pNFS Capabilities in the Real World

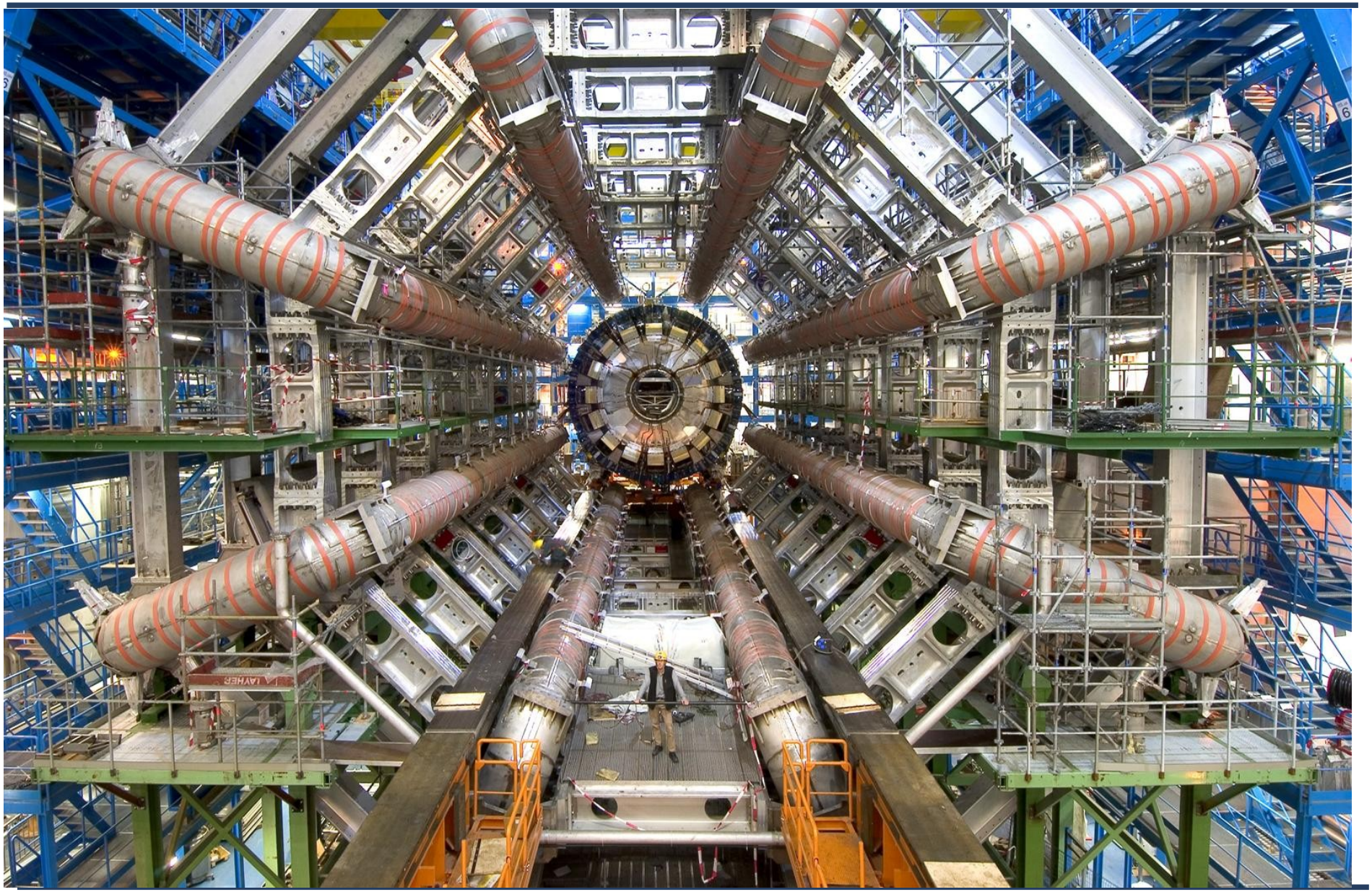
or

pNFS under fire

