STUDIO E TEST DI FILE-SYSTEM DISTRIBUITI PER HEP

Giacinto Donvito
INFN-BARI

OUTLOOK

- New trends on open source storage software
- Overview on Lustre
  - Lustre architecture and features
  - Some Lustre examples
  - Tier3 on Lustre
  - Future developments
- Hadoop: concepts and architecture
  - Feature of HDFS
  - Few HDFS examples
  - Tier3 on HDFS
- Hepix FS-WG and CMS specific test and results
- Conclusions

TRENDS ON STORAGE SOFTWARE

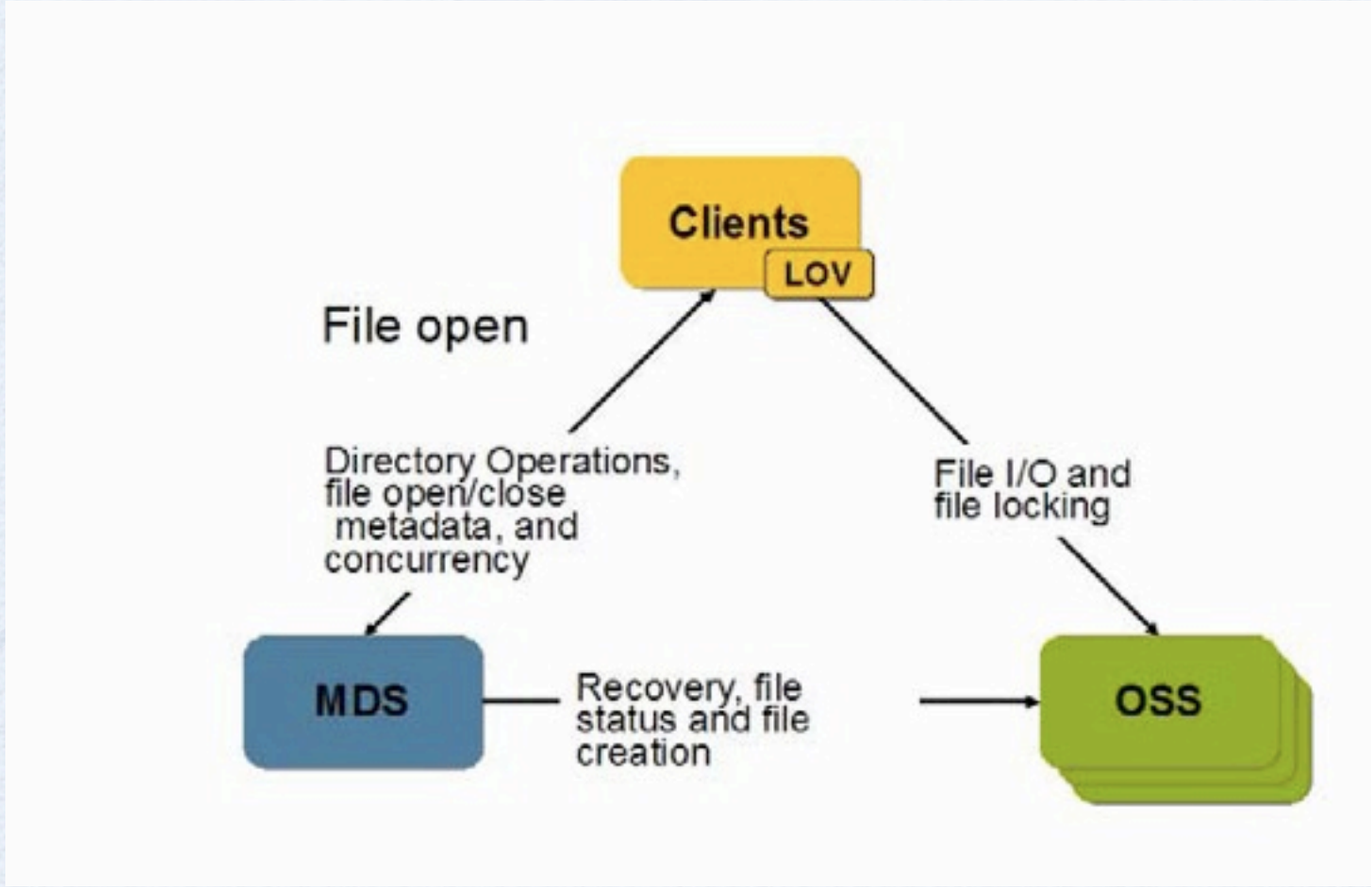
Requirements:

- CPUs are always much more eager of data, and the performance of disks are not growing as much as CPUs
- Very often the users requires native posix file system
  - FUSE helps a lot in providing a layer that could be used to implement “something like” posix filesystem
- Scalability is the main issues: what is working with 10 CPUs surely may experience problems with 1000 CPUs
- ... physics analysis is a particular use case

LUSTRE

TYPICAL LUSTRE INFRASTRUCTURE

- Lustre file-system is a typical parallel file-system in which all the client are able to use standard posix call to access files
- The architecture is designed in order to have 3 different function that can be spitted among different host or joined in the same machine:
  - MDS: this service hosts the metadata information about each file and its location
    - There could be basically one active MDS per file-system
  - OSS: is the service that hosts the data
    - There could be up to 1000 OSS
  - Clients: are hosts that are able to read lustre file-system
    - There could be up to 20000 clients in a cluster



LUSTRE 1.8.3

- All administrative operations can be done using few command line utilities and the "/proc/" file-system
  - The interface is very "admin-friendly"
- It is quite easy to put an OST in read-only
- It is possible to make snapshots and backups using standard linux tool and features like LVM and rsync
- It is possible to define easily how many stripes should be used to write each file and how big they will be (this could be configured at a file or directory level)
- Using SAN it is possible to serve the same OST with two servers and enable the automatic fail-over
- Very fast metadata handling
- In case of an OST failure only files (fully or partially) contained in that partition becomes unavailable
  - it is still possible to read partially the file in case it is split on few devices

LUSTRE 1.8.3

- It is possible to have a “live copy” of each device (for example using DRDB and heartbeat)
  - it is feasible for both data and metadata
- The client caches both data and metadata in kernel space
- (temporarily) failure of a server are not disruptive in case of repetitive operation
- The cache buffer on the client is shared: this is an advanced if several processes read the same file
  - the size of this buffer could be tuned (by */proc/* file-system)
- It is easy (and scriptable) to understand which OST hosts each file
- The performance obtained by the application does not depend on the version of the library used (this could help when old experiment framework is still used)
- It is possible to tune the algorithm used in order to distribute the files among the OSTs, giving more or less importance to the space available on each OST itself

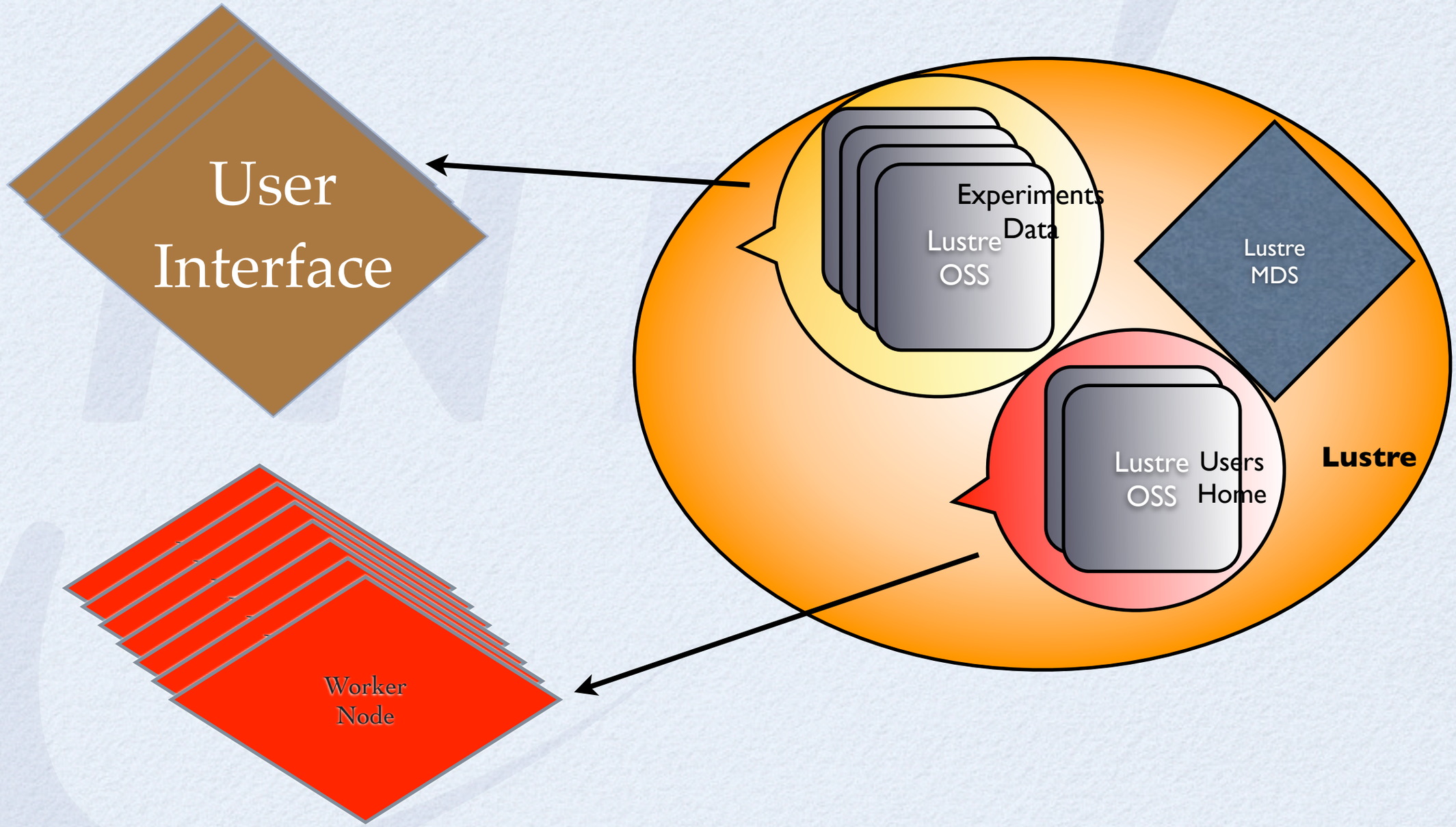
LUSTRE 1.8.3

- Using ext4 backend, it is possible to use 16TB OST.
- INFINIBAND supported as network connection
- Standard Posix ACLs are supported: it is possible to use standard unix tool to manage them
  - The ACLs should be enabled “system-wide” (on or off for the whole cluster)
- On the OSS, ~~it is mandatory to recompile the kernel or~~ it is possible to use (RedHat) kernels provided from the official web-site
  - On the client it is not strictly required
  - The "Patchless" client could work basically on every distribution
    - Not all the kernel release are fully supported (2.6.16> kernel <= 2.6.30)
  - http://wiki.lustre.org/index.php/Lustre_Release_Information#Lustre_Support_Matrix

LUSTRE 1.8.3

- **OSS Read Cache:**
  - It is now possible to cache read-only data on an OSS
  - It uses a regular Linux “pagecache” to store the data
  - OSS read cache improves Lustre performance when several clients access the same data set
- **OST Pools**
  - The OST pools feature allows the administrator to name a group of OSTs for file striping purposes
  - an OST pool could be associated to a specific directory or file and automatically will be inherited by the files/directory created inside it
- **Adaptive Timeouts:**
  - Automatically adjusts RPC timeouts as network conditions and server load changes.
  - Reduces server recovery time, RPC timeouts, and disconnect/reconnect cycles.

