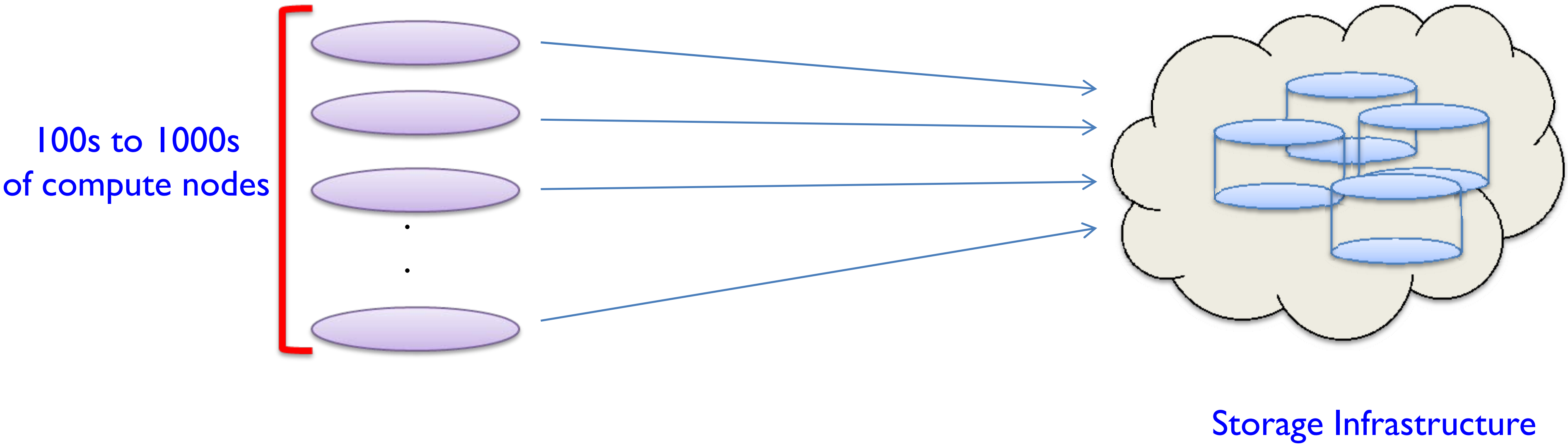
pNFS: Blending Performance and Manageability

Lisa Week and Piyush Shivam

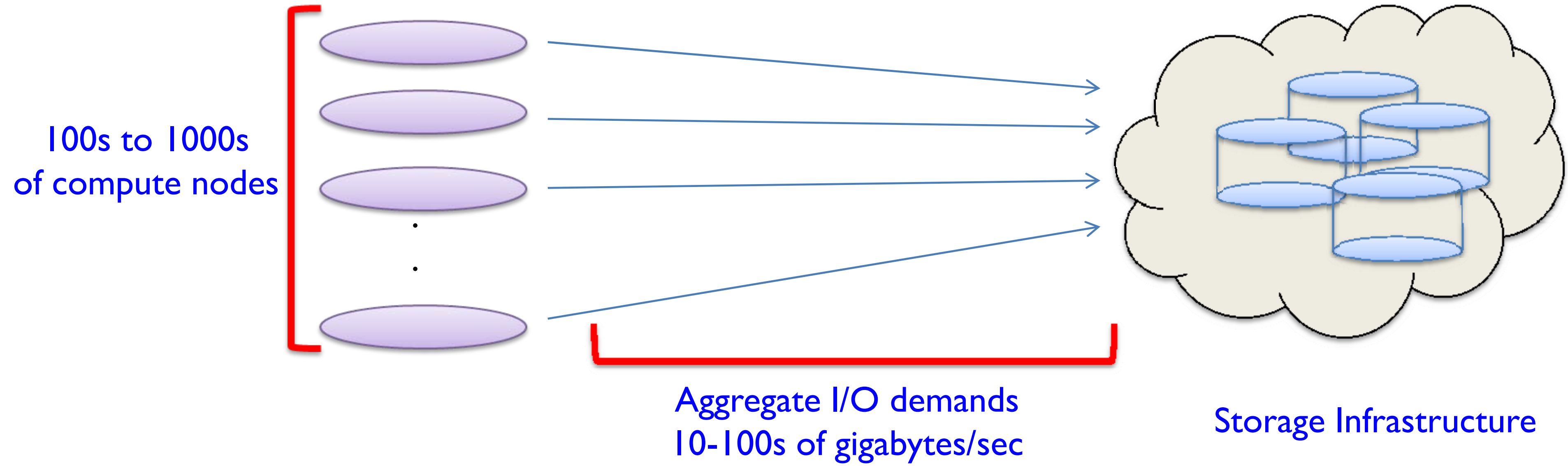
Sun Microsystems

Data-Intensive Applications

- Examples: Data mining, oil and gas, weather modeling

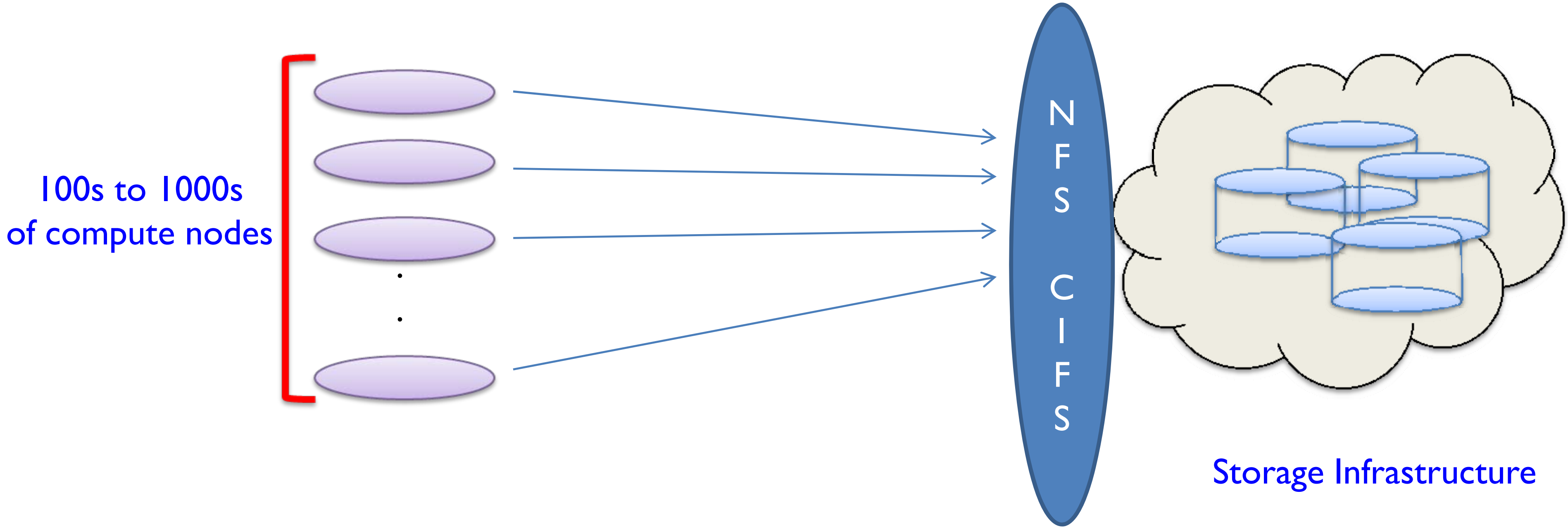


Data-Intensive Applications



Challenge: Provide scalable data access

Data-Intensive Applications



Access restricted via a single server (network endpoint)
Storage service cannot deliver its aggregate capacity

Typical Solutions *[Goodson et al. '07]*

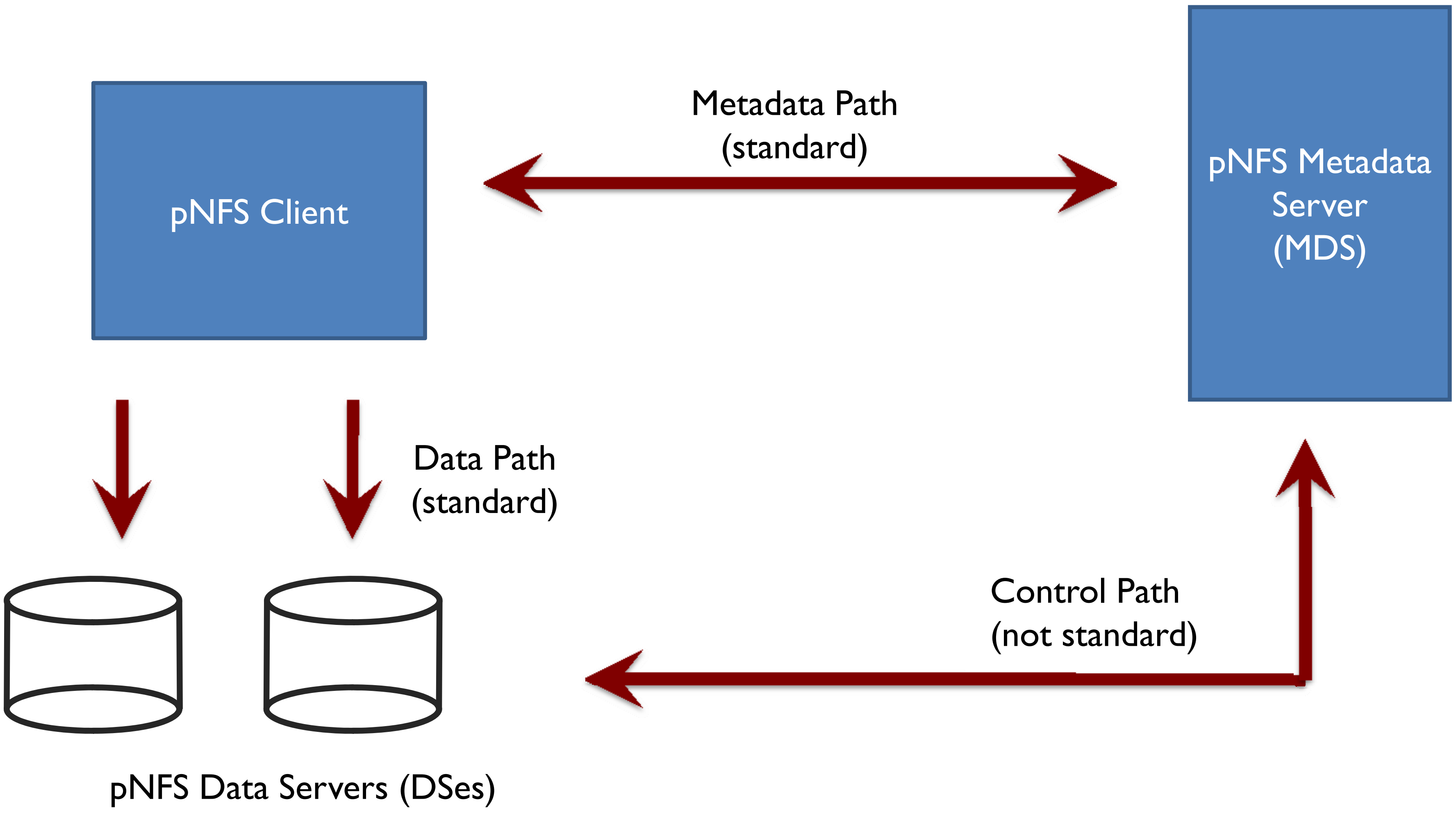
- Manual partitioning of file namespace
  - Unmanageable with data growth, no file-level parallelism
- Employ a request routing layer [Anderson et al. '00]
  - Cumbersome for stateful protocols such as NFS v4
- New protocol for parallel data access at file-level
  - Issue I/O ops in parallel to the data servers
  - Several cluster file systems, e.g., Lustre, GPFS
  - No open protocol standards for parallel data access
  - Lack of interoperability between storage architectures

