Emerging Technologies: Hadoop

Brian Bockelman

OSG Storage ‘09

Introducing Hadoop

- Hadoop is a data processing system that follows the MapReduce paradigm for scalable data analysis.
    - Largest install is at Yahoo, a major contributor.
    - 14PB of online disk.
    - Larger clusters are planned

Hadoop and HDFS

- To do large-scale data processing, you need an underlying file system.
- But to do this affordably, you need a distributed FS designed for commodity hardware.
  - I.e., stuff all your worker nodes full of disks.

HDFS

- HDFS is a scalable file system with two major components:
  - Namenode: central metadata server.
  - Datanode: file servers for data.
- Lots of design decisions in HDFS will look familiar to WLCG sites.

HDFS Design

- Big subject! See the Hadoop whitepapers
  - http://hadoop.apache.org/core/docs/current/hdfs_design.html
- The filesystem keeps all namespace information persisted in a journal and merges the journal once every hr or 64MB.
  - All operations that do not alter namespace are guaranteed to be RAM-only.
  - Benchmarked at 50k ops / sec for reads, 5k ops / sec for writes.
  - Central metadata trivial to back up - very important!
- HDFS is regarded as optimized for sequential reads, but does well for random reads (which is HEP's most frequent operation).
  - Block decomposition of files removes hot-spots.

