# Overview of the new technologies and evolution of storage systems for handling large volume of data

Giacinto Donvito - INFN-BARI

SuperB R&D Workshop -- Ferrara 2010

# OUTLOOK

- New trends on open source storage software
- Overview on Lustre
  - Lustre architecture and features
  - Some Lustre examples
  - Future developments
- Hadoop: concepts and architecture
  - Feature of HDFS
  - Few HDFS examples
- CEPH: a new concept for the storage
  - Key Features
  - status and future plans
- Conclusions

# Trends on Storage Software

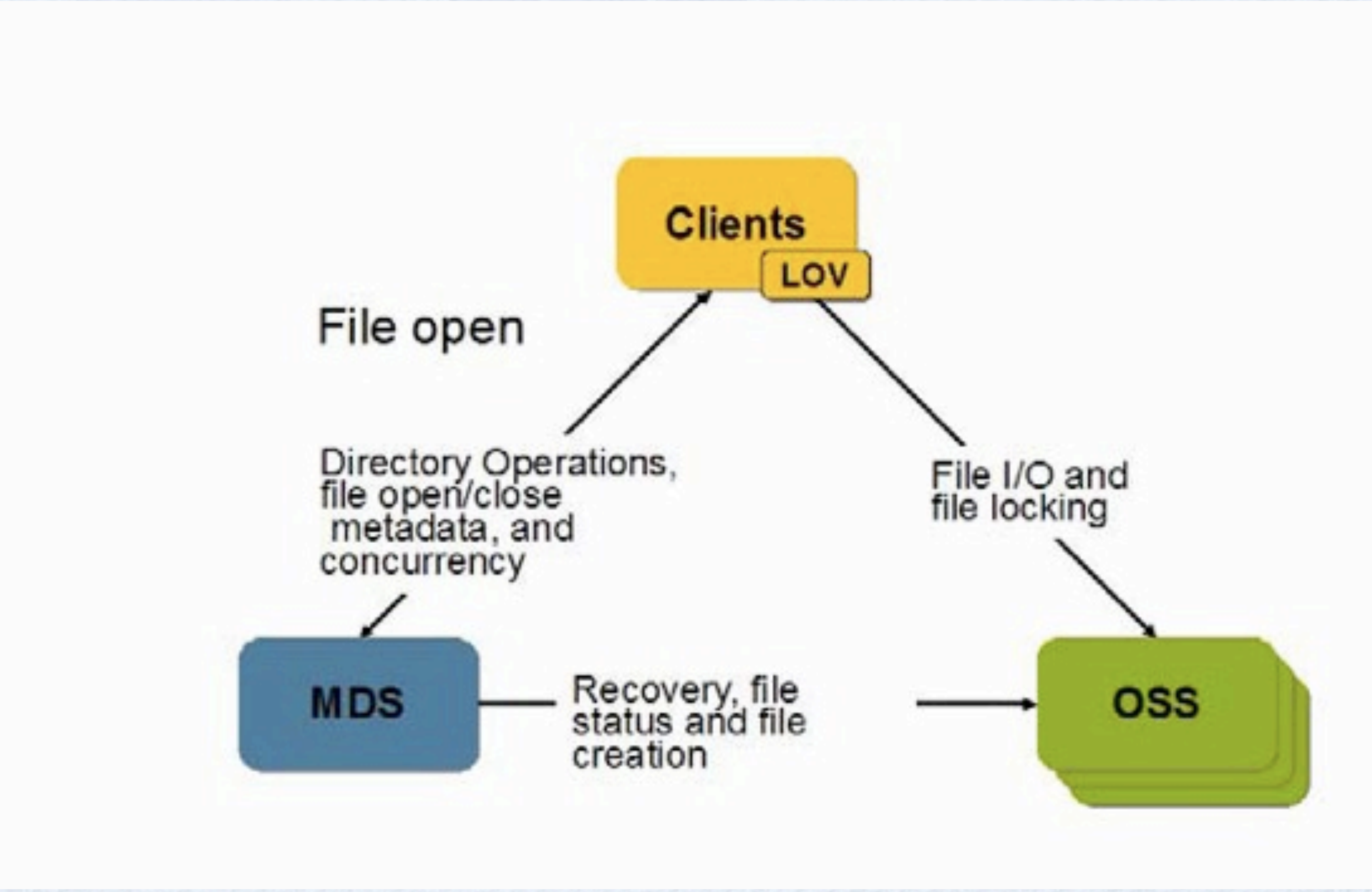
**Requirements:**

- 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
- But 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 1.8.2

- 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 1.8.2

- 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 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

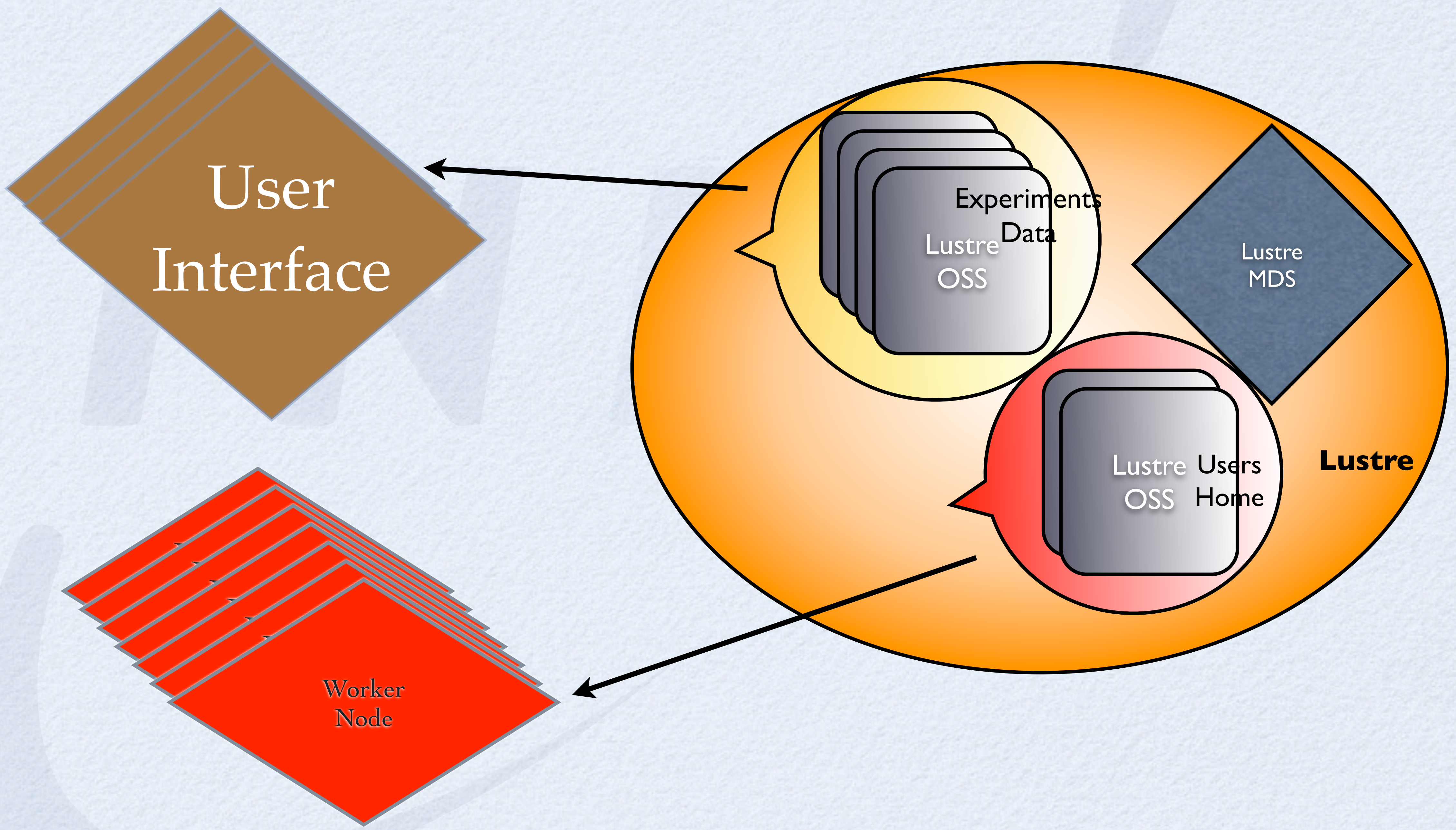
# LUSTRE 1.8.2

- 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 one of few 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

