More Results …

dCache NFS 4.1 evaluation done by :

Yves Kemp
Tigran Mkrtchyan
Dmitri Ozerov

Presenter : Patrick Fuhrmann

Funding

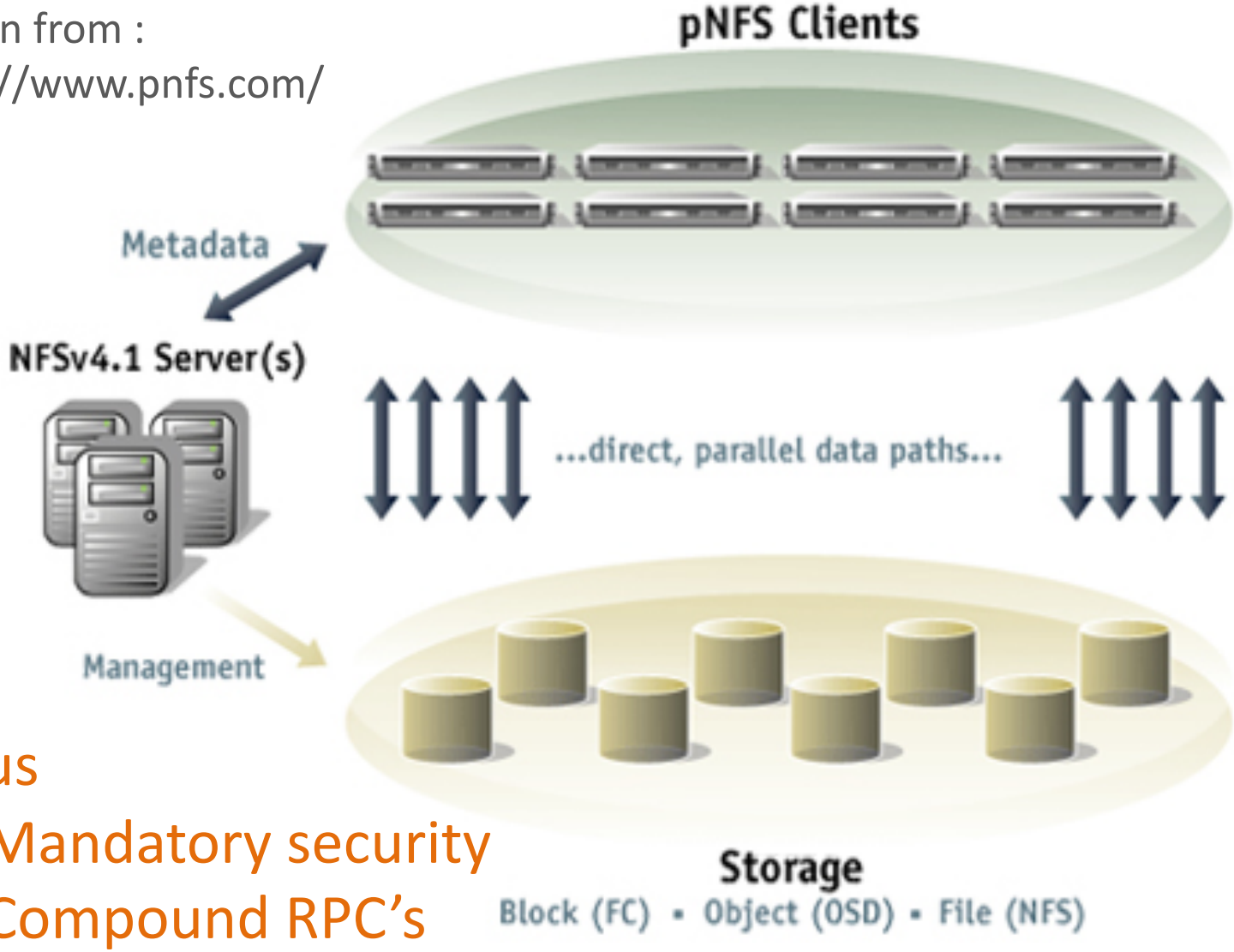
dCache.org
DESY
HELMHOLTZ ASSOCIATION
European Middleware Initiative
LCG

Content

- How differs NFS 4.1 (pNFS) from previous NFS'es.
- Who is behind the NFS 4.1 initiative
- NFS 4.1 evaluation in dCache.

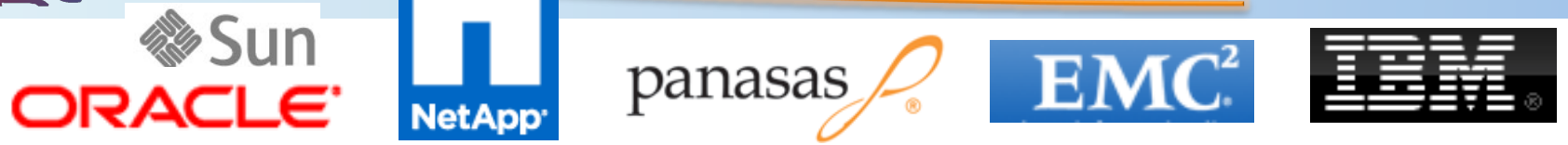
How does NFS 4.1 (pNFS) work ?

Stolen from :
http://www.pnfs.com/



Plus
✓Mandatory security
✓Compound RPC's

What is the NFS 4.1 initiative ?



- Industry initiative between all the major storage and OS vendors.
- Coordinated by CITI at the University of Michigan

- It is an WLCG demonstrator.

- Funded effort within the European Middleware Initiative

- Major effort in dCache

  - For non LCG communities
  - Hopefully for HEP as well

NFS 4.1 working group

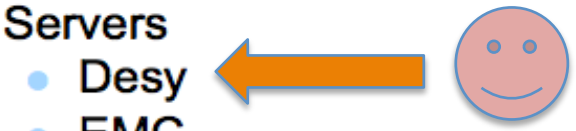
Tanja Baranova (dCache.org)
Jean-Philippe Baud (CERN, DPM)
Johannes Elmsheuser (LMU Munich, Atlas HammerCloud)
Yves Kemp (DESY, National Analysis Facility)
Maarten Litmaath (CERN)
Tigran Mkrtchyan (dCache.org)
Dmitri Ozerov (DESY)
Ricardo Rocha (CERN, DPM)
Andrea Sciaba (CERN)
Hartmut Stadie (DESY, CMS)

Who is behind NFS 4.1 (pNFS) ?

Stolen from : http://www.pnfs.com/

Industry Support - Implementations

- Clients
  - Linux
  - Sun (Solaris)
- Servers
  - Desy
  - EMC
  - IBM
  - Linux
  - NetApp
  - Panasas
  - Sun (Solaris)



Presented at SC'08

Several other implementations have been tested at Bake-a-thons and Connectathons

Why are they interested ?

