First results of NFS v 4.1 (pNFS) in dCache

Johannes Elmsheuser, LMU Munich, Atlas
Patrick Fuhrmann
Yves Kemp
Tigran Mkrtchyan

With contributions by Andrei Maslennekov

And many thanks to Johannes E. for modifying the Hammer-cloud system to run as stand alone as possible.

Goal of this presentation :

To provide first prove that

NFS 4.1 (pNFS) (client and server)

is close to be used in production and already provides very competitive performance and stability results.

However :

All numbers presented here have been collected during the last 5 days. Although they all have been double-checked, the one or other mistake might have sneaked in.

Quick reminder of NFS 4.1 advantages

Compared to WLCG protocols (dCap/rfio/xroot)

- NFSv4.1 is an industry standard (will be supported by many vendors)
- NFSv4.1 clients are provided and maintained by other people
- Client caching is coming for free (regular file system cache)
  - ✓Caching algorithms are designed by file system experts.
  - ✓Security of files in cache is consistent.

Compared to previous NFS protocol versions

- pNFS makes use of highly distributed data (client redirect, layout)
- Compound RPC calls (multiple ops, one rpc call)
- Security gss api defined in spec and not added later. (Secure)

Using NFS 4.1 with dCache

dCache can be mounted on your WN as any other network file system and dCache data can be directly accessed through this protocol.