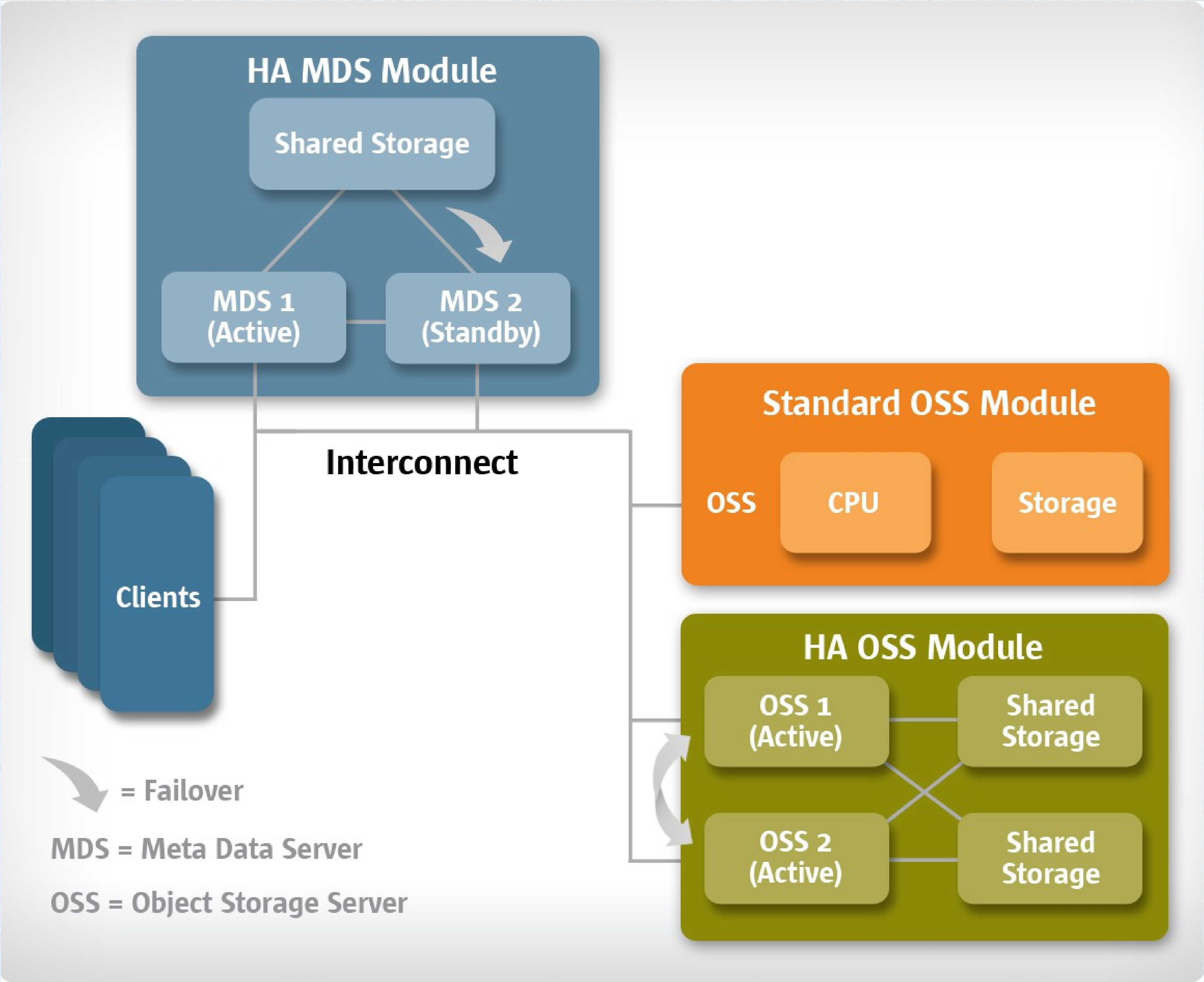
# Lustre 1.8.2

- **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 1.8.2 -- Example



# LUSTRE -- HA AND HP

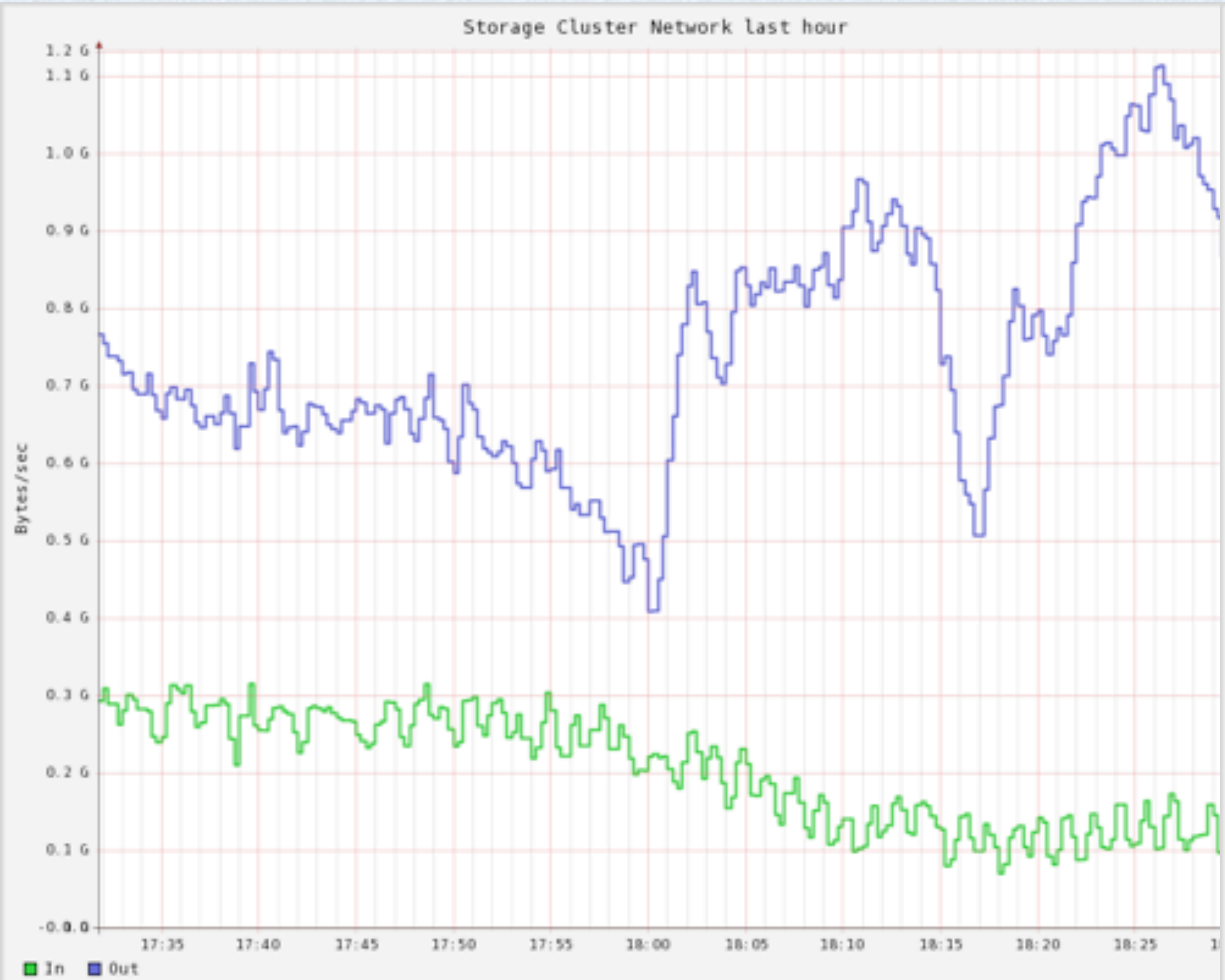


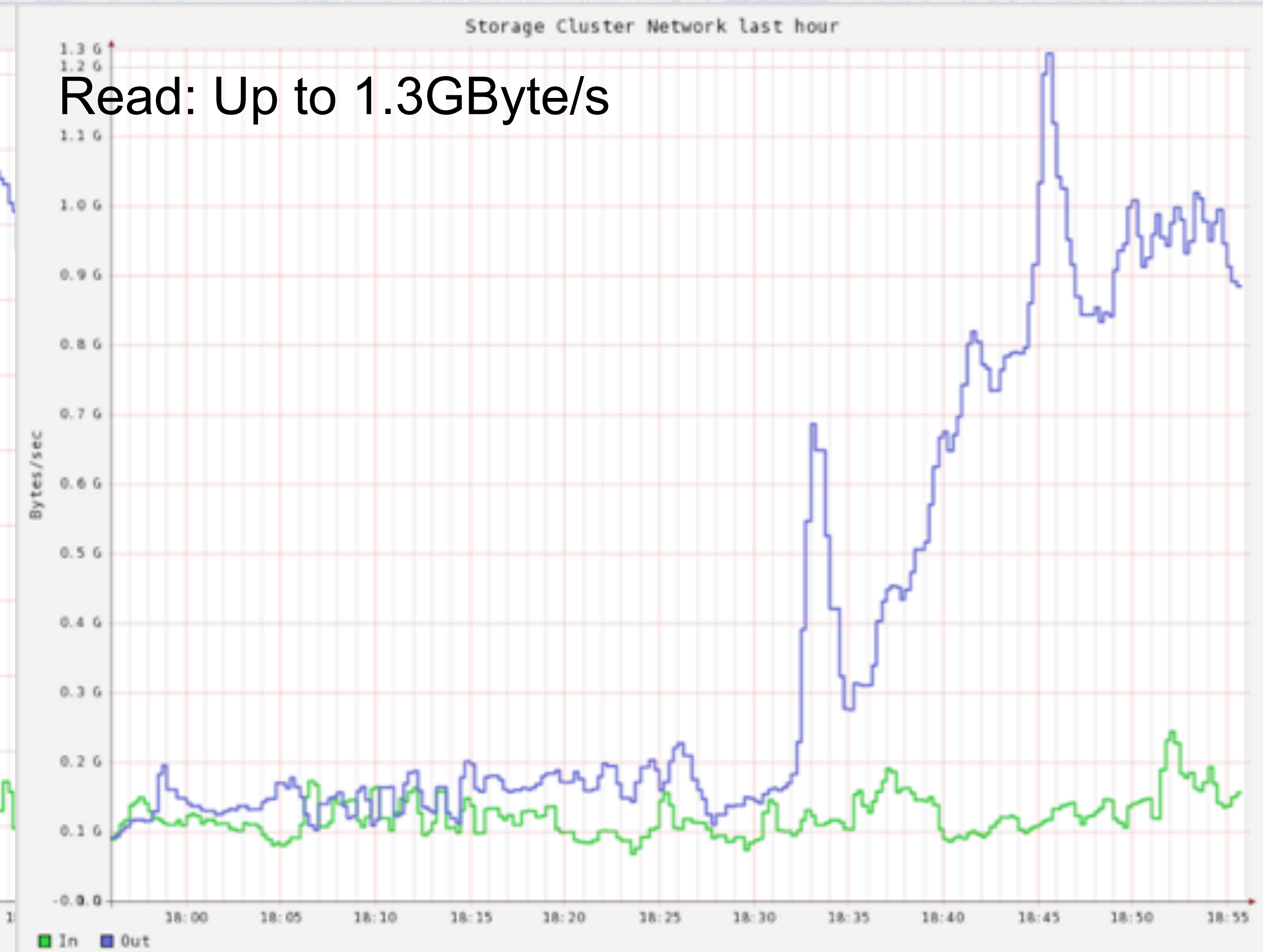
# HEP TIER2

250TB

10 server

800 concurrent jobs




- It is possible to use the file system to run job hosting both input and output files
- The rate are measured with real “root” analysis jobs.
- SRM/gridftp layer provided by StoRM