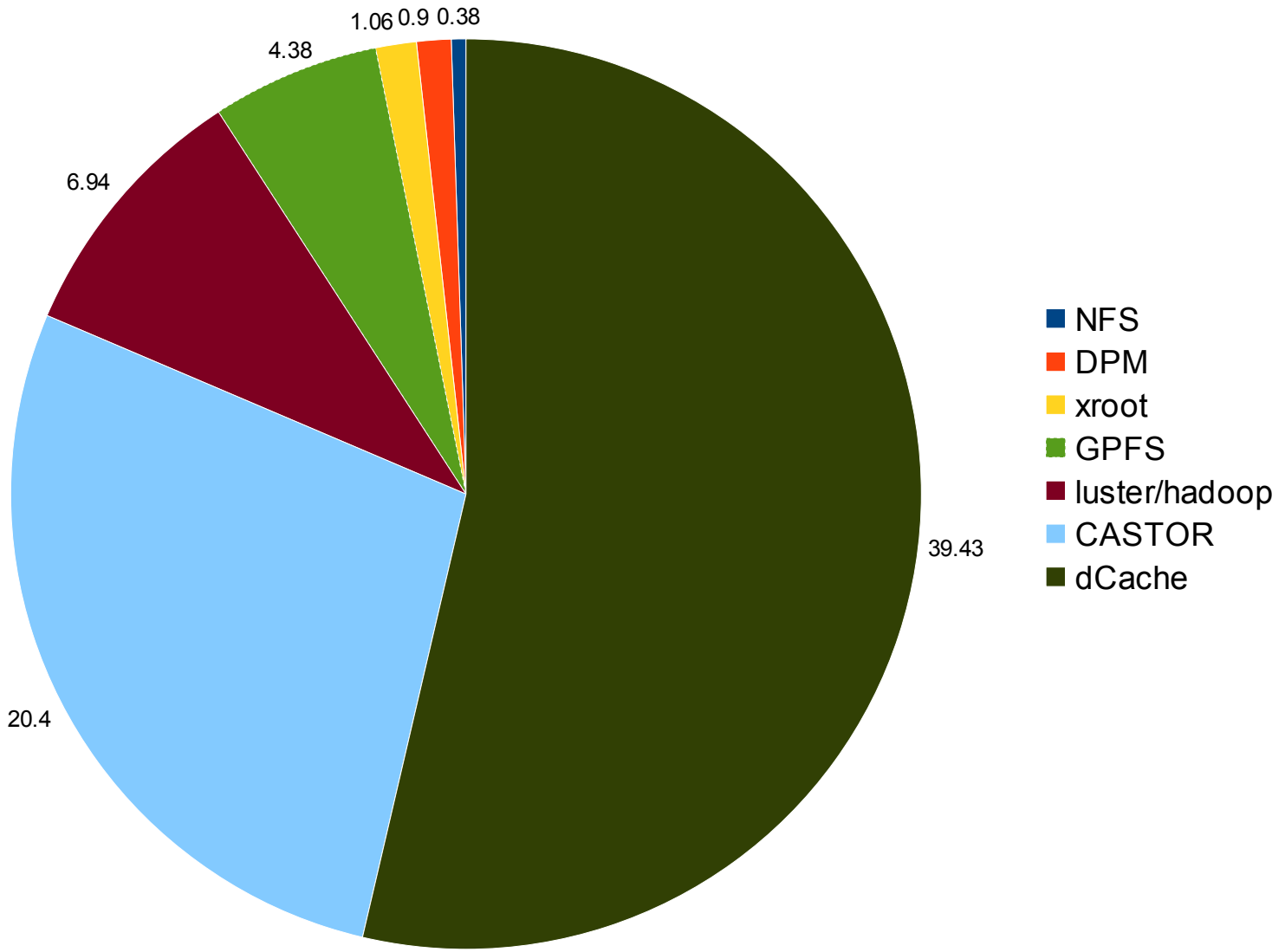
(update)

Tigran Mkrtchyan for dCache team.