pNFS

SDC
STORAGE DEVELOPER CONFERENCE
SNIA SANTA CLARA, 2009

- Parallel NFS, standards-based network protocol
- Part of NFSv4.1 IETF specification
- Classic "data-metadata" split architecture
  - Separate servers for file metadata and file data

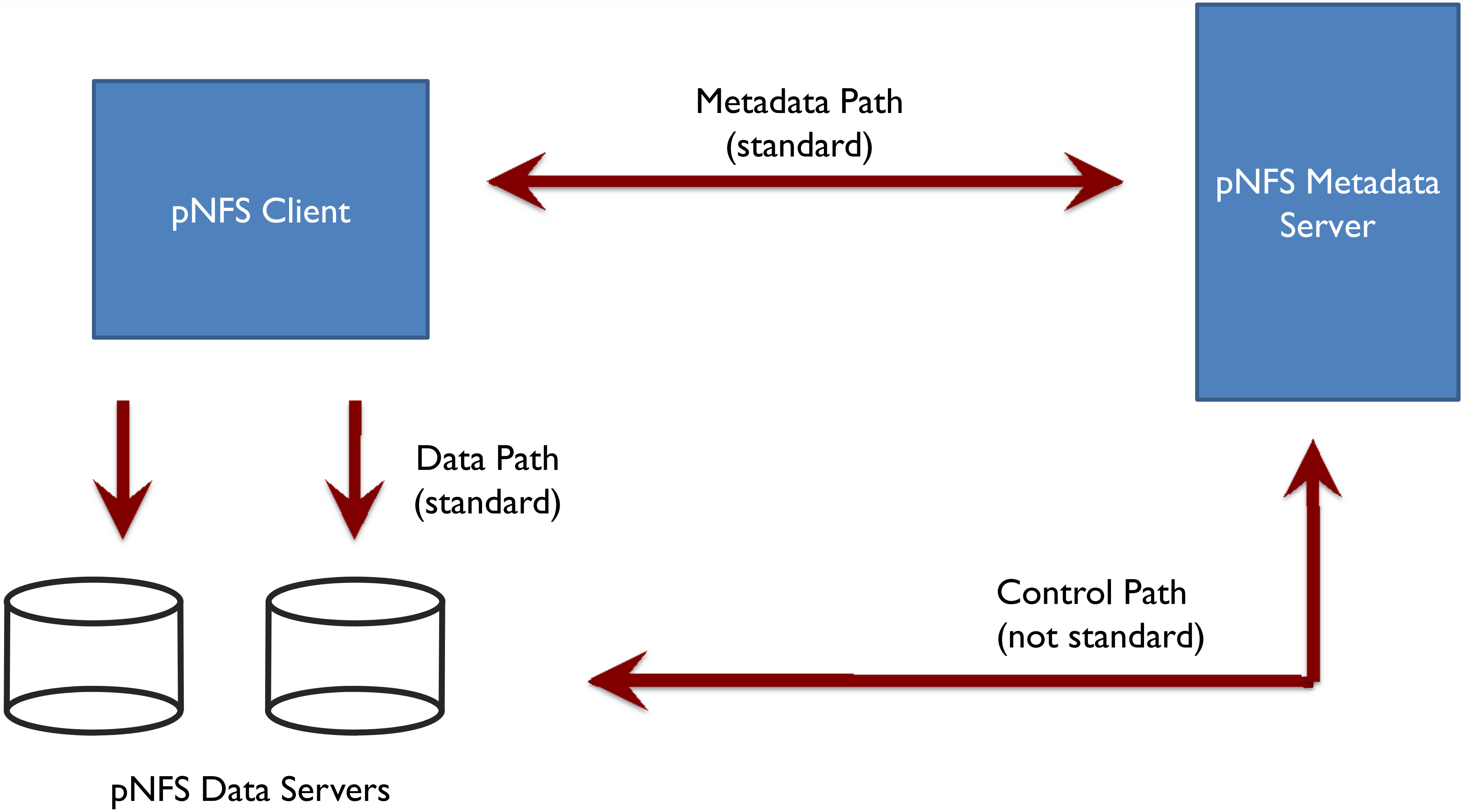
pNFS Architecture



Outline

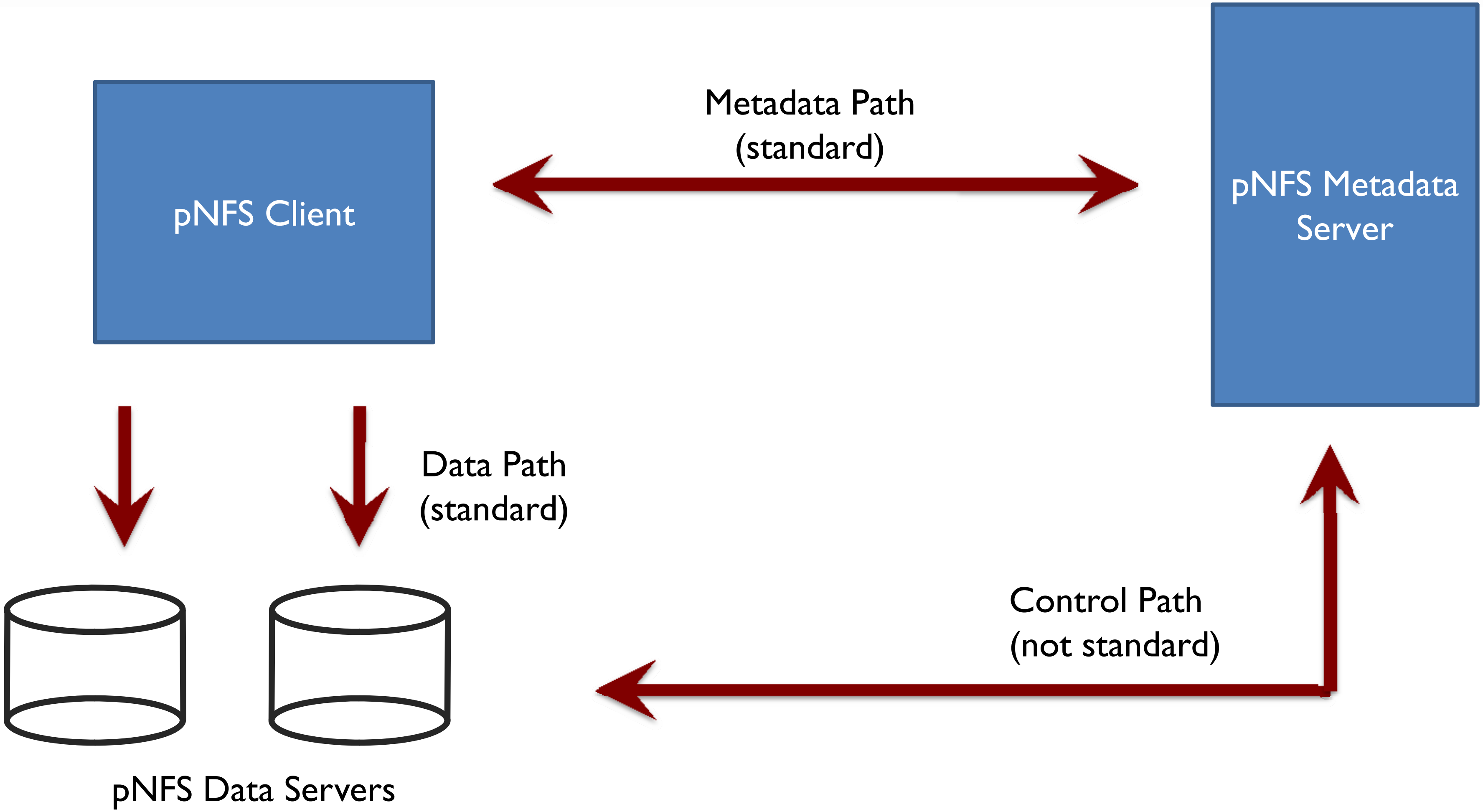
- Background and Motivation
- Key features of pNFS architecture
- Building blocks and fundamental operations in the protocol
- Performance and manageability features of Sun's implementation