Stolen from : http://www.pnfs.com/

Benefits of Parallel I/O

- Delivers Very High Application Performance
- Allows for Massive Scalability without diminished performance

Benefits of NFS (or most any standard)

- Ensures Interoperability among vendor solutions
- Allows Choice of best-of-breed products
- Eliminates Risks of deploying proprietary technology

Why is HEP interested ?

- Don't have to care about client software anymore.
- No specific ROOT drivers (dCap,rfio,xroot). Just 'open /foo/blah'
- Less software components to maintain.
- Can be used by unmodified applications (e.g. Mathematica®)

Interesting For System administrators

- regular mount-point as any other FS e.g. /afs, /pnfs.
- File/Block caching algorithms provided by professional computer scientists within the OS kernel.

More more arguments see :

"11 reasons you should care" by Gerd Behrmann

At dCache.org/manuals

Who is supporting/funding it in HEP

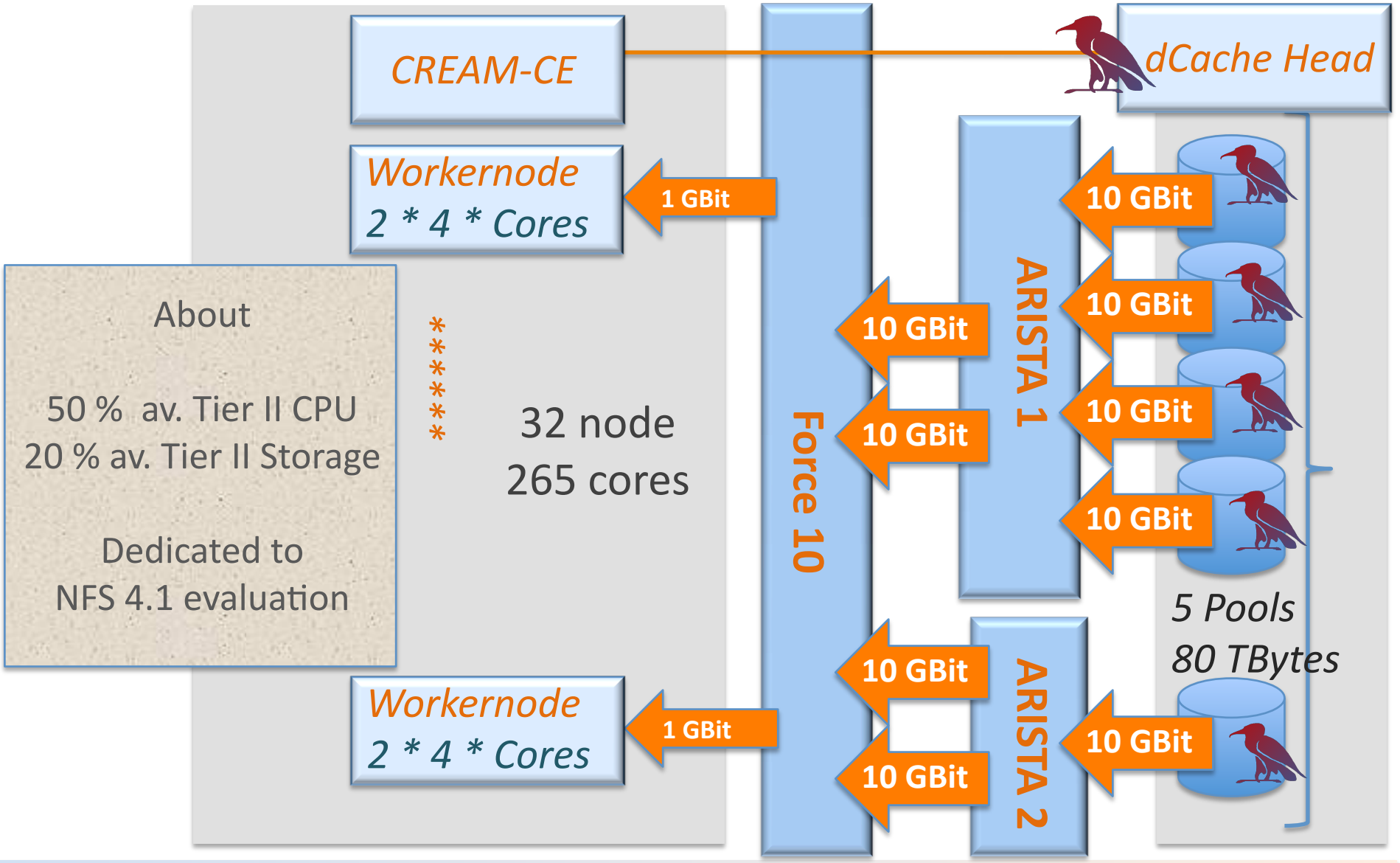
Within the European Middleware Initiative, DPM, dCache and very likely StoRM will provide an NFS 4.1 (pNFS) interface.

Imposed by the EC : EMI will only fund standards.

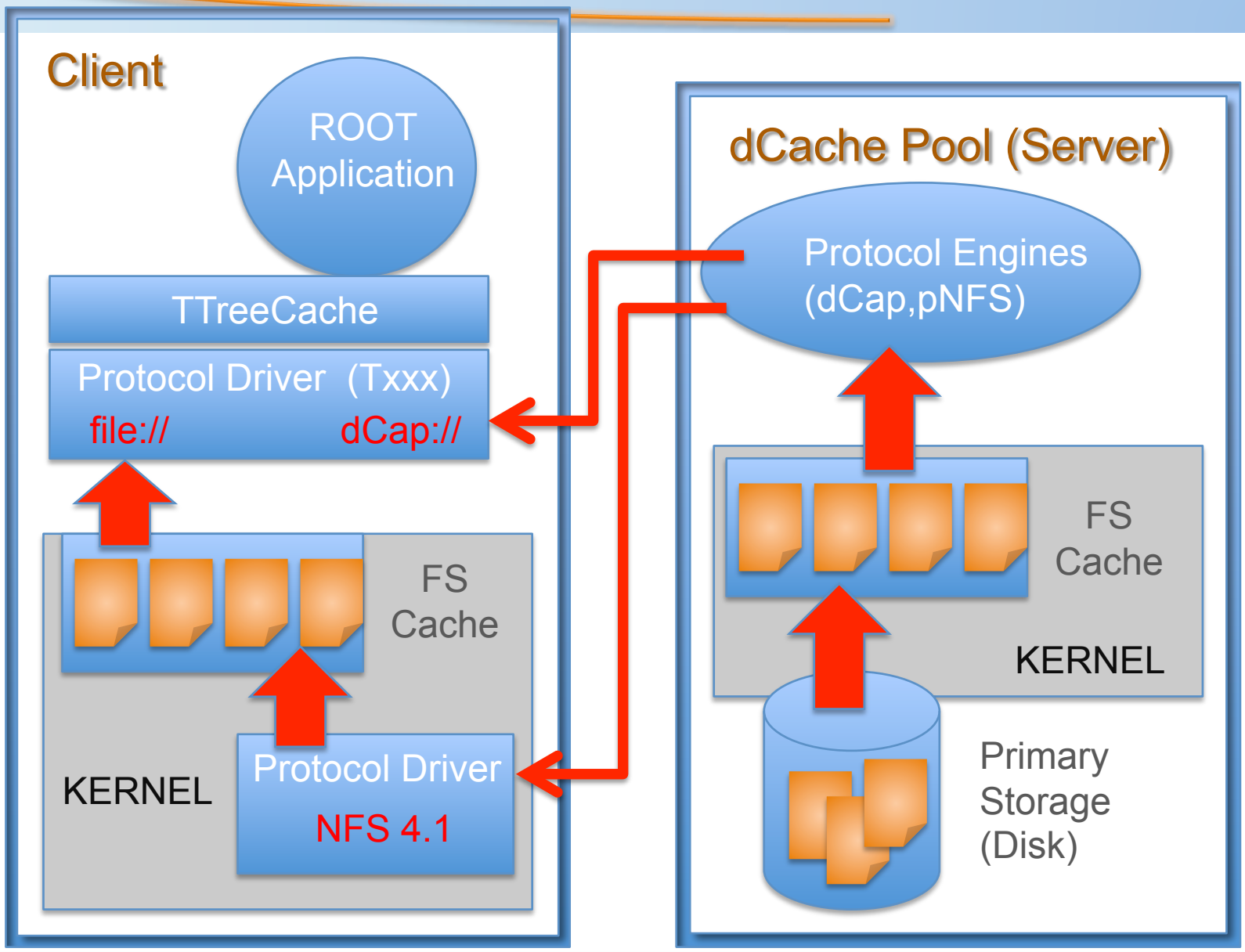
dCache production ready : 1.9.10
DPM : pNFS being finished these days.

