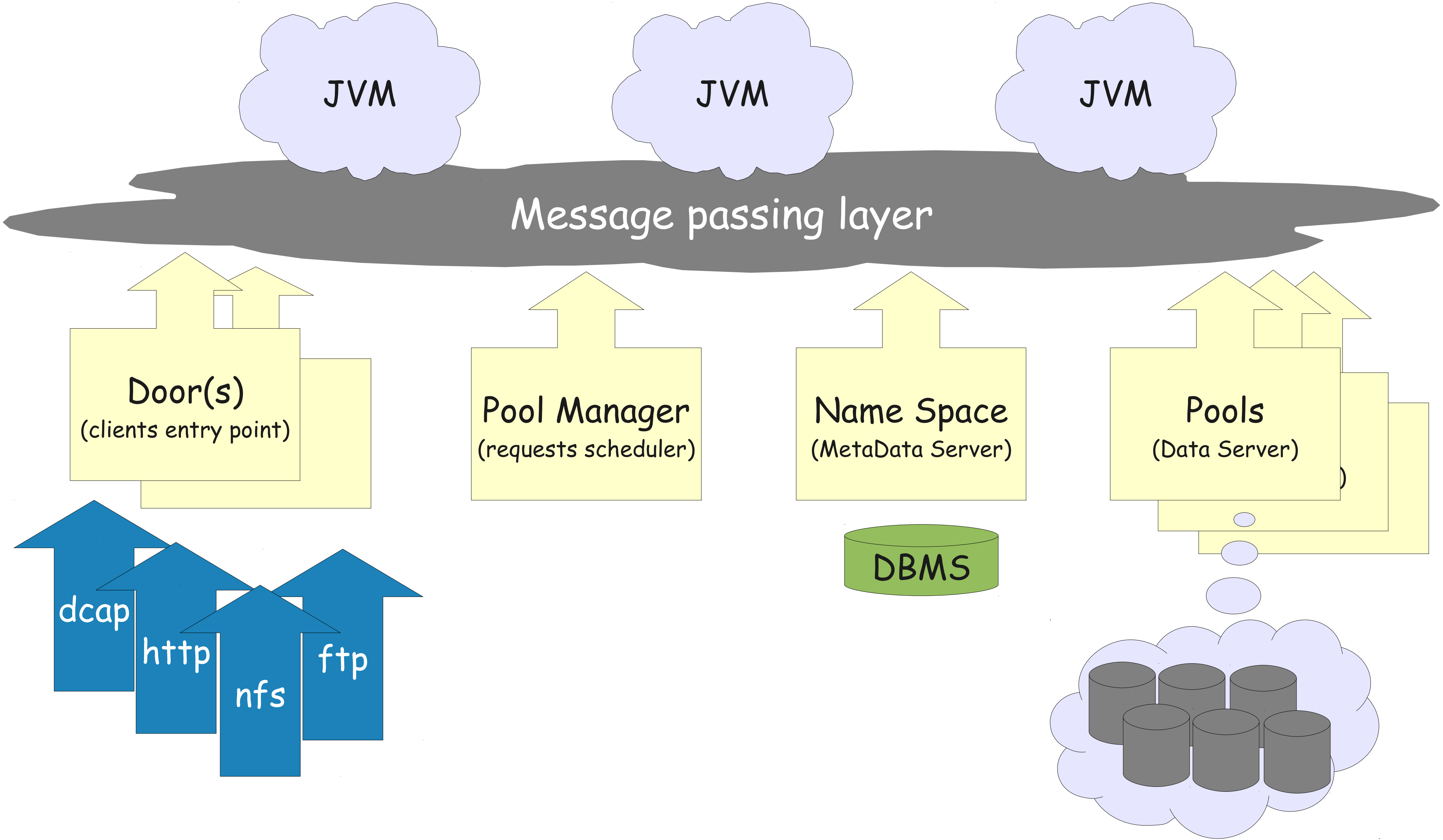
pNFS Capabilities in the Real World

or

pNFS under fire

Tigran Mkrtchyan for dCache team.

dCache in one slide



JVM

JVM

JVM

Message passing layer

Door(s)
(clients entry point)

Pool Manager
(requests scheduler)

Name Space
(MetaData Server)

Pools
(Data Server)