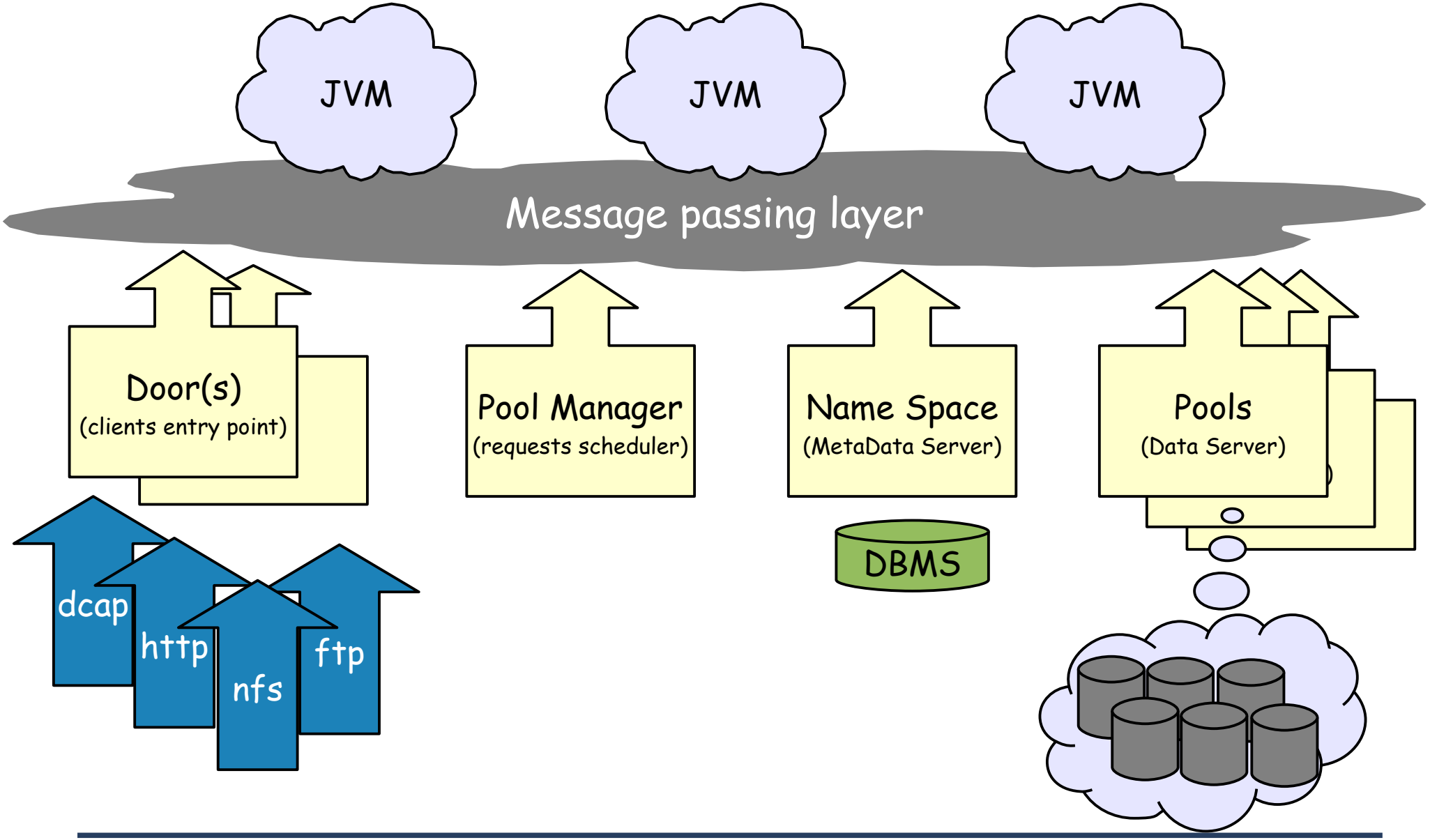
Storage type share (in PB) @ HEP



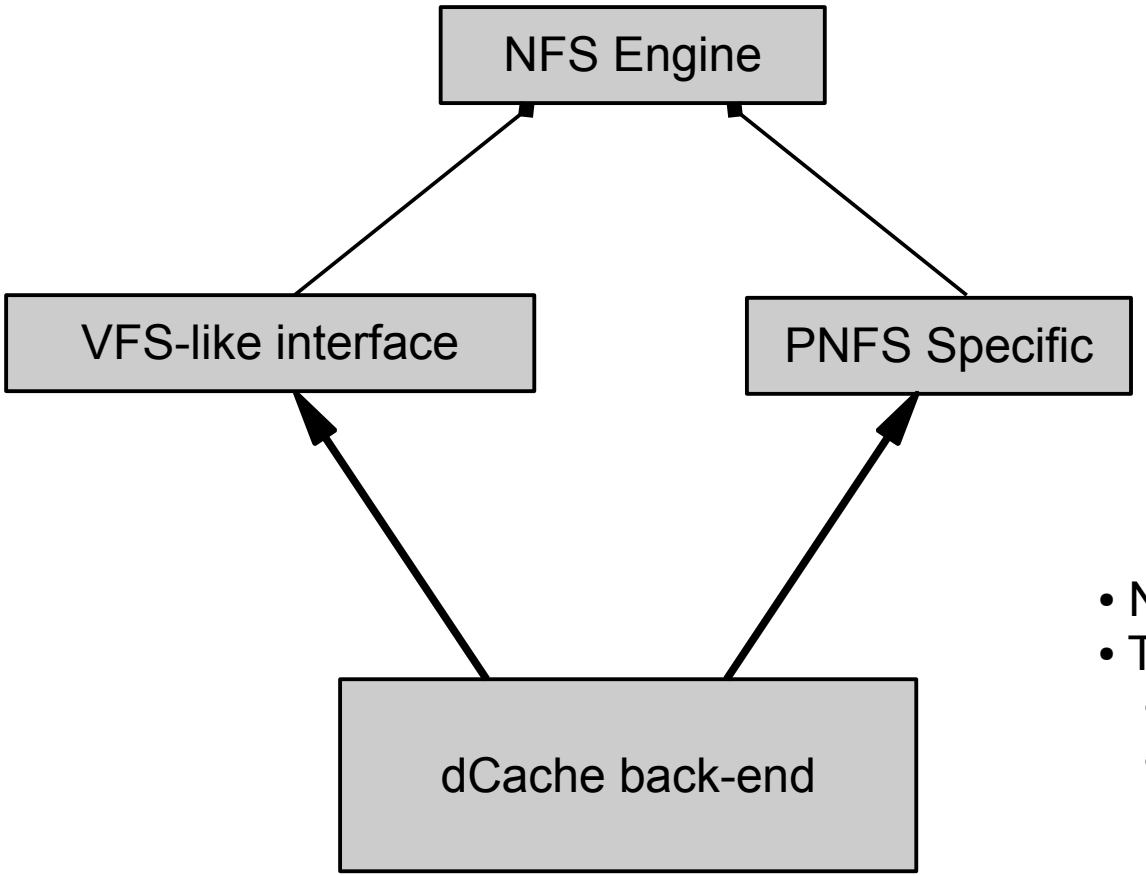
Storage type share (in PB) @ HEP



dCache in one slide

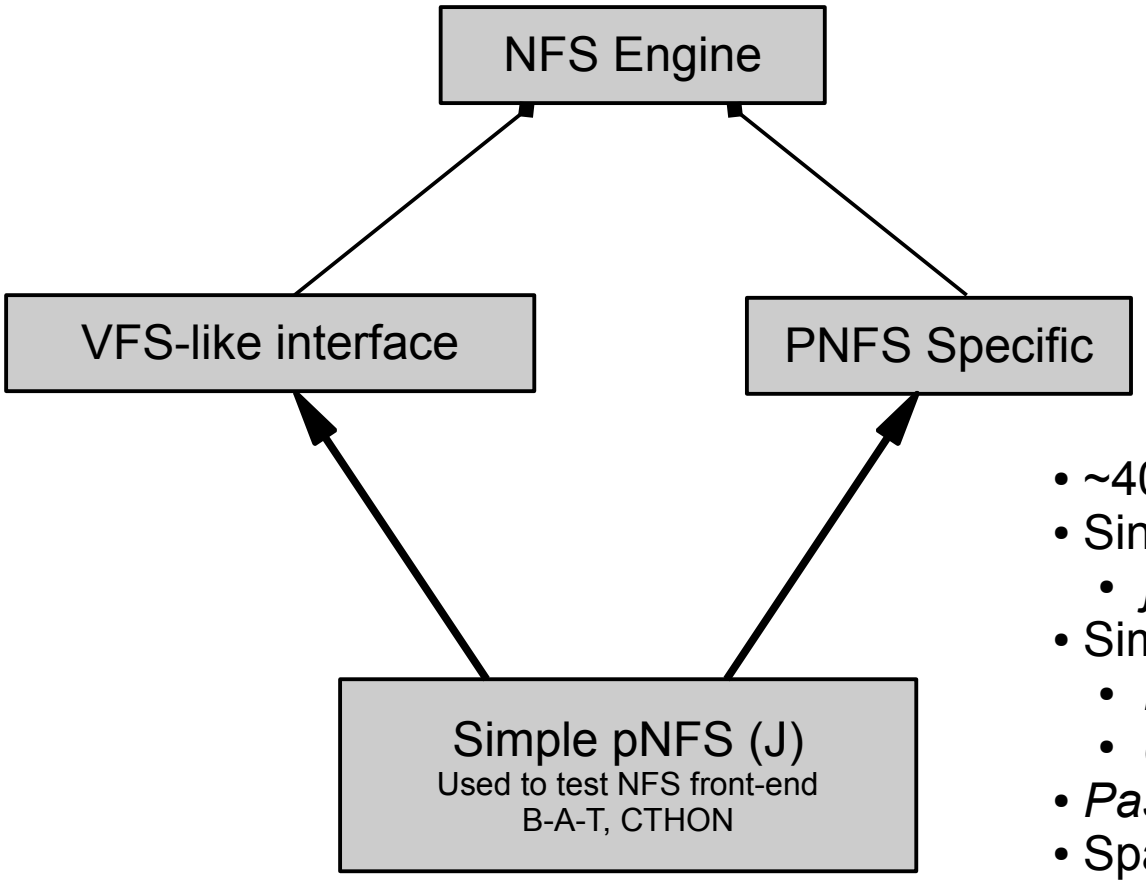


NFS module



- NFS server not aware of backend
- Two interfaces to implement
  - Metadata operations
  - Layout management

NFS module



- ~400 LOC
- Singe jar file to start
  - *java -jar server.jar*
- Simple config
  - *mds.devices=ds1:2049,ds2:2049*
  - *ds.base=/tmp/pNFS*
- *Passes CTHON*
- Sparse striping
- Turns cluster FS in pNFS
- A prototype for other implementations

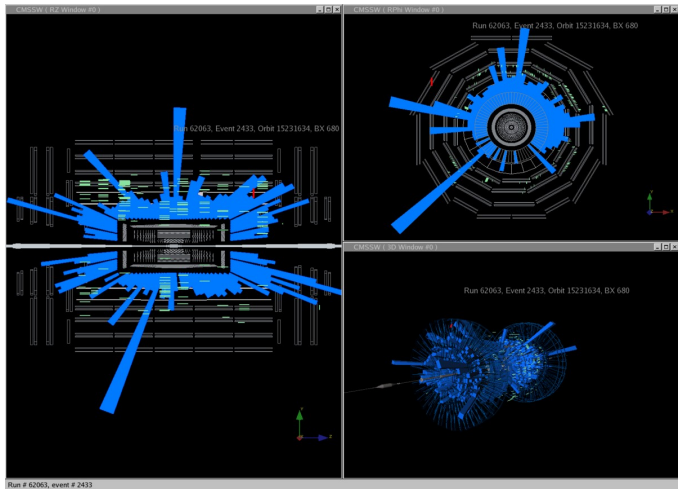
dCache

- Immutable data (PUT/GET/REMOVE/METADATE UPDATE)
  - Common use-case today?
- Policy based file migration
  - Simple interface to tape systems
- Metadata query optimized namespace (MDS)
  - RDBMS back-end with SQL access
- Aggregates simple building blocks into distributed storage
  - **Don't panic:** we use your boxes as SAN (fc) or DAS (JBOD)
  - Java VM and filesystem required (typically, xfs and zfs)

dCache + pNFS

- No file striping in upcoming (1.9.12) version
  - (sparse) Striping on read in test phase
  - (dense) Striping on WRITE come later
- No proxy-io through MDS
  - Will be for complete IO only
- No LOCKs – all files are immutable
- LAYOUTRECALL on close
  - but we work on changing this
- WE DO NOT EXPORT EXISTING FILE SYSTEMS
  - we are the file system