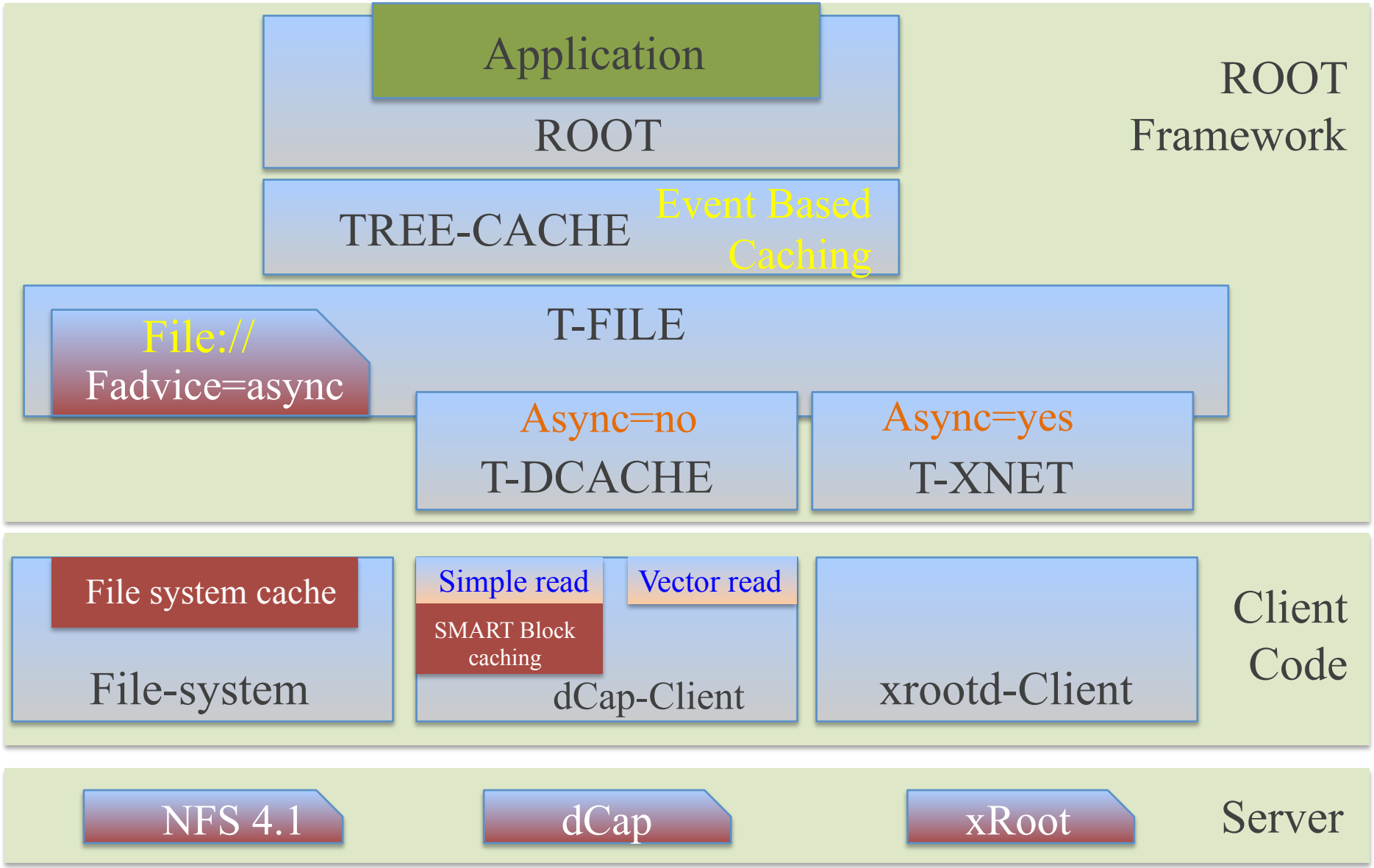
Status : dCache NFSv4.1 Server

- Nameserver & I/O implemented since 1.9.5
- Immutable files only (as for any other dCache I/O protocol)
- Security
  - Kerberos in code-trunk. (expected for 1.9.9)
  - X509 not yet clear but might be possible
  - Full NFS 4 ACL support
  - BUT : Still only through admin interface, ‘setacl’ will follow very soon.
- Automatic Tape restore disabled (to protect ‘tape system’ meltdown)

Status : NFS 4.1 (pNFS) Linux client

- NFS 4.1 and the linux kernel
  - NFS 4 already in SL5
  - NFS 4.1 in 2.6.32
  - NFS 4.1 plus pNFS in 2.6.33/34
- Kernel 2.6.34 will be in Fedora 13 and RH6 Enterprise (summer)
- NFS 4.1 (pNFS) Kernel available in Fedora 12 (NOW)
- Windows Client expected 4Q10.
- DESY grid-lab is testing with :
  - SL5 and 2.6.33 kernel plus some special RPM. (mount tools)
  - See our wiki for further information

ROOT Framework I/O structure



It is very difficult for us to know which optimization the ROOT frameworks tries to use for different Applications and different files.

Different tuning might be required for different access patterns.

Mechanisms to speed up wire protocols

Read Ahead

Read more (linear) data than requested by the application.
Of advantage only for linear reads in forward direction and if reasonable amounts of the file are read in.

Vector Read

A list of (offset,size) values are send to the server which returns the requested data in one chunk.
Very efficient but client has to wait till the entire 'chunk' arrived.

Asynchronous Vector Read

A list of (offset,size) values are send to the server which returns the requested data asynchronously, while the application can already read data as it arrives.
Very efficient.

Async Vector Read and Posix

There is no posix call allowing Vector Read. So it be used directly on the file system. However there is an fadvice posix call, which is implemented in Linux, which allows to ask the file system to read a list of (non continues) blocks in the background. Tigran added this call to the Tfile driver with the following result :

Quote Fons Rademakers :

implement TFile::ReadBufferAsync() using posix_fadvise() which tells the kernel which blocks we are going to read, so it can start loading these blocks in the buffer cache. Works only on Linux for the time being.
Minimum speedup about 15%-20%.

So, other than often claimed, the file system layer, and with that NFS4.1, can make use of the ROOT vector read optimization.

Library Block Caching (Smart Caching)

The client library holds a list of cached data blocks which are replaced by newly incoming blocks based on e.g. an LRU algorithm.
Similar to file system approach. Helps with jumping read pointers e.g. non optimized ROOT files.