# Lustre Future

- ZFS back-end support:
  - end-to-end data integrity
  - SSD read cache
- 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

# 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

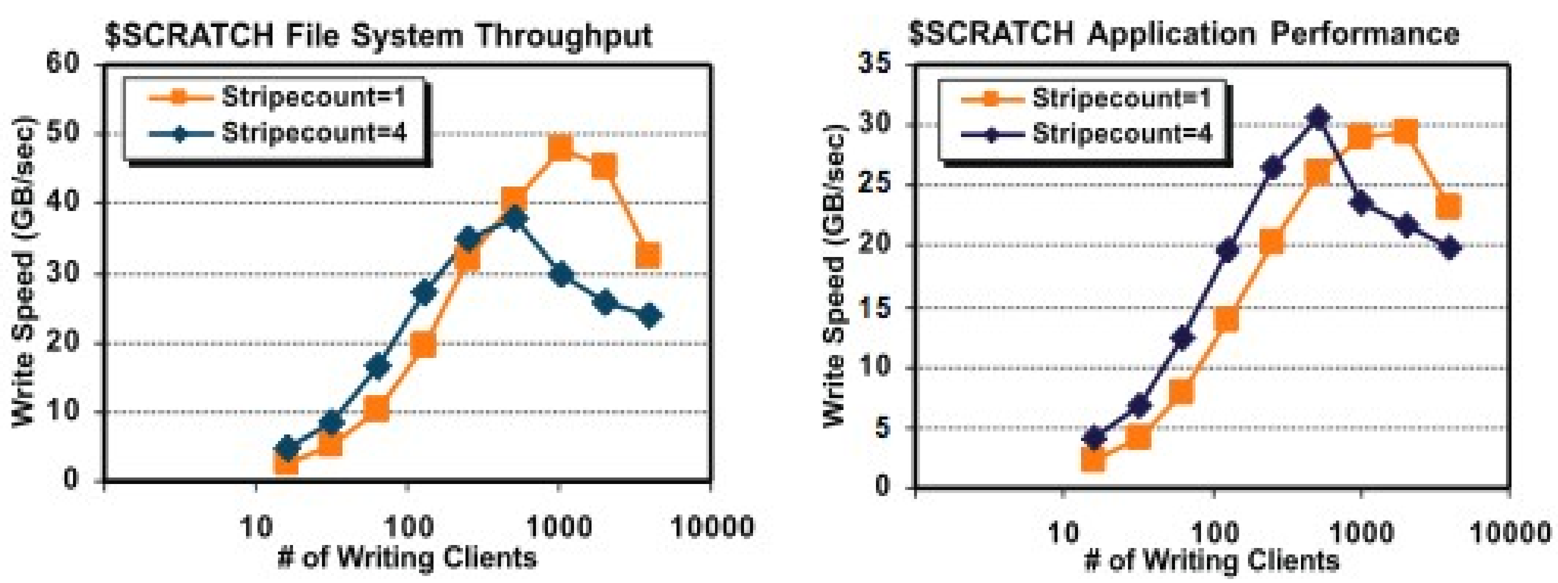



*Figure 12. Lustre file system performance at TACC.*

- "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."

# 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

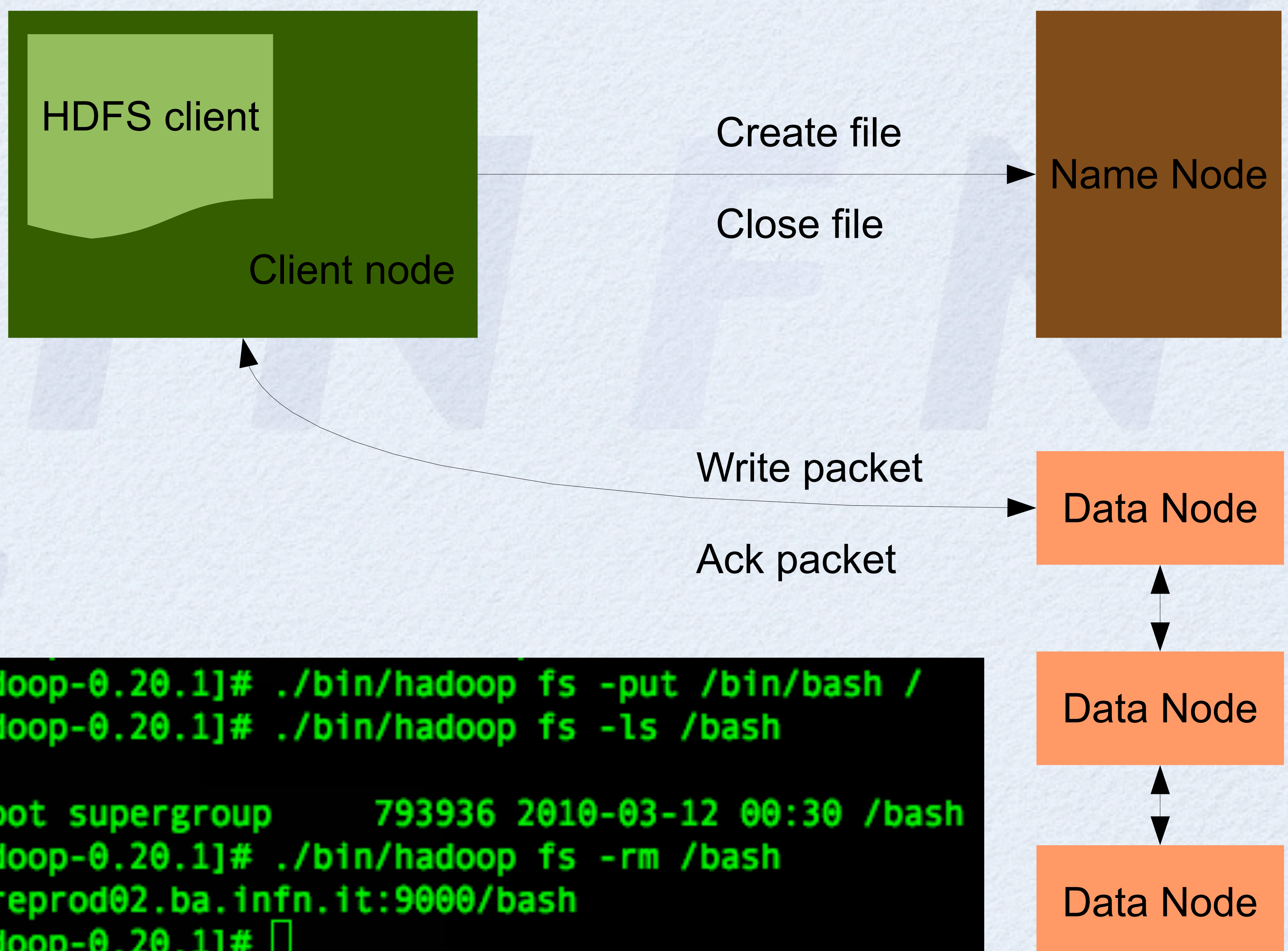
- A9.com
- AOL
- Booz Allen Hamilton
- EHarmony
- Facebook
- Freebase
- Fox Interactive Media
- IBM
- ImageShack
- ISI
- Joost
- Last.fm
- LinkedIn
- Metaweb
- Meebo
- Ning
- Powerset (now part of Microsoft)
- Proteus Technologies
- The New York Times
- Rackspace
- Veoh
- Twitter

# Hadoop: highlight

- Using FUSE => some posix call supported
  - Basically “all read operation” and only “serial write operations”
- Web interface to monitor the HDFS system
- Java APIs to build code 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