DBMS

dcap

http

nfs

ftp

dCache + pNFS

- No file striping
  - but this will change for reads in near future
- No LOCKs – all files are immutable
- LAYOUTRECALL on close (if client is greedy)
  - but we work on changing this
- WE DO NOT EXPORT EXISTING FILE SYSTEMS
  - we are the file system

Real life application

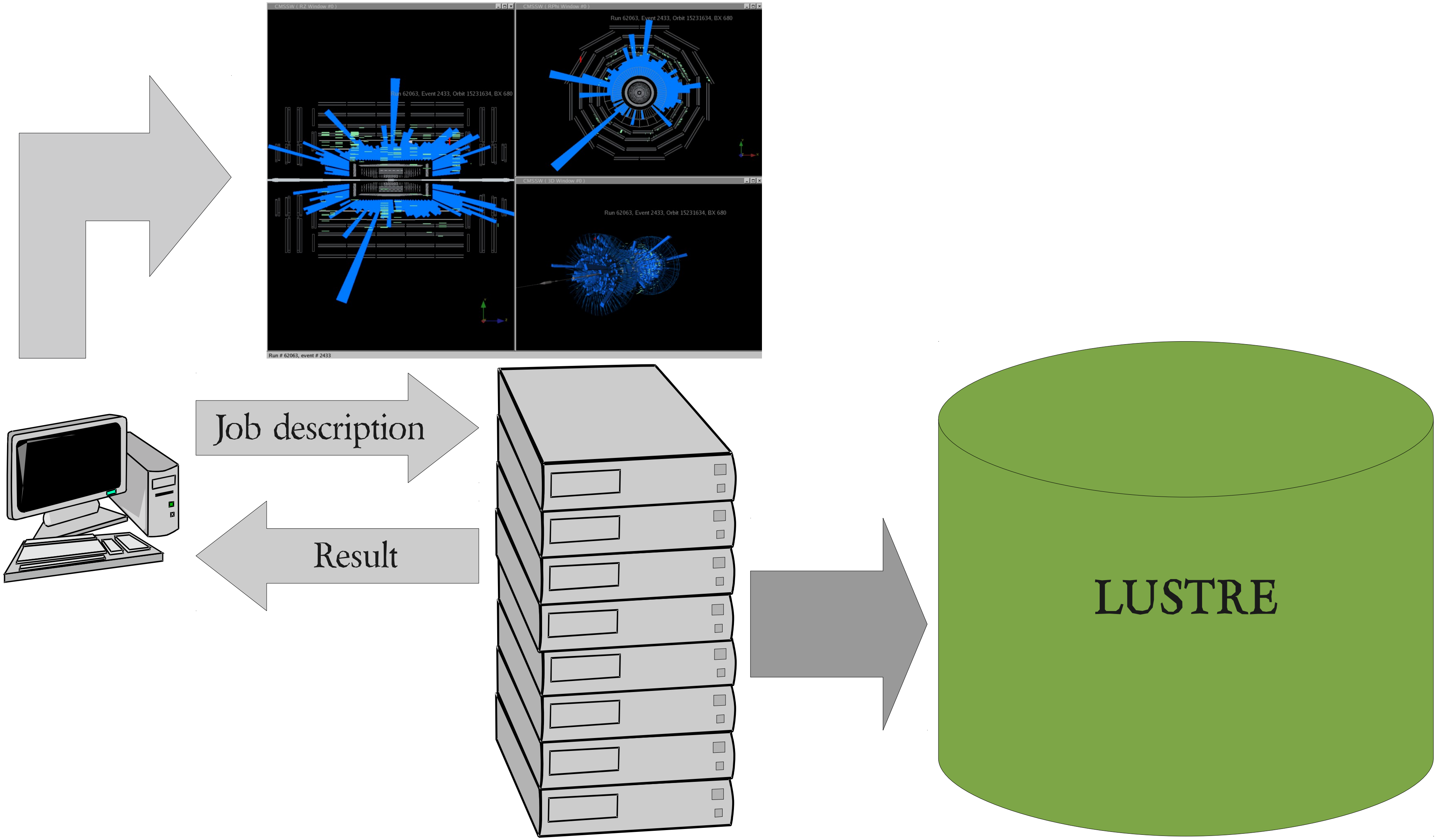
To convince end-users to use pNFS based system cthon tests are not enough.

Real parallel analysis job used at Karlsruhe Grid School in September 2009 was taken.