File System caching

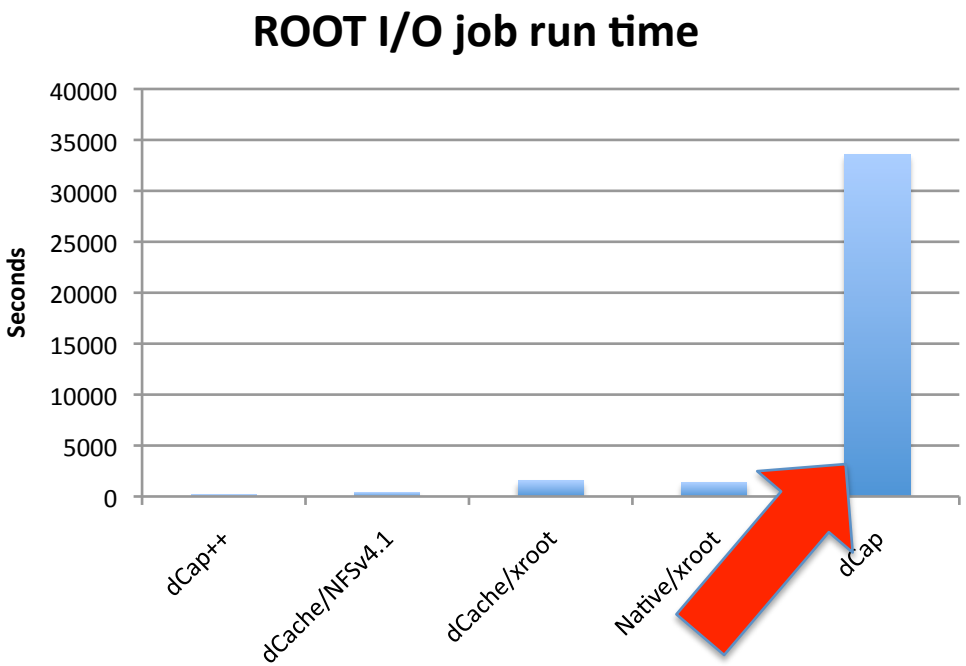
Partition wide, process wide, caching of often used blocks. OS providers have been putting big efforts in optimizing this part.
Very efficient.

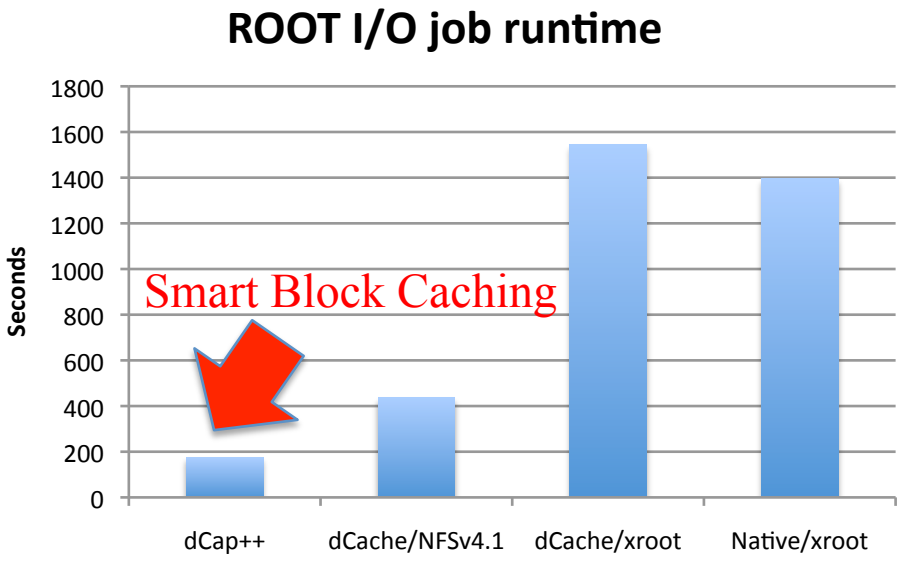
Mechanisms to speed up wire protocols

Mechanism	dCap/Client	NFS v 4.1/Client	Xroot/Client
Read Ahead	Yes	Small	Yes
Vector Read	Yes	No	Yes
Async Vector Read	No	Yes	Yes
Event Caching	ROOT	ROOT	ROOT
Block Caching	Only dCap++	Yes	Yes
File System Caching	No	Yes	No

(*)dCap++ is a dCap version modified by Günter Duckeck (Munich) providing smart block Caching.