pNFS: Scalable



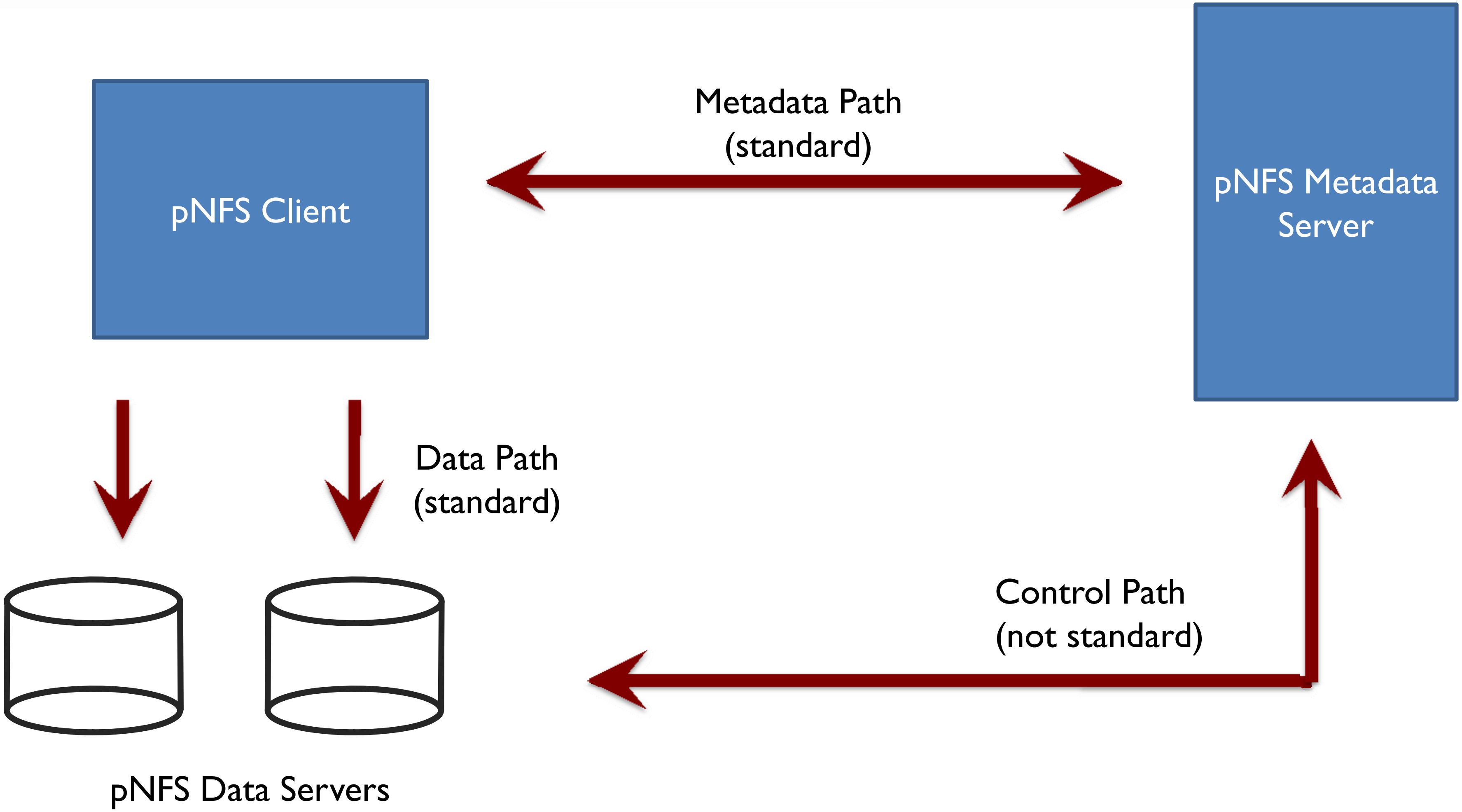
Parallel data access to data servers at the file-level

pNFS: Flexible



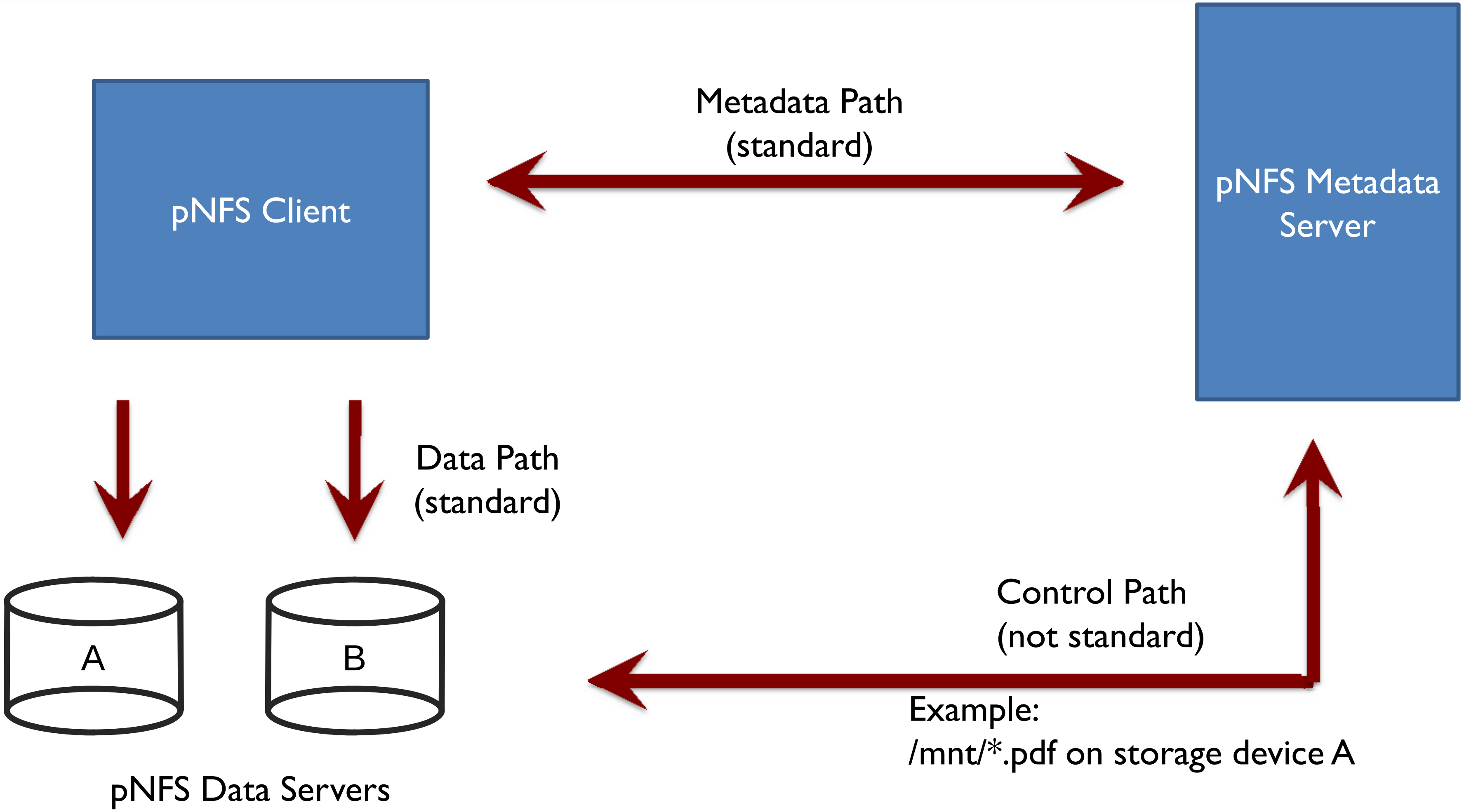
Protocol supports data access via files, blocks, and objects

pNFS: Namespace-Location Indp.



Namespace decoupled from its storage, single mount can span multiple storage devices

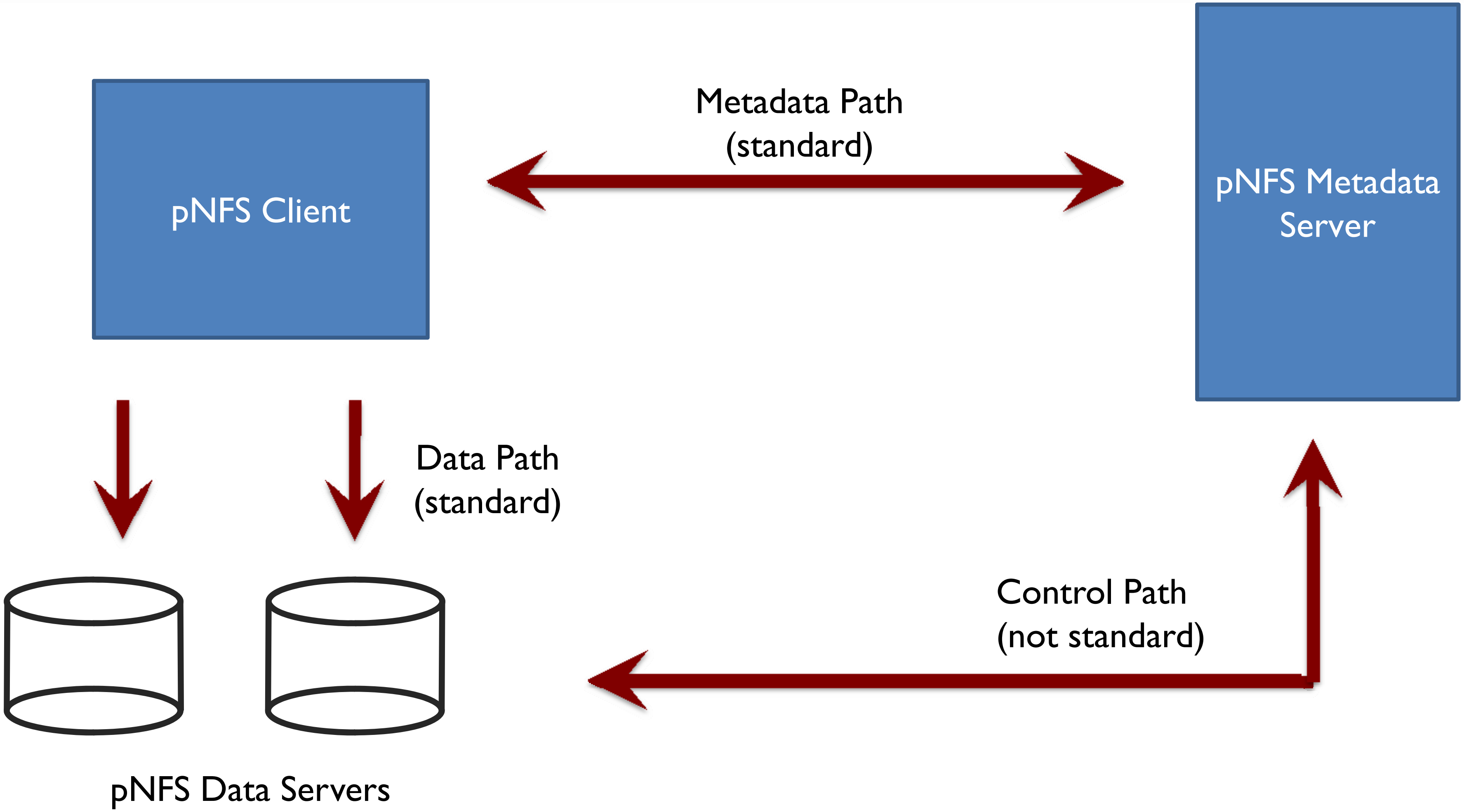
pNFS: Namespace-Location Indp.