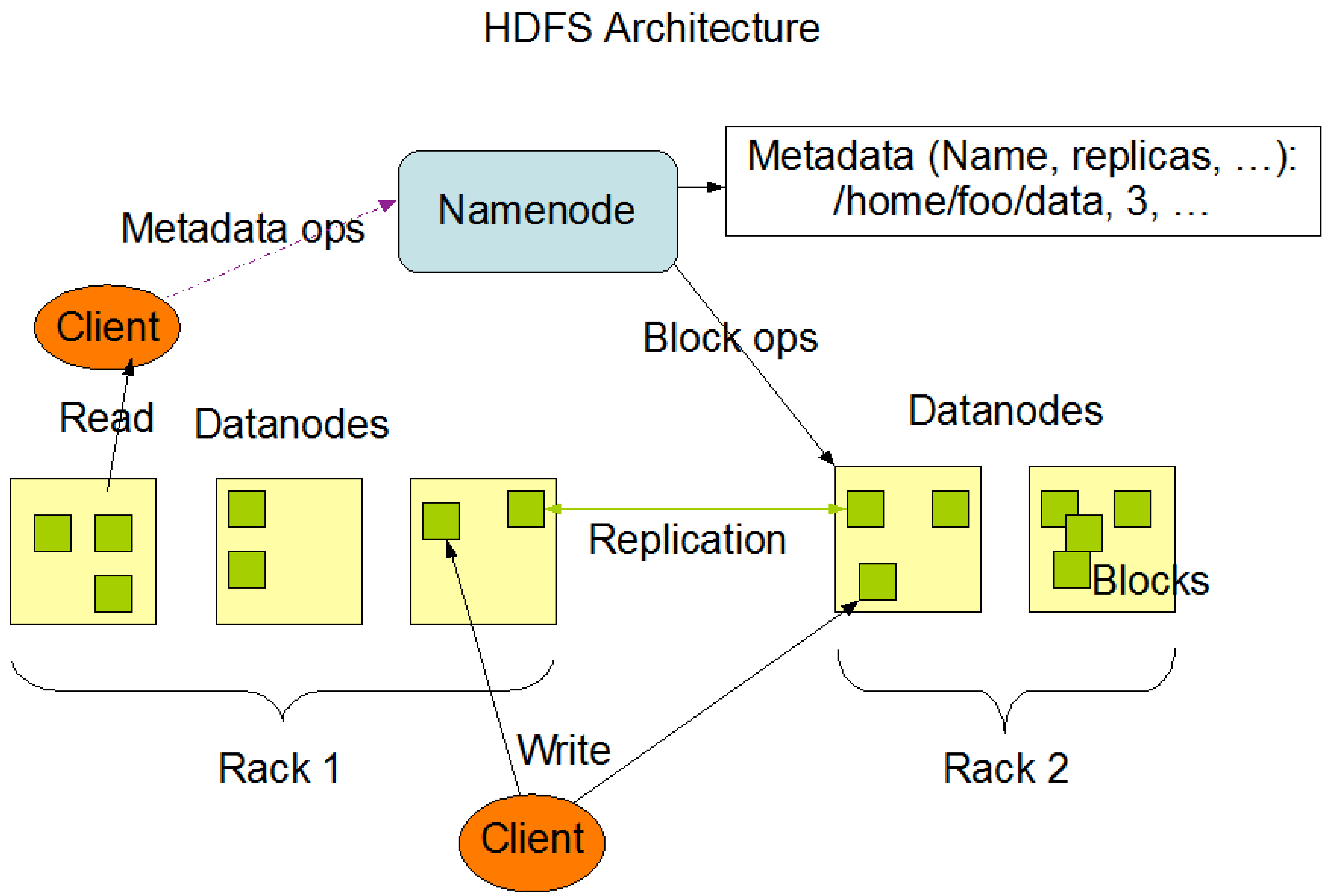
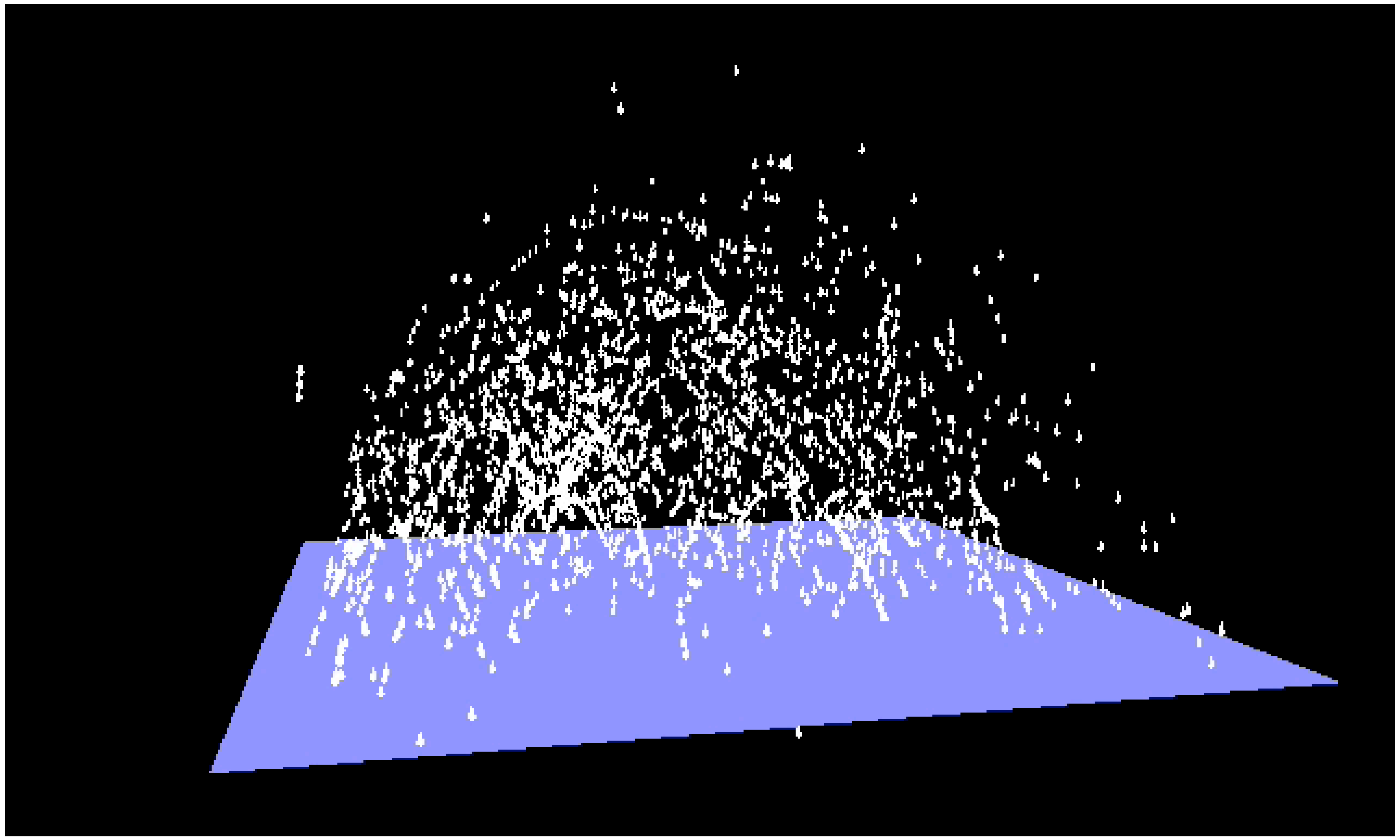