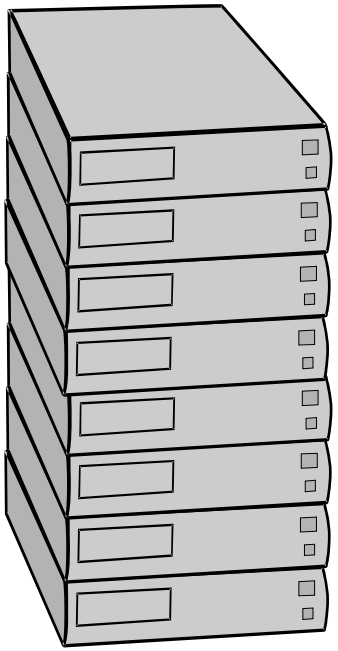
Cthon-2010



- 9 cores
- 9 hosts (old HW)
- 9 jobs
- 8 DS
- 1GB

Job description

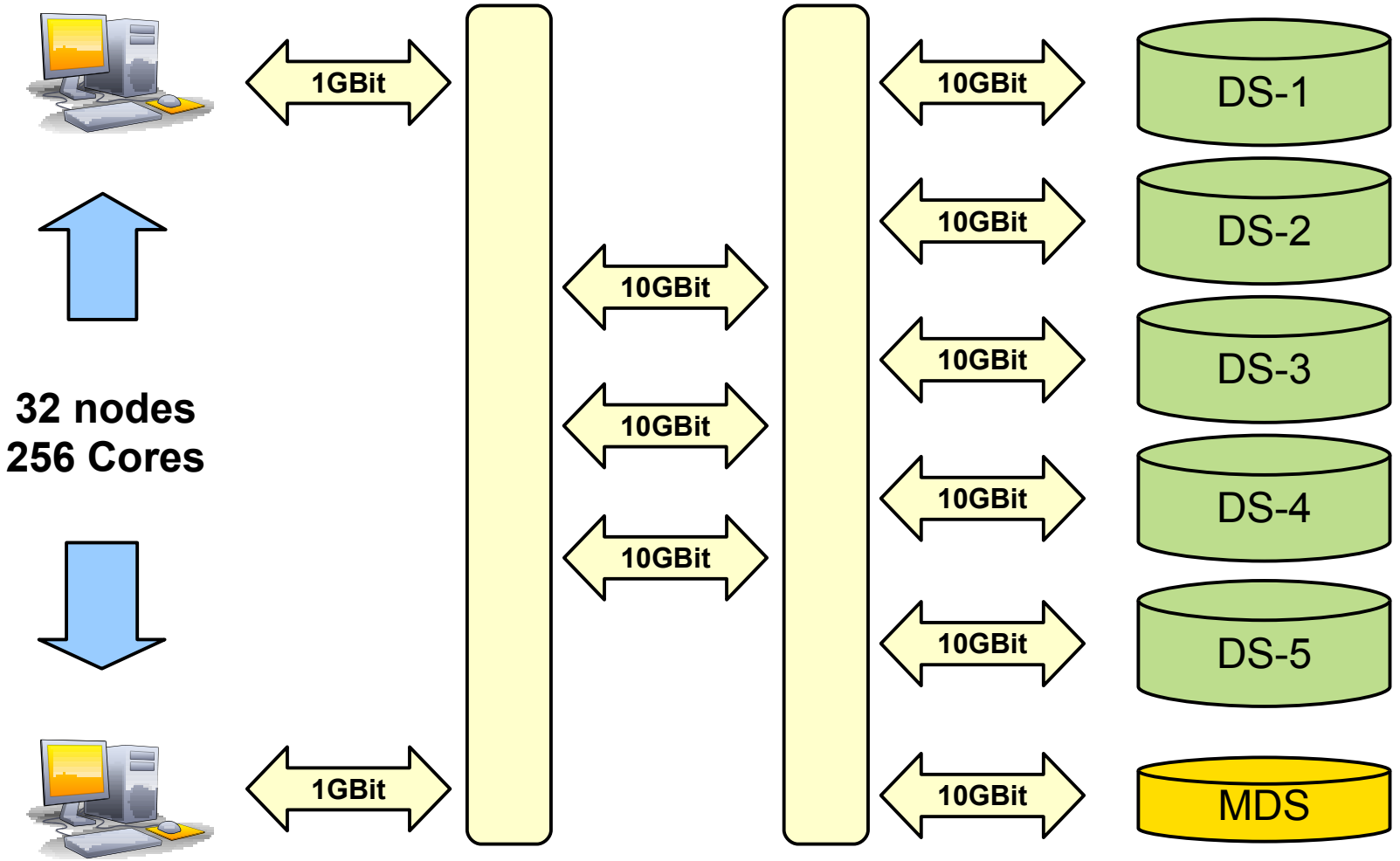
Result



~~LUSTRE~~

pNFS (dCache)

The testbed (2011)



Environment

Production like environment in a size of small Lab:

Servers : dcache-1.9.10-2, kernel-2.6.18
- MDS + 5xDEll R510
  - 8 cores, 12 GB RAM, 12x2TB SATA, 80TB total
- 10 GB/s uplink

Clients:
- 32x 8 core, 16GB RAM, RHEL5.3 x64 + kernel-2.6.37-pnfs
  - 256 job in total