The Job

- 9238 files and a total of ~270 GB
- Each physical node has three workers
  - A total of 24 workers on eight physical machines
- Workers read data, analyze it and put results to the master
- Master plots the results
  - Typical HEP analysis workflow

The Job



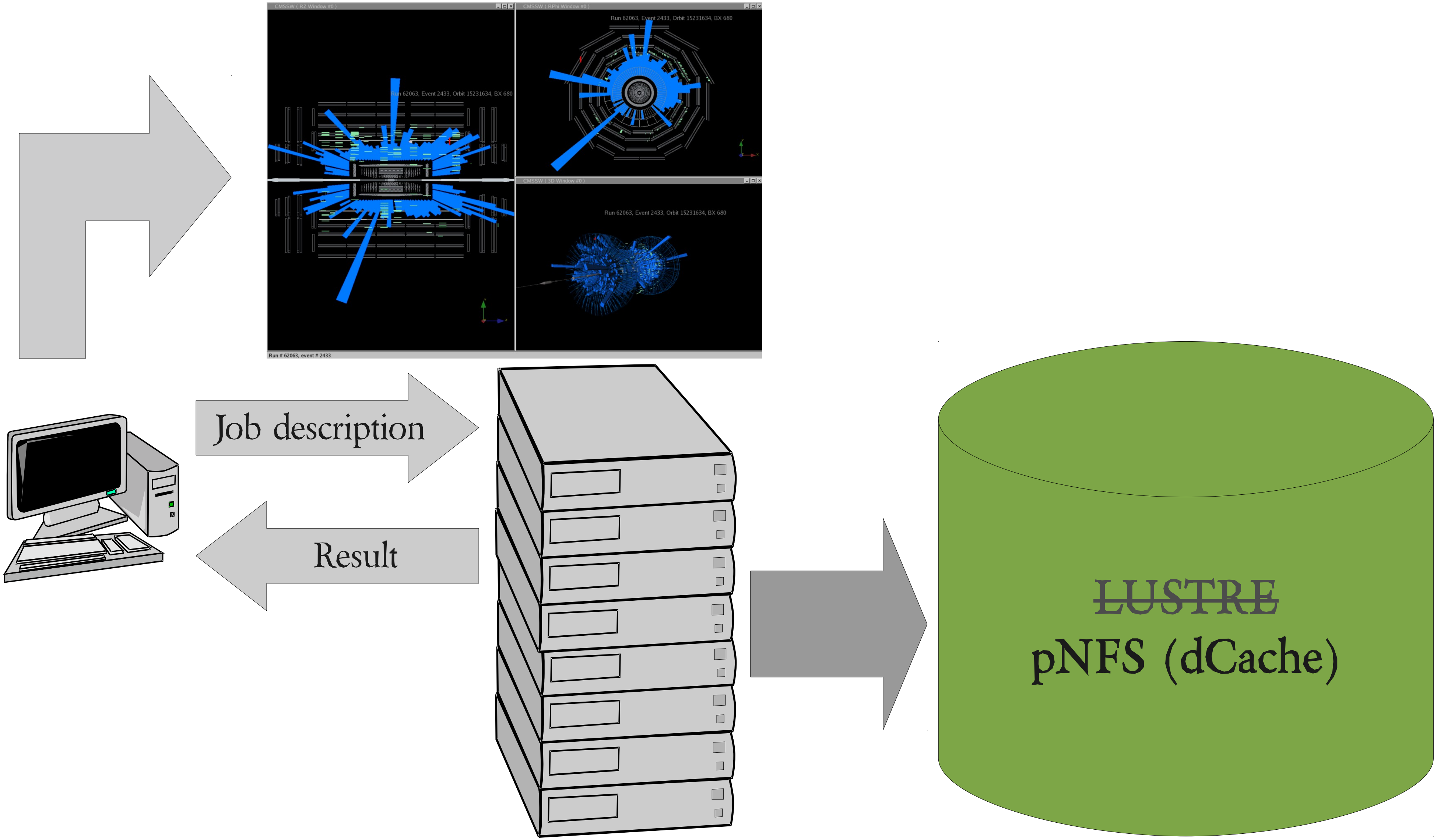
Job description

Result

LUSTRE

CMSSW (RZ Window #0)
CMSSW (RPhi Window #0)
CMSSW (3D Window #0)
Run 62063, Event 2433, Orbit 15231634, BX 680
Run # 62063, event # 2433