LUSTRE 1.8.X -- EXAMPLE

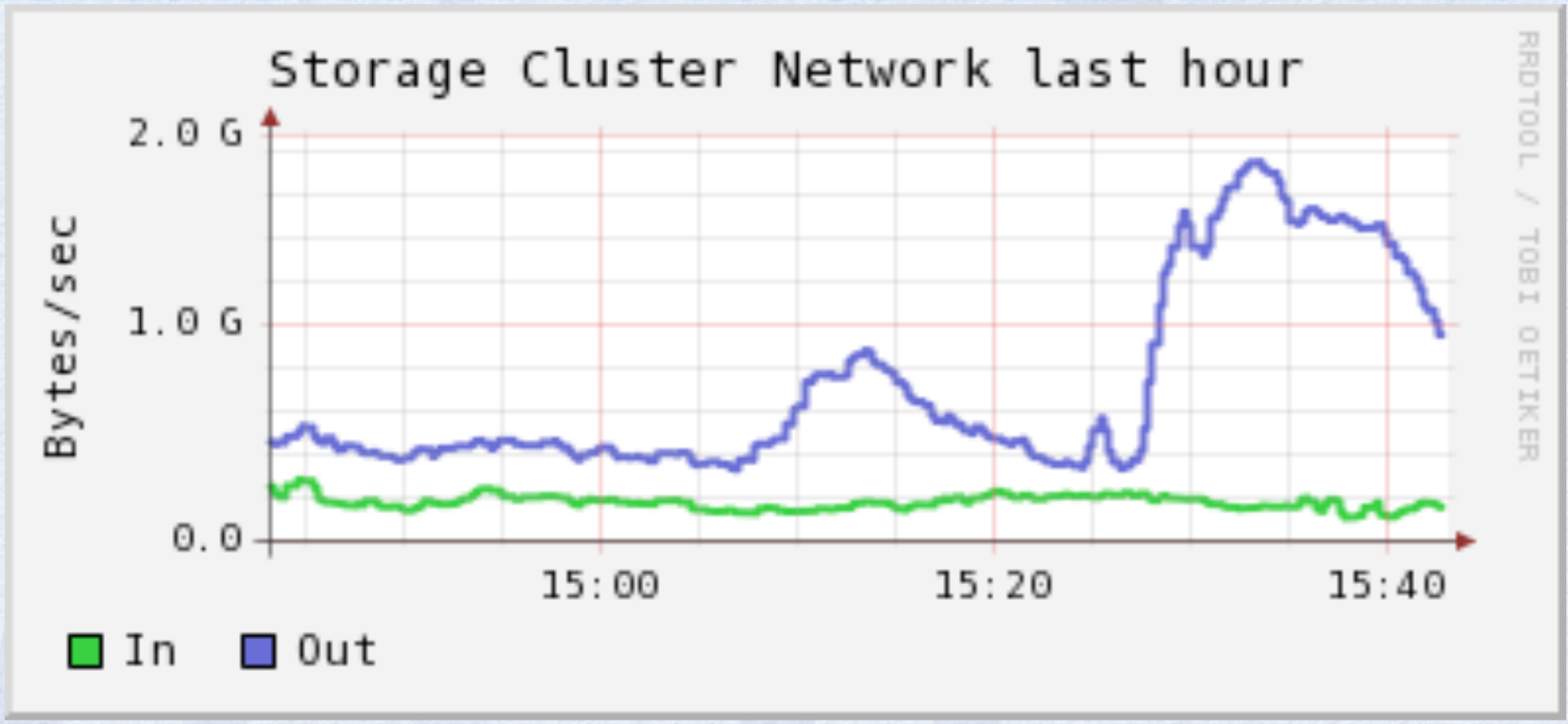


LUSTRE/STORM PERFORMANCE HEP TIER2

~500 CMS + Phedex WAN transfers

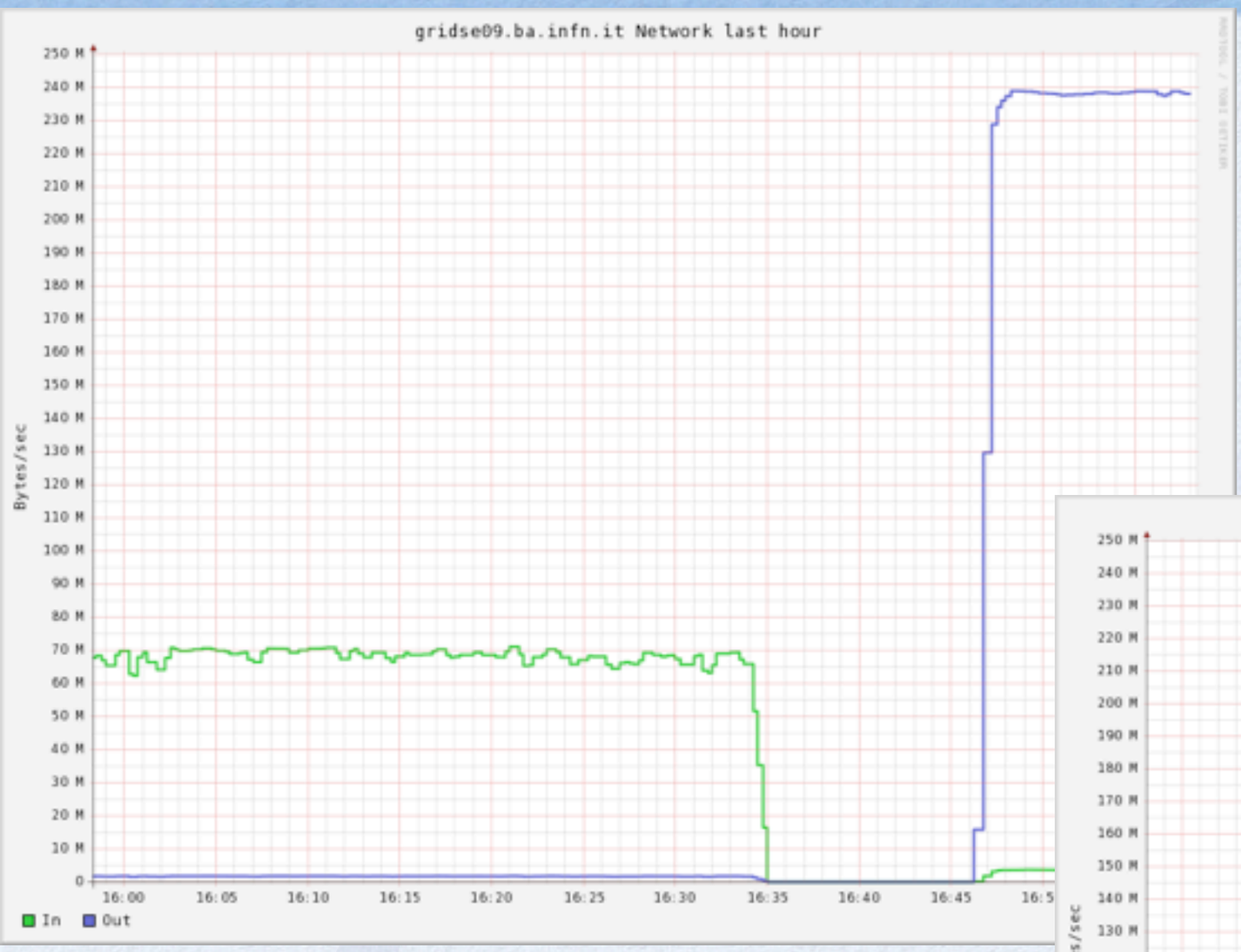
~4MB/s per job slot

15 disk servers

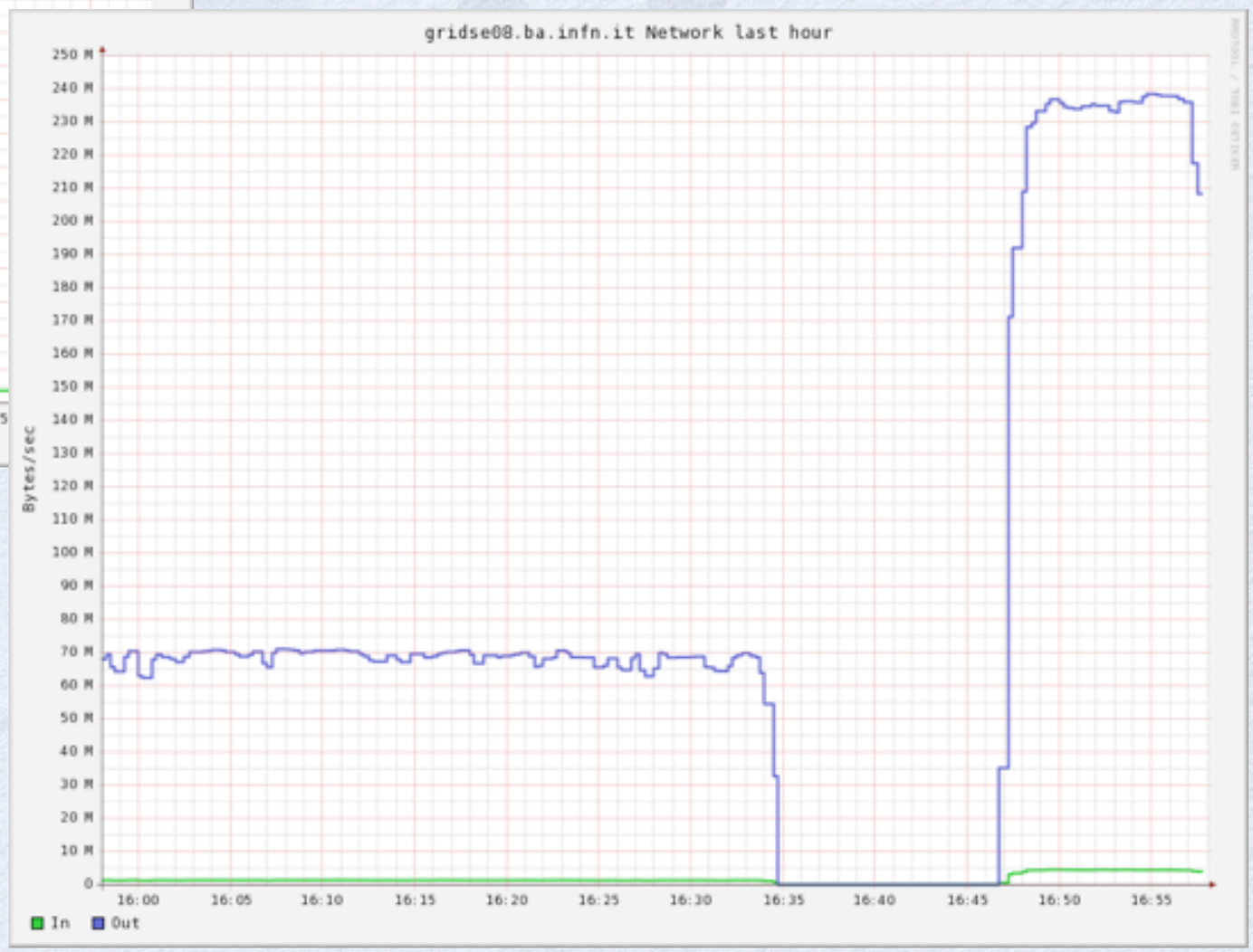


- The rate are measured with real CMS analysis jobs.
- SRM/gridftp layer provided by StoRM

