pNFS over OSD

Benny Halevy
bhalevy@panasas.com

Linux Storage and File System Workshop
February 13, 2007

Background

What problem are we trying to solve?

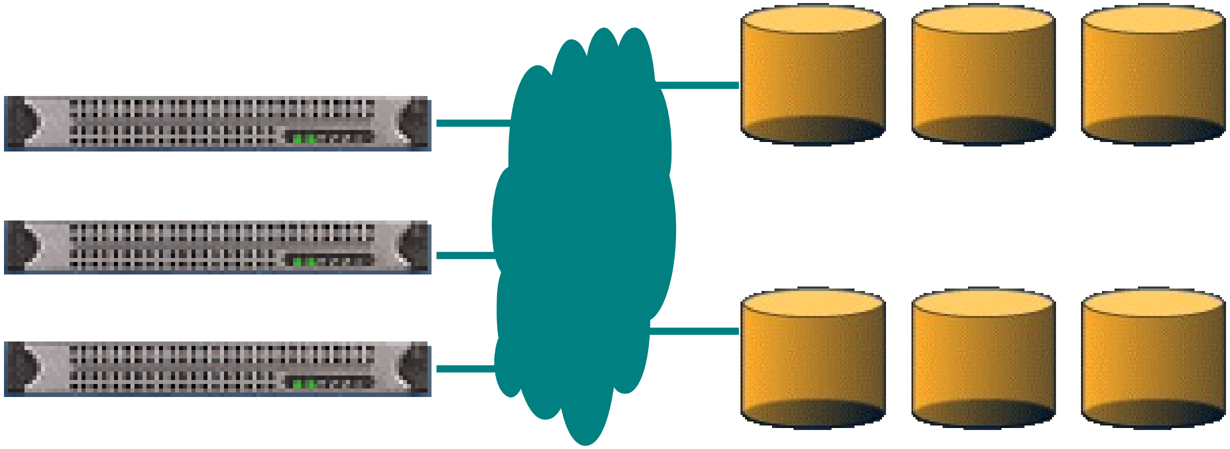
- Essentially, scaling out I/O, so that:

What problem are we trying to solve?

- Essentially, scaling out I/O, so that:
  - Many clients

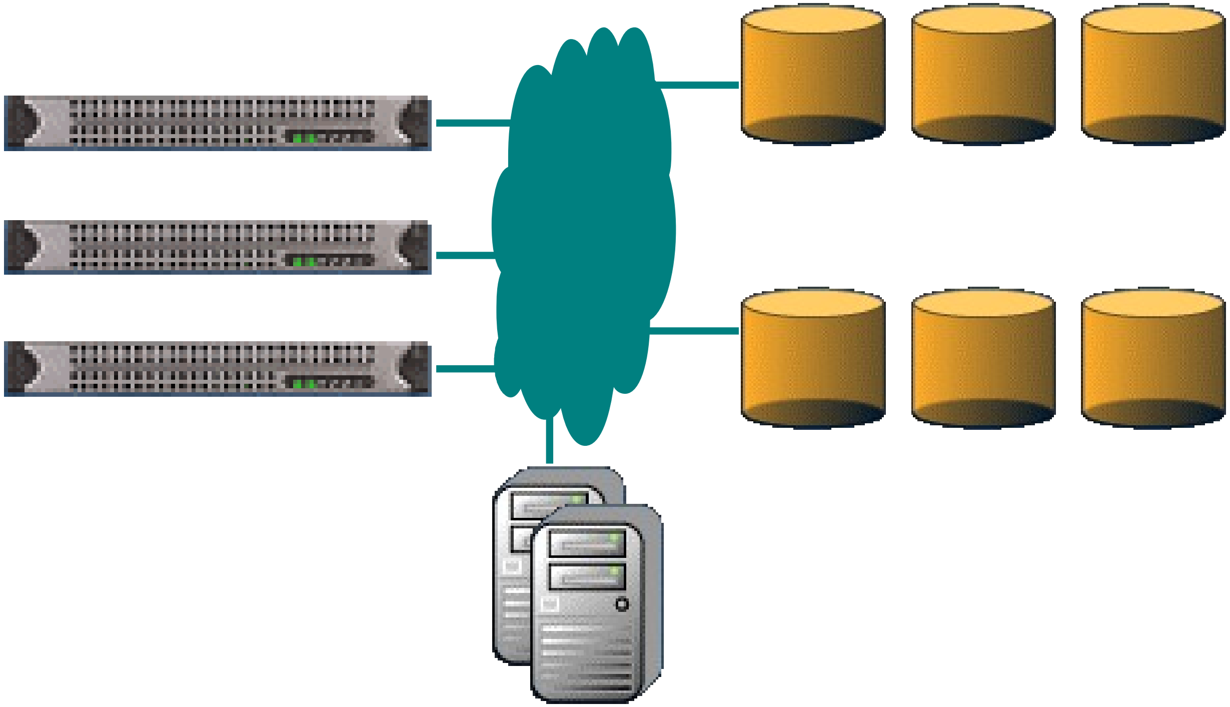
What problem are we trying to solve?