First results under '*developers conditions*'





No optimization, no caching, no read ahead, no vector read

Access : reading every 100th event out of 52804 events from an non optimized Atlas event file

Not optimized 'atlas' file results in reading of small portions of the file in rather random fashion and a lots of jumping forth and back within the file.

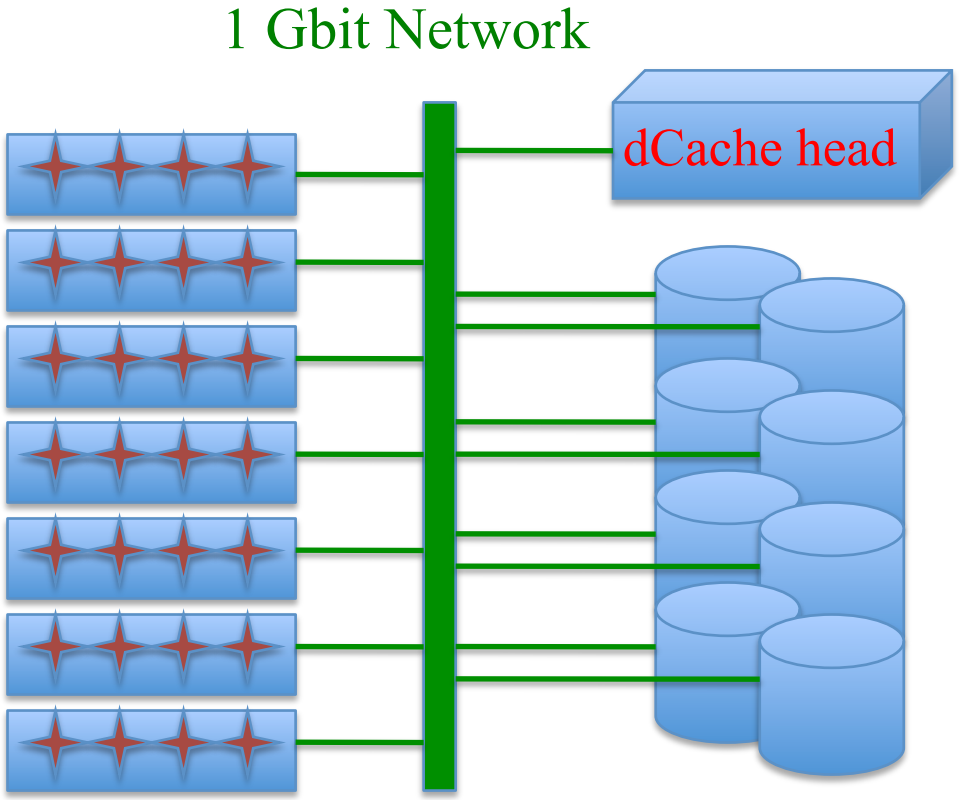
Stability/Stress tests

Numbers provided by Yves Kemp using *DESY grid-lab*

dCache NFS 4.1 setup (DESY grid-lab)

7 Workernodes
2 CPU * 2 Cores
each

SL5
2.6.33 kernel
NFS 4.1 (pNFS)