TEST ON STORAGE HW AND SW: FEW RESULTS

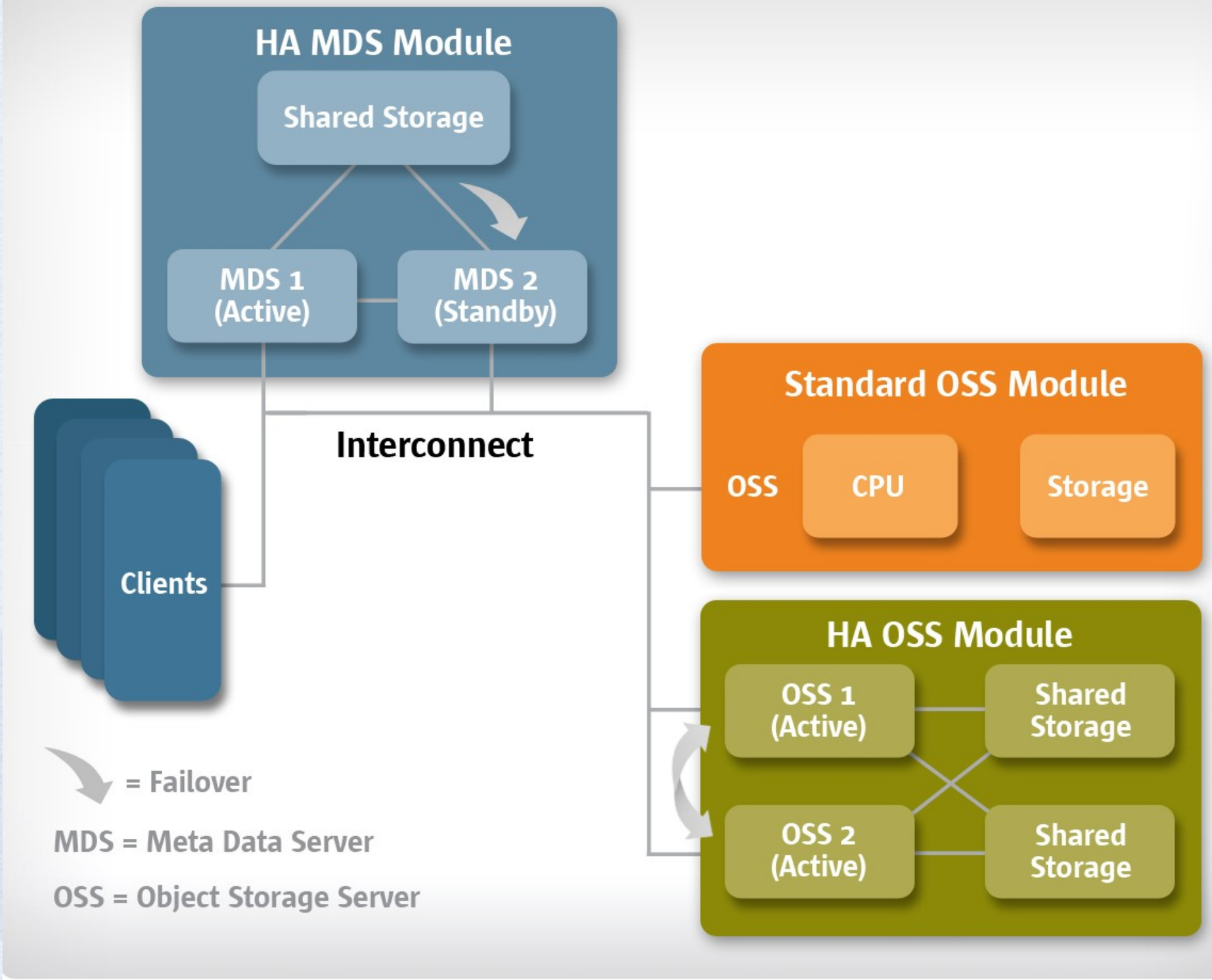


- xyratex 2 FC controller, 48+48 disk
- up to 96 TB RAW
- 2 disk servers

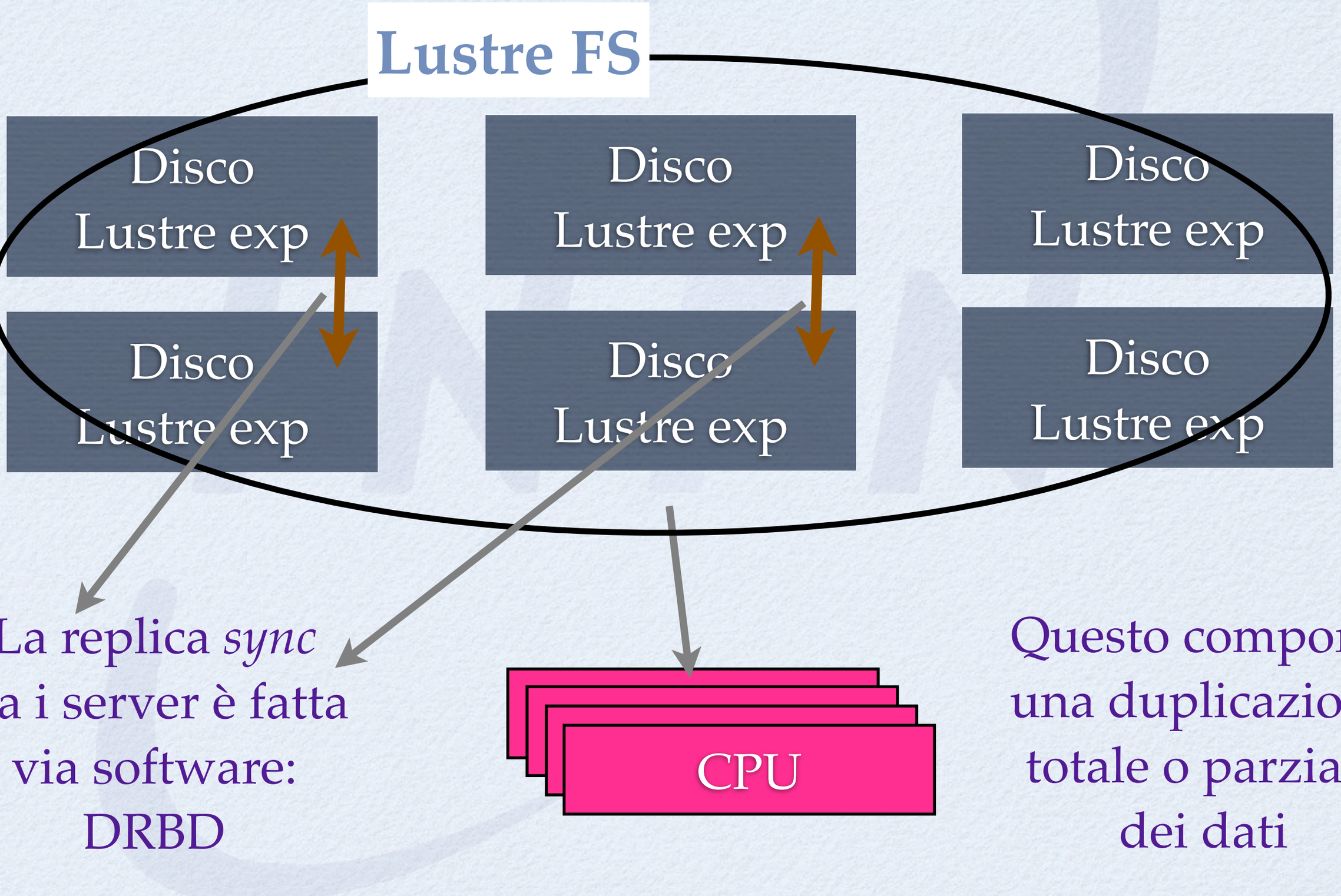


- it is possible to achieve HA configuration (see next slide)
- an aggregate of **~480MB/s**

LUSTRE CONFIGURATIONS



LUSTRE CONFIGURATIONS



La replica *sync* fra i server è fatta via software: DRBD

Questo comporta una duplicazione totale o parziale dei dati

LUSTRE -- AT A SUPERCOMPUTING CENTRE

Lustre at TACC Performance

- TACC ranger system – has observed 46 GB/sec throughput
- They use 50 Sun Fire X4500 servers as OSS
- A single app achieved 35 GB/sec throughput

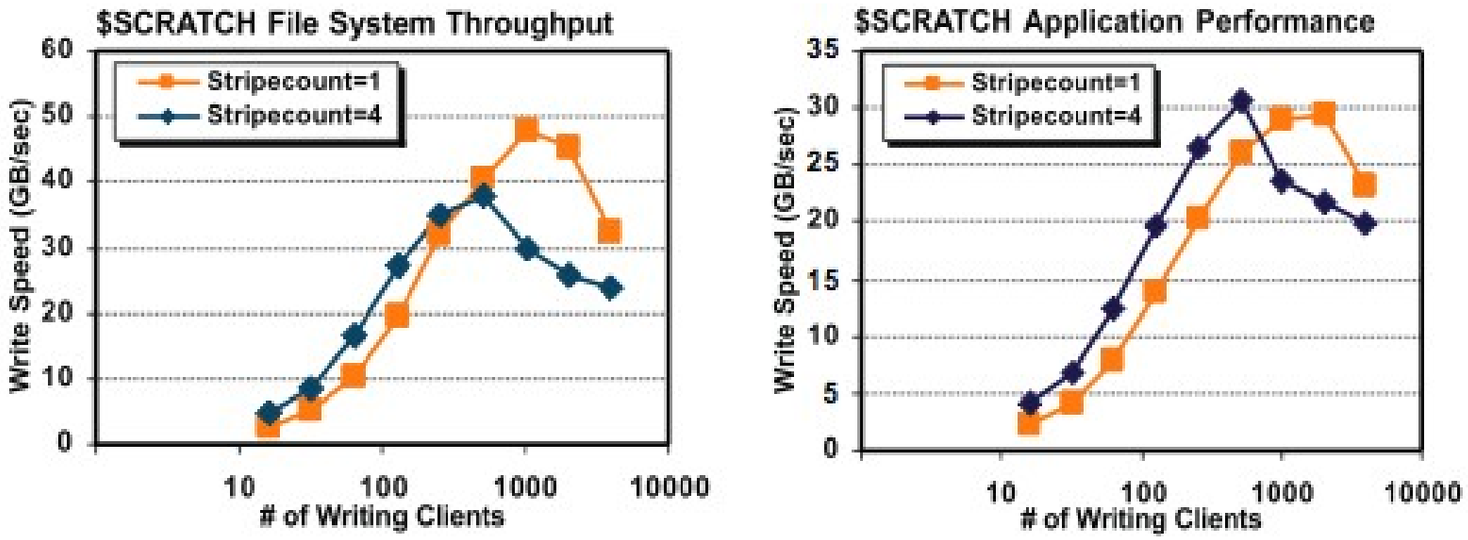


Figure 12. Lustre file system performance at TACC.

- "Typical numbers for a high-end MDT node (16-core, 64GB of RAM, DDR IB) is about 8-10k creates/sec, up to **20k lookups/sec** from many clients."

LUSTRE FUTURE (2.0)

- ZFS back-end support:
  - end-to-end data integrity
  - SSD read cache
- HSM support
  - with home made plugin
- Changelogs
  - Record events that change the filesystem namespace or file metadata.
- lustre_rsync
  - provides namespace and data replication to an external (remote) backup system without having to scan the file system for inode changes and modification times

HADOOP

HADOOP: CONCEPTS AND ARCHITECTURE

- Moving data to CPU is costly
  - Network infrastructure
  - And performance => latency
  - **Moving computational to data could be the solution**
- Scaling the storage performance, following the increase of computational capacity, is hard
  - **Increasing the number of disks together with the number of CPU could help the performance**
- There is the need to take into account machines failures in a computing centre
- DB also could benefit from this architecture

HADOOP: HIGHLIGHT

- It is developed till 2003 (born @google)
- It is a framework that provide: file-system, scheduler capabilities, distributed database
- Fault tolerant
  - Data replication
  - DataNode failure is ~transparent
  - Rack awareness
- Highly scalable
  - It is designed to use the local disk on the worker nodes
- Java based
- XML based config file

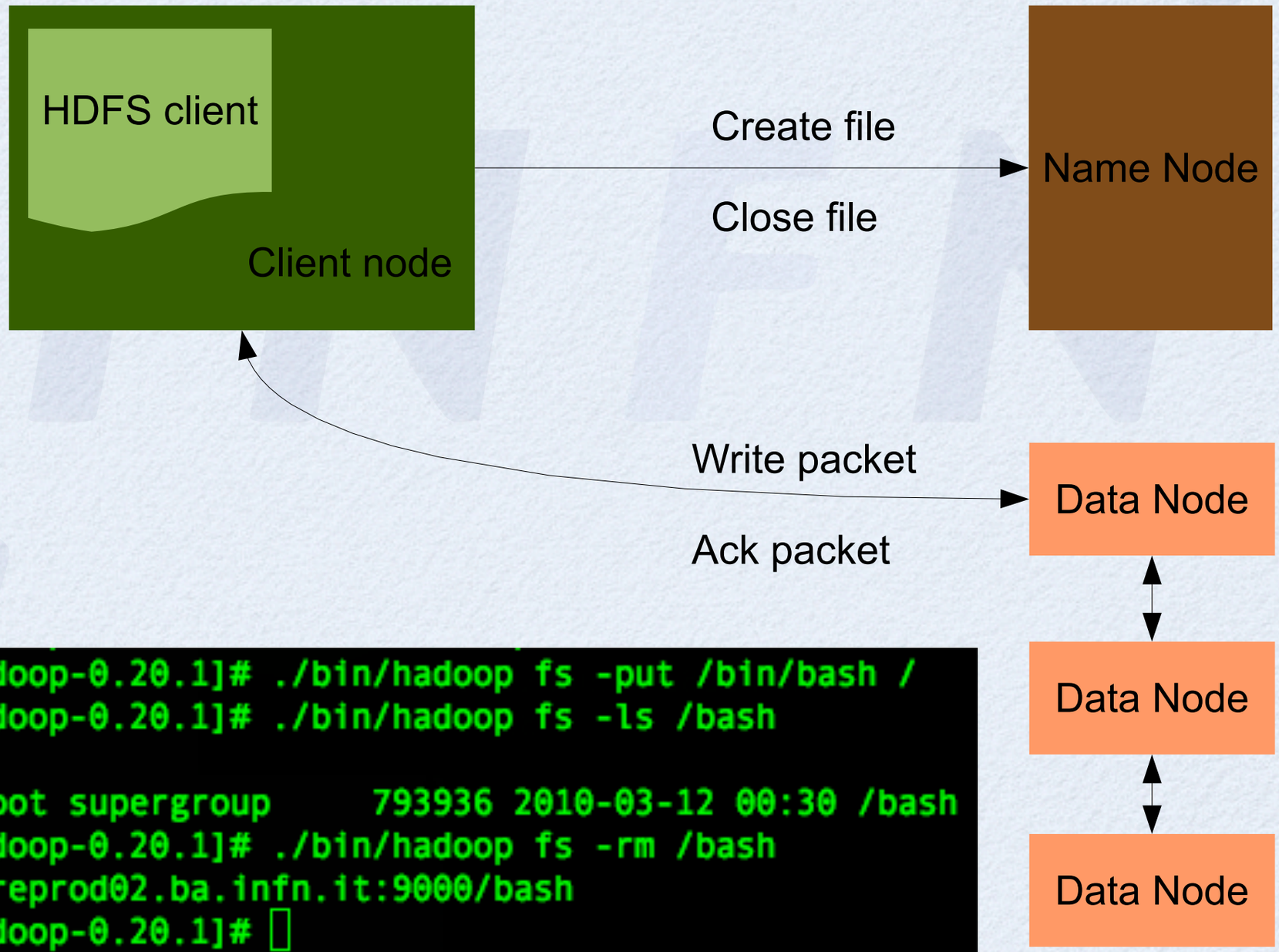
- A9.com
- AOL
- Booz Allen Hamilton
- EHarmony
- Facebook
- Freebase
- Fox Interactive Media
- IBM
- ImageShack
- ISI
- Joost
- Last.fm
- LinkedIn
- Metaweb
- Meebo
- Ning
- Powerset (now part of Microsoft)
- Proteus Technologies
- The New York Times
- Rackspace
- Veoh
- Twitter