Example:
/mnt/*.pdf on storage device A

/mnt/*.jpeg on storage device B

pNFS: Secure



Kerberos security for all paths (strong authentication, privacy, and integrity)

Outline

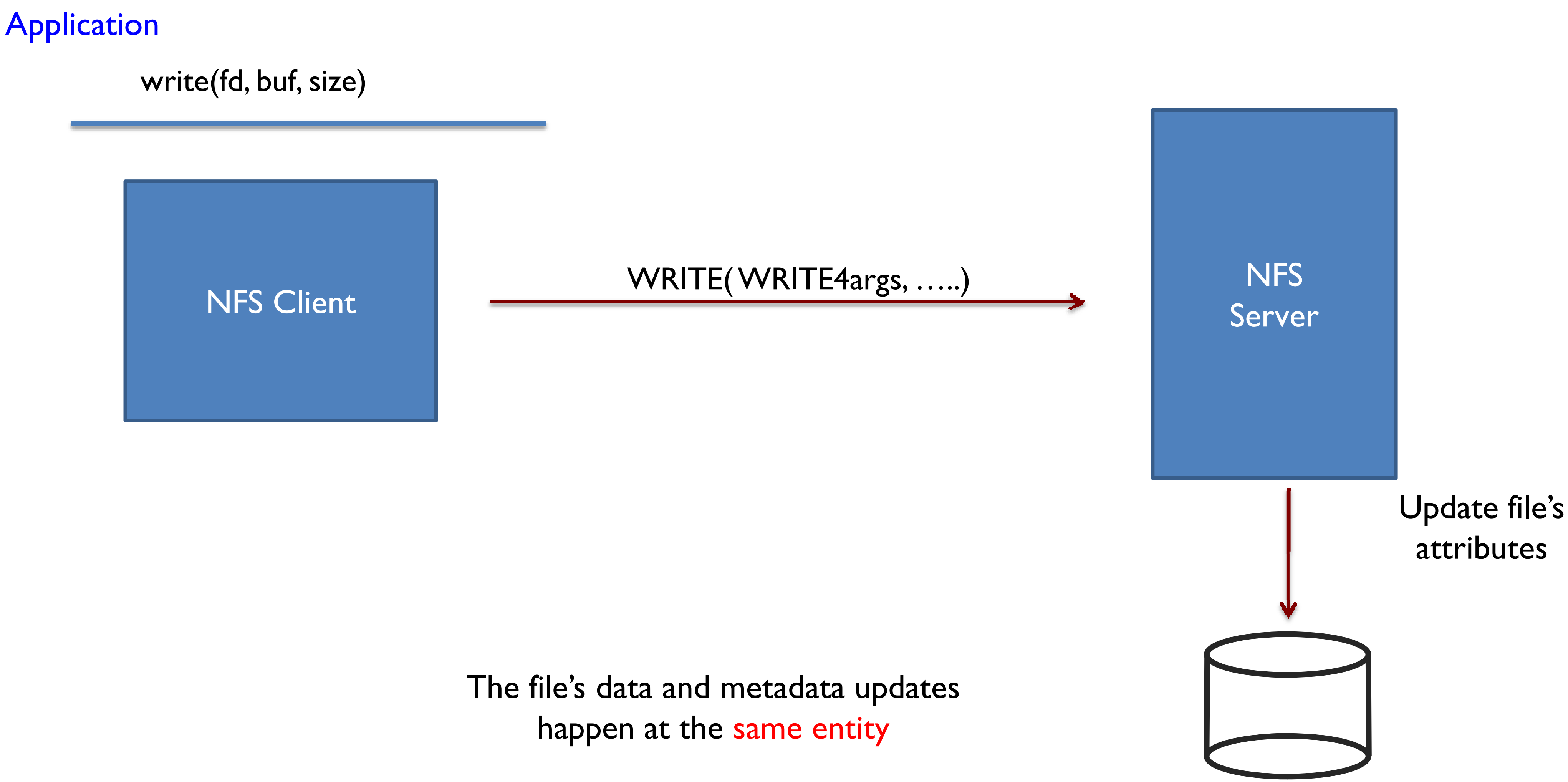
- Background and Motivation
- Key features of pNFS architecture
- Building blocks and fundamental operations in the protocol
- Performance and manageability features of Sun's implementation

Fundamental Block: LAYOUTS

- A file layout determines
  - "how" a file is stored – striping pattern
  - "where" a file is stored – address of the storage devices

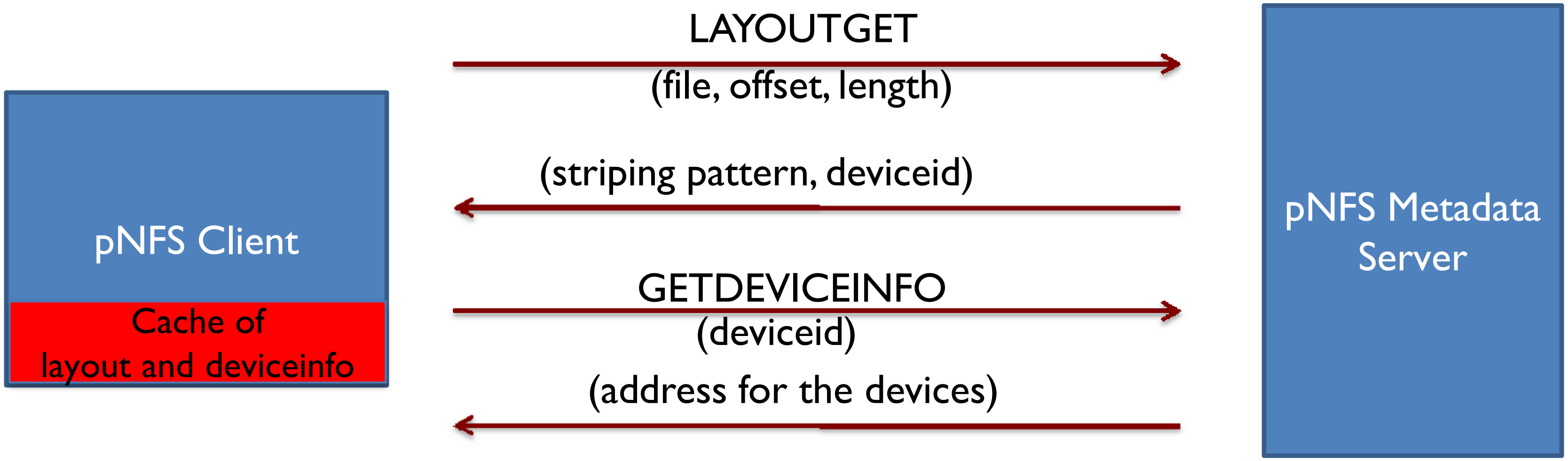
- A client asks for a layout from MDS to access a file directly from the storage devices

Example: Writes in NFSv4.0 and non-pNFS NFSv4.1



The file's data and metadata updates happen at the same entity

Example: Writes in pNFS

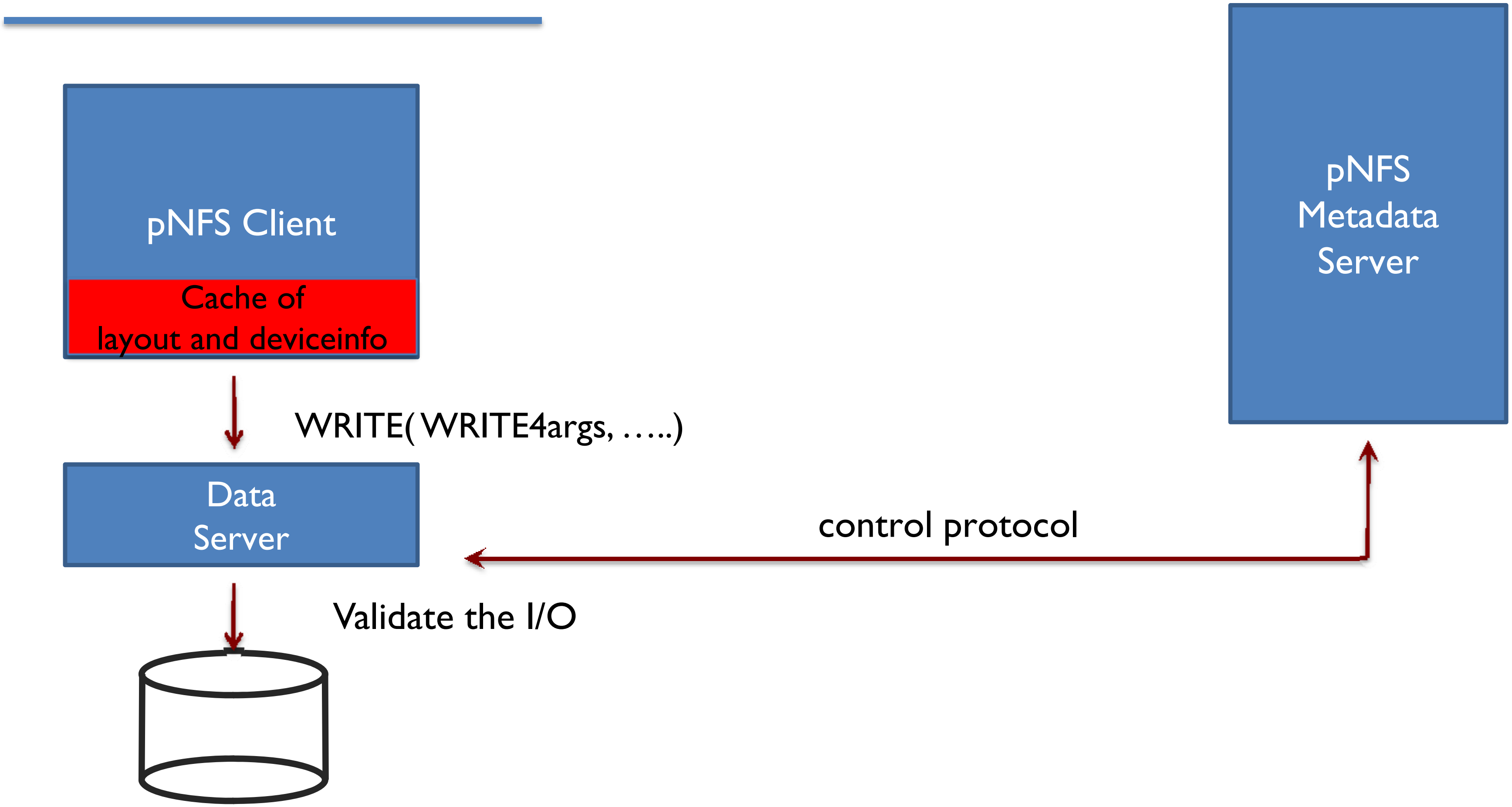


The client asks for layout and device info only if it does not already have it cached