NFS 4.1 (pNFS) evaluation In dCache

Our NFS 4.1 (pNFS) small Tier II ?



NFS 4.1 / dCap evaluation logic



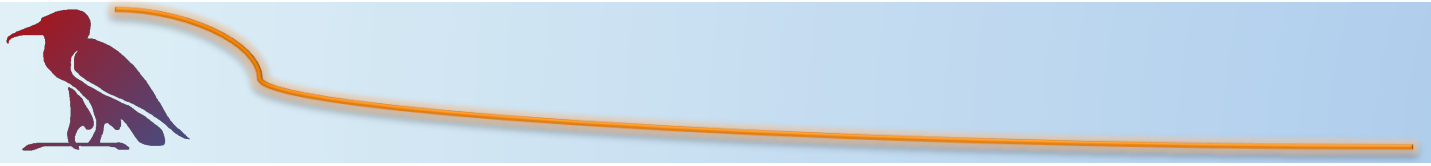
Class of test

- Stability evaluation
- Simple I/O testing
- ROOT tests
- ATLAS HammerCloud

All tests done with :

dCache 1.9.10

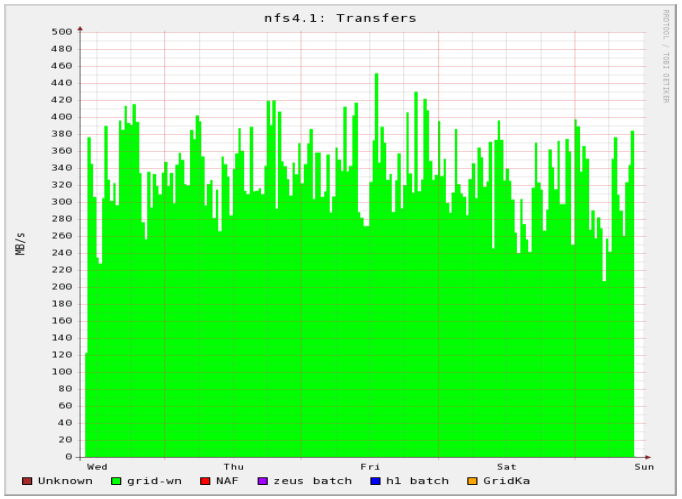
SL 5.3 2.6.36-rc3.pnfs

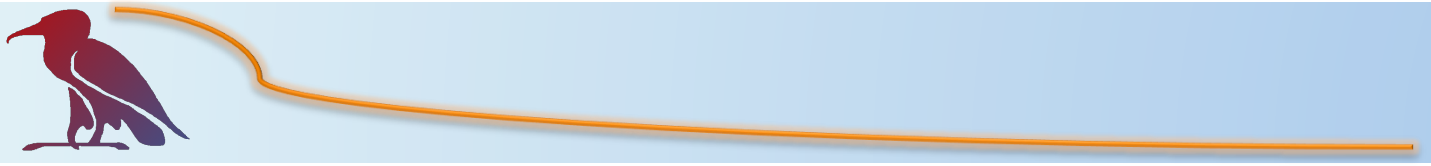


Stability

Stability

- CFEL Production Transfers from SLAC to DESY
  - 13 TBytes over 10 days
  - 100 GBytes average file size
  - No crash, no unexpected behaviour
- Un-taring Linux Kernel into NFS 4.1
  - No crash
- High-latency test
  - Recursive ‘ls –l’ over 60.000 files via DSL from home.
  - Finished w/o problem.
- 4 days at 330 MB/sec sustained Hammercloud. (stopped after 4 days)
- 128 Processes writing into the same file
  - Client nodes get stuck
  - Server was still ok





Simple I/O

Simple I/O Setup

Either

dccp <filename> /dev/null'

Or

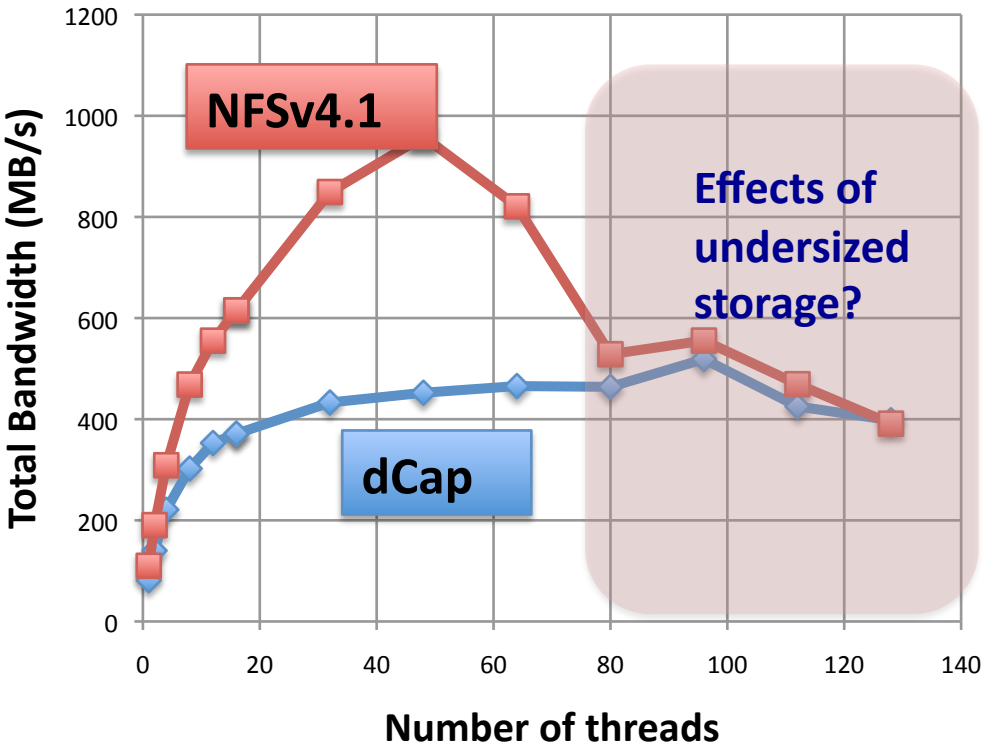
cat <filename> /dev/null

Only interested in protocol performance.
Preventing any client side caching effect.