HADOOP: HIGHLIGHT

- Using FUSE => some posix call supported
  - roughly “all read operation” and only “serial write operations”
- Web interface to monitor the HDFS system
- Java APIs to build code that is “data location aware”
- CKSUM at file-block level
- SPOF => metadata host
- HDFS shell to interact natively with the file system
- Metadata hosted in memory
  - sync with the file-system
  - it is easy to do back-up of the metadata

HADOOP: CONCEPTS AND ARCHITECTURE

Anatomy of a file write

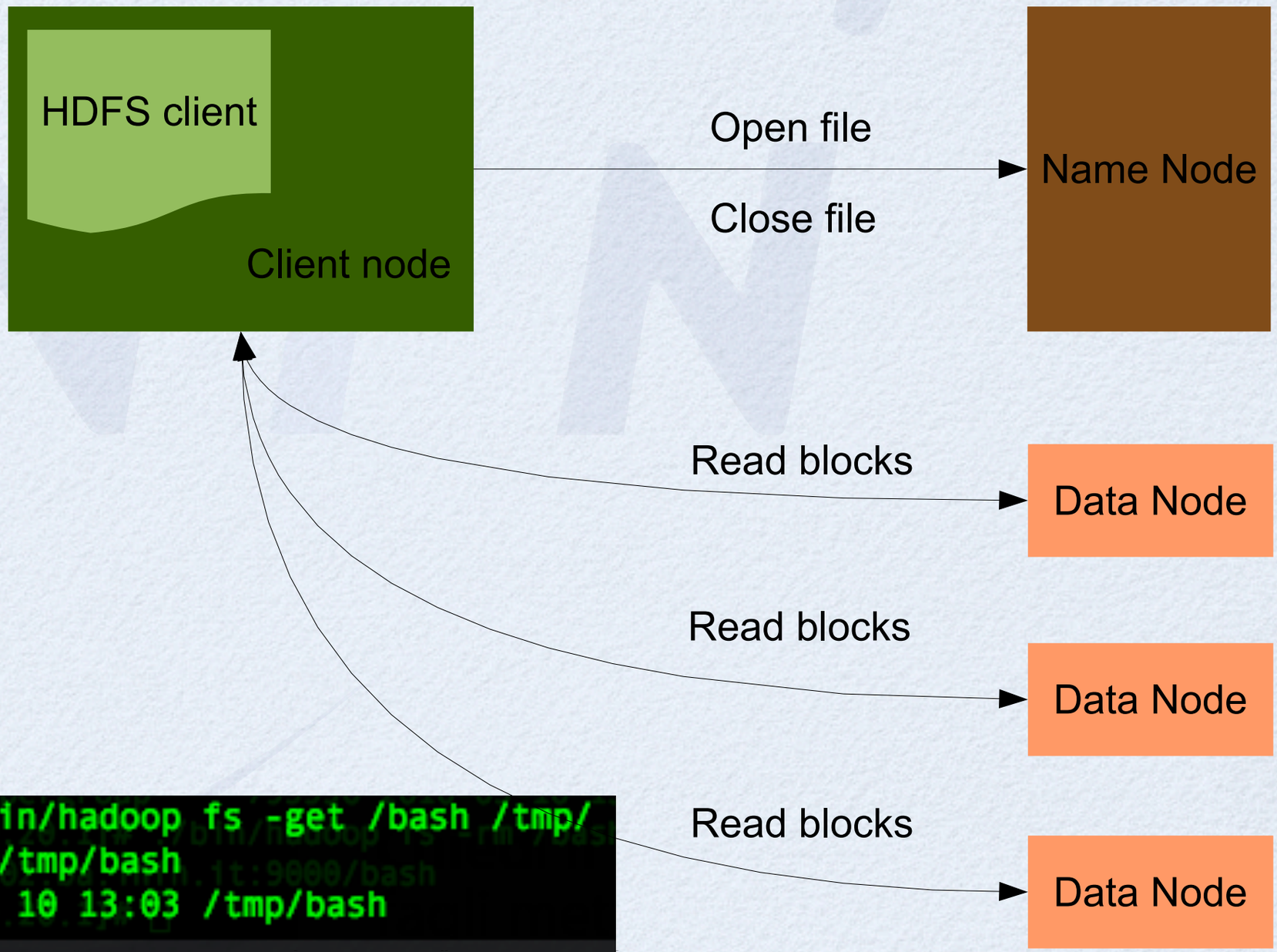



```
[root@pccms64 hadoop-0.20.1]# ./bin/hadoop fs -put /bin/bash /
[root@pccms64 hadoop-0.20.1]# ./bin/hadoop fs -ls /bash
Found 1 items
-rw-r--r--   3 root supergroup     793936 2010-03-12 00:30 /bash
[root@pccms64 hadoop-0.20.1]# ./bin/hadoop fs -rm /bash
Deleted hdfs://preprod02.ba.infn.it:9000/bash
[root@pccms64 hadoop-0.20.1]#
```

HADOOP: CONCEPTS AND ARCHITECTURE

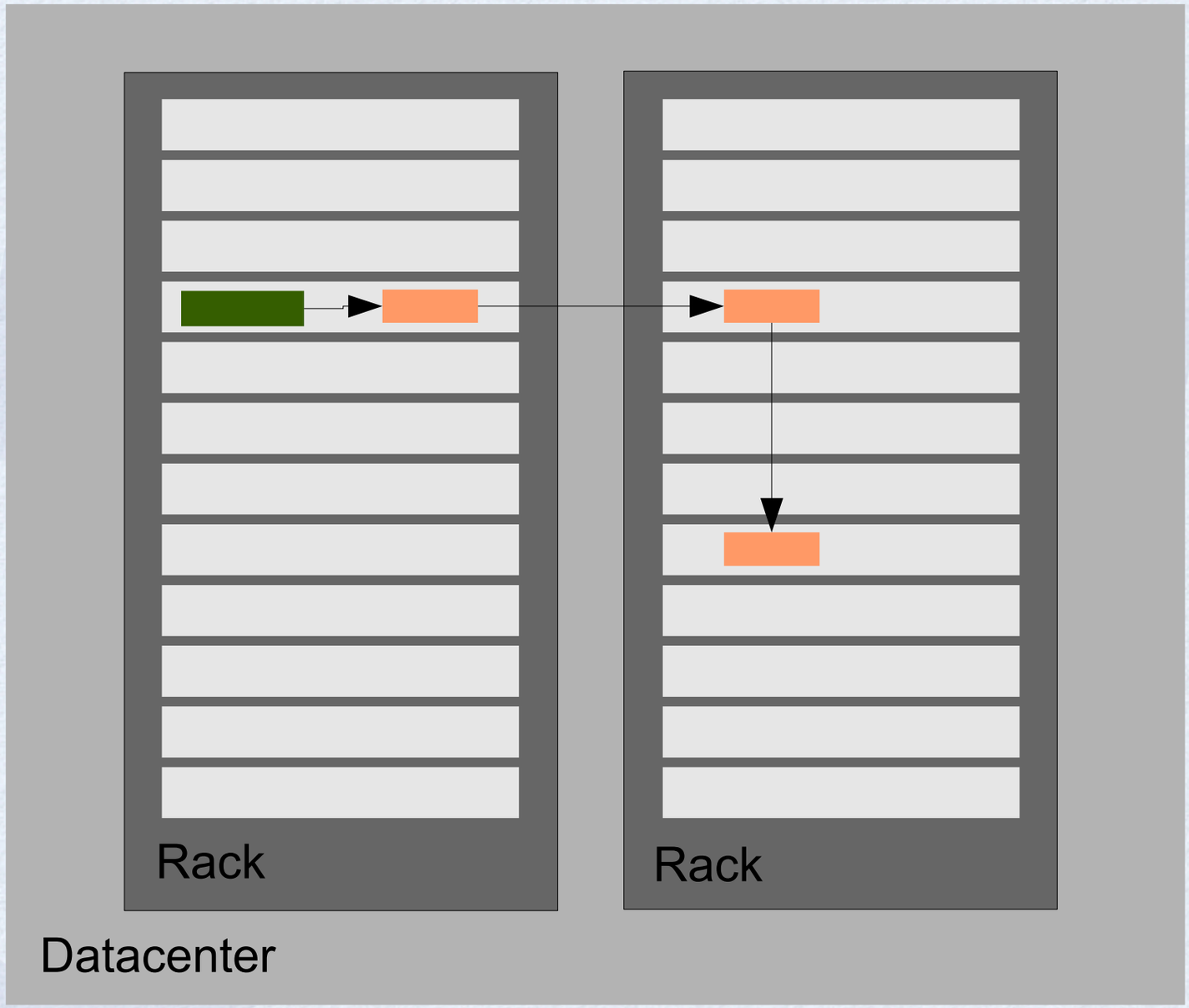
Anatomy of a file read

- Splitting files in different pools may give performance benefit when reading them back
- having the data replicated could be of help