The Job



Run 62063, Event 2433, Orbit 15231634, BX 680
Run # 62063, event # 2433

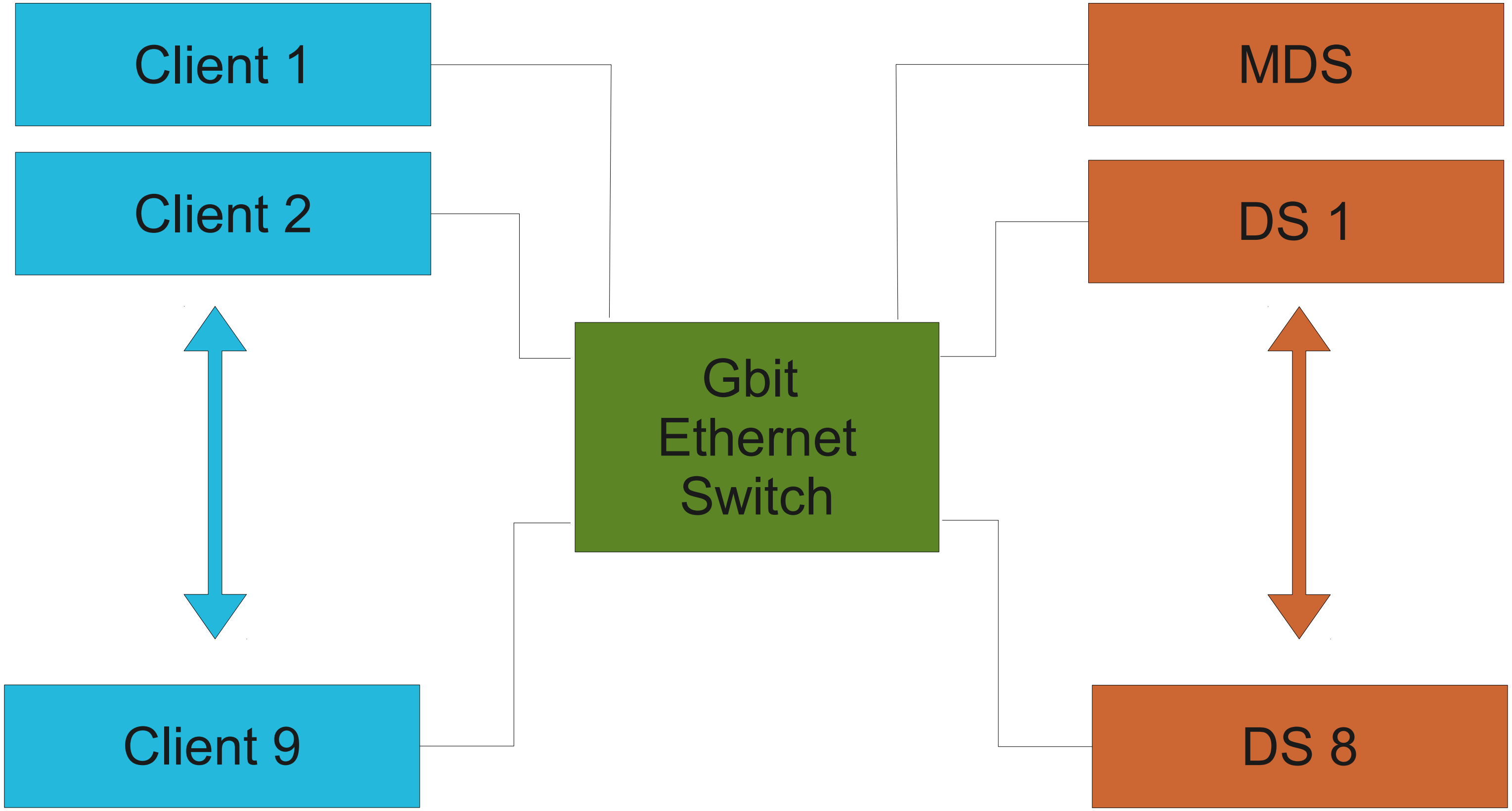
Job description

Result

~~LUSTRE~~

pNFS (dCache)