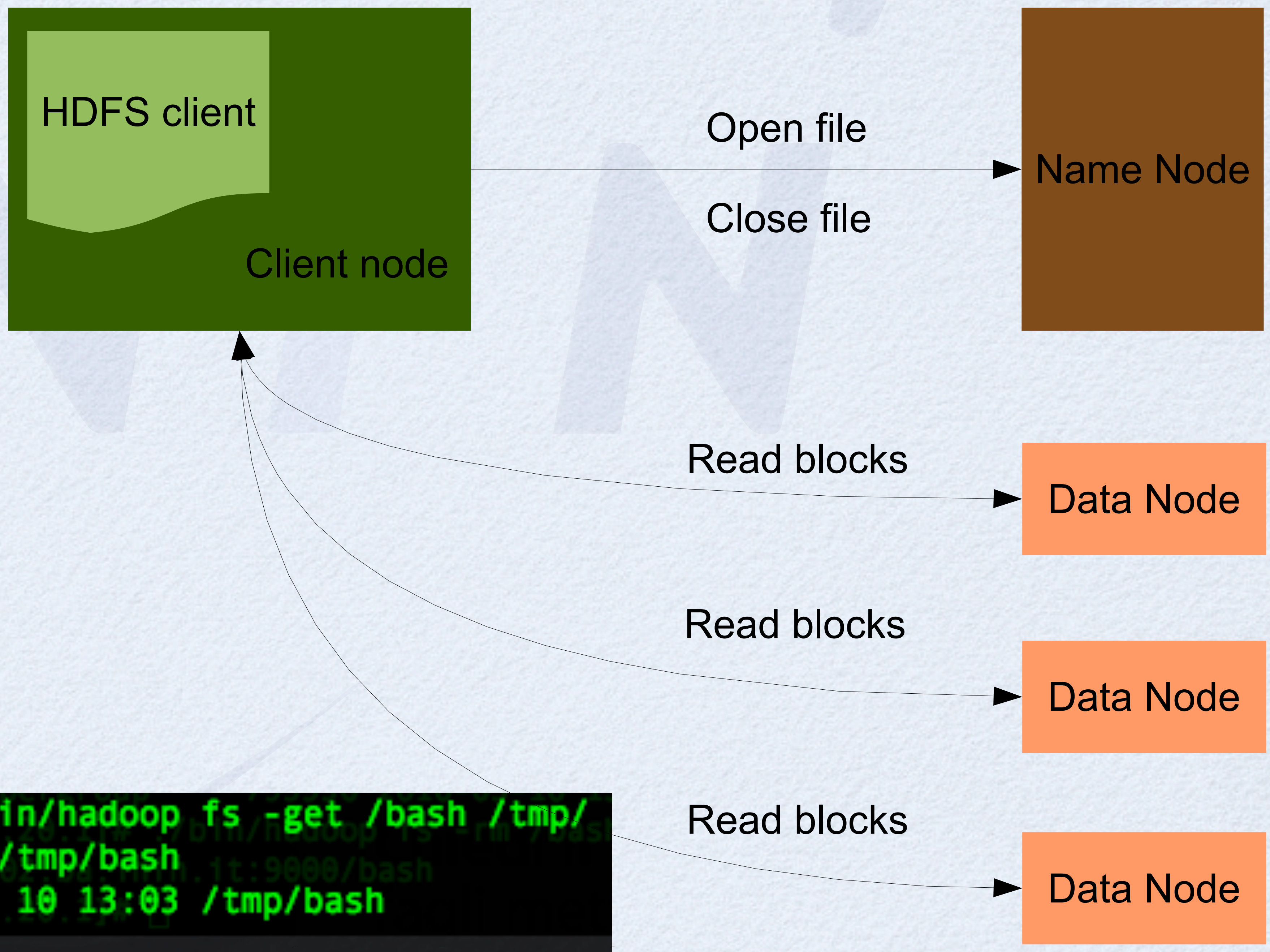



```
[root@pccms64 hadoop-0.20.1]# ./bin/hadoop fs -put /bin/bash /
[root@pccms64 hadoop-0.20.1]# ./bin/hadoop fs -ls /bash
Found 1 items
-rw-r--r--   3 root supergroup     793936 2010-03-12 00:30 /bash
[root@pccms64 hadoop-0.20.1]# ./bin/hadoop fs -rm /bash
Deleted hdfs://preprod02.ba.infn.it:9000/bash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

# Hadoop: Concepts and Architecture

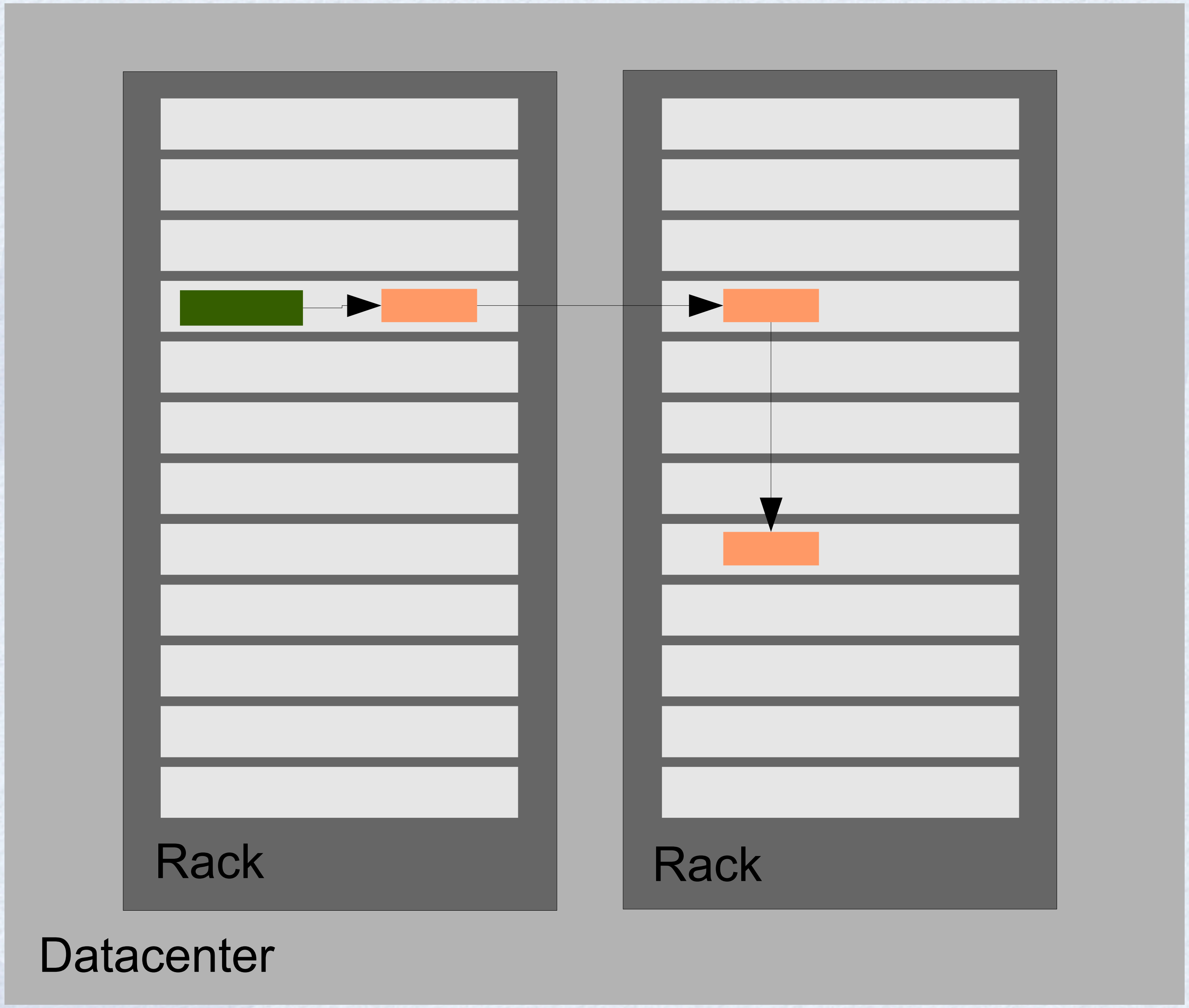
## Anatomy of a file read

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

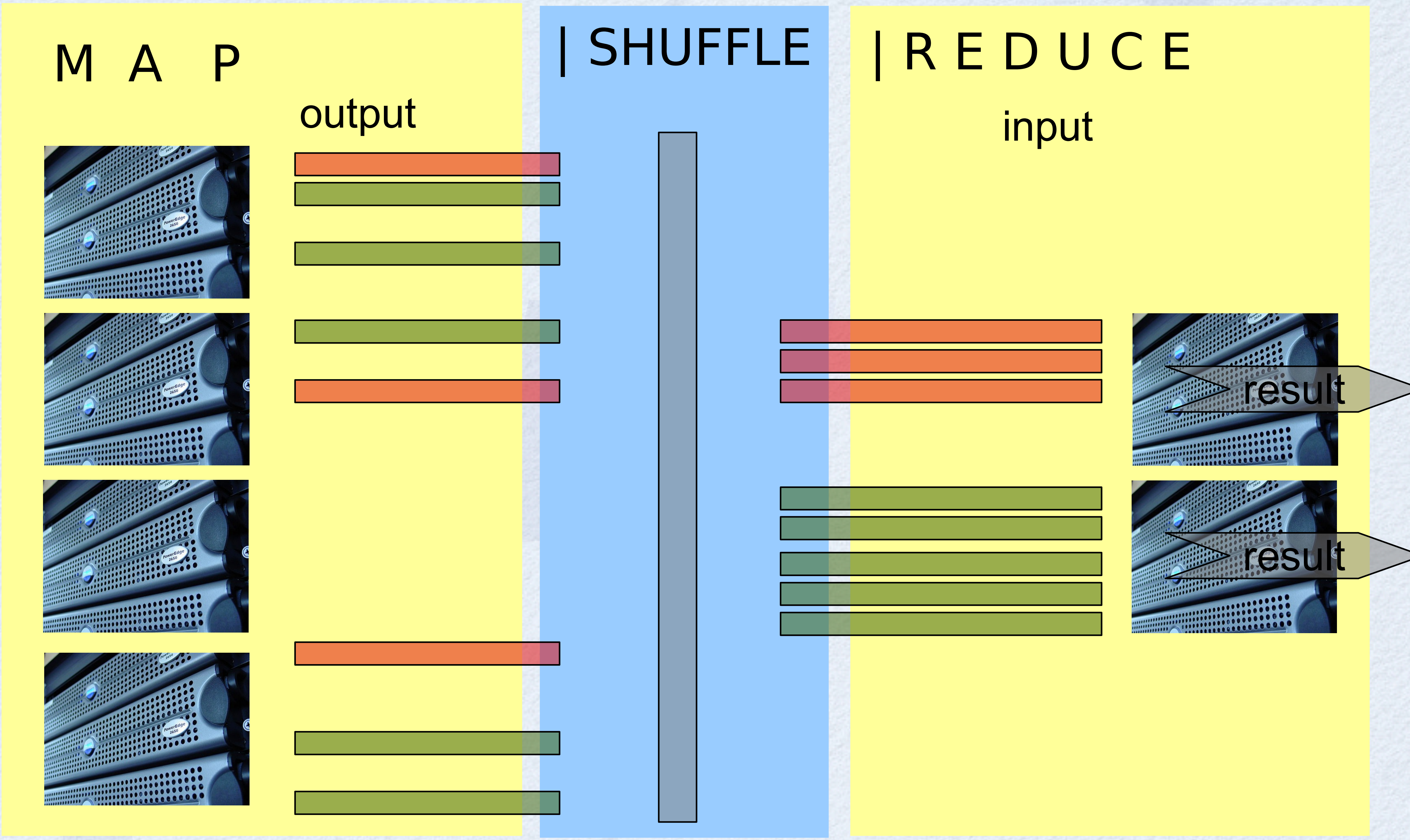