Network:
- 3GB/s worker nodes to storage

Software:
- Standard analysis jobs provided by physicists

Environment (II)

Recompiled for RHEL5 x64 packages from fedora-14 repository

- nfs-utils-lib
- nfs-utils
- libtirpc
- libgssglue

Yum repository available at:

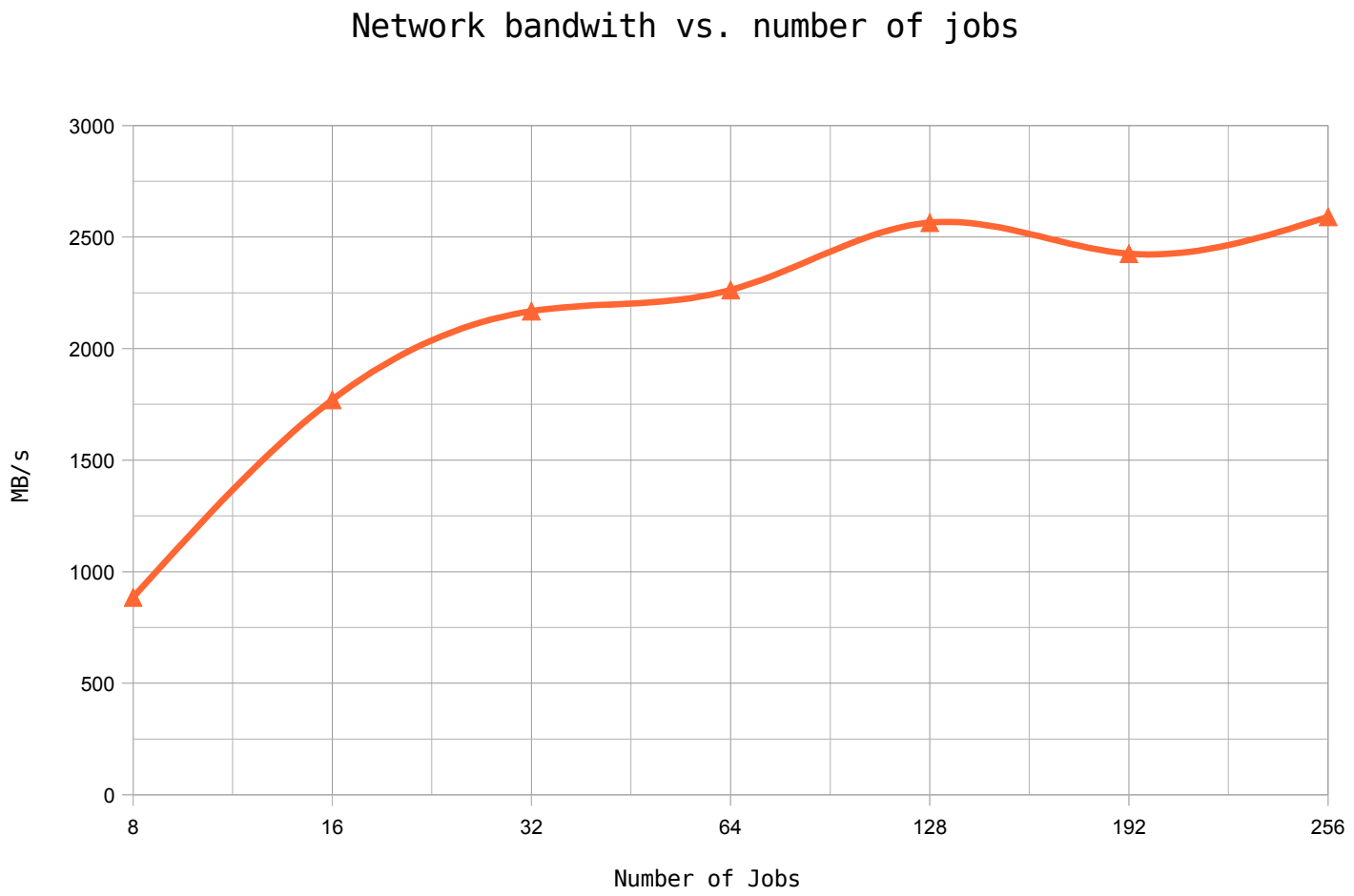
http://download.opensuse.org/repositories/home:/tigranm:/nfs-utils/RedHat_RHEL-5/home:tigranm:nfs-utils.repo