Environment



Test Environment

18 hosts: Xeon 2x2 Core 3GHz , 8GB Ram, 70 GB local disk, RHEL5.3 x86_64

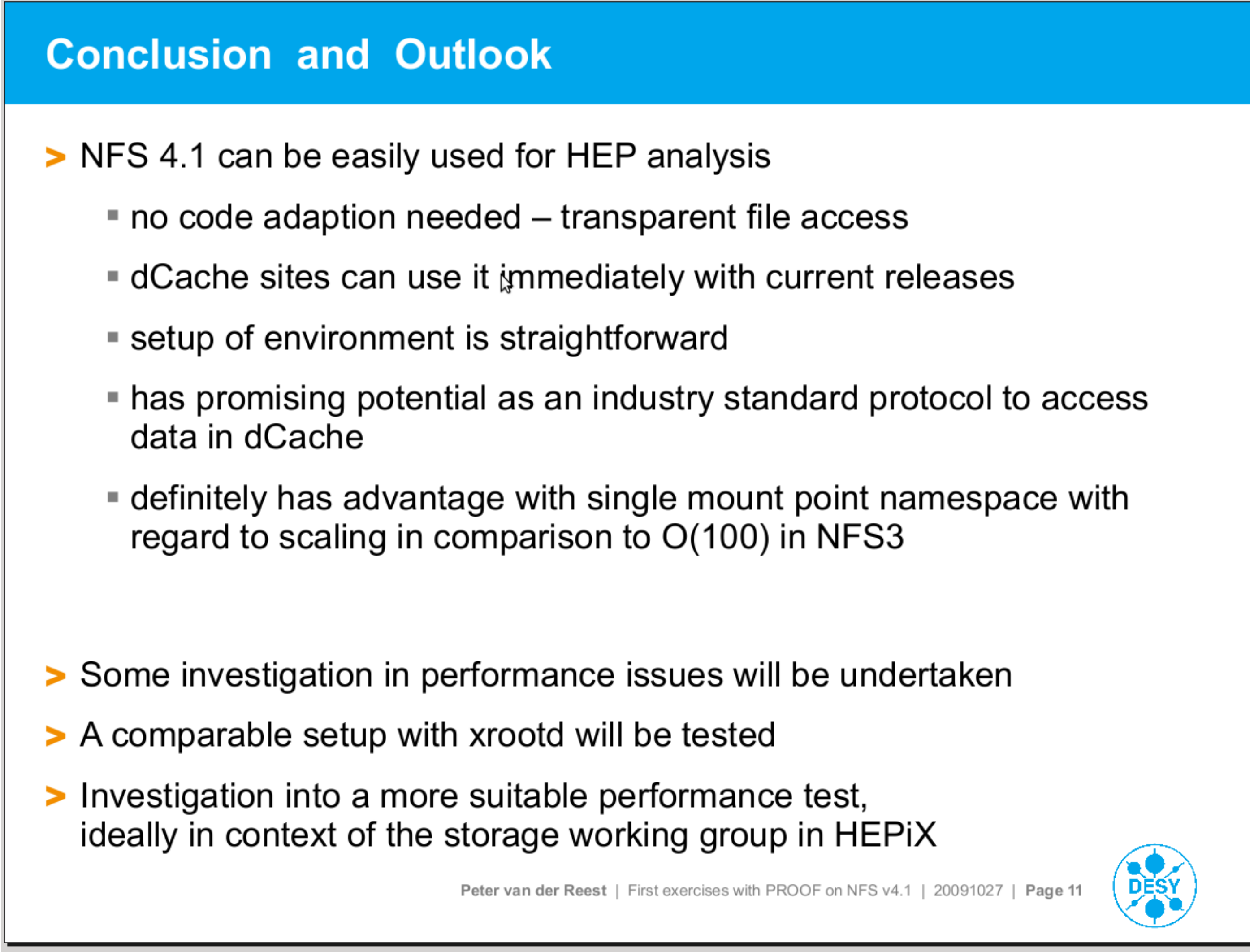
Servers : dcache-1.9.6, kernel-2.6.18

- MDS + 8xDS

Clients:

- 9x pnfs enabled kernel-2.6.32

First Results



Conclusion and Outlook

- NFS 4.1 can be easily used for HEP analysis
  - no code adaption needed – transparent file access
  - dCache sites can use it immediately with current releases
  - setup of environment is straightforward
  - has promising potential as an industry standard protocol to access data in dCache
  - definitely has advantage with single mount point namespace with regard to scaling in comparison to O(100) in NFS3
- Some investigation in performance issues will be undertaken
- A comparable setup with xrootd will be tested
- Investigation into a more suitable performance test, ideally in context of the storage working group in HEPiX

Peter van der Reest | First exercises with PROOF on NFS v4.1 | 20091027 | **Page 11**