✓Reading each file only once.
✓Reading files sequentially only.

Yves reported at CHEP'10

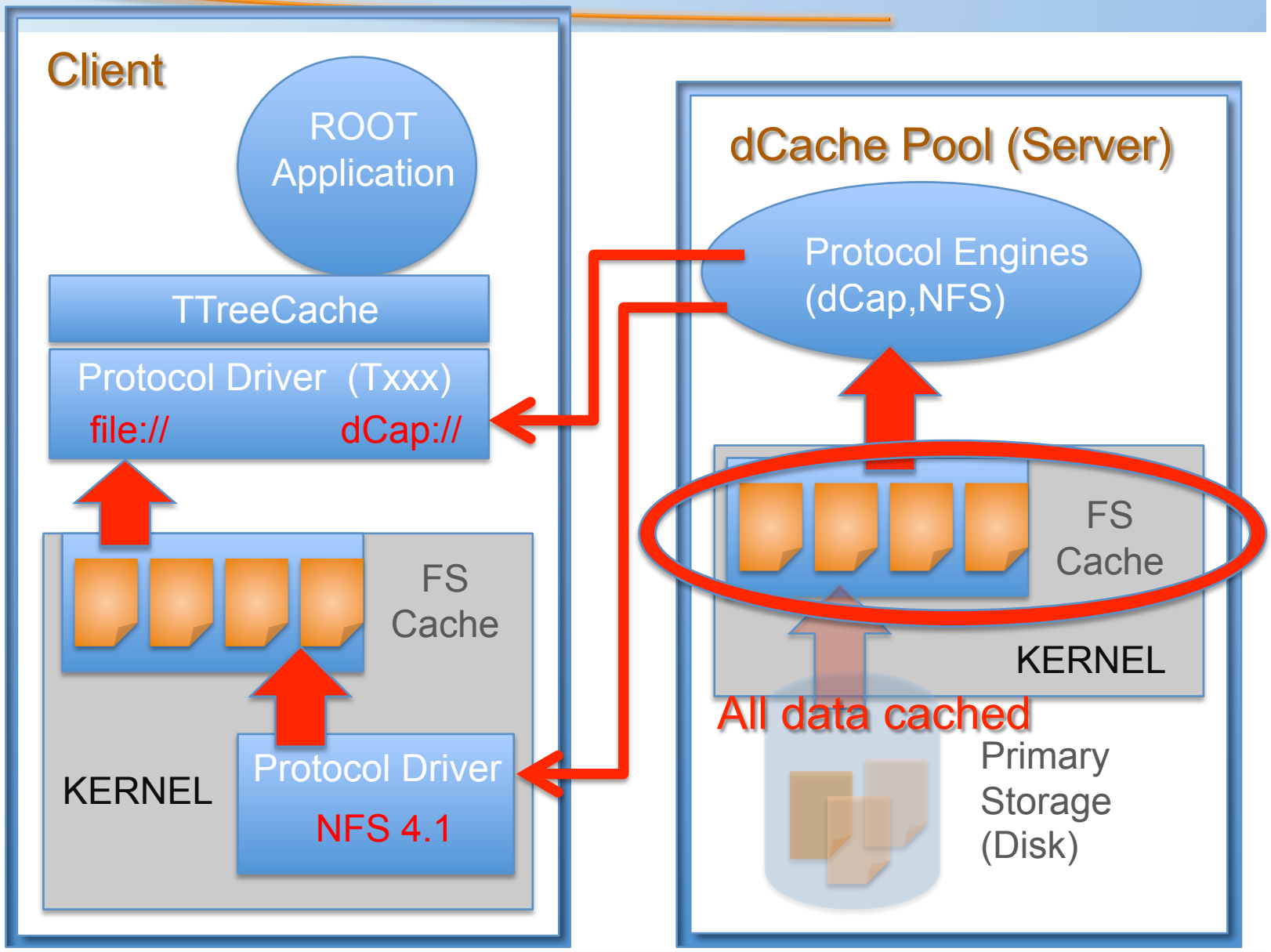
- Simple I/O
  - Reading file to /dev/null
  - No caching (read once, not jumping around in file)
  - A maximum of 128 clients (16 nodes)



- NFS behaves better than dCap up to a certain limit
- We have no definite answer for this effect, suppose congestion on the server
  - Probably due to undersized storage