Stability

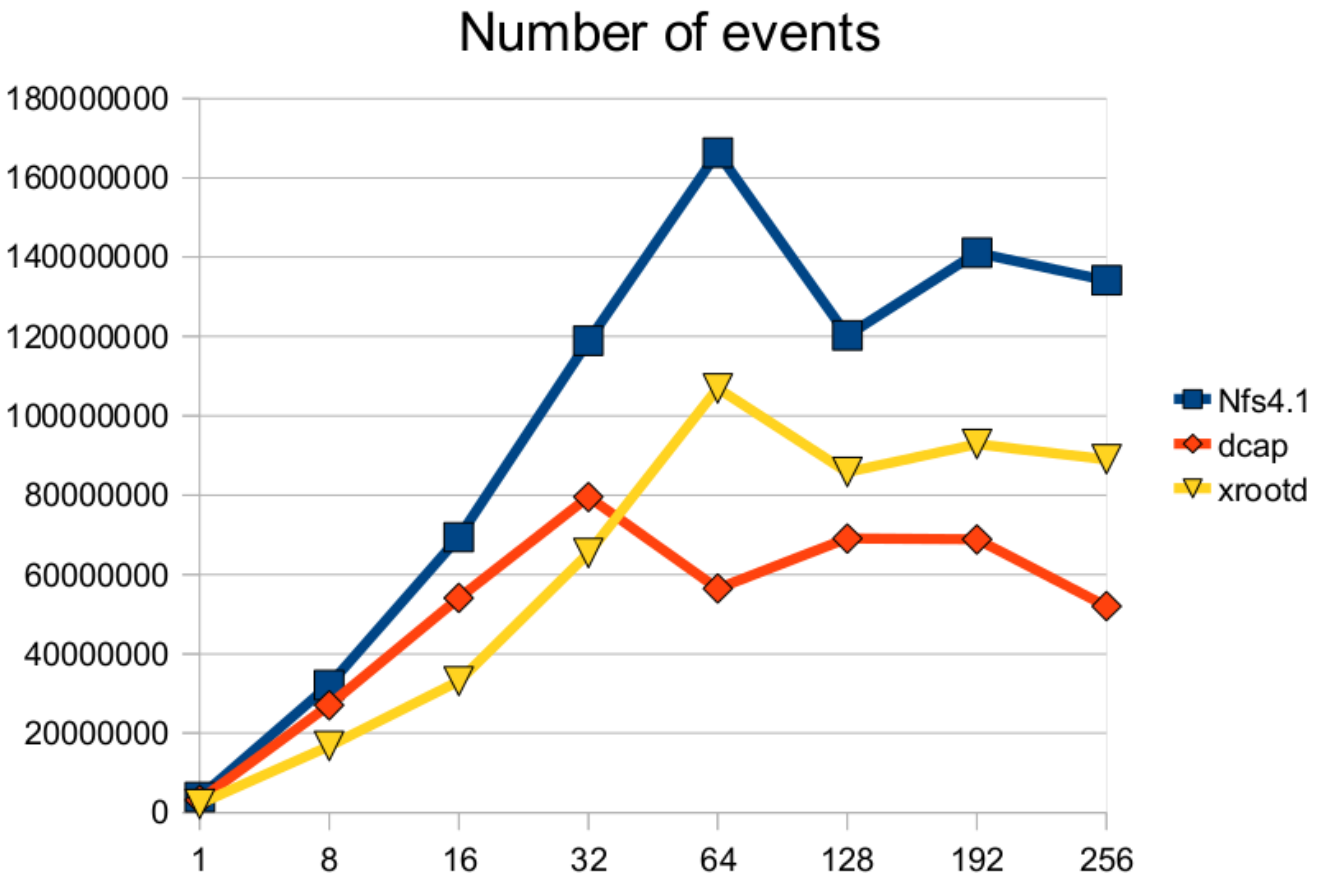
- 13 TB over WAN, ~100GB each file
- Kernel-2.6.xx.tar.bg2 unpack
- 'ls -lR' over slow connection
- Number of concurrent jobs from one to 256
- Each job was running for 24 hours
- None of the files was used twice