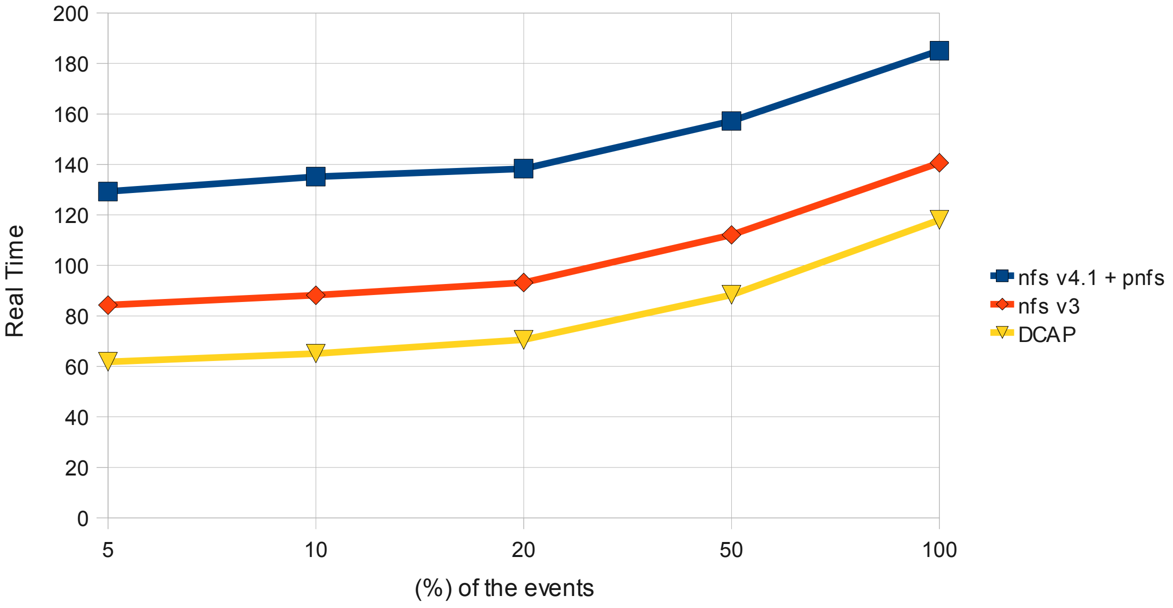
Reported at HEPiX in fall 2009

Questions?

Are we ready?

Some Numbers

(%)	MB	Real Time v4	Real time v3	Real Time DCAP	V4 reads	V3 reads	CPU Time
5	101.41	**129.3**	**84.26**	**61.81**	49565	46485	33.13
10	202.83	**135.16**	**88.18**	**65.08**	49565	46485	36.19
20	405.6	**138.3**	**93.2**	**70.57**	49565	46485	42.12
50	1014.09	**157.21**	**112.11**	**88.34**	49565	46485	59.91
100	2028	**185.11**	**140.67**	**118.06**	49565	46485	89.69



While the test turned out to be irrelevant it have shown that with nfsv3 client send less read requests than with v4.1.

Typical small site

- ~5000 CPUs + ~2000 desktops
- 1040 'data servers' on ~240 hosts (~3000 physical disks)
- ~1.5 PB data on disks
- ~3 PB on tape
- ~14*10^6 files
- different storage systems for different purposes
  - dCache – long living immutable data
  - Home directory (AFS) – mutable data with backup
  - Scratch space (NFS)– short living mutable data

TODO for Mike Eisler

add QoStorage into next spec

What to test

Typical site setup highlighted following use cases:

- Multiple clients

  as a state full protocol NFSv4.1 puts extra load on MDS

- Multiple servers

  client nodes usually mounts multiple storage systems at the same time

- DS crash

  daily disk and data server crashes are a 'normal' for big data centers

- MDS crash

Case #1: Multiple clients

- Mount a single MDS with multiple client a the same time. To test NFSv4.1 sessions and server stability.

- We tested only with 9 clients (0.4% of expected clients).