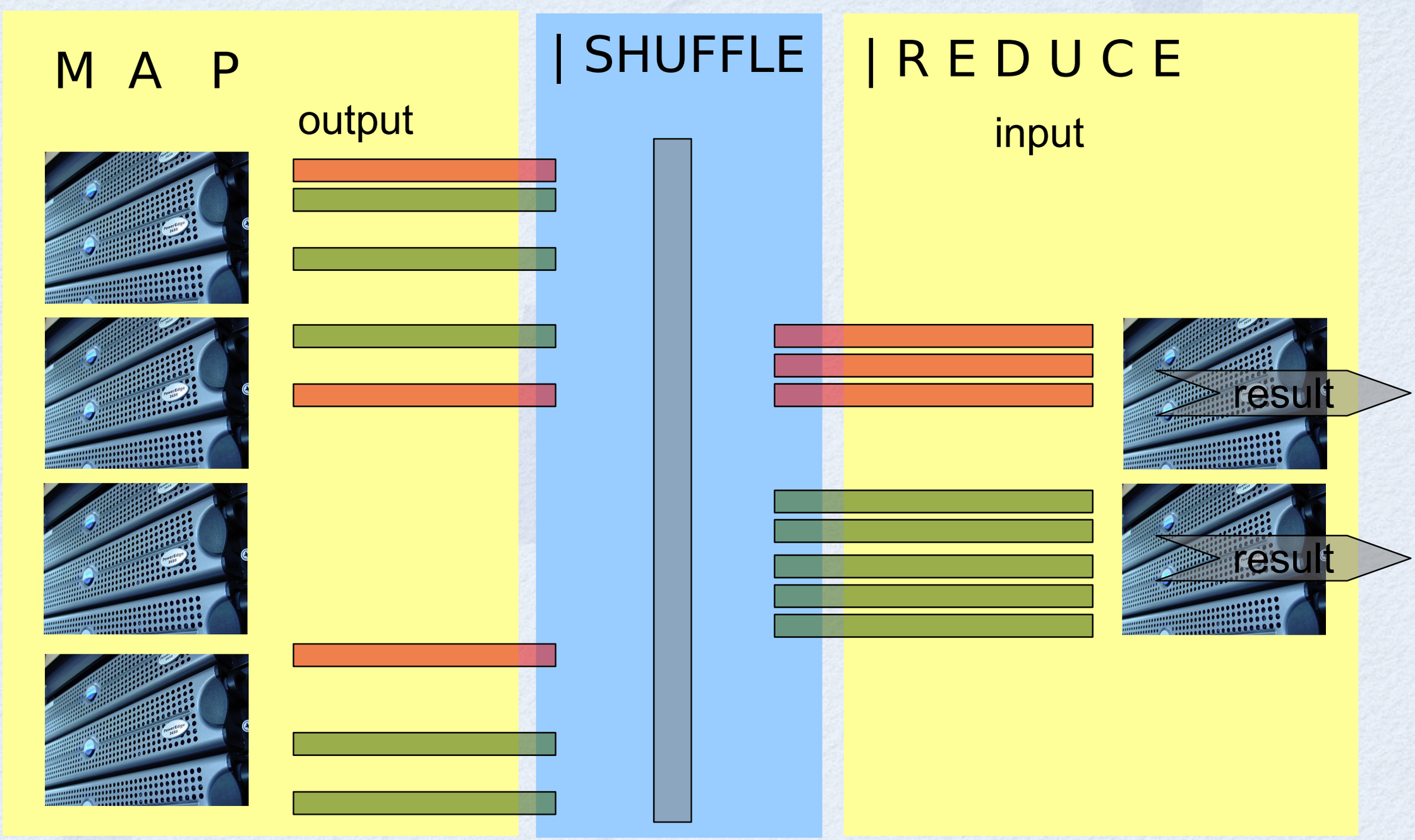



```
[root@pccms64 hadoop-0.20.1]# ./bin/hadoop fs -get /bash /tmp/
[root@pccms64 hadoop-0.20.1]# ll /tmp/bash
-rw-r--r-- 1 root root 793936 Mar 10 13:03 /tmp/bash
[root@pccms64 hadoop-0.20.1]#
```

HDFS Replication Strategy



HADOOP: CONCEPTS AND ARCHITECTURE



Local to data.
Outputs a lot less data.
Output can cheaply move.

Shuffle sorts input by key.
Reduces output significantly.

HADOOP: FEW EXAMPLES

"SORT EXERCISE"

Bytes	Nodes		Replication	Time
500,000,000,000	1406		1	59 seconds
1,000,000,000,000	1460		1	62 seconds
100,000,000,000,000	3452	10x data ~6x time	2	173 minutes
1,000,000,000,000,000	3658		2	975 minutes

Per node: 2 quad core Xeons @ 2.5ghz, 4 SATA disks, 8G RAM (upgraded to
16GB before petabyte sort), 1 gigabit ethernet.
Per Rack: 40 nodes, 8 gigabit ethernet uplinks.

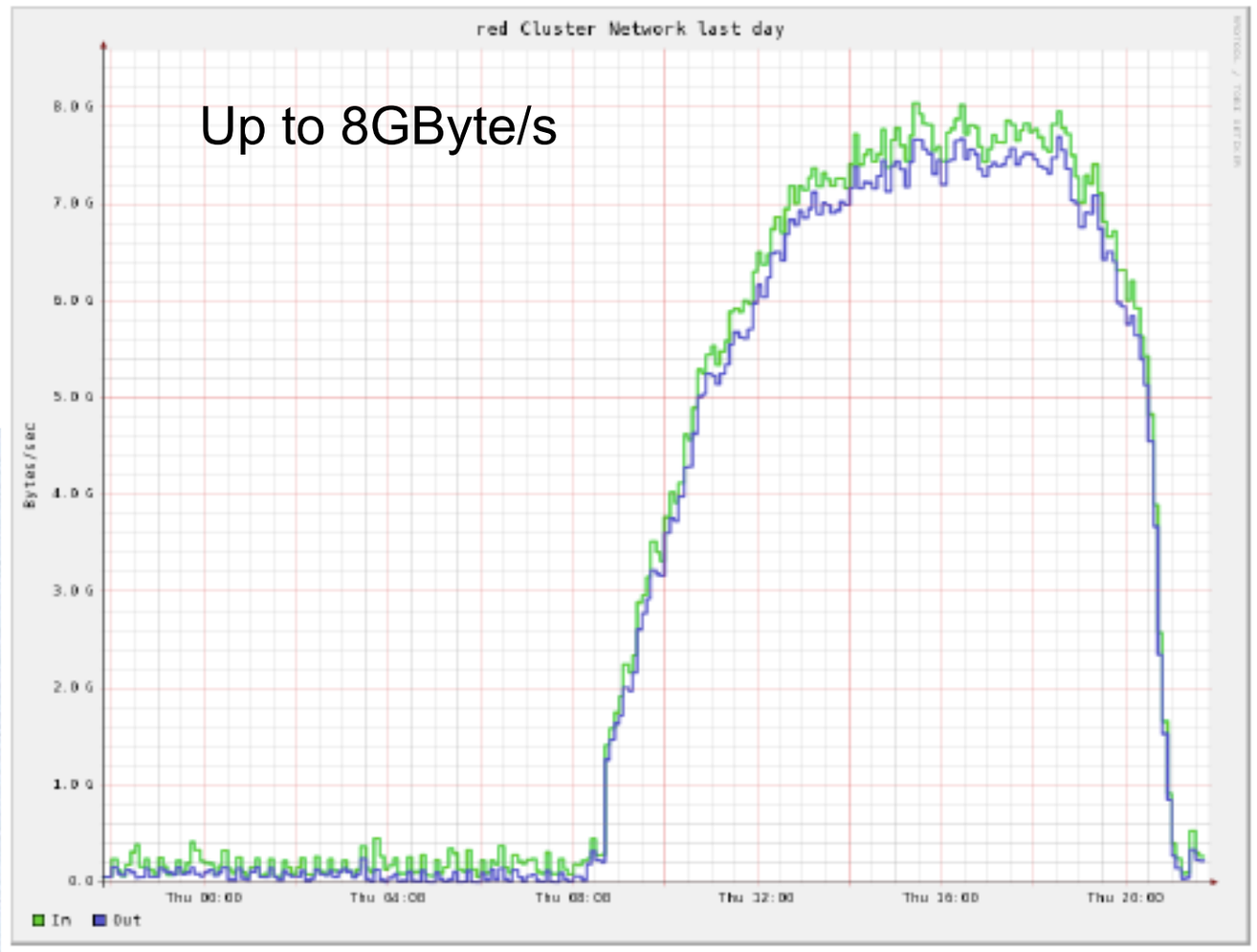
HADOOP: FEW EXAMPLES "CMS US EXAMPLE"

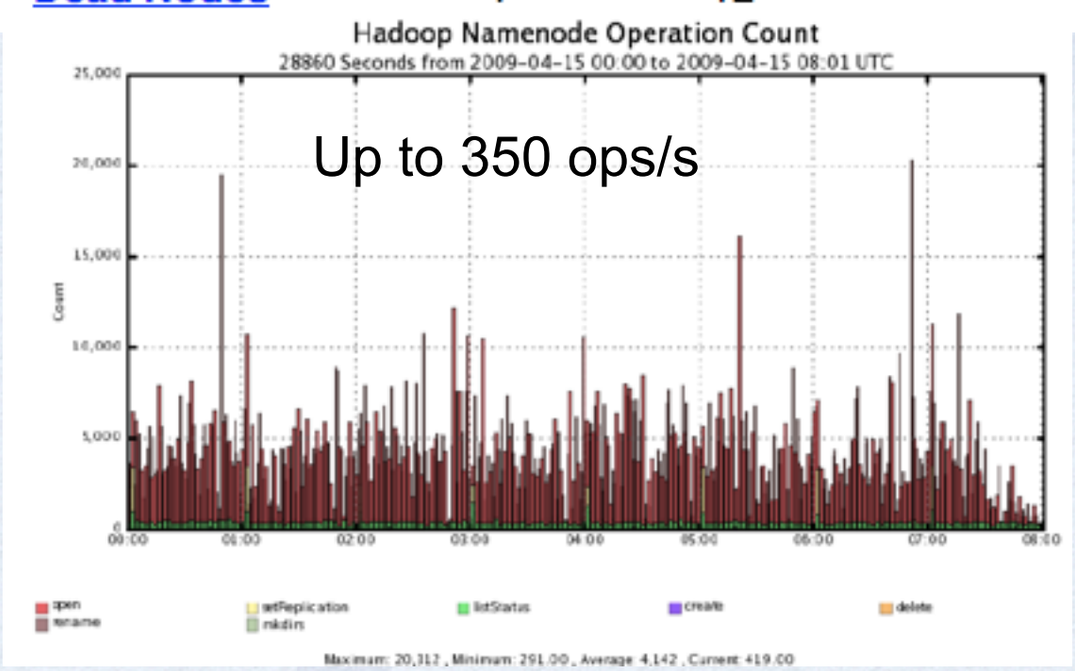
Cluster Summary

575761 files and directories, 2204886 blocks = 2780647 total. Heap Size is 3.86 GB / 7.11 GB (54%)

Configured Capacity	:	859.96 TB
DFS Used	:	424.82 TB
Non DFS Used	:	8.46 TB
DFS Remaining	:	226.7 TB
DFS Used%	:	64.37 %
DFS Remaining%	:	34.35 %
Live Nodes	:	142
Dead Nodes	:	12

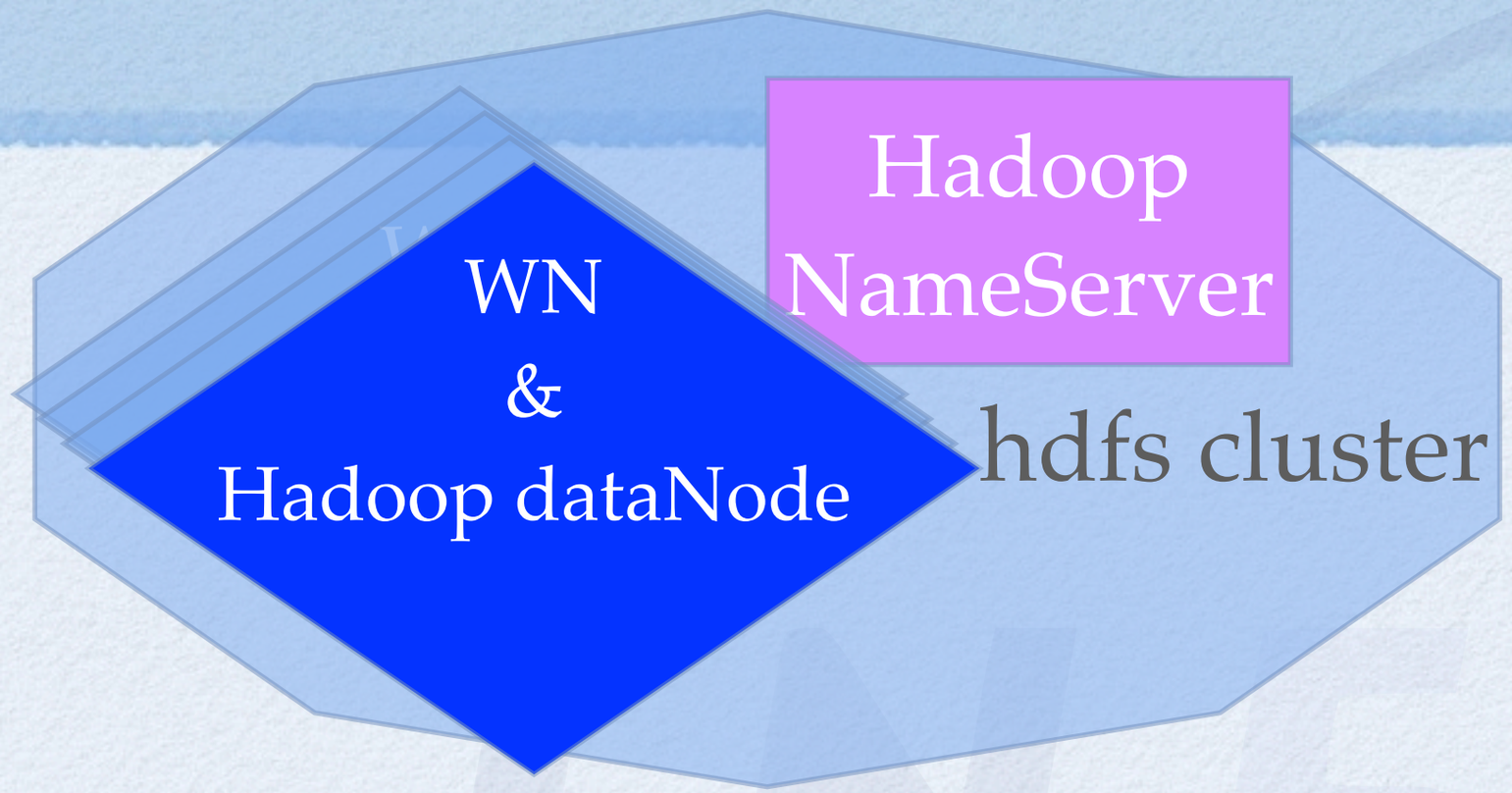
=> 800TB





- 2.5TB < Each DataNode < 21TB
- ~600 Core
- SRM/gridftp layer provided by FUSE and BestMan