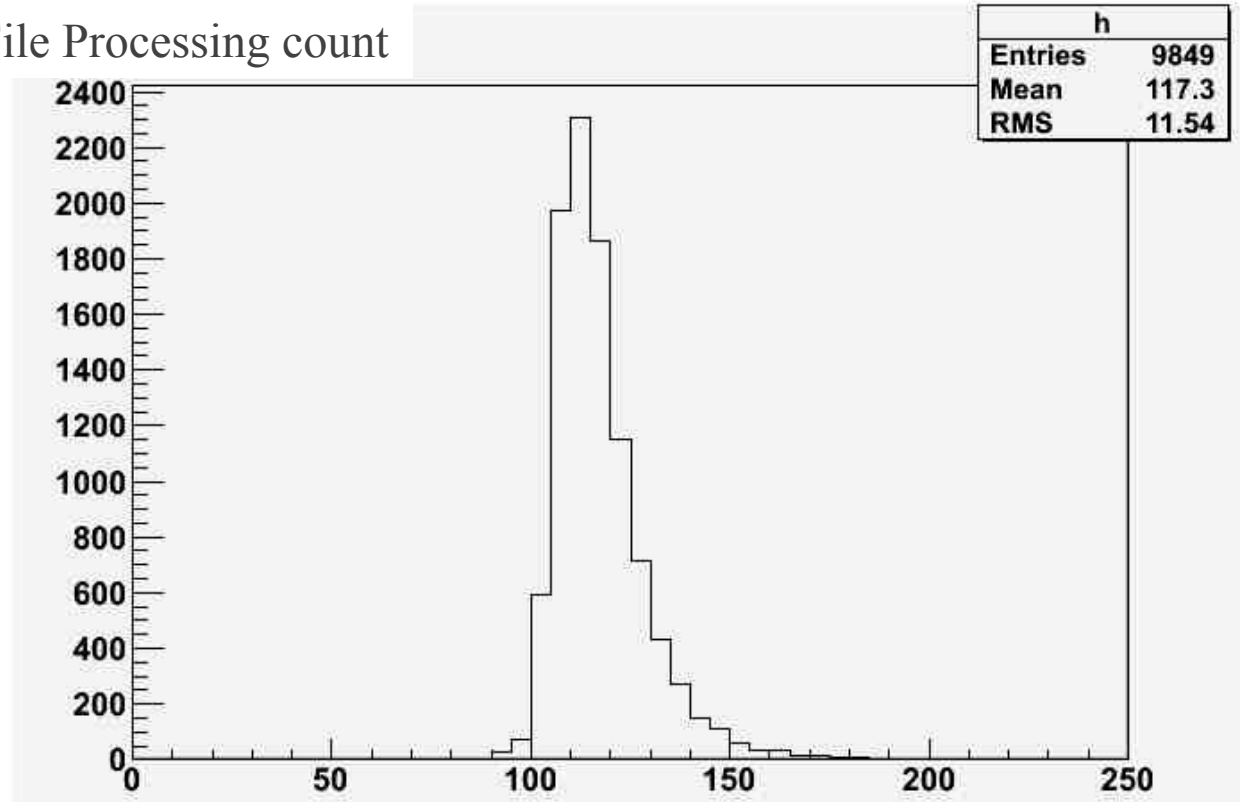
1 headnode
8 pool nodes

“Real” ROOT stress

To be as close as possible to Real analysis

- ✓Test lasted for about 12 hours
- ✓7 WN’s *4 Jobs accessing 9849 files on 8 pool nodes
- ✓11 Tbytes transferred with an average of 260 MB/sec
- ✓Compressed data to detect possible corruption
- ✓Disadvantage : High CPU to uncompress
- ✓No client kernel panic, no server error
- ✓No I/O error reported by ROOT