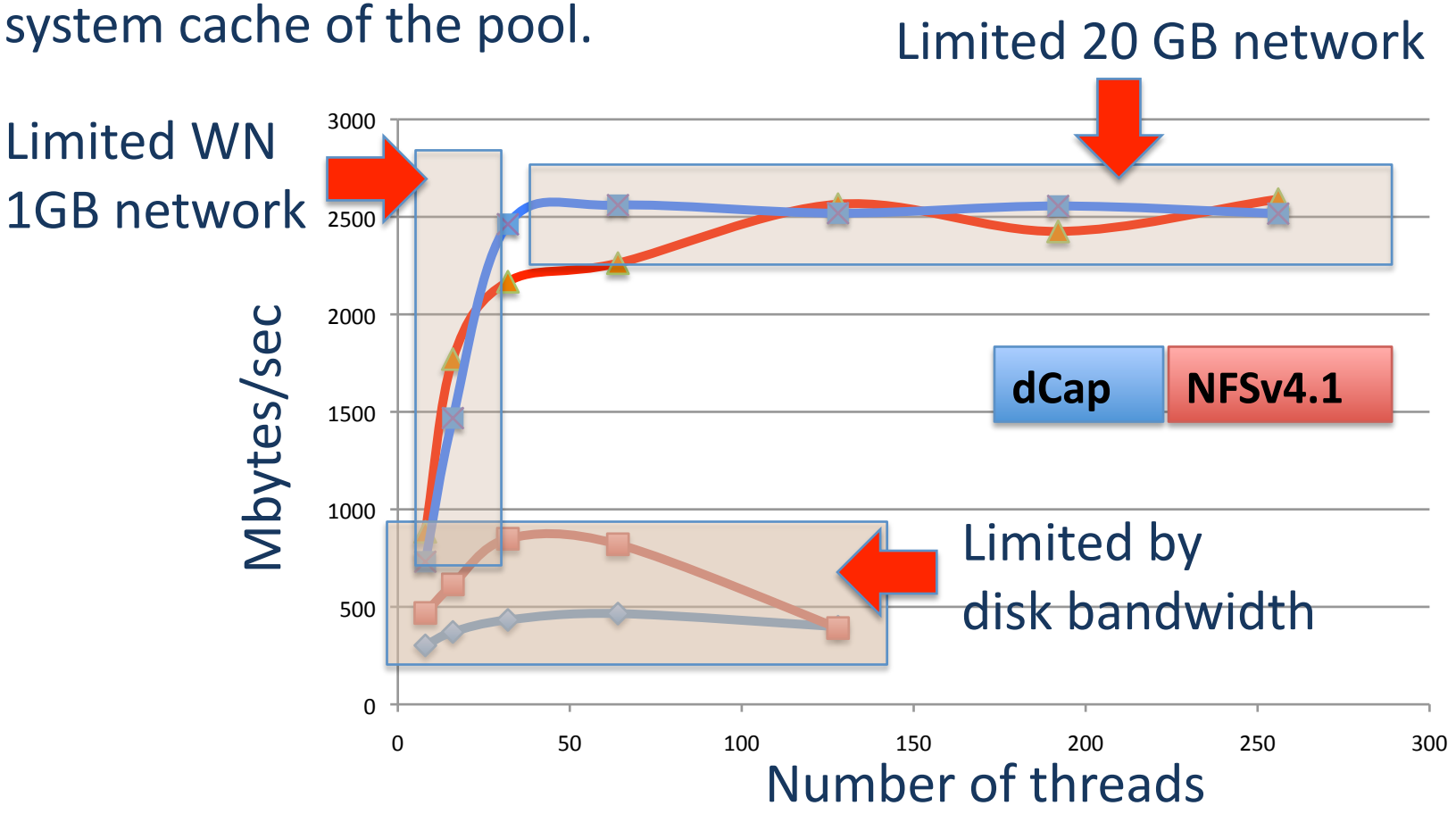
➔ **Needs further investigation**

Remove disk resp. server side FS effects

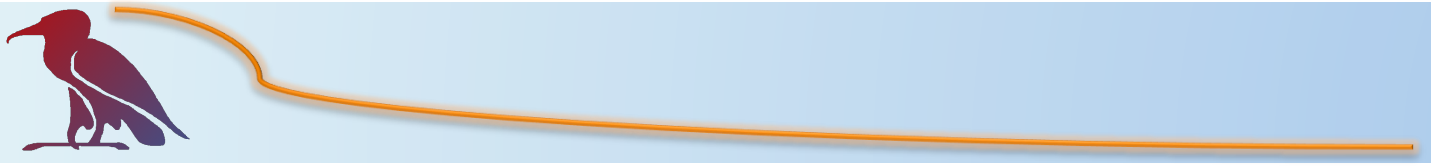


Limits

Removing server disk congestion effect by keeping all data in file system cache of the pool.



Total throughput doesn't depend on the protocol.