HADOOP T3 TEST

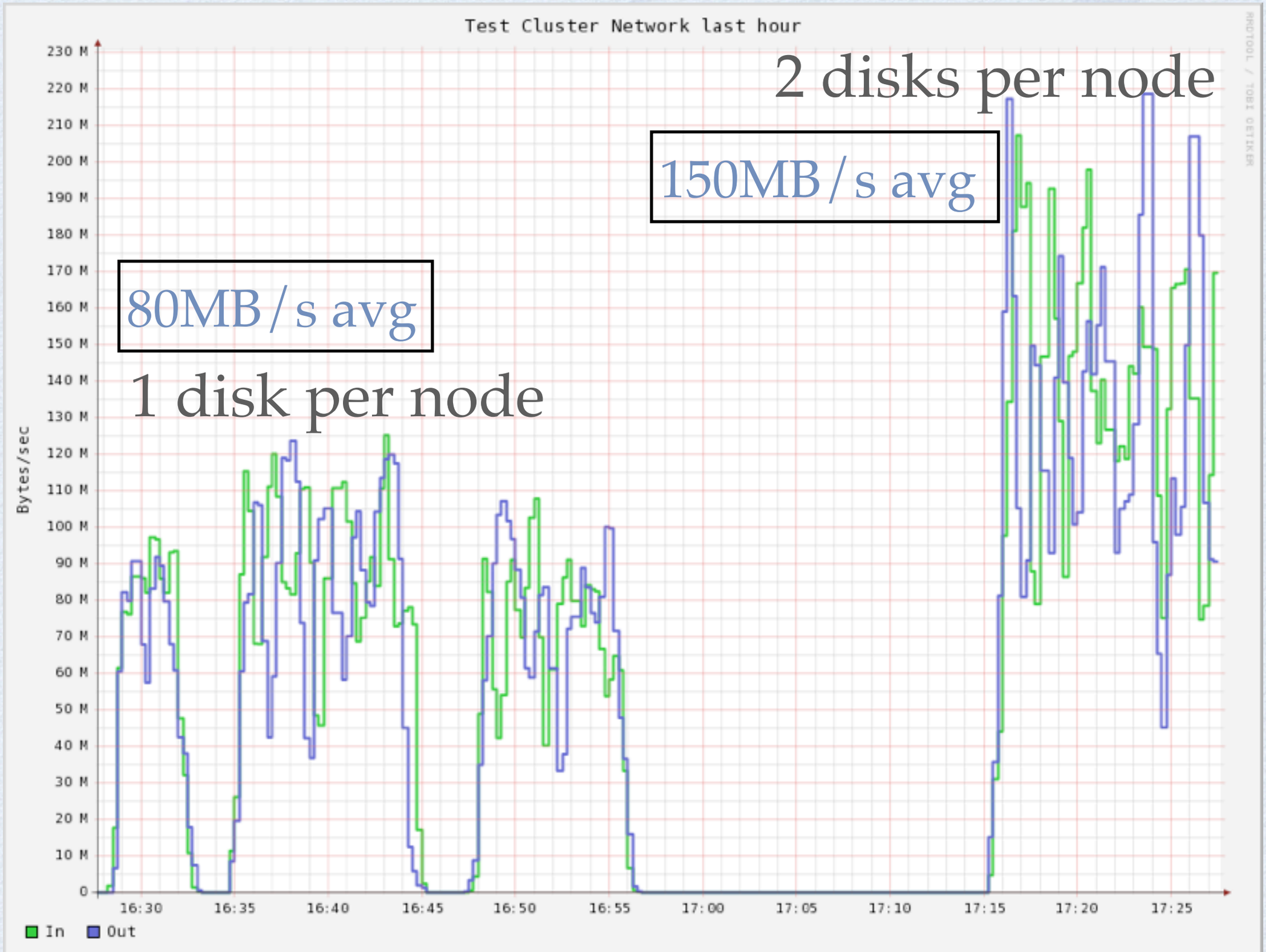


- up to **2** concurrent **node failed** w/o any service interruption

- Using 7 **old** test machine:
  - 2xXeon CPU
  - 4GB RAM each
  - 2x120GB HD each
  - 1Gbit/s eth
- 1 Admin node + WN
- 6 data node + WN

- 0.8TB of **redundant** storage
- **14** concurrent I/O **processes**
- **150 MB/s** of aggregate bandwidth

HDFS DUMMY SCALABILITY TEST

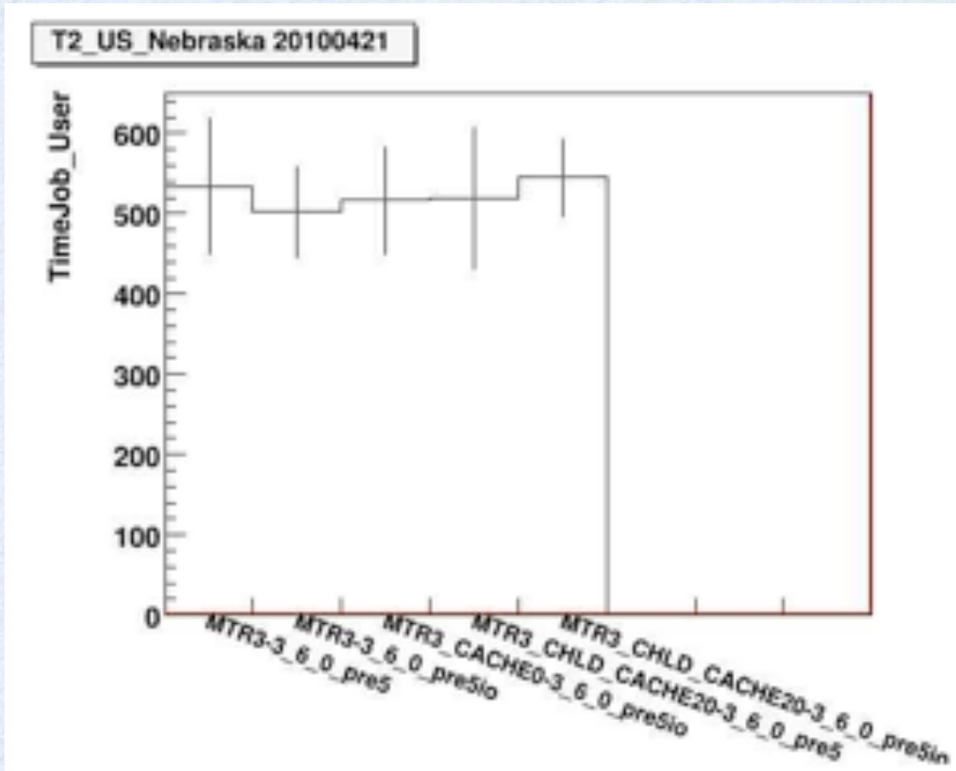


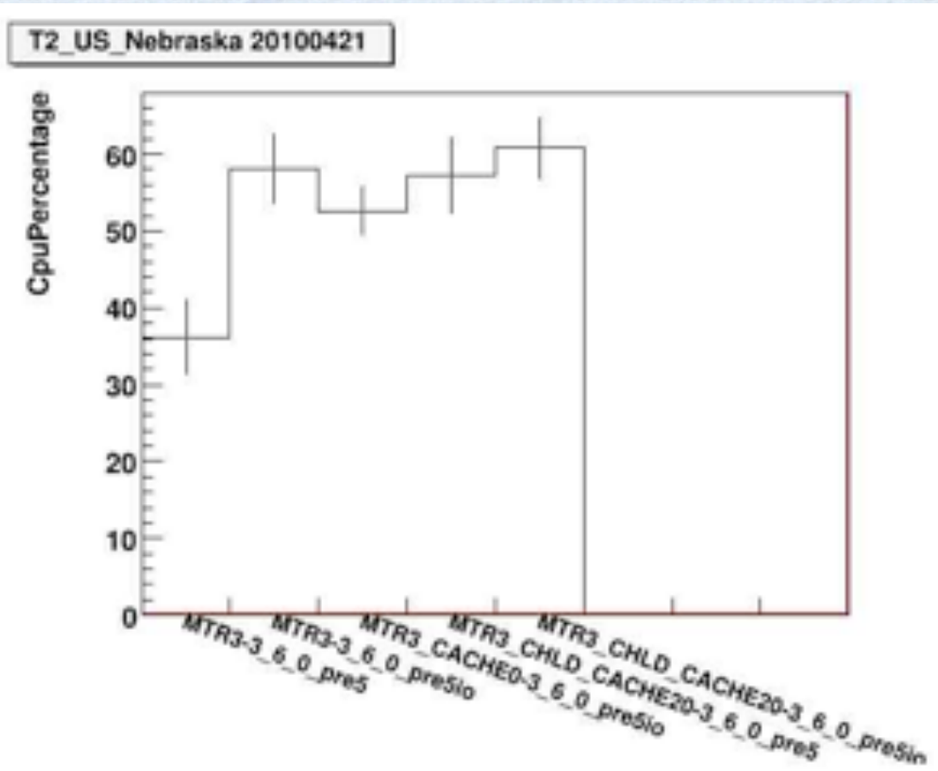
HADOOP: FUTURE

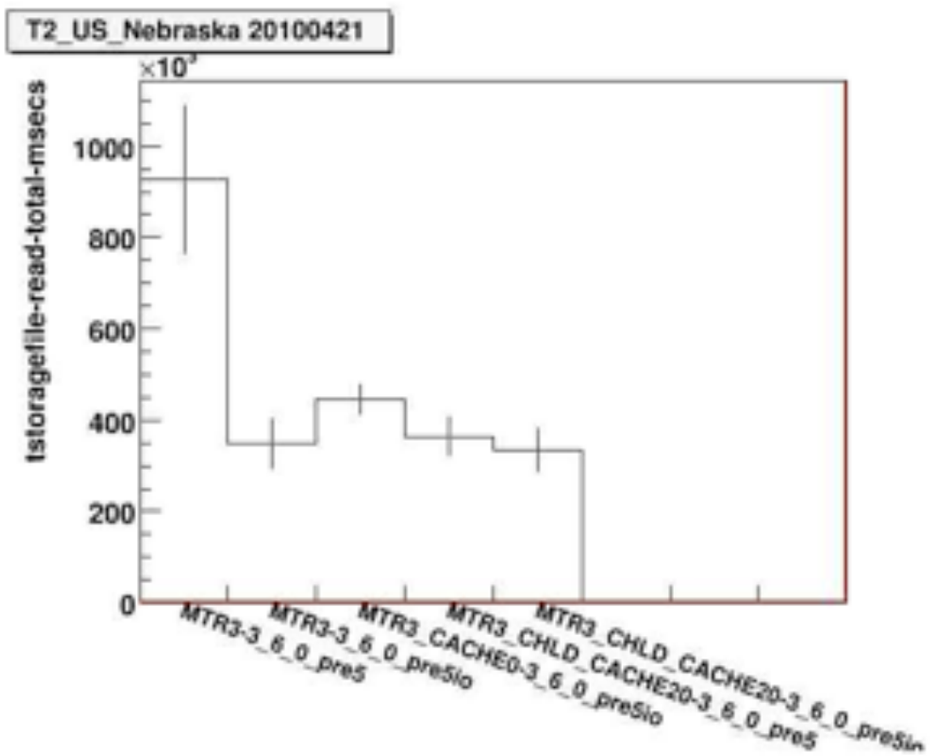
- Support for “append”
- Support for “sync” operation
- Cluster NameNode