```
[root@pccms64 hadoop-0.20.1]# ./bin/hadoop fs -get /bash /tmp/
[root@pccms64 hadoop-0.20.1]# ll /tmp/bash
-rw-r--r-- 1 root root 793936 Mar 10 13:03 /tmp/bash
[root@pccms64 hadoop-0.20.1]#
```

# 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 | 2 | 173 minutes |
| 1,000,000,000,000,000 | 3658 | 2 | 975 minutes |

10x data
~6x time

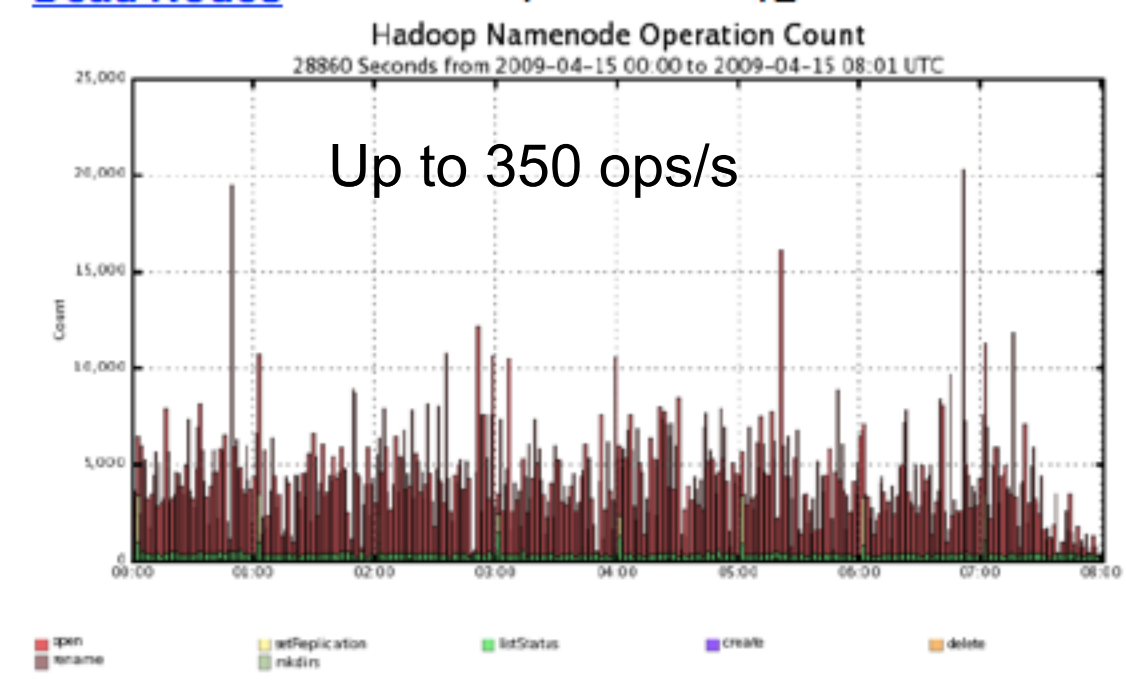
Per node: 2 quad core Xeons @ 2.5ghz, 4 SATA disks, 8G RAM (upgraded to
16GB before petabyte sort), 1 gigabit ethernet.
Per Rack: 40 nodes, 8 gigabit ethernet uplinks.

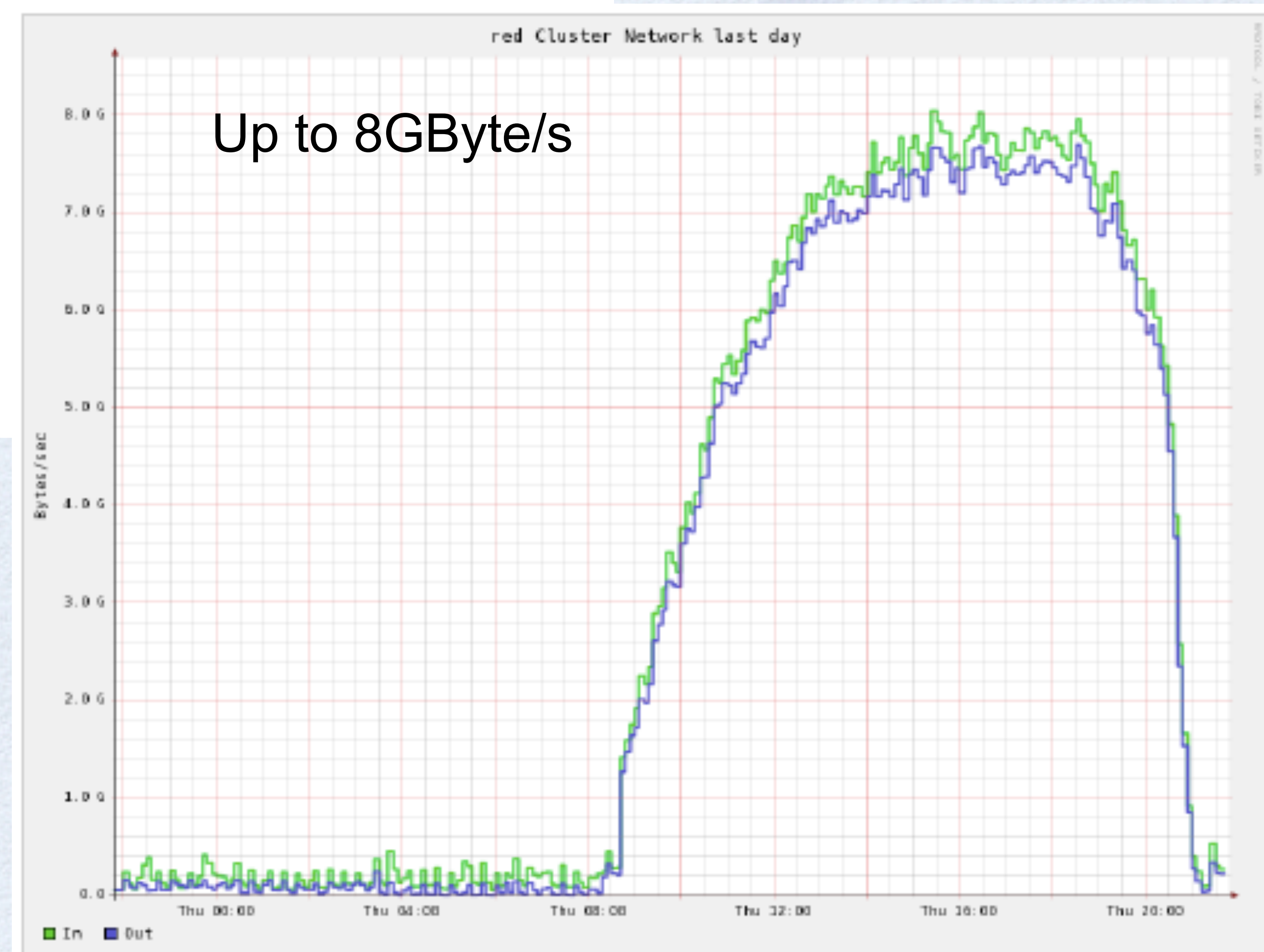
# Hadoop: few examples "CMS example"

**Cluster Summary**

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

| | | |
|---|---|---|
| Configured Capacity | : | 659.98 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 |




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

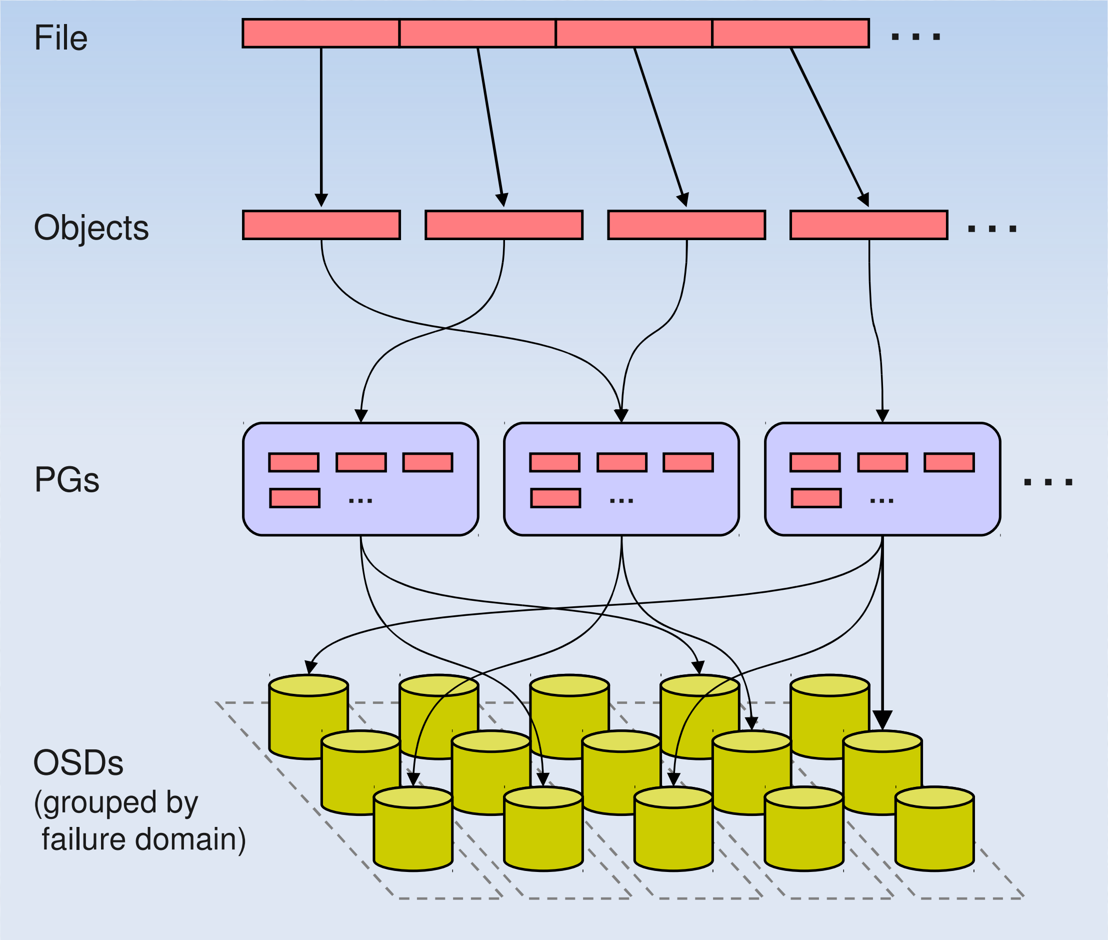
# HADOOP: FUTURE

- Support for "append"
- Support for "sync" operation
- Cluster NameNode

# CEPH

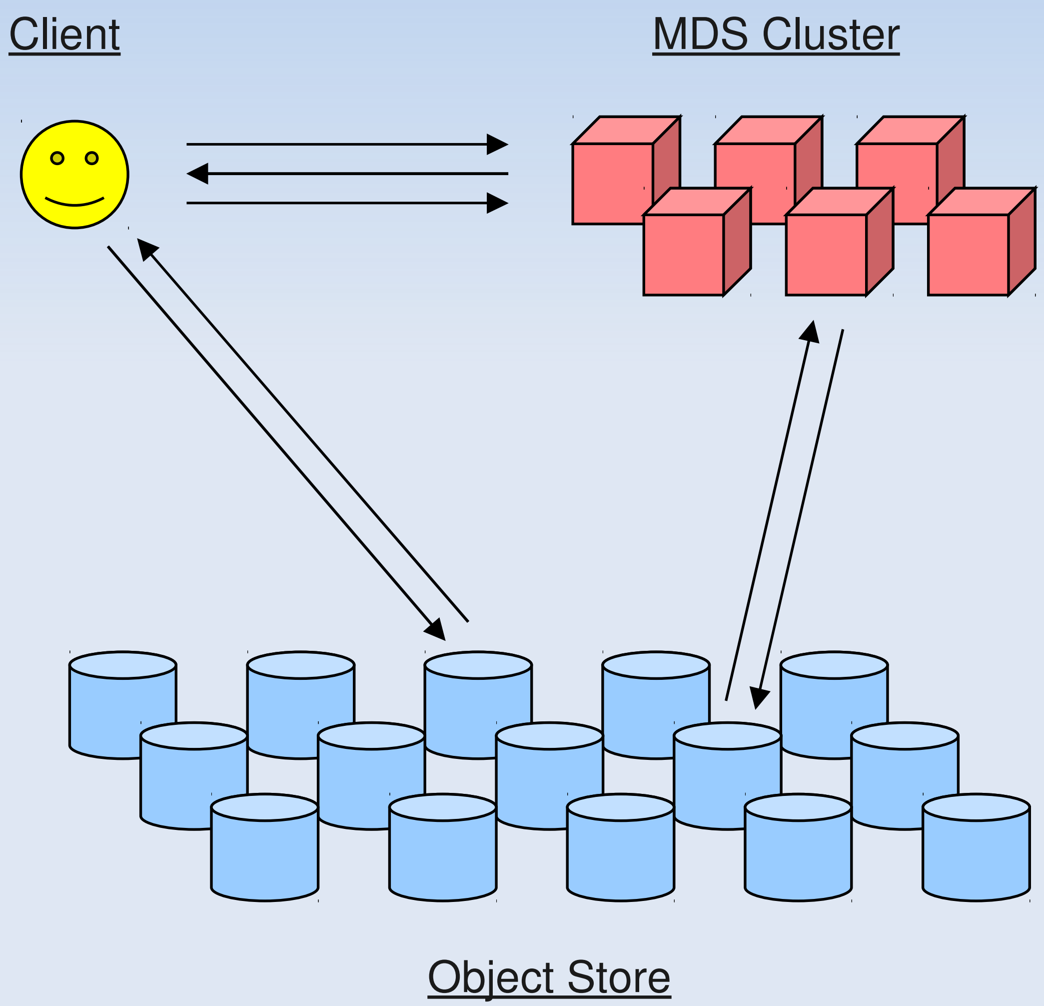
# CEPH: CONCEPT AND ARCHITECTURE

- Designed to be scalable, reliable, fast
  - avoid SPOF
  - avoid shared disk (SAN, etc => too expensive)
- Data Placement is realized by means of “hash functions”:
  - Location of data is calculated => no lookup tables
    - this means: unstable mapping and adding disk servers means reshuffling
    - “Rules” driven by replica: “three replica should be in different cabinet”



# Ceph: Concept and Architecture

- fd=open("/foo/bar", O_RDONLY)
  - Client: requests open from MDS
  - MDS: reads directory /foo from object store