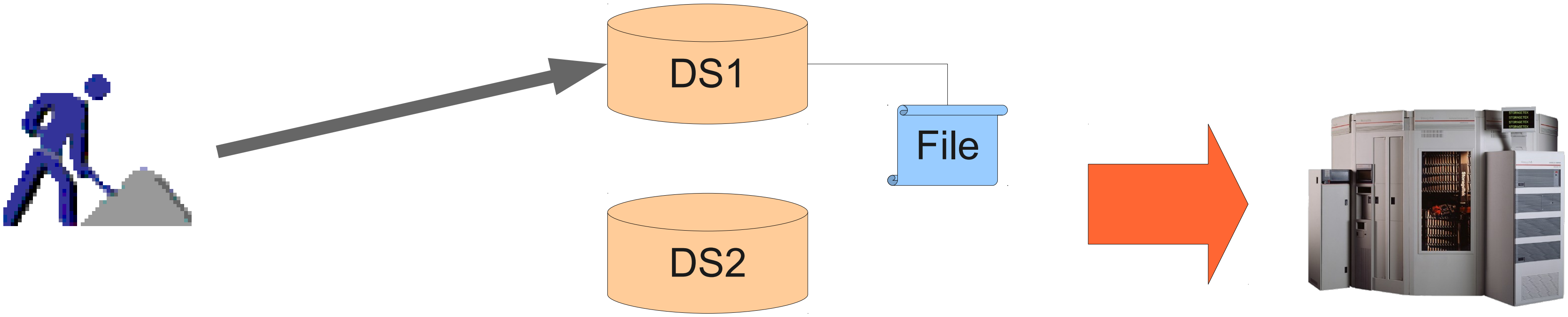
Case #2: Multiple servers

- Mount multiple NFSv4.1 servers on a single client a the same time. Check transfers between them.