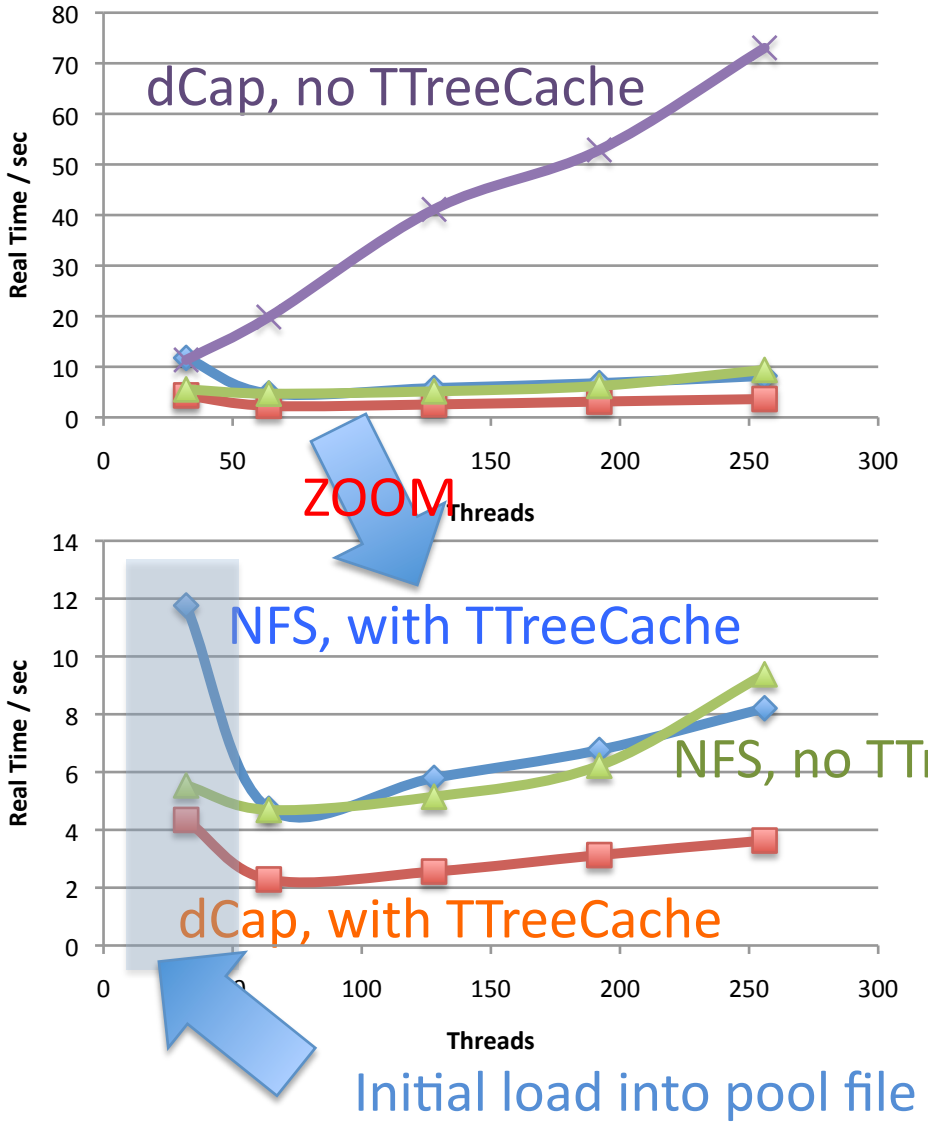
ROOT

ROOT Setup

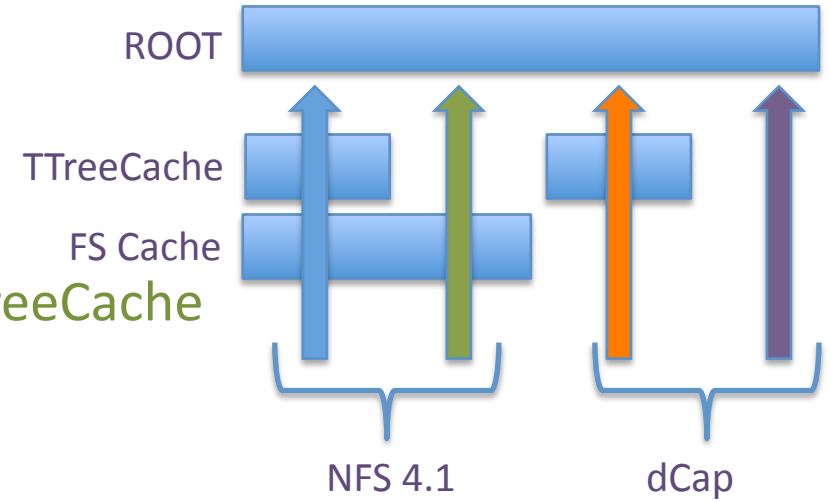
- New ROOT version 5.27.06, compiled with dCap support
- Files provided by René Brun: atlasFlushed.root (re-organized files with optimized buffers) and AOD.067184.big.pool_4.root (some other original file) (optimized: 1GByte, original 1.3 GByte)
- Test script provided by René: simple script reading events: taodr.C
- Different test runs:
  - Reading via NFS or dCap
  - Reading with 60MByte TreeCache, or with 0Byte TreeCache
  - Reading all branches or only 2 branches
  - 32, 64, 128, 192 or 256 jobs running in parallel

- Last minute-result! Have not spoken with ROOT people!

ROOT : Non optimized files, 2 trees only



- ✓Non optimized files
- ✓Reading only 2 trees.
- ✓TTreeCache does vector read with dCap.
- ✓VR = fadvise disabled in ROOT for NFS.



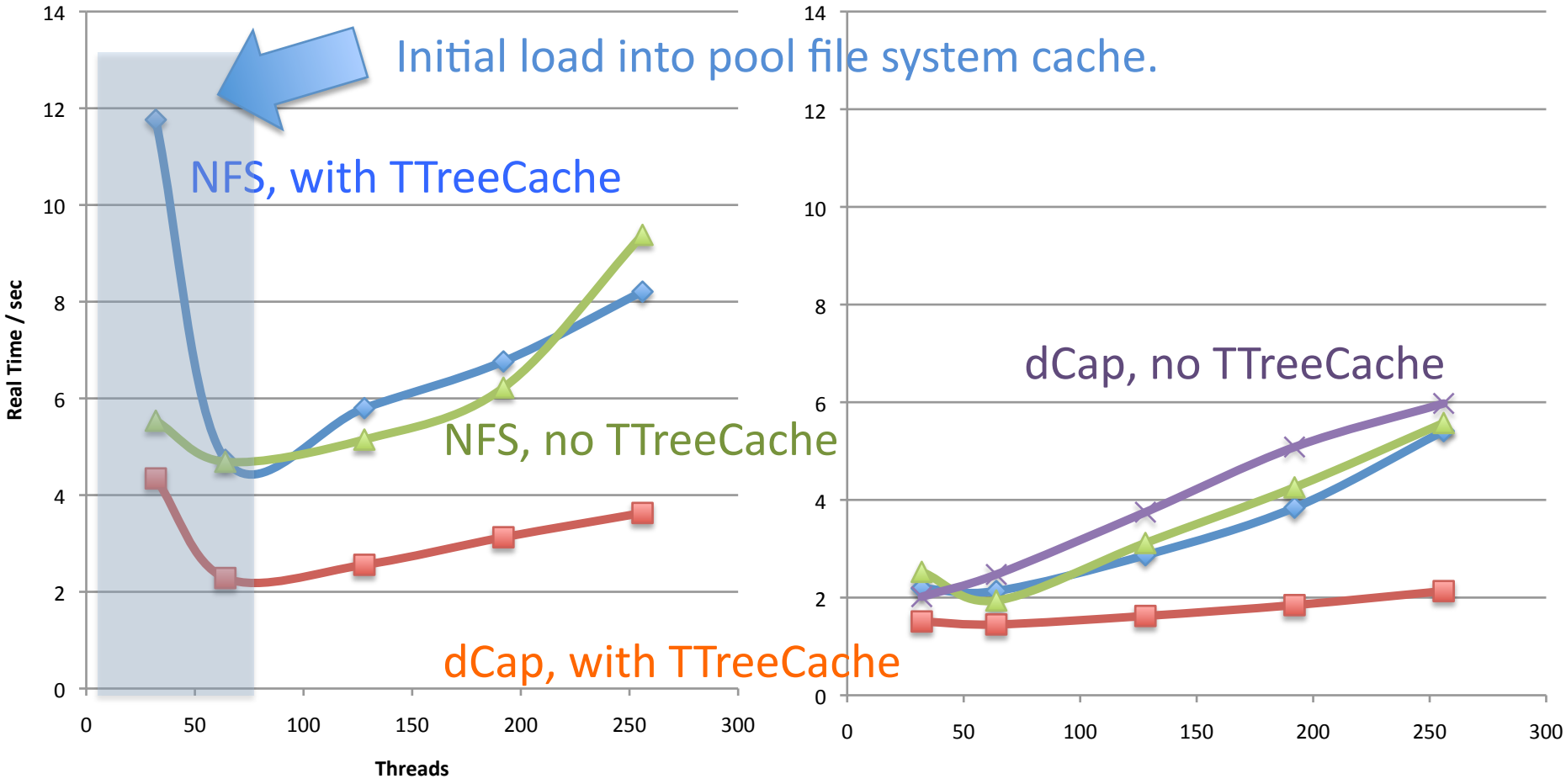
Initial load into pool file system cache.