  - MDS: issues capability for file content
- read(fd, buf, 1024)
  - Client: reads data from object store
- close(fd)
  - Client: relinquishes capability to MDS



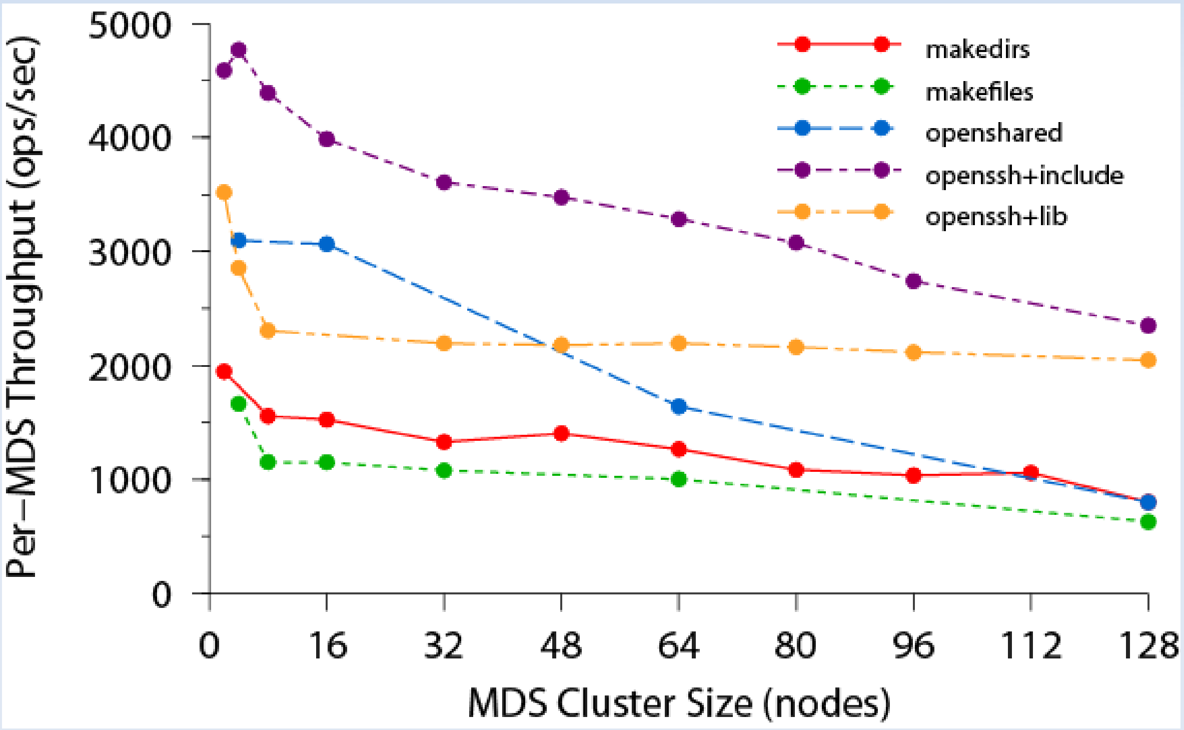

  - MDS out of I/O path
  - Object locations are well known–calculated from object name

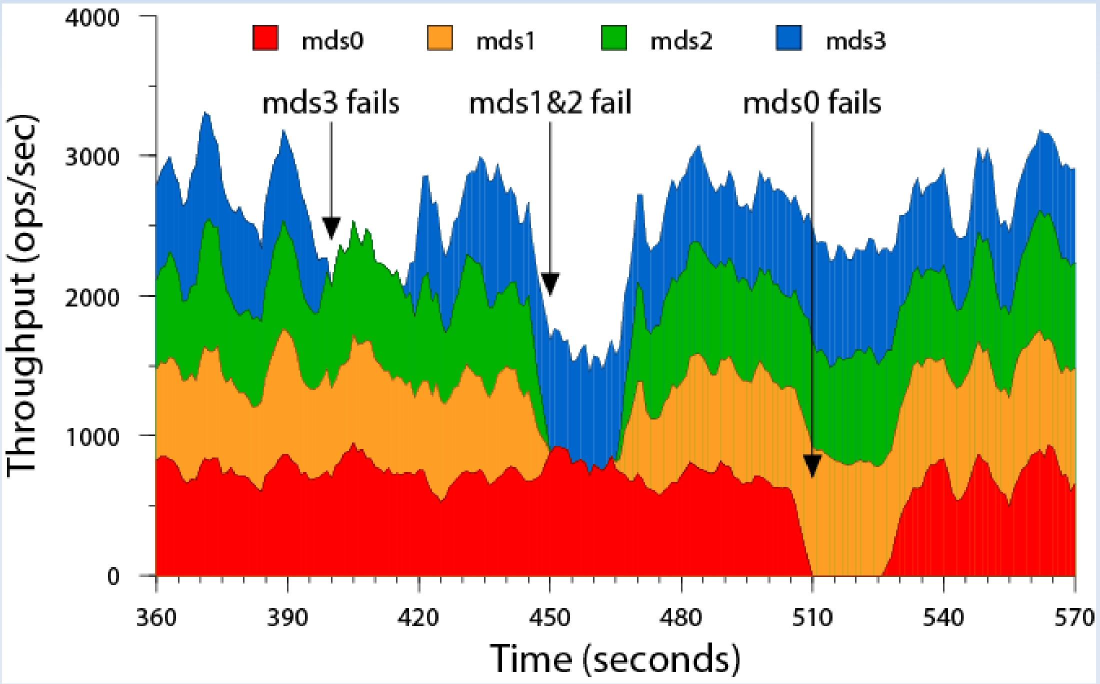
# Ceph: concept and architecture

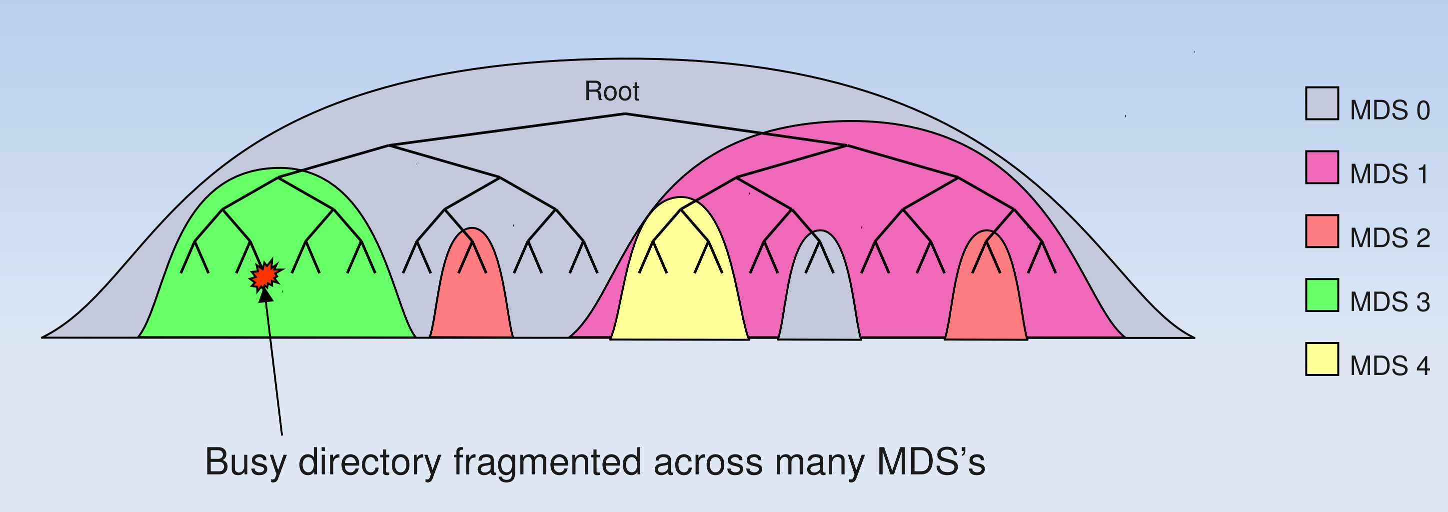
- Intelligent server: replicate data, migrate object, detect node failures
  - this could happen because everyone know where object belongs
- inodes are stored together with the directory object: you can load complete directory and inodes with a single I/O ("find" or "du" are greatly faster)
- It is easy to build a cluster of metadata servers (MDS)
  - Than it is scalable and adaptive
    - The work is moved from busy servers to idle ones

# Ceph: Concept and Architecture

- Up to 128 MDS nodes and 250kops/s
- I/O rates of potentially many TB/s
- File system containing many petabytes of storage



# Ceph: concept and architecture

Overall, things are looking good. If you've been standing on the sidelines waiting for something more stable to test, now is a good time to try things out. There are some lingering OSD performance problems (see below), and we are still a long ways off from something we would recommend for use in a production environment, but otherwise this release is looking pretty good for evaluation purposes.

- Subtree based usage accounting (half the work of a quota system)
- Near-posix, strong consistency
- Support snapshots
- kernel > 2.6.25 is required
  - or is there a FUSE client