- Essentially, scaling out I/O, so that:
  - Many clients
  - Can talk to many storage devices, in parallel



What problem are we trying to solve?

- Essentially, scaling out I/O, so that:
  - Many clients
  - Can talk to many storage devices, in parallel
  - Without having to go through the server



Sounds Familiar?

- Well, quite a few clustered file systems were built this way...
- So why not keep doing that?
  - Proprietary protocols are bad
  - Interoperability is good for everybody
  - I don't know anyone that enjoys chasing linux (well, maybe Boaz does actually ;-)

So this is how pNFS was born

- pNFS is now in the IETF NFSv4.1 draft
- Sun implemented it on Solaris
- CITI, IBM, EMC, Netapp, Panasas working on linux
- DESY doing it in Java
- CMU doing research obout it
- Connectathon tests are passing with nfsv4.1 pnfs and sessions prototypes as of last week.

But, you talked about SAN Filesystems...

- And Panasas is doing objects...
- Hmm, and what Sun and Netapp are there for?
- Well, we figured out we all want to solve the same problem but we just happen to use different types of storage.
- So pNFS comes in three different basic flavors:
  - Files (NFSv4.1)
  - Blocks (SCSI SBC)
  - Objects (SCSI OSD)

So how do you do this?

- LAYOUTGET, LAYOUTCOMMIT, and LAYOUTERETURN carry layout_type specific metadata (defined in other WG RFCs)
- CB_LAYOUTRECALL kindly asks the client to return layouts.
- GETDEVICELIST and GETDEVICEINFO save the admin whole lot of trouble
- FILE_LAYOUT_HINT is an attribute that can be set on CREATE.

How does the layout look like?

- Here's a glimpse into the pnfs-obj layout:
  http://www.nfsv4-editor.org/draft-08/draft-ietf-nfsv4-minorversion1-08.txt:

```
struct layout4 {
    offset4                    lo_offset;
    length4                    lo_length;
    layoutiomode4              lo_iomode;
    layouttype4                lo_type;
    opaque                     lo_layout<>;
};
```

- http://www.ietf.org/internet-drafts/draft-ietf-nfsv4-pnfs-obj-02.txt:

```
struct pnfs_osd_layout4 {
    pnfs_osd_data_map4      map;
    pnfs_osd_object_cred4   components<>;
};
```

Object-based layout map

- The map describes how the file is striped

```
struct pnfs_osd_data_map4 {
    length4                         stripe_unit;
    uint16_t                        group_width;
    uint16_t                        group_depth;
    uint16_t                        mirror_cnt;
    pnfs_osd_raid_algorithm4        raid_algorithm;
};

enum pnfs_osd_raid_algorithm4 {
    PNFS_OSD_RAID_0     = 1,
    PNFS_OSD_RAID_4     = 2,
    PNFS_OSD_RAID_5     = 3,
    PNFS_OSD_RAID_PQ    = 4
};
```