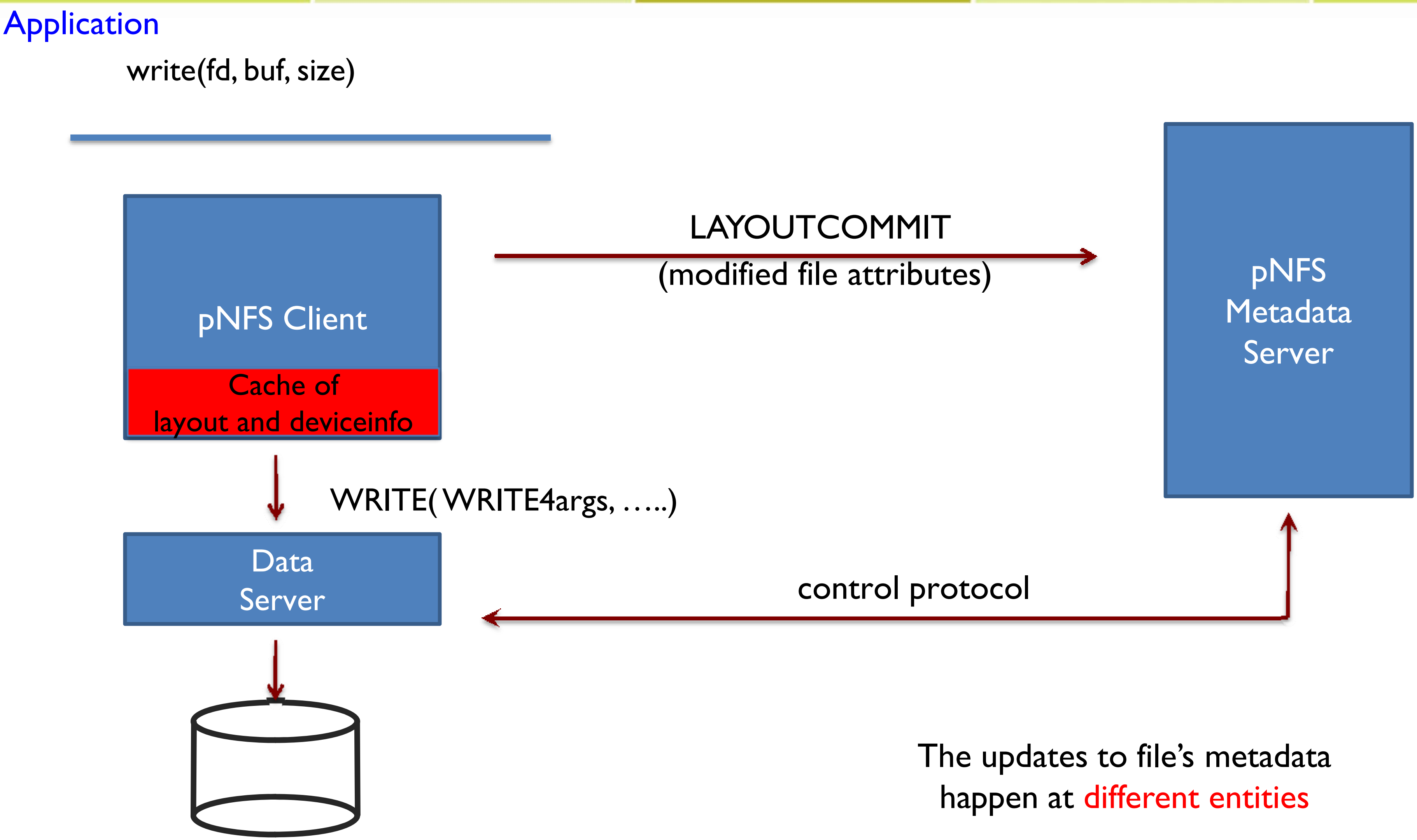
Example: Writes in pNFS

Application

write(fd, buf, size)