- Done a minimal testing with two servers

Case #3: DS Fail over

- Re-request layout on DS shutdown/failure

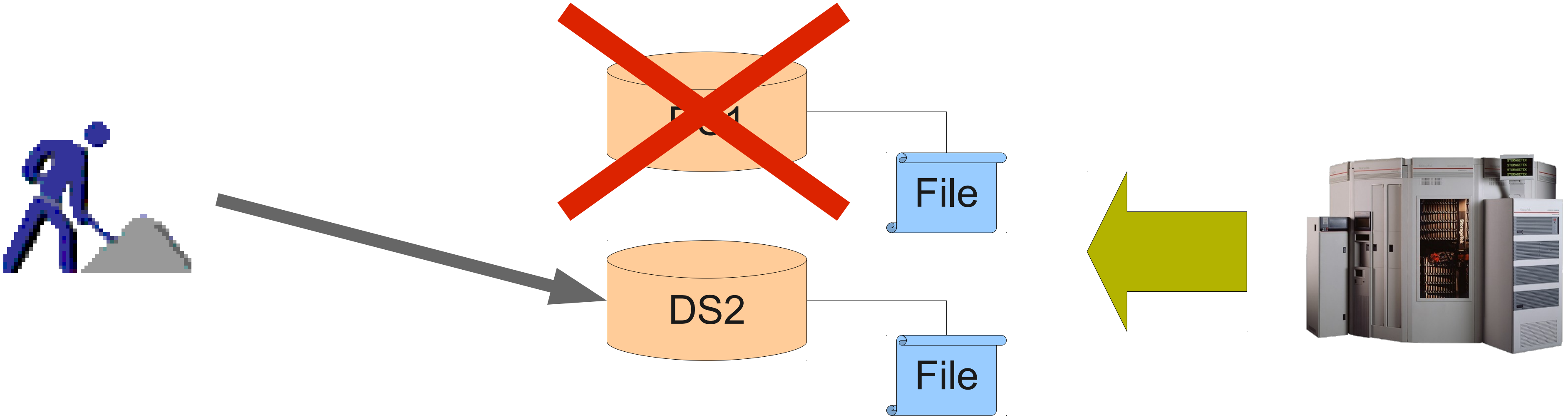


Tigran Mkrtchyan

Case #3: DS Fail over

- Re-request layout on DS shutdown/failure

- Failed



Case #3: DS Fail over

- Re-request layout on DS shutdown/failure
- Failed

This is why I am here!
e
DS2
File

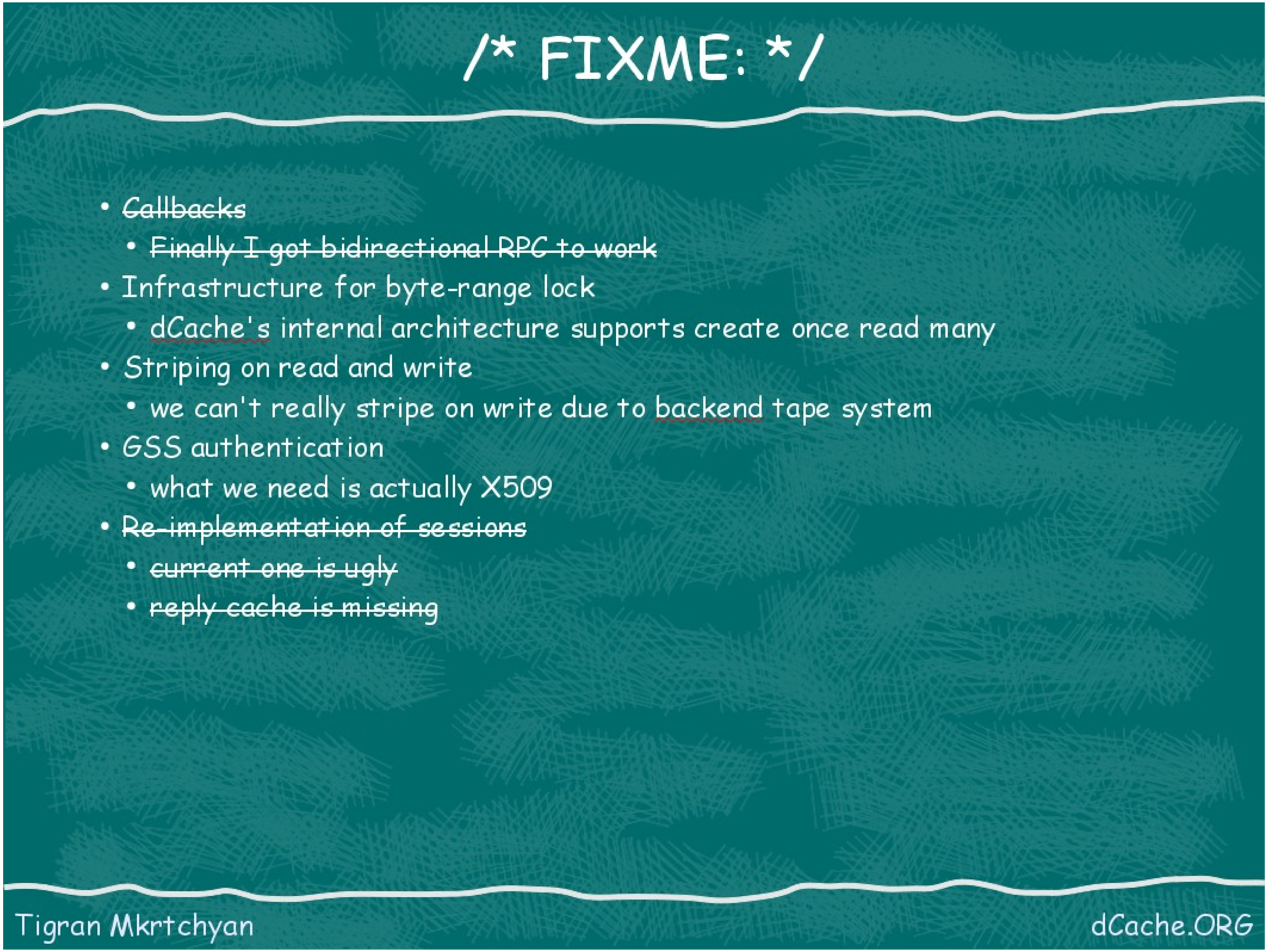
Of course proxy-IO is an option, but .....

Case #4: MDS restart

- Client have to re-establish connection to server. IO operations have to proceed

- Mostly fine, but have to be more robust.

diff -r cthon09



/* FIXME: */

- ~~Callbacks~~
  - ~~Finally I got bidirectional RPC to work~~
- Infrastructure for byte-range lock
  - dCache's internal architecture supports create once read many
- Striping on read and write
  - we can't really stripe on write due to backend tape system
- GSS authentication
  - what we need is actually X509
- ~~Re-implementation of sessions~~
  - ~~current one is ugly~~
  - ~~reply cache is missing~~

Tigran Mkrtchyan

dCache.ORG

Conclusions

Of course every one have his own idea on when pNFS is ready for production. But it obvious that end-users as well as data-centers operating teams have there own description for 'readiness'.

To meet there needs we have to provide an access to pNFS based installations. This will allow us, as a developers, get an early feedback and them to get a feeling what pNFS can do for them.