Object-based Storage

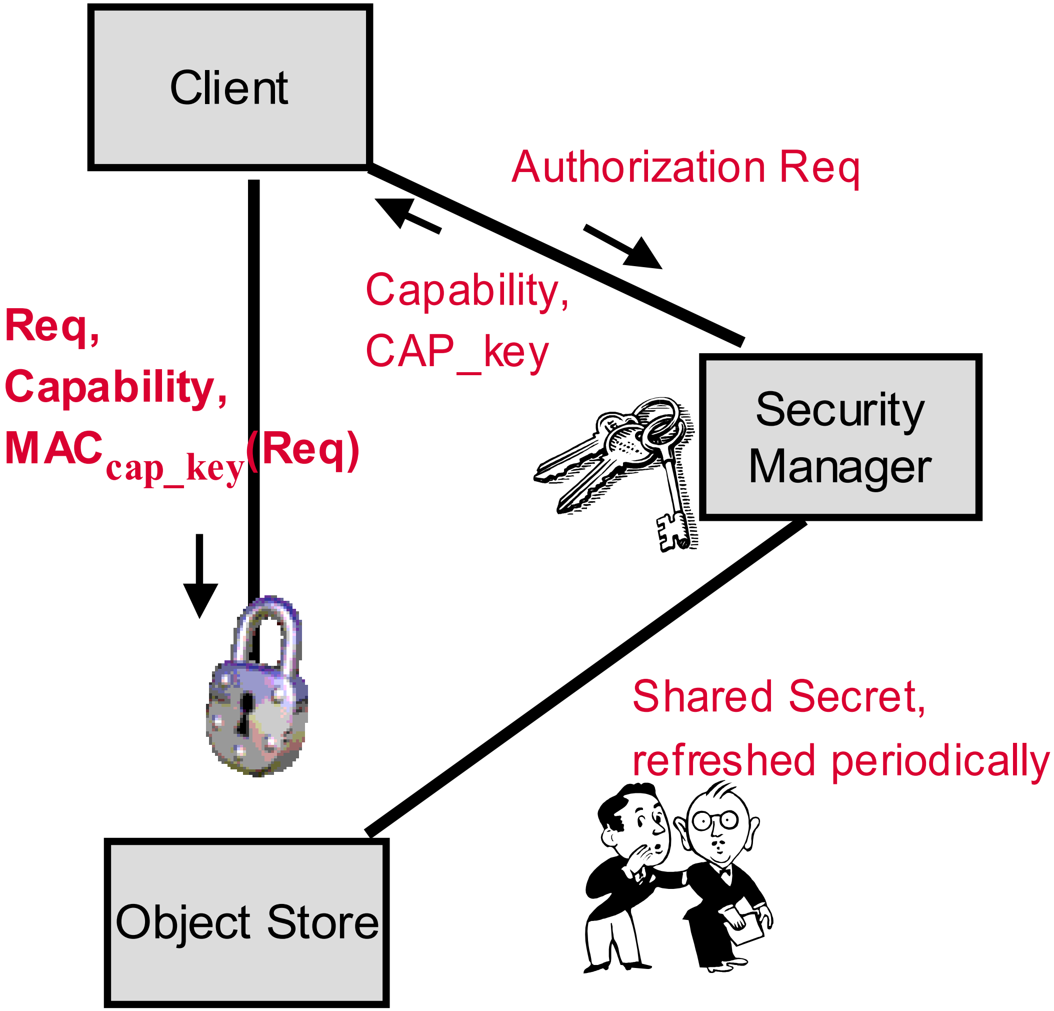
Why do I like Object-based Storage

- First, it's doing local block allocation
  - Dividing the problem this way really helps you scale
- Second, they're easy to manage
  - When you have to manage thousands of storage devices you want little management overhead
- Third, somebody thought about their security model seriously
  - No user authentication, capability based security
  - The file server decides about the access policy
  - And it can fence off clients easily and securely

Oh, one more thing

- Did I say OSDs are cool? :)
- Well, they are...
  - and it doesn't mean they are necessarily wrong
- It's about time to start putting some intelligence in front of these spinning disks...

OSD Security Model



1. Client asks for access authorization.
2. Security manager returns credential (cap + CapKey) signed over cap, system ID, secret key.
3. Client presents cap and signs the request using the CapKey
4. OSD verifies request signature using the secret key.

OSD Commands are a bit chubby

- Long identifiers, capabilities, etc, make OSD CDBs 200 bytes long.

	7	6	5	4	3	2	1	0
8	SERVICE ACTION (8806h)							
9								
10	OPTIONS BYTE							
11	Reserved		GET /SET CDBFMT		Reserved			
12	TIMESTAMPS CONTROL							
13	Reserved							
15								
16	PARTITION _ID							
23								
24	USER _OBJECT _ID							
31								
32	Reserved							
35								

	7	6	5	4	3	2	1	0
36	LENGTH							
43								
44	STARTING BYTE ADDRESS							
51								
52	Get and set attributes parameters							
79	...							
80	Capability							
159	...							
160	Security parameters							
199	...							

OSD commands can set and get attributes

- This makes them inherently bi-directional
- For example: a WRITE command can send on the data out phase also a list of attributes to set and a list of attributes to get
- The data in phase sends data back, plus optional attributes

Kernel support for OSD

- Linux wants bi-directional SCSI commands for other reasons
- We also need support for large, variable length CDBs
- Good responses for the patches we sent for review to the block, scsi, and iscsi layers.
  - Done some cleanup along the way
  - Tested successfully on iscsi -> IET and IBM OSD initiator -> IBM OSD target simulator

More on the patches

- The main idea was to add an API to access the current I/O related information as uni-directional with little or no change to existing code, and to have a similar API to access bi-directional read and write buffers.
- The SCSI layer helps setting up bi-directional block requests
- Varlen CDBs are pointed at
- Scsi lib prep function makes a scsi_cmnd out of the request
- Scsi transports such as iscsi make a PDU out of it.

To do

- Some minor cleanups
- Bidi residual bytes
- OSD initiator library

- No need for testing since it will all just work™ ;-)

The Design

pNFS Software Stack

- (p)NFS client
- pnfs-obj layout driver (layout and device cache)
- OBJ RAID
- Flow control (global and per-device)
- OSD initiator
- SCSI stack
- iscsi_tcp | iser | fc | ...

Want to read more?

- http://www.nfsv4-editor.org/draft-08/draft-ietf-nfsv4-minorversion1-08.txt
- http://www.ietf.org/internet-drafts/draft-ietf-nfsv4-pnfs-obj-02.txt
- http://www.ietf.org/internet-drafts/draft-ietf-nfsv4-pnfs-block-01.txt
- http://www.t10.org/ftp/t10/drafts/osd/osd-r10.pdf