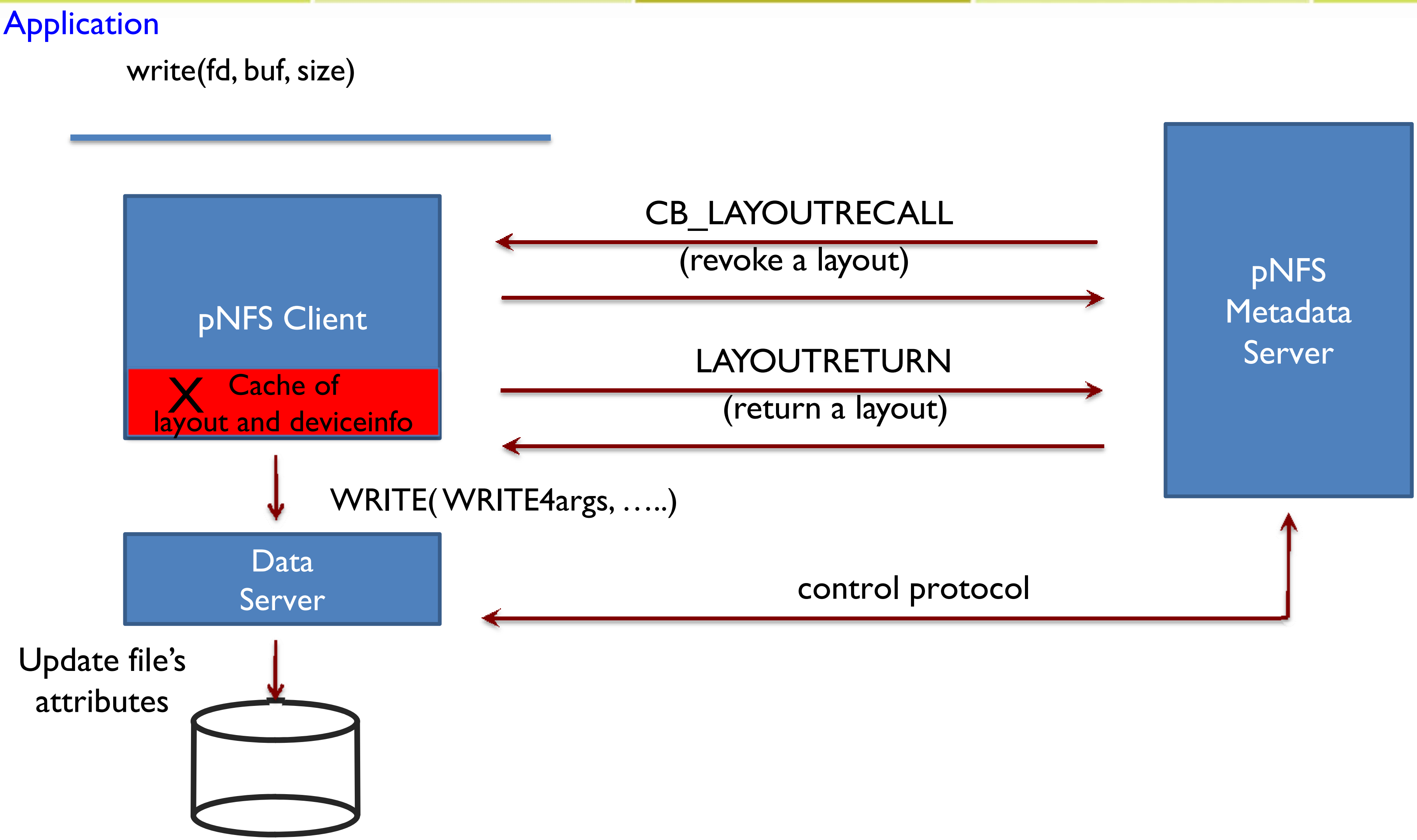


File Attribute Synchronization

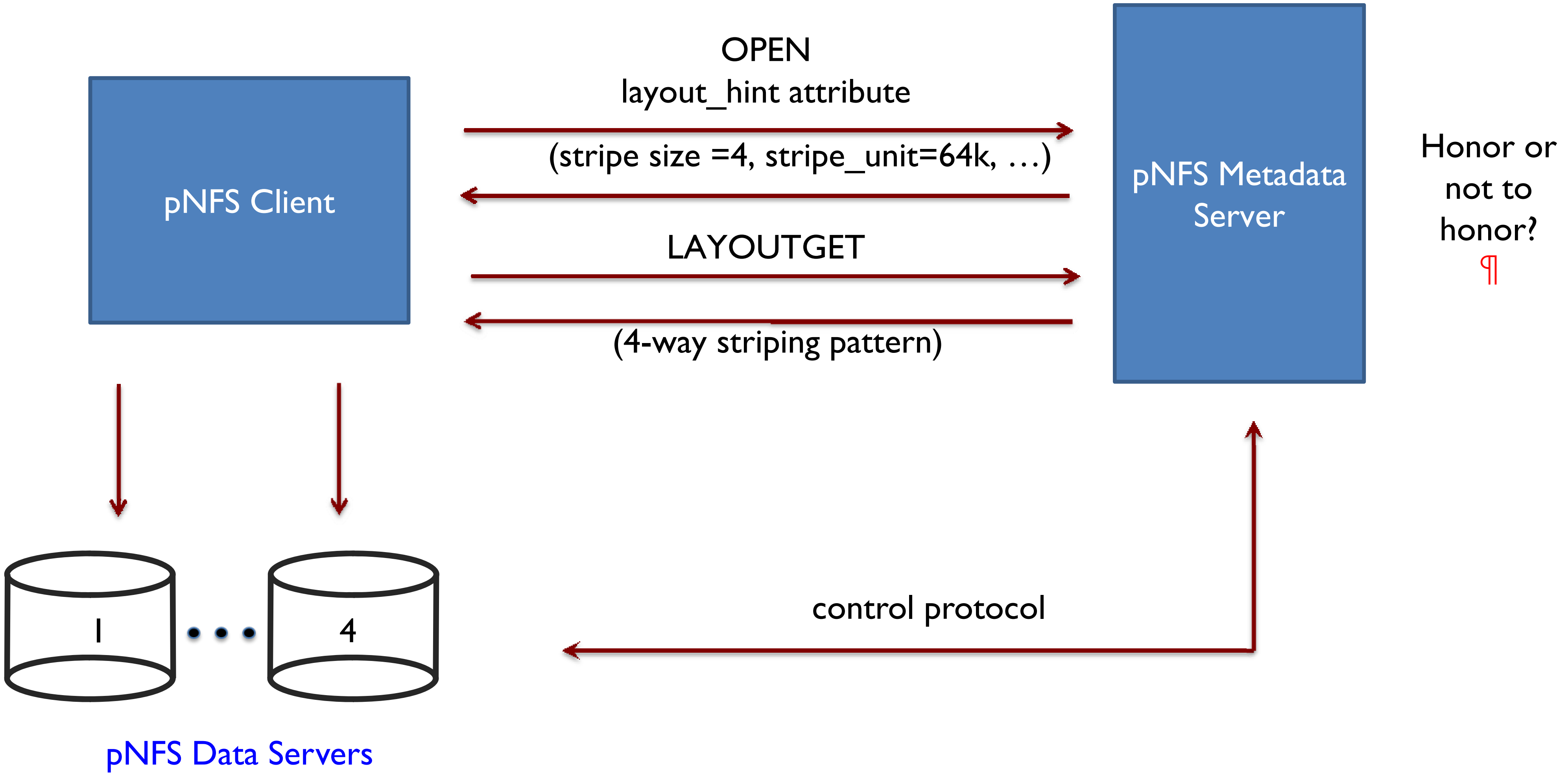


The updates to file's metadata happen at different entities

Layout Synchronization



Client Layout Hints



Outline

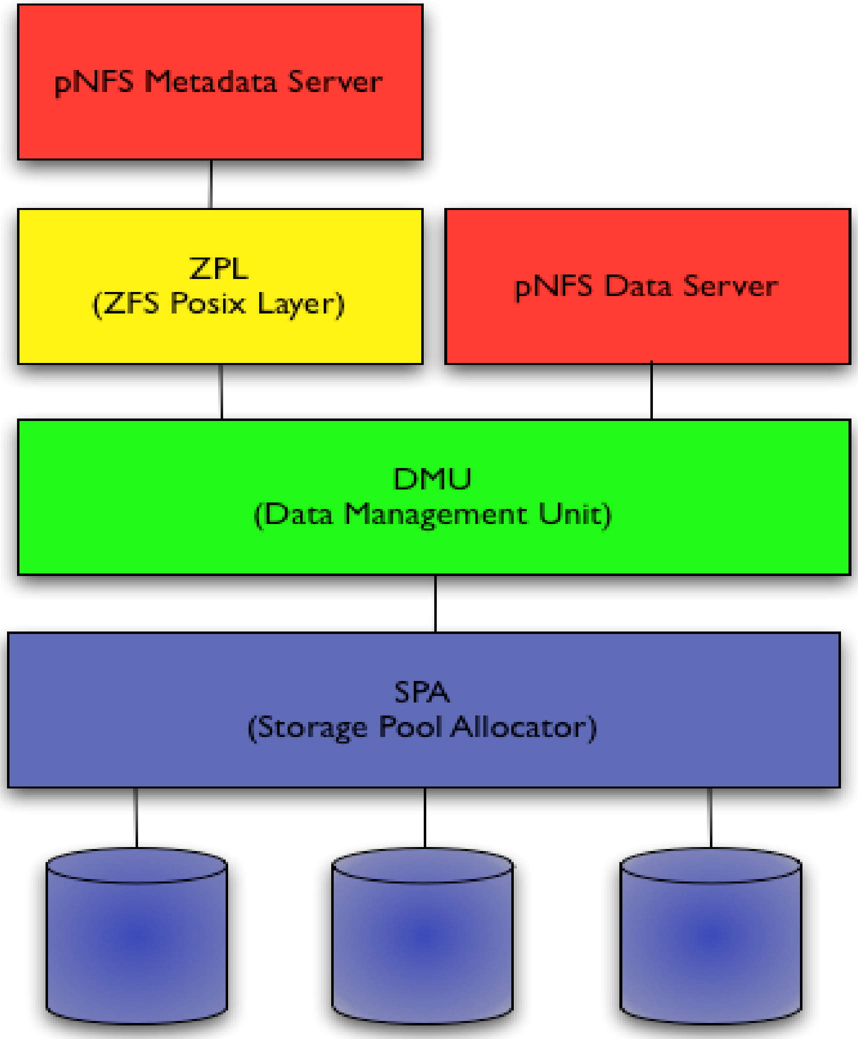
- Background and Motivation
- Key features of pNFS architecture
- Building blocks and fundamental operations in the protocol
- Performance and manageability features of Sun's implementation

Performance and manageability features of Sun's pNFS

- Files based implementation exploiting ZFS capabilities
- Policy Based Management
- Multi-transport support
  - TCP/IP/Ethernet
  - Increased performance via NFS/RDMA/IB

ZFS

- pNFS benefits from use of ZFS
  - Pooled storage
  - Transactional object system
  - End to end data integrity
  - Highly scalable
- MDS layered on ZPL
- DS layered on DMU

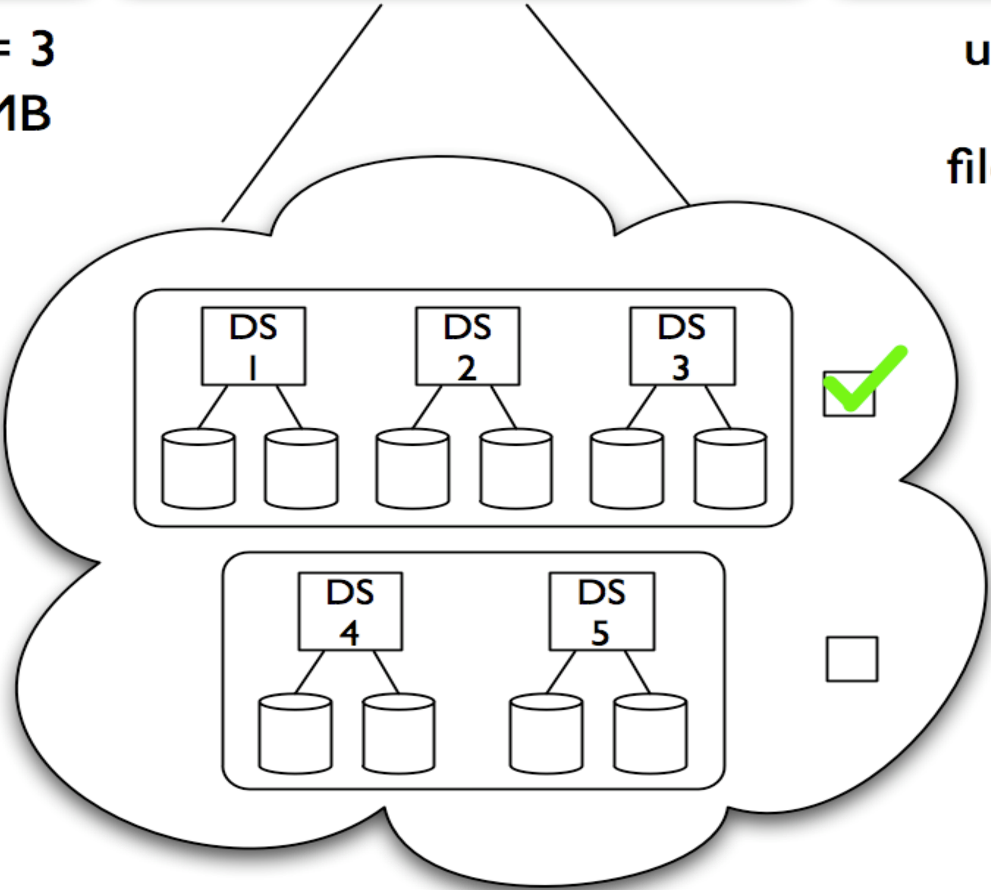


Policy Based Management

- Policy Driven Layouts
  - Layouts determine “how” and “where” a file is laid out
- Problem: Who makes that decision?
  - No standard mechanism for user/admin to specify layout
- Current pNFS prototype: Simple Policy Engine (SPE)