ROOT : optimized versus non optimized files

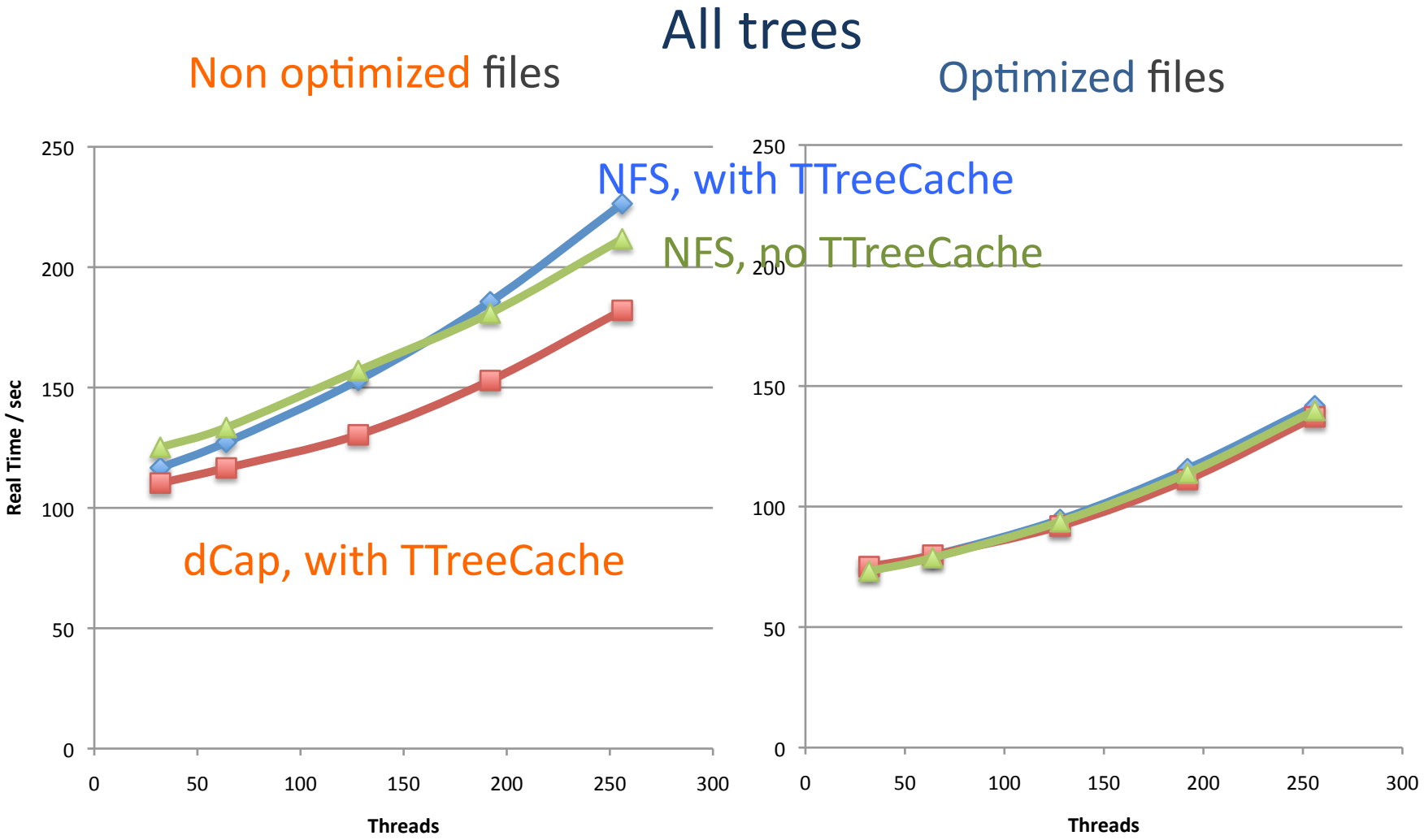
2 trees only

Non optimized files

Optimized files



ROOT : optimized versus non optimized files



On client side caching

The above evaluation doesn't at all use client side caching

But

- From evaluation (last hepix) we know that caching is 50 % of the game.
- This can be achieved by
  - TTreeCache for ROOT application
  - dCap ++ (see Patrick's talk at Lisboa Hepix) any application using dCap.
  - Or client file system cache for NFS 4.1 (pNFS)
- For ROOT application, the TTreeCache has a slight advantage, as it knows the structure of the ROOT files and can act accordingly

The vector read magic

The above evaluation demonstrates the advantages of Vector-Read by ROOT.

- Vector read can only be used through proprietary protocols (dCap,..)
- The file system semantics doesn't allow direct vector read. (bad)
- However, the is the famous 'fadvise' file system call :
  - Advised the file system (kernel) to prefetch certain portions of a file, if CPU time allows.
  - If those portions are read later, they are already available in the FS cache.
- Has been added to the 'file://' driver of ROOT and, according to Fons, improved access with 'file://' by up to 20%.
- Has been removed from the code again because it spoiled the TTreeCache I/O statistics. (very bad).

Fadvise in ROOT

Reply by Rene

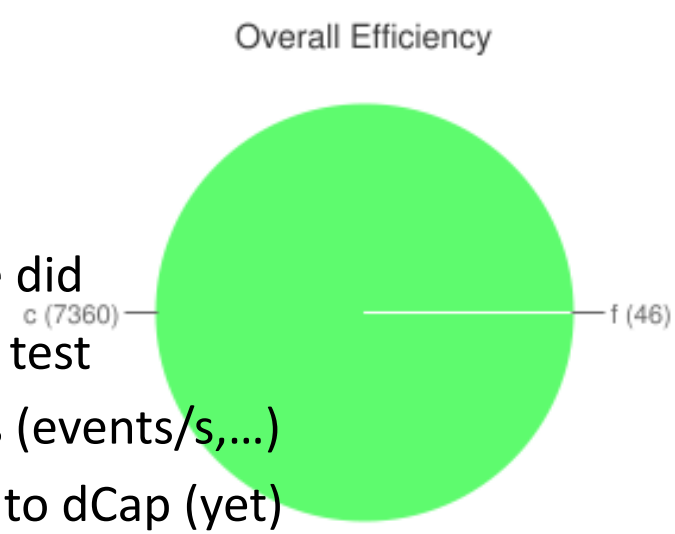
I did not disabled it because it was polluting the TTreePerfStats statistics (minor point), but mainly because in general (and in particular with optimized trees) it was introducing a performance penalty ranging from 1 to 20%.

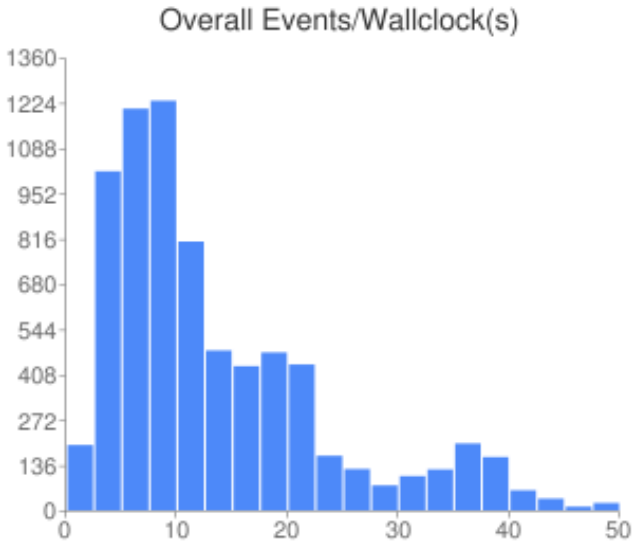
- Has been removed from the code again because it spoiled the TTreeCache I/O statistics. (very bad).