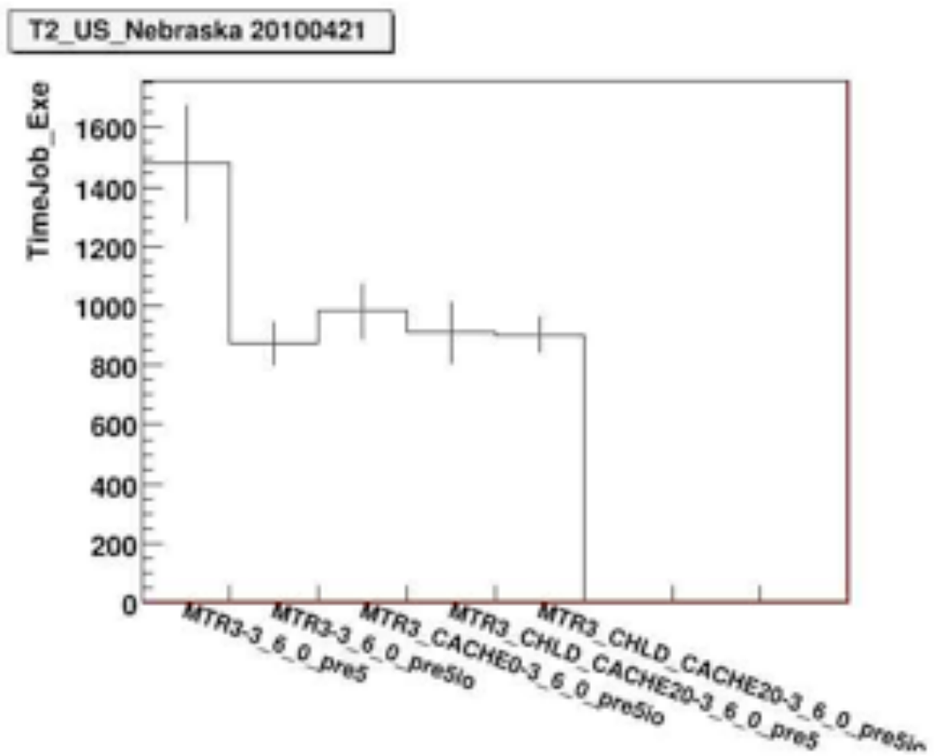
CMSSW NEW TEST

THANKS TO LEONARDO SALA!!







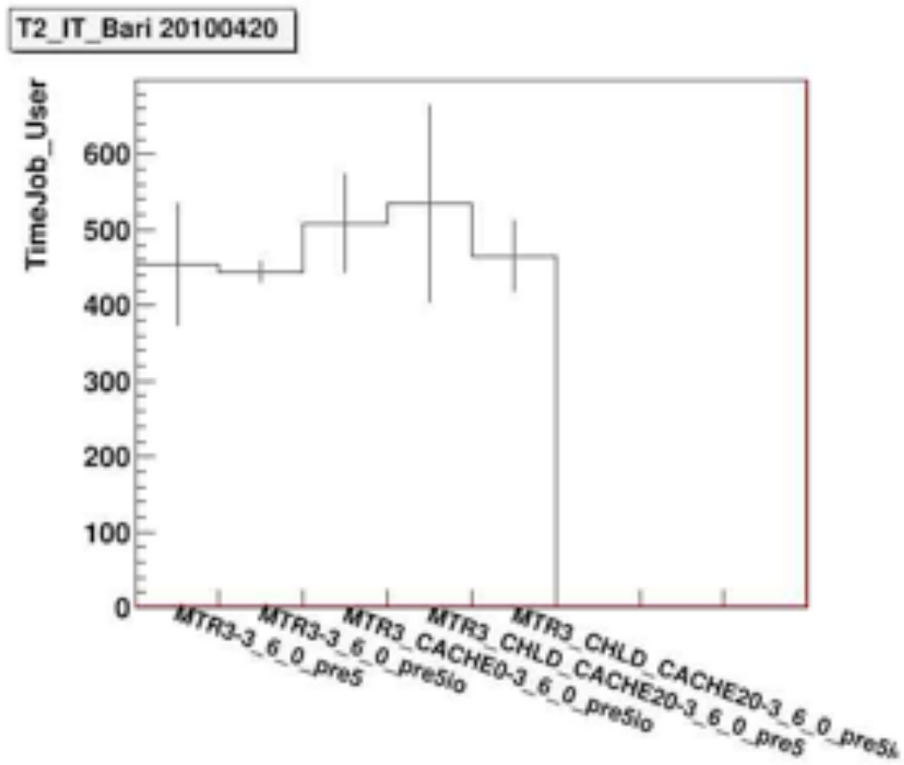


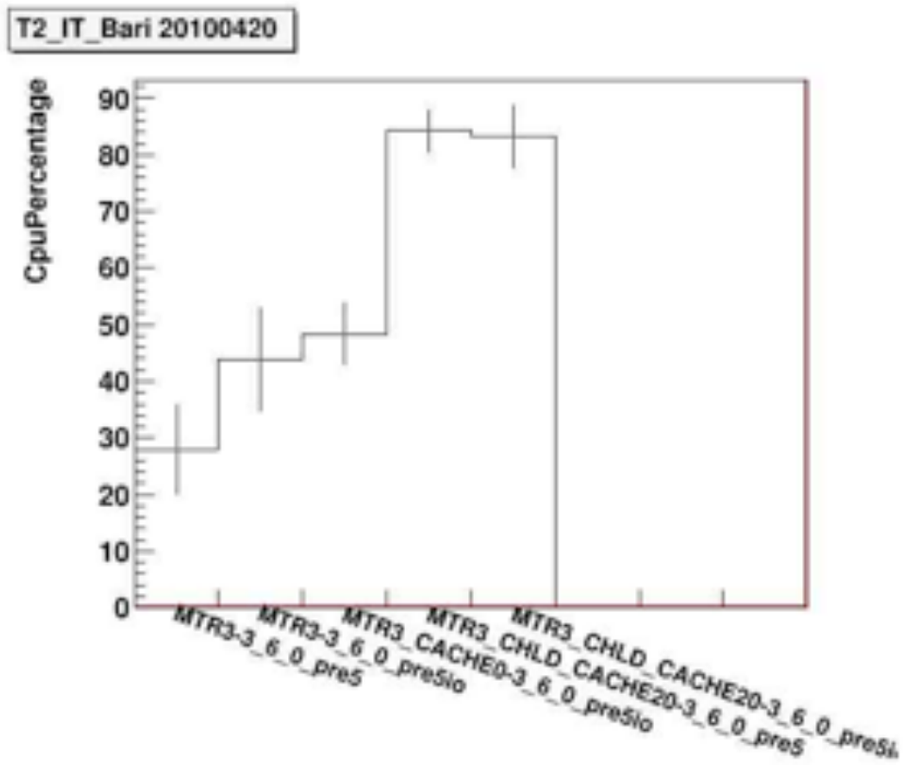
CMSSW NEW TEST

THANKS TO LEONARDO SALA!!

CMSSW NEW TEST

THANKS TO LEONARDO SALA!!





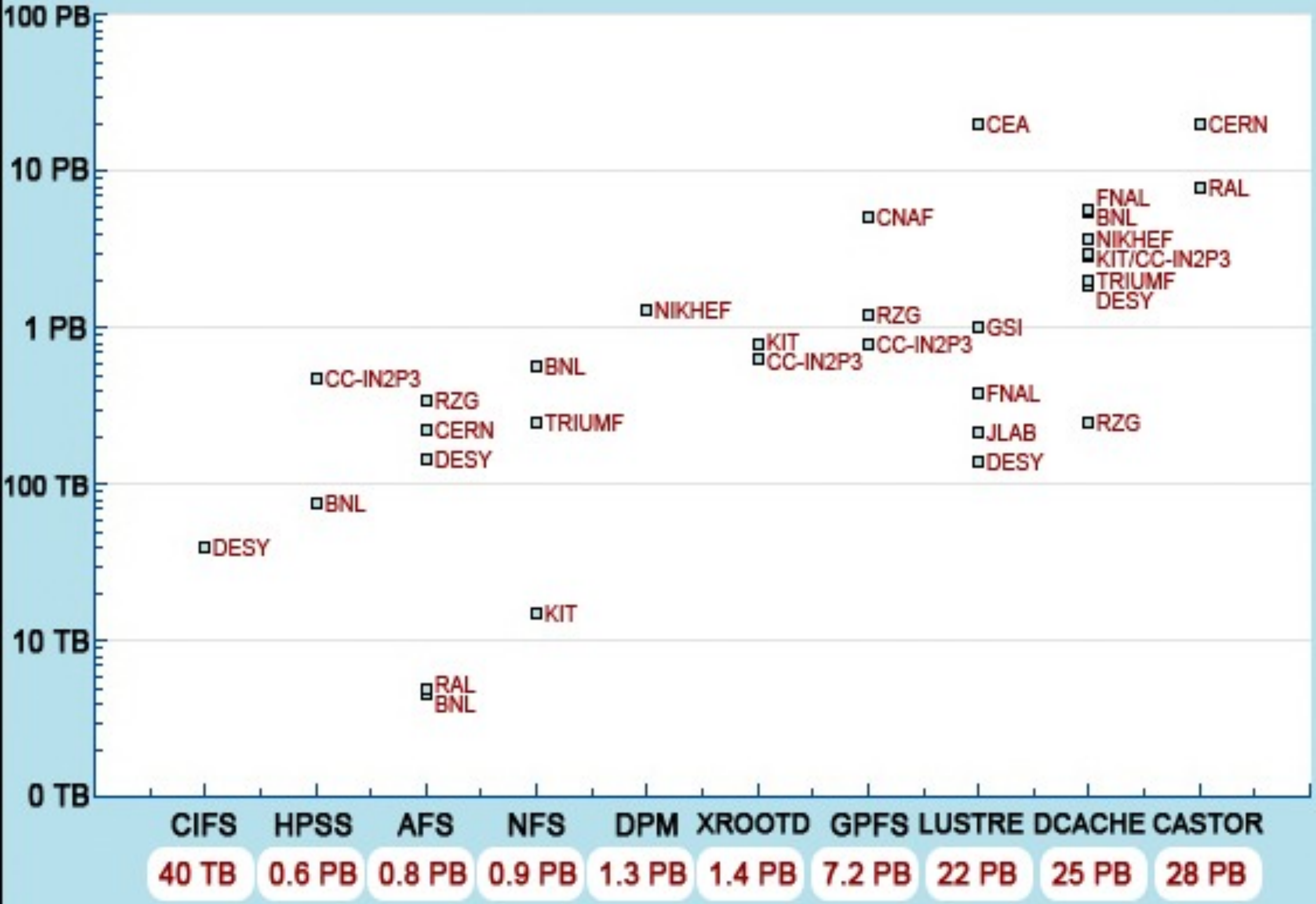
Credits for the late period

- The new test laboratory at KIT was built on the top of hardware kindly provided by Karlsruhe Institute of Technology (rack and network infrastructure, load farm) and E4 Computer Engineering (new disk server). CERN had contrubuted with some funds to cover a part of human hours.