```
$ ls -al
drwx------ 1 root root 5438384 Oct 20 14:51 ./
drwx------ 1 root root 5438387 Oct 20 14:51 ../
drwxr-xr-x 1 root root 2342034 Apr 20  2009 ghostscript/
drwxr-xr-x 1 root root  276961 Apr 20  2009 libthai/
drwx------ 1 root root 2817666 Oct 20 14:51 python-support/
drwxr-xr-x 1 root root    1723 Apr 20  2009 readline/
```

# Ceph: Future Work

- Focus on:
  - OSD performance
  - Stability
  - Reliability
  - Cluster MDS

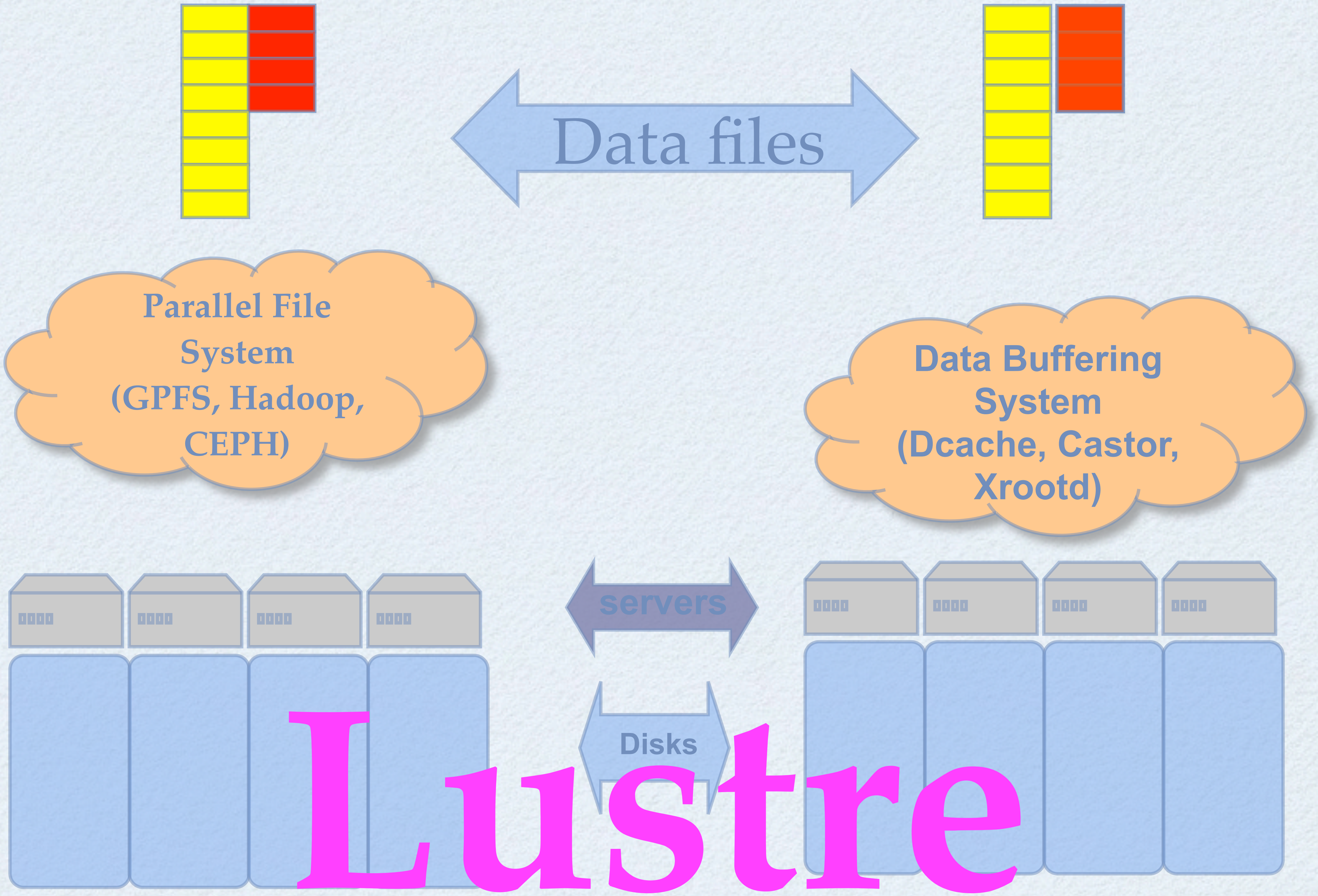
# Conclusions

| | Lustre | Hadoop | Ceph |
|---|---|---|---|
| Posix Functionalities | **True** | **Partially** | **Partially** |
| Quota | **True** | **Directory Quota** | **Not enforced** |
| Data Replica | **Not easy** | **True** | **True** |
| Metadata Replica | **Not natively** | **Not natively** | **True** |
| Resilient on SPOF | **Not natively** | **Not natively** | **True** |
| Management Cost | **Low** | **Could be costly** | **Could be costly** |
| Platform Supported | **SLC4/5 - Suse Linux** | **Every Platform** | **Debian - Suse Linux** |
| Installation procedure | **Easy** | **Quite easy** | **Not so easy** |
| Doc/Support | **Good** | **Quite good** | **Need to be improved** |
| Hep experience | **Fairly good** | **Just starting now** | **No experience** |

# Conclusions

- Lustre born in the HPC environment 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
  - not fully posix compliant
  - Is not easy to use MapReduce on HEP code, it could be an interesting development?
- CEPH is based on very good ideas and it could become a good option if it proves the needed stability and reliability

# Backup slides


Data files
Parallel File System (GPFS, Hadoop, CEPH)
Data Buffering System (Dcache, Castor, Xrootd)
servers
Disks
Lustre

# Lustre -- Installation