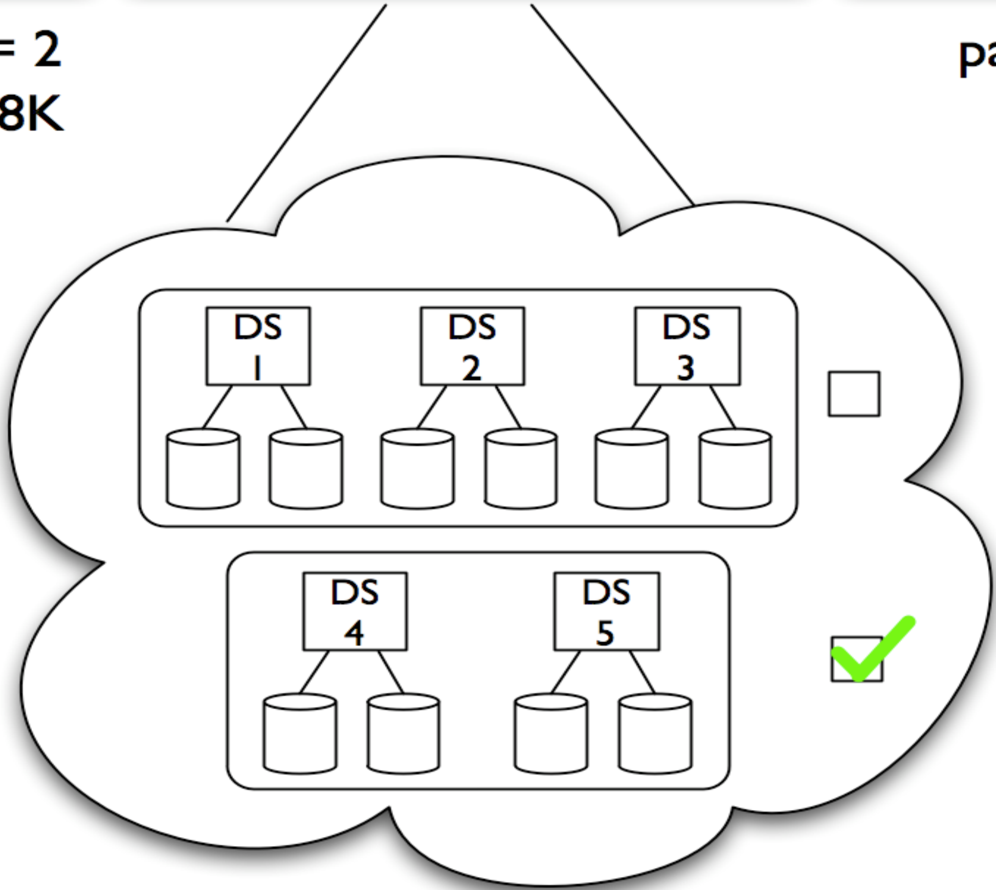
Management: Policy Example

how?	where?	why?
stripe count = 3 unit size = 1MB		user = theboss and file name = *.iso



Management: Policy Example (cont.)

how?	where?	why?
stripe count = 2 unit size = 128K		path = /var/mail



Management: SPE

- Policies are "create-time" only
- Policies can be set at client and MDS
- Policies define slightly different parameters if on client vs. server

Management: Server-side SPE

SDC
STORAGE DEVELOPER CONFERENCE
SNIA ■ SANTA CLARA, 2009

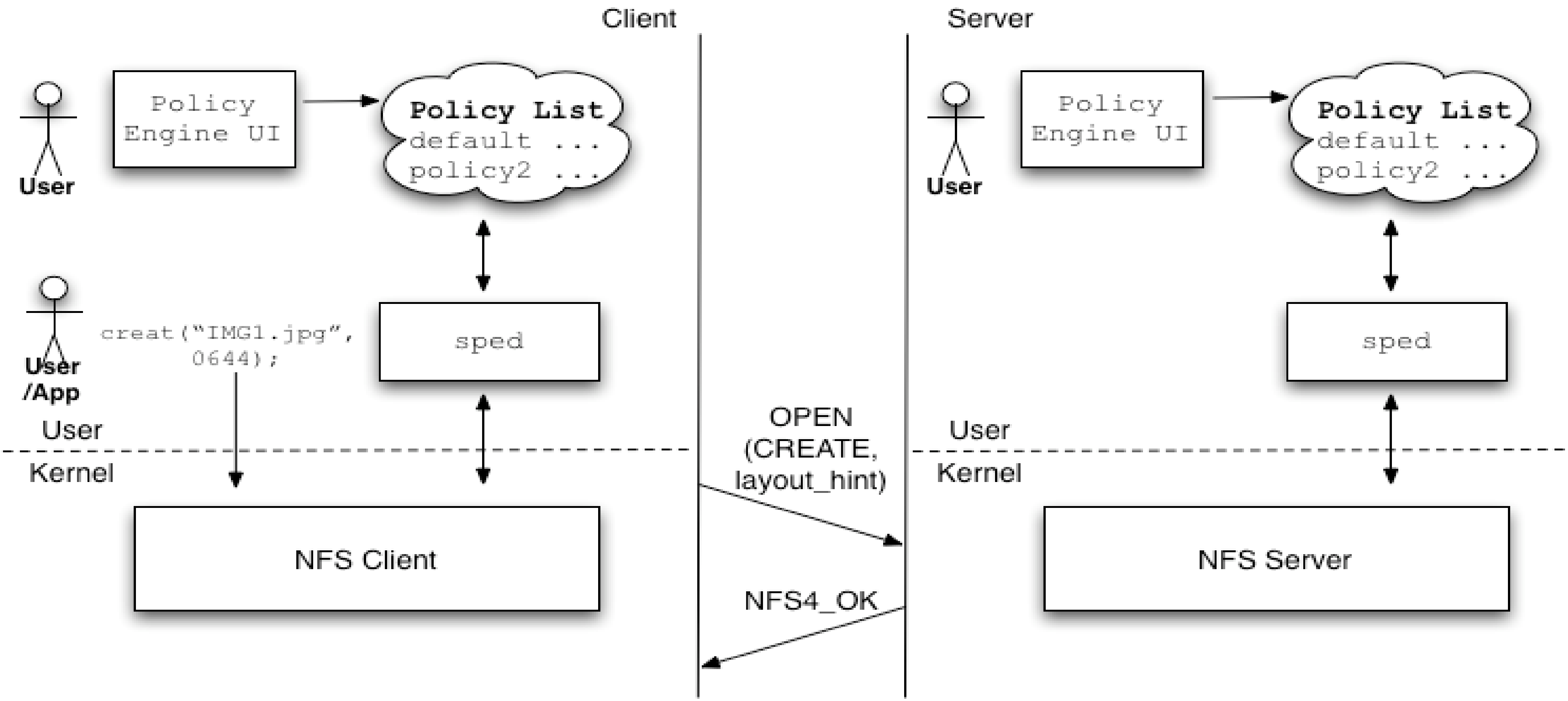
- Server side policies describe:
  - File attributes
    (e.g., path, uid/gid, file name (solaris.*, *.jpeg), create time/date)
  - Environment attributes
    (e.g., DNS subd main, fully qual. domain name, host name/IP, subnet, weekday, hour of day)
  - Striping information (e.g. stripe count, unit size)
  - Specific storage (i.e., data servers) for the file to be allocated on

Management: Client-side SPE

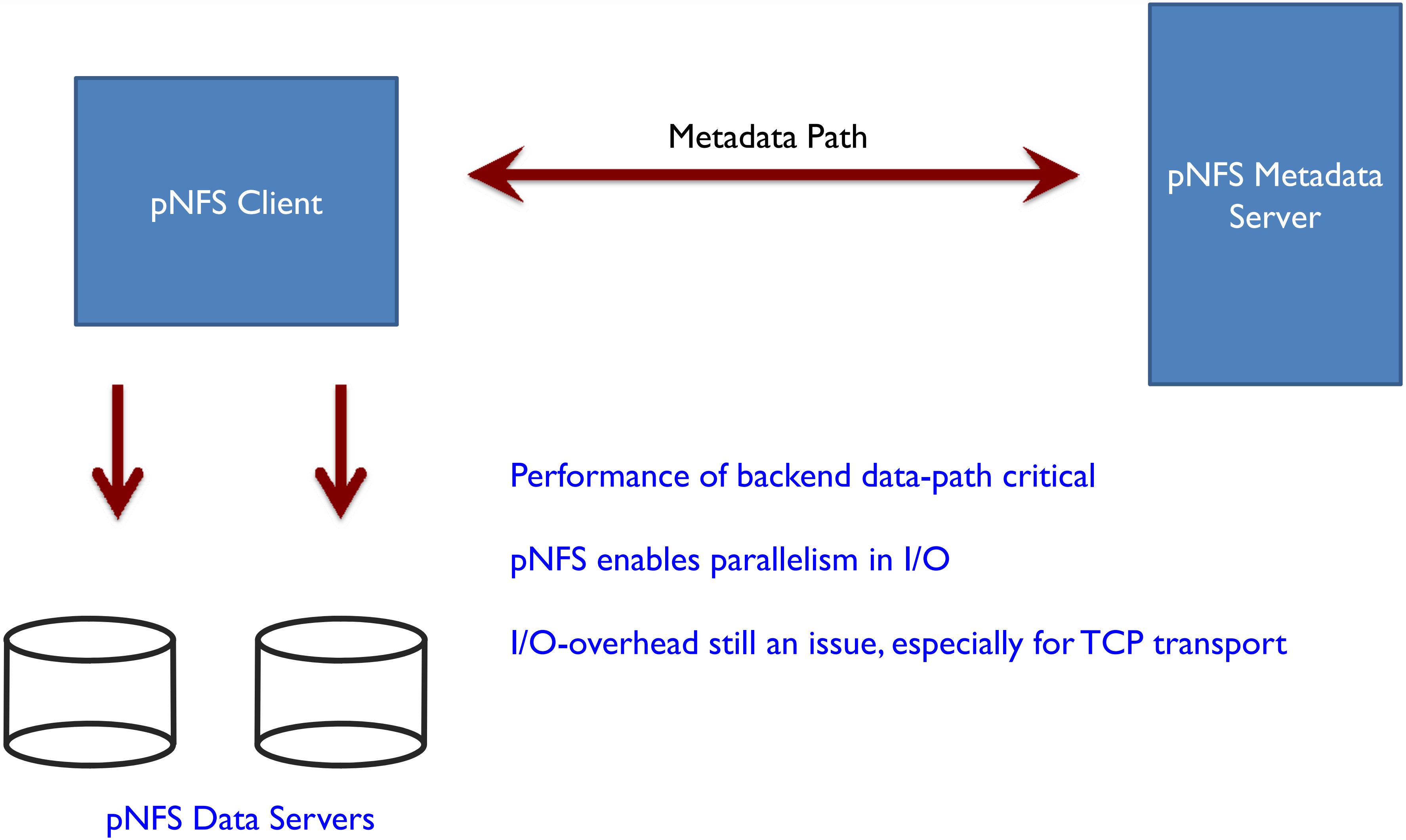
- Policies set at the client are "hints"
  - File attributes
  - Environment attributes
  - Striping information (e.g. stripe count, unit size)
  - No storage information