Operational issues

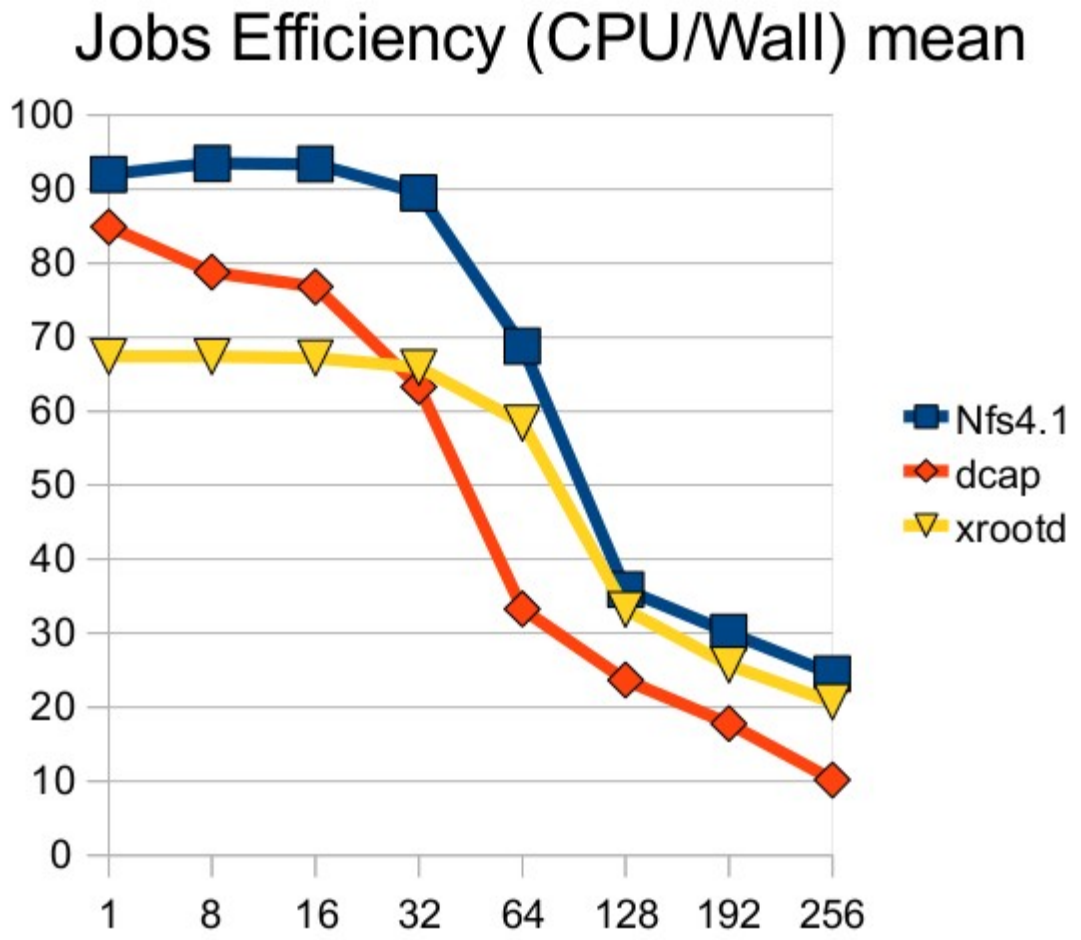
Performance



Performance (II)



Performance (III)



Conclusions #1

- We have dedicated testing environment
  - 50% CPU and 30% storage of a small Lab
- System is stable for on field testing and brave users
- NFSv41 often performs better than home grown protocols
  - See next slides
- We are able to utilize 100% of DISK/NET bandwidth
- Our applications survive upstream kernel with RHEL5

Conclusions #2

We have tested the dCache server implementation of the NFS v4.1(pnfs) protocol against the dCap protocol using LHC analysis code. Also synthetic and stability tests were performed. We see a remarkable stability of the dCache server and an overall comparable if not superior performance of NFS v4.1(pnfs). We clearly see effects and benefits from the client caching of the Linux kernel.

CHEP2010, LHC Data Analysis Using NFSv4.1 (pNFS)
Yves Kemp

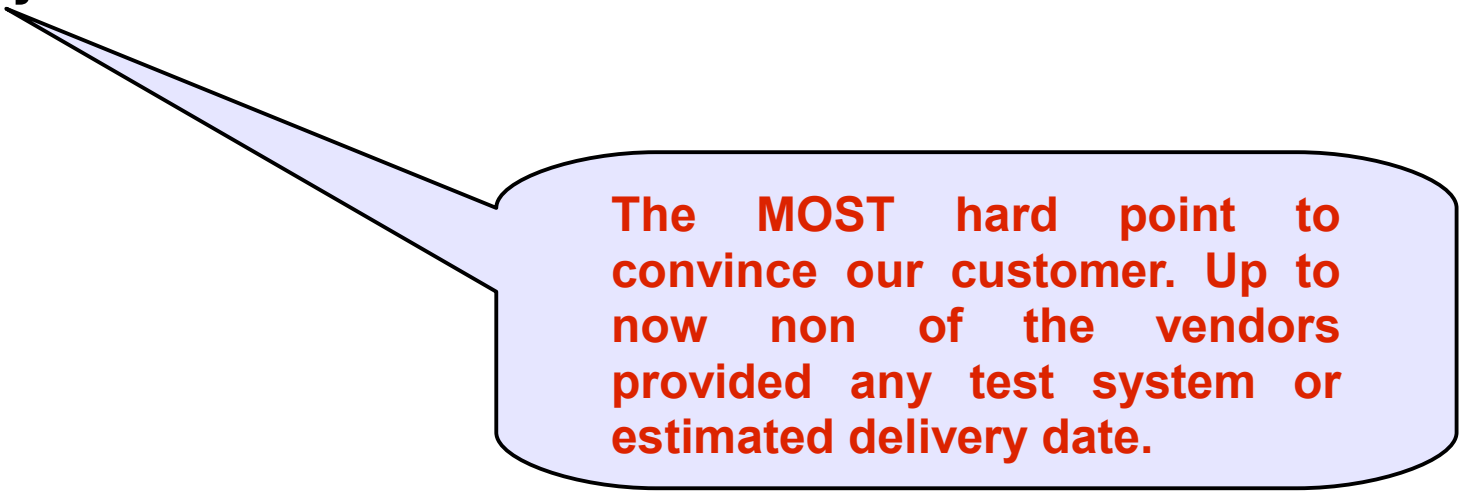
Are we ready?

Why we should use pNFS?

- No vendor lock
- Easy to move do a different solution
- No need to modify/re-link applications
- Client comes with OS (one day)
- Industry standard, rfc5661

Why we should use pNFS?

- No vendor lock
- Easy to move do a different solution
- No need to modify/re-link applications
- Client comes with OS
- Industry standard, rfc5661



Call for vendors

We have reasonable testing environment,
MAKE USE OF IT!

Work-in-progress

We are working on striping on read

- deviceid generated per file
  - may be the same in some cases
- Always empty device list
- The same FH for all DS
- Round-robin stripping pattern

Info and links

http://www.dcache.org/
http://www.eu-emi.eu/
http://www.dcache.org/downloads/nfsv41.repo