File Processing count



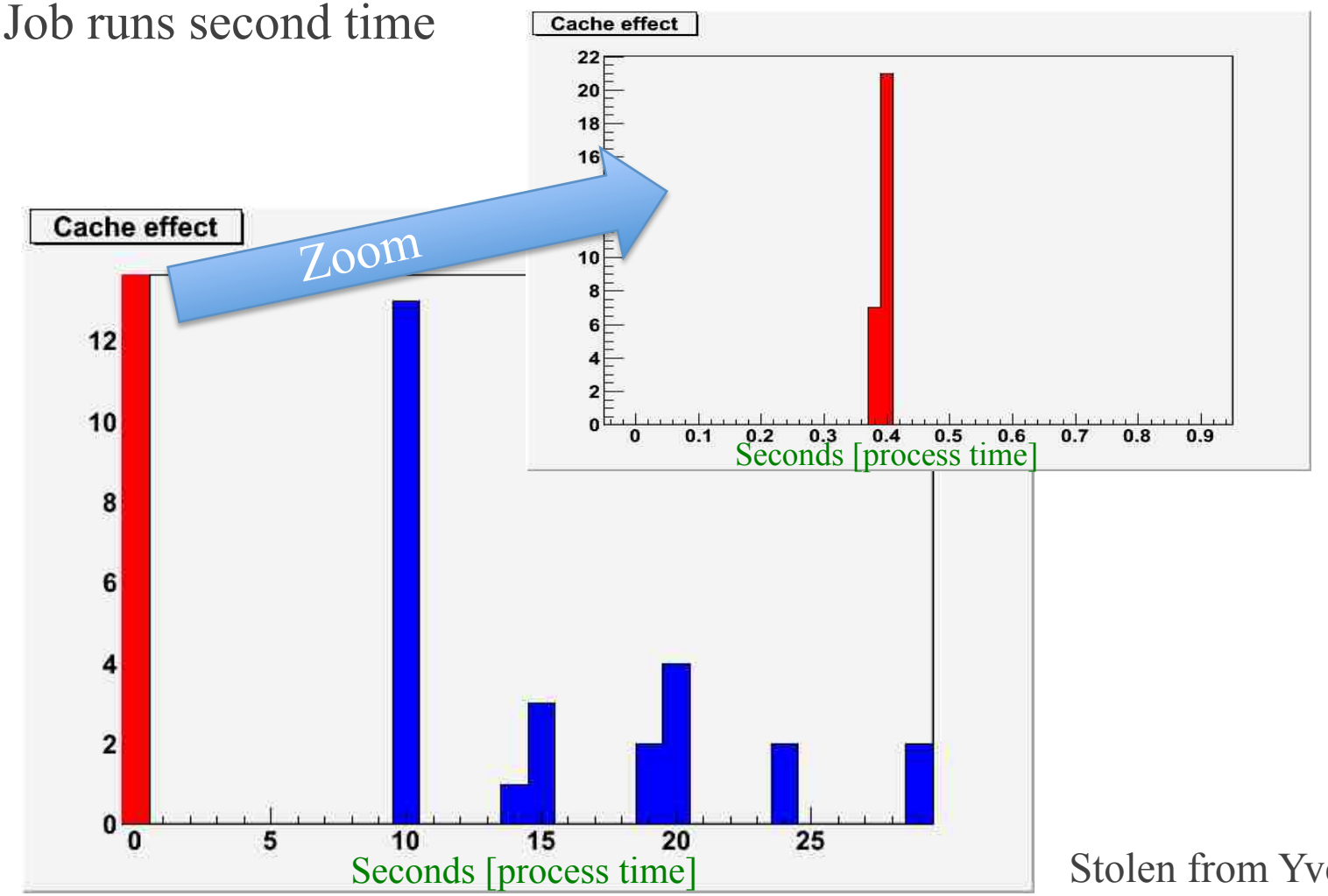
Seconds [file processing time]

Stolen from Yves Kemp

Demonstration of File System Cache Effect

X Job runs first time

X Job runs second time



Stolen from Yves Kemp

Hammer Cloud Results

Sorry, they have not been ready in time.