```
# rpm -ivh lustreldiskfs-3.0.6-2.6.9_67.0.22.EL_lustre.1.6.6smp.i686.rpm
lustre-modules-1.6.6-2.6.9_67.0.22.EL_lustre.1.6.6smp.i686.rpm kernel-lustre-
smp-2.6.9-67.0.22.EL_lustre.1.6.6.i686.rpm lustre-1.6.6-2.6.9_67.0.22.EL_lustre.
1.6.6smp.i686.rpm e2fsprogs-1.40.11.sun1-0redhat.i386.rpm
#!!!!!!reboot!!!!!!
# mkfs.lustre --fsname=lustre --mdt --mgs /dev/sdb1
# mkdir -p /mnt/test/mdt
# mount -t lustre /dev/sdb1  /mnt/test/mdt
# cat /proc/fs/lustre/devices
# mkfs.lustre --fsname lustre --ost --mgsnode=${mdt_server}@tcp0 /dev/sdc
# mkdir -p /mnt/test/ost0
# mount -t lustre /dev/sdc  /mnt/test/ost0
# mkdir /lustre
# mount -t lustre ${mdt_server}@tcp0,1@elan:/lustre /lustre
```

# Lustre -- few CLI example

```
# lfs df  [-i]
UUID                 1K-blocks      Used Available  Use% Mounted on
lustre-MDT0000_UUID   27226500   1950044  23720488    7%  /lustre[MDT:0]
lustre-OST0000_UUID  2884113492 1310544468 1427064248   45%  /lustre[OST:0]
lustre-OST0001_UUID  2402260432 1104465044 1175765056   45%  /lustre[OST:1]
......
filesystem summary:  201971633276 91551804884 100160181456   45%  /lustre

# echo '64' >  /proc/fs/lustre/llite/*/max_cached_mb
# echo '64' >  /proc/sys/lustre/max_dirty_mb

# lfs setstripe -c 10 -p pool_name -d  /lustre/directory
# lfs getstripe [-r]  /lustre/directory [/lustre/directory/file1]
# lfs getstripe -r --obd lustre-OST004e_UUID  /lustre  > /tmp/list_files

# lctl pool_new <fsname>.<poolname>
# lctl pool_add <fsname>.<poolname> <ostname indexed list>
# lctl pool_list  <fsname>[.<poolname>] | <pathname>
```

# Lustre -- few CLI example

# lfs quotaon -ug /lustre
# lfs quotacheck -ug /lustre
# lfs setquota -u [-g] <name> <block-softlimit> <block-hardlimit> <inode-softlimit> <inode-hardlimit> /lustre

# **tail -f /var/log/messages**

Dec 3 05:36:41 lustre01 kernel: LustreError: 4478:0:(import.c:909:ptlrpc_connect_interpret()) lustre-OST0048_UUID went back in time (transno 8590306664 was previously committed, server now claims 0)! See https://bugzilla.lustre.org/show_bug.cgi?id=9646

Dec 3 05:36:41 lustre01 kernel: LustreError: 4478:0:(import.c:909:ptlrpc_connect_interpret()) Skipped 1 previous similar message

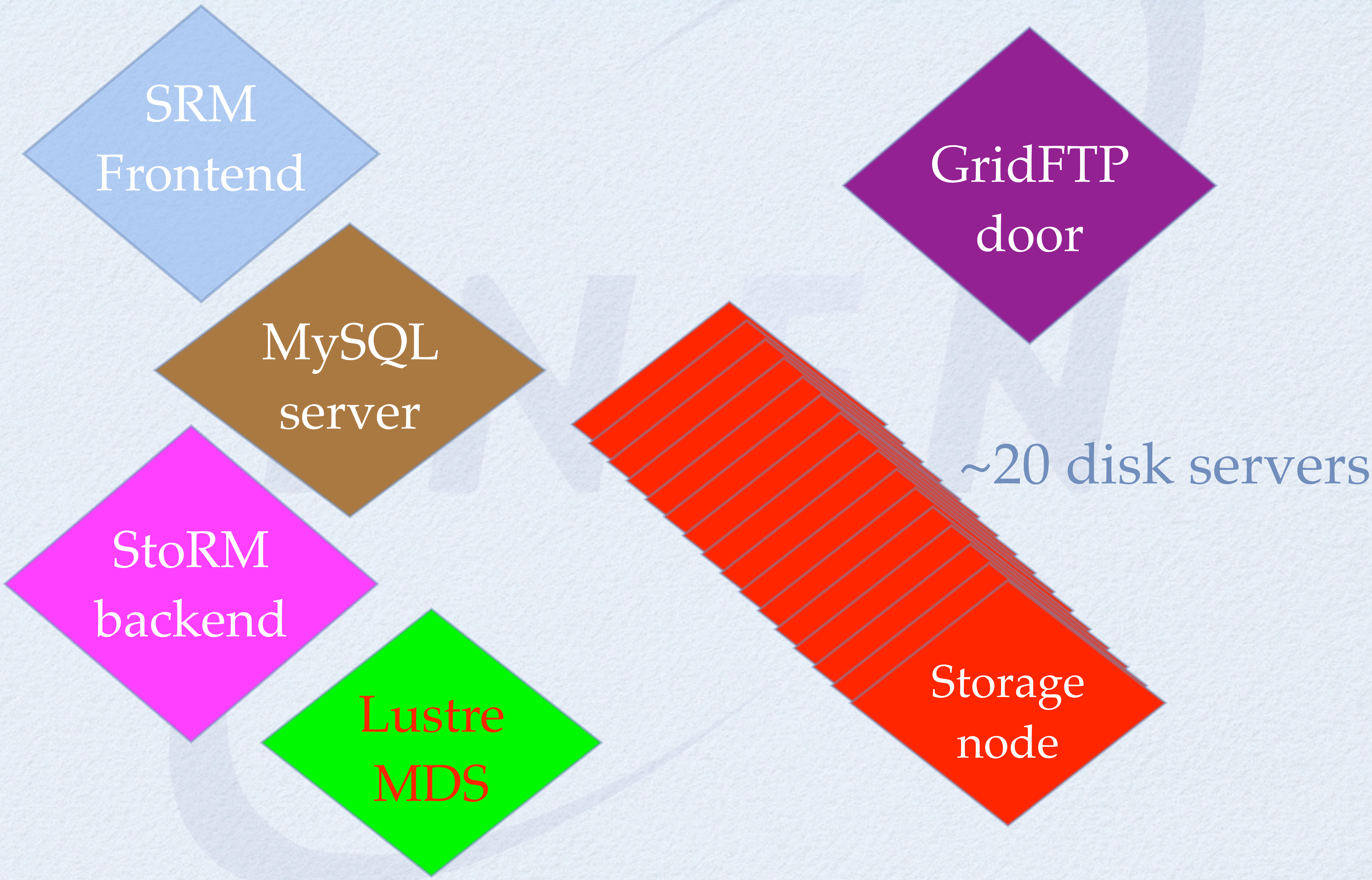
Dec 3 05:36:41 lustre01 kernel: Lustre: 4478:0:(quota_master.c:1680:mds_quota_recovery()) Only 81/79 OSTs are active, abort quota recovery

Dec 3 05:36:41 lustre01 kernel: Lustre: lustre-OST0048-osc: Connection restored to service lustre-OST0048 using nid 212.189.205.106@tcp.

# Lustre -- few CLI example

```
# pwd
/proc/fs/lustre/llite/lustre-ffff8101264f5c00
# ls
blocksize                 fstype             max_read_ahead_mb         statahead_stats
checksum_pages            kbytesavail        max_read_ahead_whole_mb  stats
contention_seconds        kbytesfree         max_rw_chunk              stats_track_gid
dump_page_cache           kbytestotal        mdc                       stats_track_pid
extents_stats             lazystatfs         offset_stats              stats_track_ppid
extents_stats_per_process  lockless_truncate  pgcache_balance          uuid
filesfree                 lov                read_ahead_stats
filestotal                max_cached_mb      statahead_max
```

# Lustre possible scenario



# Hadoop -- installation

```
# wget http://mirror.nohup.it/apache/hadoop/core/stable/
hadoop-0.20.2.tar.gz
# tar xfvz hadoop-0.20.2.tar.gz
# mkdir -p /hadoop/namenode_dir
# mkdir -p /hadoop/datanode_dir
# cd hadoop-0.20.2
(modify configuration files)
conf/hadoop-env.sh
conf/slaves
conf/hdfs-site.xml
conf/core-site.xml
conf/masters
# ./bin/hadoop namenode -format
# ./bin/start-all.sh
```

# Hadoop -- few CLI

```
# ./bin/hadoop dfsadmin -refreshNodes
# ./bin/hadoop dfsadmin -report
# ./bin/hadoop balancer
# ./bin/hadoop dfsadmin -safemode leave
# ./bin/hadoop fsck / -files
```