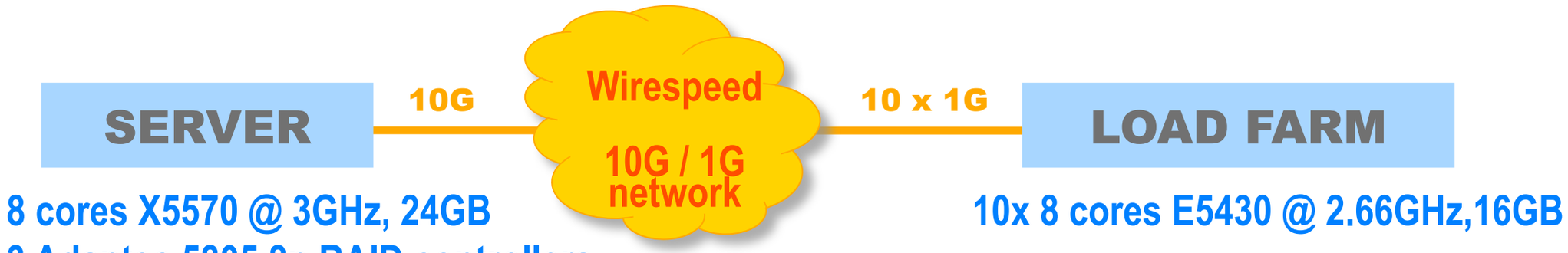
- These people participated in provisioning, funding, discussions, laboratory building, preparation of test cases and test framework, tests and elaboration of the results:

CASPUR	A.Maslennikov (Chair), M.Calori (Web Master)
CEA	J-C.Lafoucriere
CERN	B.Panzer-Steindel, D. van der Ster, R.Toebbicke
DESY	M.Gasthuber, P.van der Reest
E4	C.Gianfreda
INFN	G.Donvito, V.Sapunenko
KIT	J.van Wezel, A.Trunov, M.Alef, B.Hoeft
LAL	M.Jouvin
RZG	H.Reuter

Terabytes on disk per type of the shared area



Hardware setup 2010 at KIT



This setup reperesents well an elementary fraction of a typical large hardware installation and has basically no bottlenecks:

- Each of the three Adaptec controllers may deliver 600+ MB/sec (R6)
- Ttcp memory-memory network test (1 server – 10 clients) shows full 10G speed

(In 2009 we were limited by 4x 1G NICs and only one RAID controller)

Details of the current test environment

- RHEL 5.4/64bit on all nodes (kernel 2.6.18-164.11.1.lustre / -164.15.1)
- Lustre 1.8.2
- GPFS 3.2.1-17
- OpenAFS/OSD 1.4.11 (trunk 984)
- dCache 1.9.7

- Use Case 1: CMS “Data Merge” standalone job - fw v.3.4.0 (Giacinto Donvito)

- Use Case 2: ATLAS “Hammercloud” standalone job – fw v.15.6.1 (Daniel van der Ster)

Tunables

We report here, for reference, some of the settings that were used so far.

Diskware: three stanadlone RAID-6 arrays of 8 spindles, stripe size=1M

Lustre: No checksumming, No caching on server

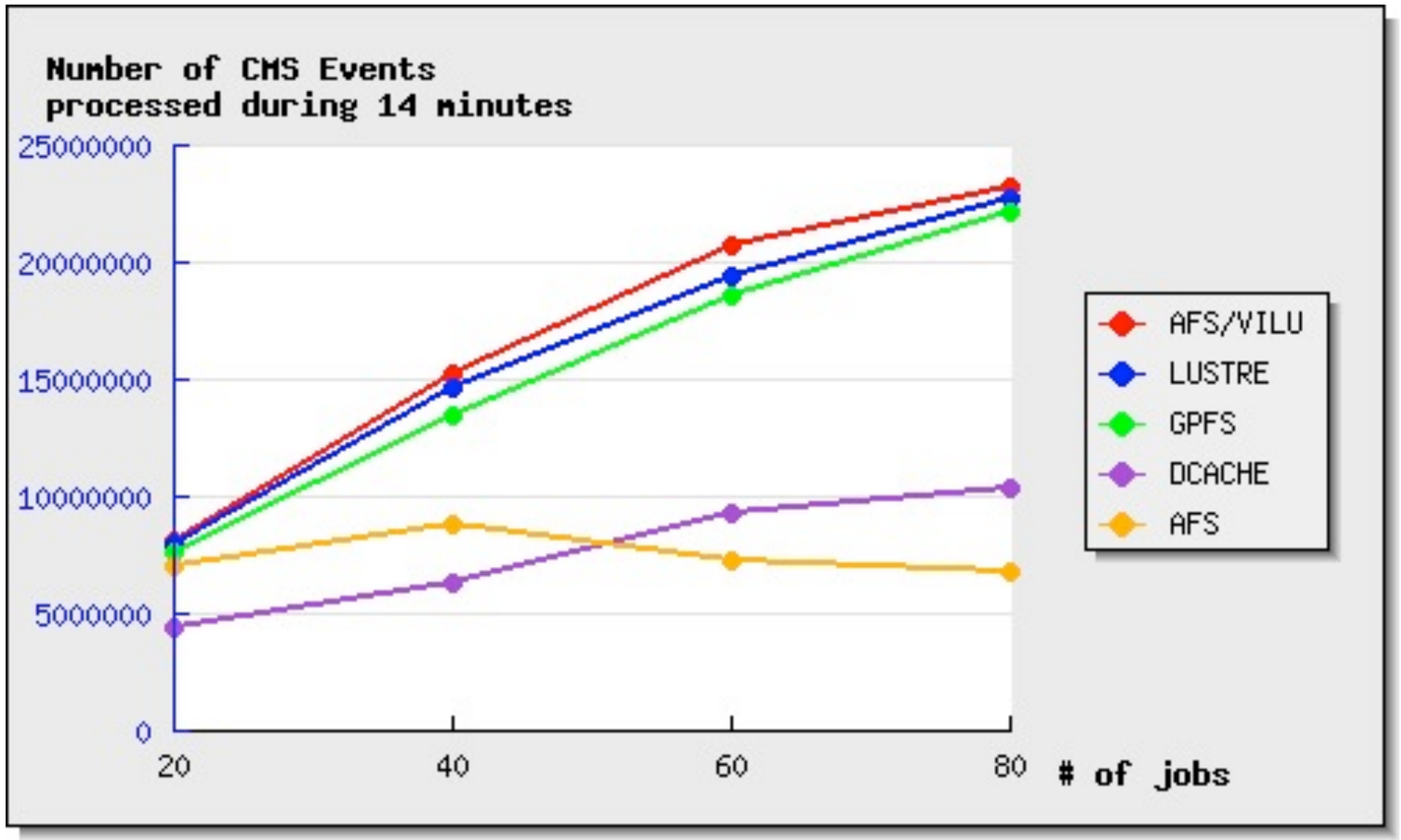
Formatted with: “-E stride=256 -E stripe-width=1536”
Data were spread over 3 file systems (1 MGS +3 MDT)
OST threads: “options ost oss_num_threads=512”
Read-aheads on clients: 4MB (CMS), 10MB (ATLAS)

GPFS: 3 NSDs, one per RAID-6 array

3 file systems (one per NSD)
-B 4M –j cluster
maxMBpS 1250
maxReceiverThreads 128
nsdMaxWorkerThreads 128
nsdThreadsPerDisk 8
pagepool 2G

AFS: 3 XFS vicep or dCache pool partitions (one per RAID array)
(dCache) Formatted with: “-i size=1024 -n size=16384 -l version=2 -d sw=6,su=1024k”
Mounted with: “logbsize=256k,logbufs=8,swalloc,inode64,noatime”
Afsd options: “memcache, chunksize 22, cache size 500MB”
Dcache options: DCACHE_RAHEAD=true, DCACHE_RA_BUFFER=(100KB-100MB)

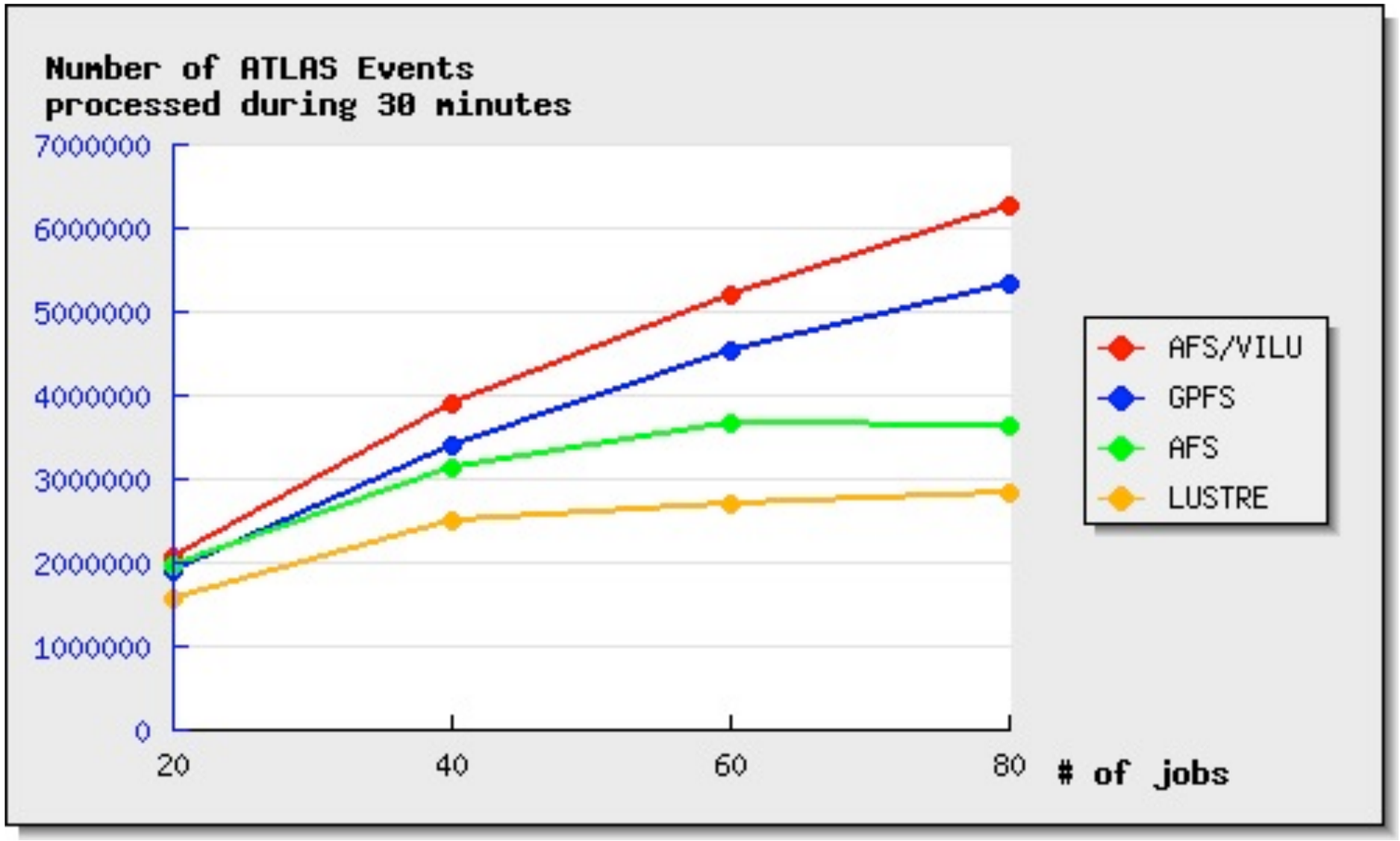
Current CMS use case results



For this test case, GPFS and Lustre are almost equally efficient. AFS/Vicep-over-Lustre looks surprisingly good.

The dCache result is very fresh and still has to be investigated. We however plot it here along with the others since the CMS test job was taken from the real life environment. The dCache team expressed an interest to verify the correctness of dCache and/or setup usage in this case, this will shortly be done in collaboration with them.

Current ATLAS use case results



The ATLAS job was prepared in the beginning of 2010; since then, ATLAS had migrated to a new data format and, consequently, to the new data access pattern. We were still using the previous version known for its high fraction of random access I/O. Thus it was of no surprise to discover that native Lustre was the most inefficient solution for this use case. However, AFS/Vicep with Lustre transport had shown the best results, like in the case of CMS. We were yet unable to run the dCache-based ATLAS test, this will be done soon.

CONCLUSIONS

	Lustre	Hadoop
Posix Functionalities	**Fully**	**Partially**
Quota	**Fully**	**Directory Quota**
Data Replica	**Not easy**	**Easy**
Metadata Replica	**Not natively**	**Not natively**
Resilient on SPOF	**Not natively**	**Not natively**
Management Cost	**Low**	**Could be costly**
Platform Supported	**SLC4/5 - Suse Linux**	**Every Platform**
Installation procedure	**Easy**	**Fairly easy**
Doc/Support	**Good**	**Fairly good**
Hep experience	**Fairly good**	**Just starting now**

CONCLUSIONS

- Lustre born in the HPC environment and it can guarantee good performance on standard servers (SAN or similar)
  - completely posix compliant
  - the scalability seems guaranteed from the biggest installation in supercomputing centres, but the use case are different from the HEP analysis
- Hadoop can provide needed performance and scalability by means of commodity hw
  - maybe it requires more man power to manage it if the installation grow too much in size
  - not fully posix compliant
  - Is not easy to use MapReduce on HEP code, it could be an interesting development for “future” experiments?