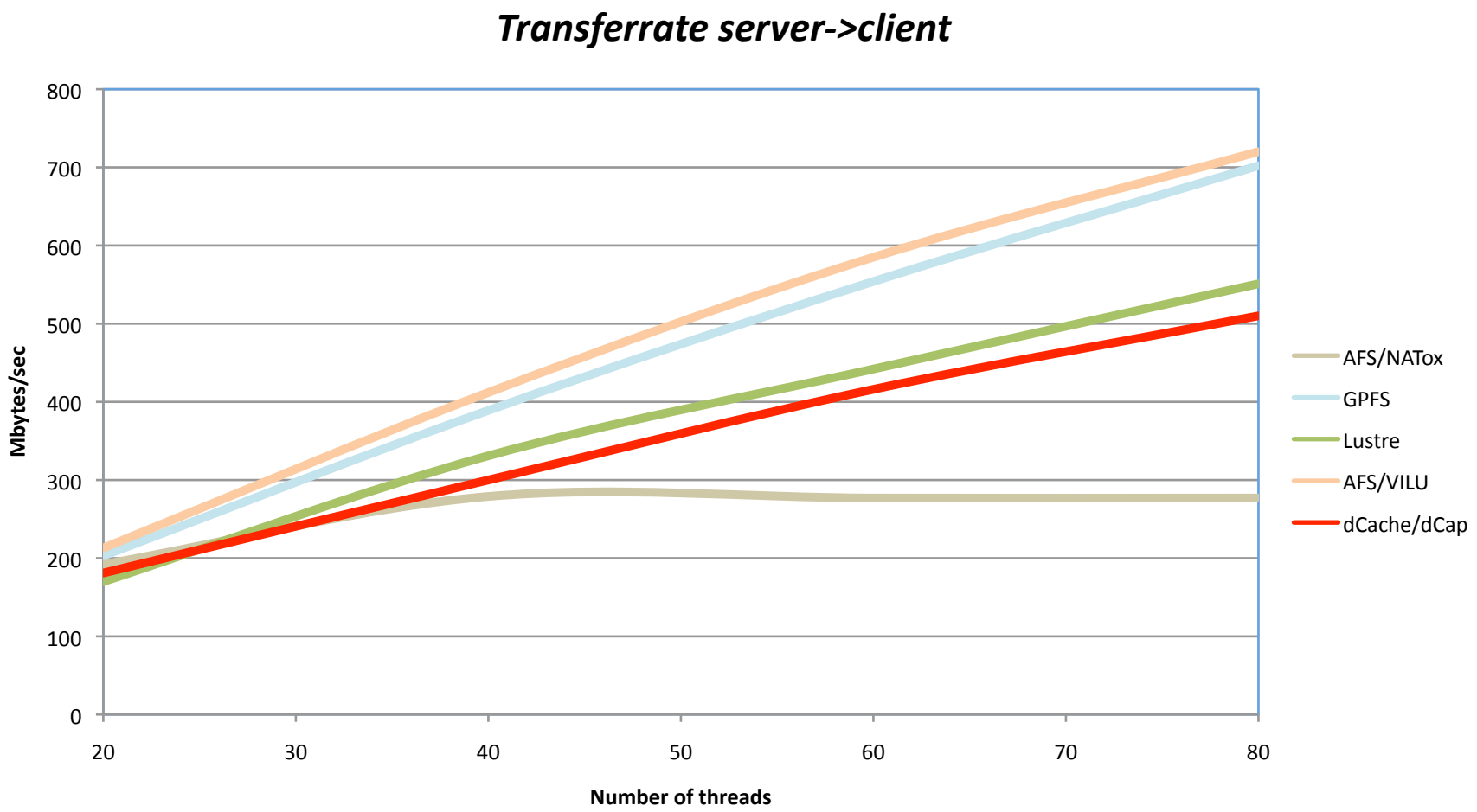
We will present those for CHEP or/and next HEPIX

Interpretation of the results from Hepix Storage working group evaluation.

- Running CMS (black-box) job for 14 minutes
- Measuring *total amount of data* transferred and
- *Number of event* processed
- 10 client machines running 20,40,60 and 80 jobs/threads
- Andrei's test give a very good impression on the advantages of file system caching.

