH, layout type, length, offset, iomode…>*
- IOmode describes client's data access intent
  - May be used differently depending on layout-type
    - E.g.: block-lyt may do allocation on *W* iomode
  - Iomodes do not conflict with share modes or locks
- No implied ordering between `OPEN` & `LAYOUTGET`
  - Layouts may be held across multiple `OPEN/CLOSE`s
- Layout hint can be provided when creating a file
  - Use `FILE_LAYOUT_HINT` attribute in `OPEN` attrs

Committing a layout

- **LAYOUTCOMMIT has two functions:**

1. **Synchronizing attributes between MDS and DSs**
   - **Writes to DSs may not alter MDS's attributes**
   - **`LAYOUTCOMMIT` is a synchronization point**
   - **After `LAYOUTCOMMIT` attrs. must be consistent**

2. **Updating layout's provisionally allocated space**
   - **In block-layout MDS must know blocks written**
   - **`LAYOUTCOMMIT` can updates MDS's layout**
   - **New layout is passed in `LAYOUTUPDATE` struct**
     - **Structure's def depends upon layout-type**

LAYOUTCOMMIT and size

- A size hint is passed into `LAYOUTCOMMIT`
  - `LAST_WRITE_OFFSET`
- The MDS may:
  1. Update file size based on this size
  2. Ignore this size and use another method
  3. Use this size as hint subject to some validation
- A client should use `SETATTR` to truncate file
- `CB_SIZECHANGED` used to notify layout holders to change in size
  - This is in preference to recalling all layouts

Recalling/returning a layout

- Layout recalled when it is no longer accurate
- Recall through `CB_LAYOUTRECALL`
  - May recall a layout segment (byte range)
  - Recall robustness:
    - The range need not match a previous `LAYOUTGET`
    - In fact, the range may never have been given out
    - Layout may be returned in multiple segments
  - May recall a single layout or all layouts for a FSID
- Layout returned via `LAYOUTRETURN`
  - Dirty data may be written with layout before return
  - Or, dirty data may be written through MDS after

Reclaiming a layout (new feature)

- **Problem: MDS fails before `LAYOUTCOMMIT`**
  - Client has written data to data servers
  - Written data may have no corresponding metadata
- **Layouts may be reclaimed during grace period**
- **`LAYOUTCOMMIT` used instead of `LAYOUTGET`**
  - Specify by setting reclaim flag
  - Reclaimed layout specified in `LAYOUTUPDATE` struct
  - Reclaimed layout subject to validation by MDS
- **For dirty data client must get new layout**
  - Usually after grace period, but depends on MDS

Writing through the MDS

- **Mandatory to support writes through MDS**
  - **But how do they interact with reads at DSs?**
- **Two guarantees must be made:**
  1. **MDS `COMMIT` makes unstable writes visibile at DS**
     - **Implementation may make writes visible sooner**
  2. **Stable MDS writes must be visible upon completion**
- **MDS must recall layouts if guarantees not met**

Outline

- pNFS architecture overview
- What's in current spec/what's not
- Layouts and operations to manipulate them
  - Obtaining a layout
  - Committing/updating a layout
  - Recalling/returning a layout
  - Reclaiming a layout
- **NFSv4 file-layout overview**
- Status and conclusions

NFSv4 File-layout

- Layout is an array of `nfsv4_file_layout` types
  - One per stripe
- `nfsv4_file_layout` is an array of devices and FHs
  - Device ID array facilitates device level multipathing

```
struct nfsv4_file_layout {                  /* Per data stripe */
          pnfs_deviceid4   dev_id<>;
          nfs_fh4          fh;
};

struct nfsv4_file_layouttype4 {             /* Per file */
          stripetype4        stripe_type;
          length4            stripe_unit;
          length4            file_size;
          nfsv4_file_layout dev_list<>;
};
```

NFSv4 File-layout: Quick overview

- Very simple layout-type
  - A single layout can easily cover an entire file
- Data servers service `READ/WRITE/COMMITS`
  - All attribute and other metadata ops go to MDS
- Global stateid requirement
  - No stateid in layout-type; use stateid from MD op.
  - Requires some validation/propagation at/to DSs
- lomode not usually required (impl. dependent)
  - Clients can default to `READ/WRITE`
- MD ops may truncate/zero extend DS files
  - e.g. on a `LAYOUTCOMMIT` or a `SETATTR`

Outline

- pNFS architecture overview
- What's in current spec/what's not
- Layouts and operations to manipulate them
  - Obtaining a layout
  - Committing/updating a layout
  - Recalling/returning a layout
  - Reclaiming a layout
- NFSv4 file-layout overview
- **Status and conclusions**

Current status

- Spec is fairly complete; please read it…
- Few things left to work out:
  - Sessions and callback races
  - EOF handling for block-layouts
- Prototyping:
  - NetApp have updated prototype to last IETF-draft
    - Demo here at Connectathon
  - Linux pNFS client in the works (CITI/IBM/NetApp)
    - Goal is to modularize layout drivers
    - NFSv4 file-layout and PVFS2 layout drivers exist

Conclusions

- We've come a long way in the last year
- Protocol balances simplicity and feature creep
- Many more corner cases and details in spec.

- Now we just need a few more prototypes…

(Soon to be outdated) IETF-WG Draft:
http://www.ietf.org/internet-drafts/draft-ietf-nfsv4-minorversion1-01.txt

My contact info: goodson AT netapp.com