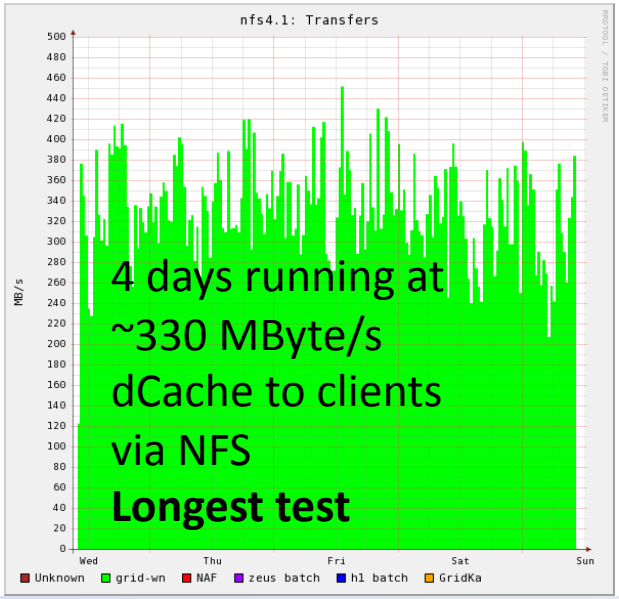
Hammer Cloud

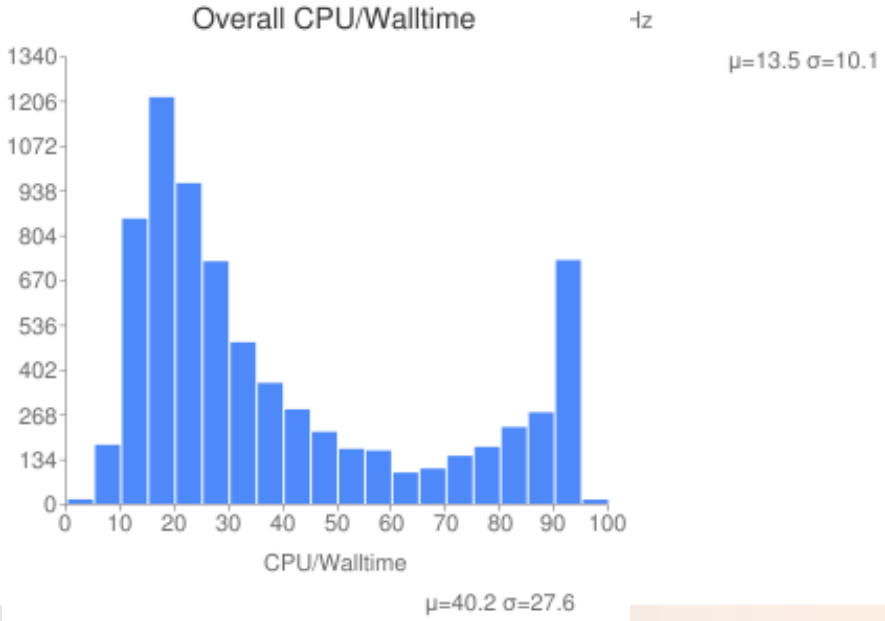
ATLAS Hammer Cloud tests

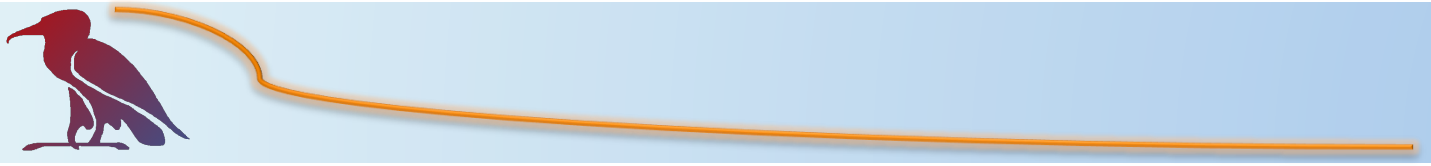
- 8248 jobs in total
- Cancelled after 4 days
- Longest single test we did
  - No trouble during test
- Reasonable outcomes (events/s,…)
- No comparison made to dCap (yet)











Client (kernel) availability

Kernel availability

- Kernel used for evaluation : 2.6.36_rc3
- NFS 4.1 (pNFS) kernels expected in SL6.(>2)
- 2.6.36 back-port to SL5 available from DESY
  - Plus ‘mount tools’ RPM.
  - Kernel will very likely not cover all hardware setups.
- With a Joined Effort (e.g. CERN, FNAL, DESY), we would be able to provide an SL5 with NFS 4.1 (pNFS) kernel within months. (If we really want)

Kernel availability

commit a4dd8dce14014665862ce7911b38cb2c69e366dd
Merge: b18cae4 411b5e0
Author: Linus Torvalds <torvalds@linux-foundation.org>
Date: Tue Oct 26 09:52:09 2010 -0700

Merge branch 'nfs-for-2.6.37' of
git://git.linux-nfs.org/projects/trondmy/nfs-2.6.git

* 'nfs-for-2.6.37' of git://git.linux-nfs.org/projects/trondmy/nfs-2.6:
net/sunrpc: Use static const char arrays
nfs4: fix channel attribute sanity-checks
NFSv4.1: Use more sensible names for 'initialize_mountpoint'
NFSv4.1: pnfs: filelayout: add driver's LAYOUTGET and GETDEVICEINFO infrastructure
NFSv4.1: pnfs: add LAYOUTGET and GETDEVICEINFO infrastructure
NFS: client needs to maintain list of inodes with active layouts
NFS: create and destroy inode's layout cache
NFSv4.1: pnfs: filelayout: introduce minimal file layout driver
NFSv4.1: pnfs: full mount/umount infrastructure
NFS: set layout driver
NFS: ask for layouttypes during v4 fsinfo call
NFS: change stateid to be a union
NFSv4.1: pnfsd, pnfs: protocol level pnfs constants
SUNRPC: define xdr_decode_opaque_fixed
NFSD: remove duplicate NFS4_STATEID_SIZE

First part of pNFS now in 2.6.37

Next Steps

- More details at CHEP'10 by Yves and Dmitri.
- More investigation with various different ROOT setups.
- Working with the CMS official test-case.
- Investigating X509 Certificate/Proxy security.
- Wide area transfer evaluation. (DPM, dCache, DESY, CERN)
- Setting up a regular NFS 4.1 (pNFS) system e.g. : NetApp and Pillar.
- Evaluation by the HEPIX working group.
- Trying to find groups as guinea-pigs for NFS4.1 production.

Conclusion

- Stability is much better than expected : Production ready.
- Kernel situation : short term solution for SL5 would be available, if we want.
- pNFS is partially already in 2.6.37
- Performance already comparable with existing solutions.
- Nevertheless : more evaluation on ROOT framework interaction needed. (vector read, fadvise)
- Efforts will continue within the EMI/dCache.org framework.
- You want to volunteer ?
  - Get dCache 1.9.10 from dCache.org
  - Get nfs enabled kernel : http://www.dcache.org/chimera/x86_64/

dCache birthday

Stolen from Dmitry Litvintsev OSG presentation.