- The "hints" are sent to the server in a protocol specified attribute: layout_hint

Management: Prototype SPE Architecture



Performance: On the Data Path

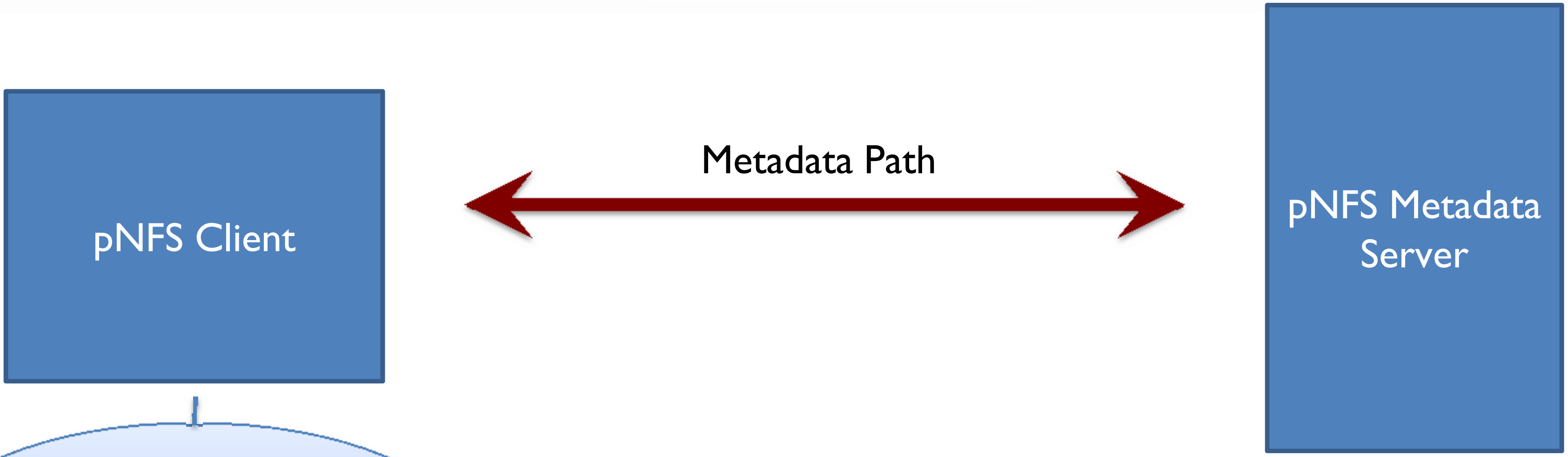


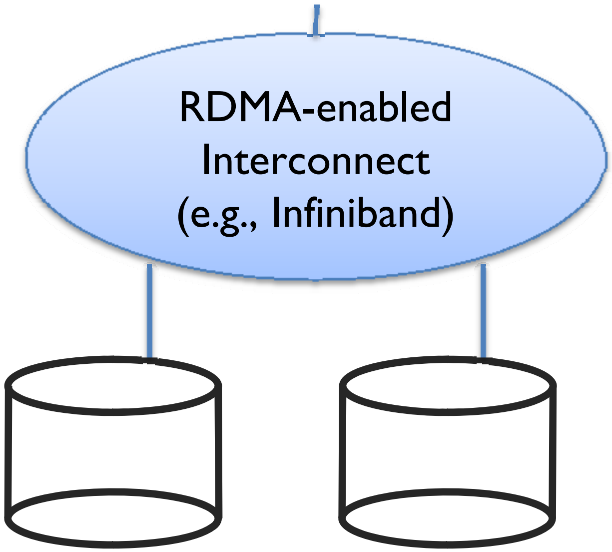
Performance of backend data-path critical

pNFS enables parallelism in I/O

I/O-overhead still an issue, especially for TCP transport

Performance: pNFS over RDMA





RDMA transport enables low-overhead I/O

- Low CPU utilization
- Reduced latency
- Increased throughput

pNFS will do I/O calls over RDMA transport

NFS over RDMA for NFSv3/v4 already in OpenSolaris distribution

pNFS Data Servers

Performance: NFS-over-RDMA

- NFS/RDMA project (integrated in 2008)
  - OpenSolaris implementation of RPC for InfiniBand
  - http://opensolaris.org/os/project/nfsrdma

- Reads - ~2.8 GB/sec, Writes - ~1.9 GB/sec

- Single client against SunFire x4170 server
  - PCIe gen2 and 2x Intel Xeon @ 2.27 Ghz CPUs
  - 12 GB memory
  - Mellanox ConnectX HCAs
  - QDR network
  - Solaris onnv b117, backing file system: zfs
  - Filebench workload generator

pNFS Standardization

- One (among many) of the features in NFSv4.1
  - Protocol standardized in the IETF
    - NFSv4.1 Internet Draft has been approved for RFC publication (but no RFC number given as of yet)

- http://www.ietf.org/id/draft-ietf-nfsv4 minorversion1-29.txt

pNFS Contributors

BlueArc	NetApp
CITI	Ohio SuperComputer Center
CMU	Panasas
EMC	Seagate
IBM	StorSpeed
LSI	Sun Microsystems
OSU	Desy

pNFS Clients and Servers

- Clients
  - Sun (Files)
  - Linux (Files / Blocks / Objects)
  - Desy / dCache (Java-based / Files)
- Servers
  - Sun (Files)
  - Linux (Files)
  - NetApp (Files)
  - EMC (Blocks)
  - LSI (Blocks)
  - Panasas (Objects)
  - Desy / dCache (Java-based / Files)

Current Status: pNFS at Sun

- Sun has demonstrated Solaris pNFS client and server prototypes to help complete IETF work.
  - Interoperability demonstrated at 02/09 Connectathon (in CA) and 06/09 Bakeathon (in CA).
  - Next Bakeathon event 10/09 (in TX).
  - Next Connectathon event 02/10 (in CA)
    - http://www.connectathon.org

Summary

- pNFS is the basis for innovations at Sun to provide storage solutions that are:
  - Standards based
  - Policy-driven (SPE)
  - High performing (pNFS-over-RDMA)

- More info: http://opensolaris.org/os/project/nfsv41/
  - nfsv41-discuss@opensolaris.org

BACKUP SLIDES

Management: Prototype SPE (Example)

- Each policy is of the form:
  - id, stripe count, unit size, npools, attribute expression

```
[root@pnfs-4-05 ~]> more /etc/policies.spe
10, 8, 16k, swimming:diving:wading:default, path == /pnfs1/nfs41
20, 4, 128k, swimming:diving, path == /pnfs1/pnfs
30, 4, 2k, diving:swimming, path == /pnfs1/default
40, 3, 8k, wading:diving, path == /pnfs2/nfs41
50, 4, 4k, swimming:wading, path == /pnfs2/pnfs

[root@pnfs-4-05 ~]> more /etc/npools.spe
default pnfs-4-09:pnfs1/alt_ds7 pnfs-4-06:pnfs1/ds1 pnfs-4-09:pnfs2/alt_ds8
pnfs-4-06:pnfs2/ds2
swimming pnfs-4-07:pnfs1/ds3 pnfs-4-08:pnfs1/ds5
diving pnfs-4-07:pnfs2/ds4 pnfs-4-08:pnfs2/ds6
wading pnfs-4-09:pnfs2/ds8 pnfs-4-09:pnfs1/ds7
```

Policies in Action: 3-way striped


```
[root@pnfs-4-03 ~]> nfsstat -l  /mnt/pnfs2/nfs41/file1
Number of layouts: 1
….
Layout [0]:
        Layout obtained at: Thu Sep 10 12:39:53:181651 2009
        iomode: LAYOUTIOMODE_RW
        offset: 0, length: EOF
        num stripes: 3, stripe unit: 8192
        Stripe [0]:
                tcp:pnfs-4-07.Central.Sun.COM:10.1.233.17:47009 OK
        Stripe [1]:
                tcp:pnfs-4-08.Central.Sun.COM:10.1.233.18:47009 OK
        Stripe [2]:
                tcp:pnfs-4-09.Central.Sun.COM:10.1.233.19:47009 OK
```

Policies in Action: 4-way striped


```
[root@pnfs-4-03 ~]> nfsstat -l /mnt/pnfs2/pnfs/file2
Number of layouts: 1
…
Layout [0]:
    Layout obtained at:Thu Sep 10 12:41:03:882393 2009
    iomode: LAYOUTIOMODE_RW
    offset: 0, length: EOF
    num stripes: 4, stripe unit: 4096
    Stripe [0]:
        tcp:pnfs-4-07.Central.Sun.COM:10.1.233.17:47009 OK
    Stripe [1]:
        tcp:pnfs-4-08.Central.Sun.COM:10.1.233.18:47009 OK
    Stripe [2]:
        tcp:pnfs-4-09.Central.Sun.COM:10.1.233.19:47009 OK
    Stripe [3]:
        tcp:pnfs-4-09.Central.Sun.COM:10.1.233.19:47009 OK
```

Policies in Action: 8-way striped


```
[root@pnfs-4-03 ~]> nfsstat -l /mnt/pnfs1/nfs41/file3
…..
    offset: 0, length: EOF
    num stripes: 8, stripe unit: 16384
    Stripe [0]:
        tcp:pnfs-4-06.Central.Sun.COM:10.1.233.16:47009 OK
    Stripe [1]:
        tcp:pnfs-4-06.Central.Sun.COM:10.1.233.16:47009 OK
    Stripe [2]:
        tcp:pnfs-4-09.Central.Sun.COM:10.1.233.19:47009 OK
    Stripe [3]:
        tcp:pnfs-4-09.Central.Sun.COM:10.1.233.19:47009 OK
    Stripe [4]:
        tcp:pnfs-4-07.Central.Sun.COM:10.1.233.17:47009 OK
    Stripe [5]:
        tcp:pnfs-4-08.Central.Sun.COM:10.1.233.18:47009 OK
    Stripe [6]:
        tcp:pnfs-4-07.Central.Sun.COM:10.1.233.17:47009 OK
    Stripe [7]:
        tcp:pnfs-4-08.Central.Sun.COM:10.1.233.18:47009 OK
```