Stolen from Andrei Maslennikov

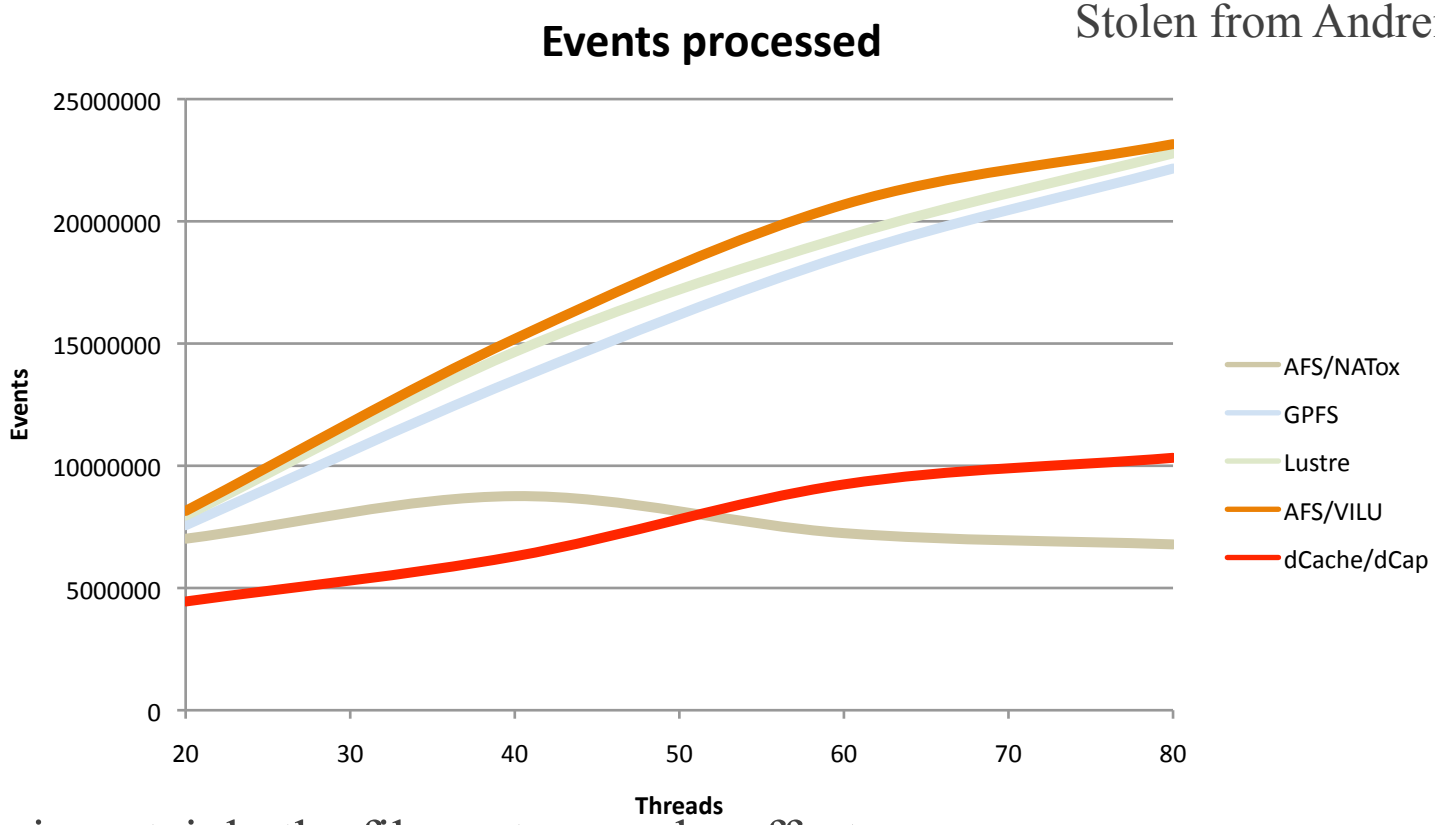
Data Rate between server and client



Stolen from Andrei Maslennikov

Events process within 14 Minutes

Stolen from Andrei Maslennikov



This is certainly the file system cache effect, as

- See Tigran's and Yves results from above.
- The bytes received from the various servers are nearly identical.
- dCap is the only protocols (in this set) which can't make use of the fs cache.
- We would expect dCache/NFS4.1 do behave as GPFS/Lustre …

Summary on NFS 4.1 (pNFS) in dCache

- pNFS is now an agreed industry standard
- It comes with the advantages of a mountable file system (as e.g. GPFS, Lustre) plus the benefits of a standard.
- Clients are available or will be available soon
- dCache provides an NFS 4.1 server which we believe is
  - Reliable and
  - Provides competitive performance
- Although special tuning of an I/O system might give better results than generic once, the vast possible ways, ROOT applications do already access the underlying I/O system will spoil all attempts to find specific solutions.

Not to forget :

Even more standards in dCache

WebDav

Status : WebDav (s)

- Available since 1.9.6 (3)
- Tested with
  - Mac OSX
    - Mac OSX doesn't support WebDav with Certificates (needs stunnel)
    - Writing only possible with 1.9.8 (expected)
  - Windows(XP) OK
  - SuSE11.2 (Gnome, KDE) OK
- Write via 'redirect' or if not supported by client via 'proxy'.
- Security
  - Plain or x509 (user, password in preparation)
  - On redirect, only control line is encrypted.

Further Reading

www.dCache.org