Image courtesy of Hadoop website

Hadoop Visualization

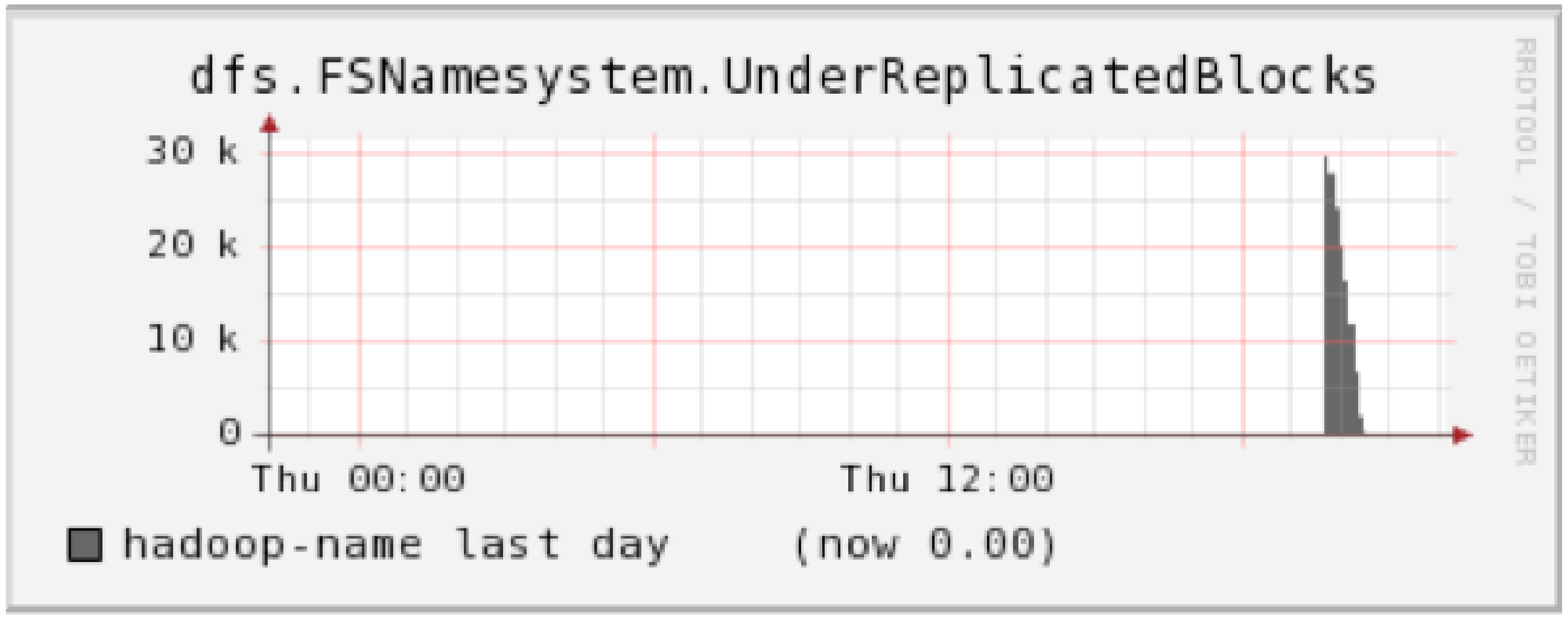


HDFS Replication

- Replication built into core of system.
- Default replication policy:
  - First replica to local datanode
  - Second replica to a node on a different rack
  - Third replica to yet another rack.

Replication Example

- Our current policy is that any node that does not have a heartbeat in 10 minutes is declared dead.
- At that point, namenode will start creating new replicas, assuming the node is dead.
- Example below: 1.5TB HDD failed; "danger zone" passed in ~ 1 hr.



Replication

- At no point when a HDD fails does a client fail!

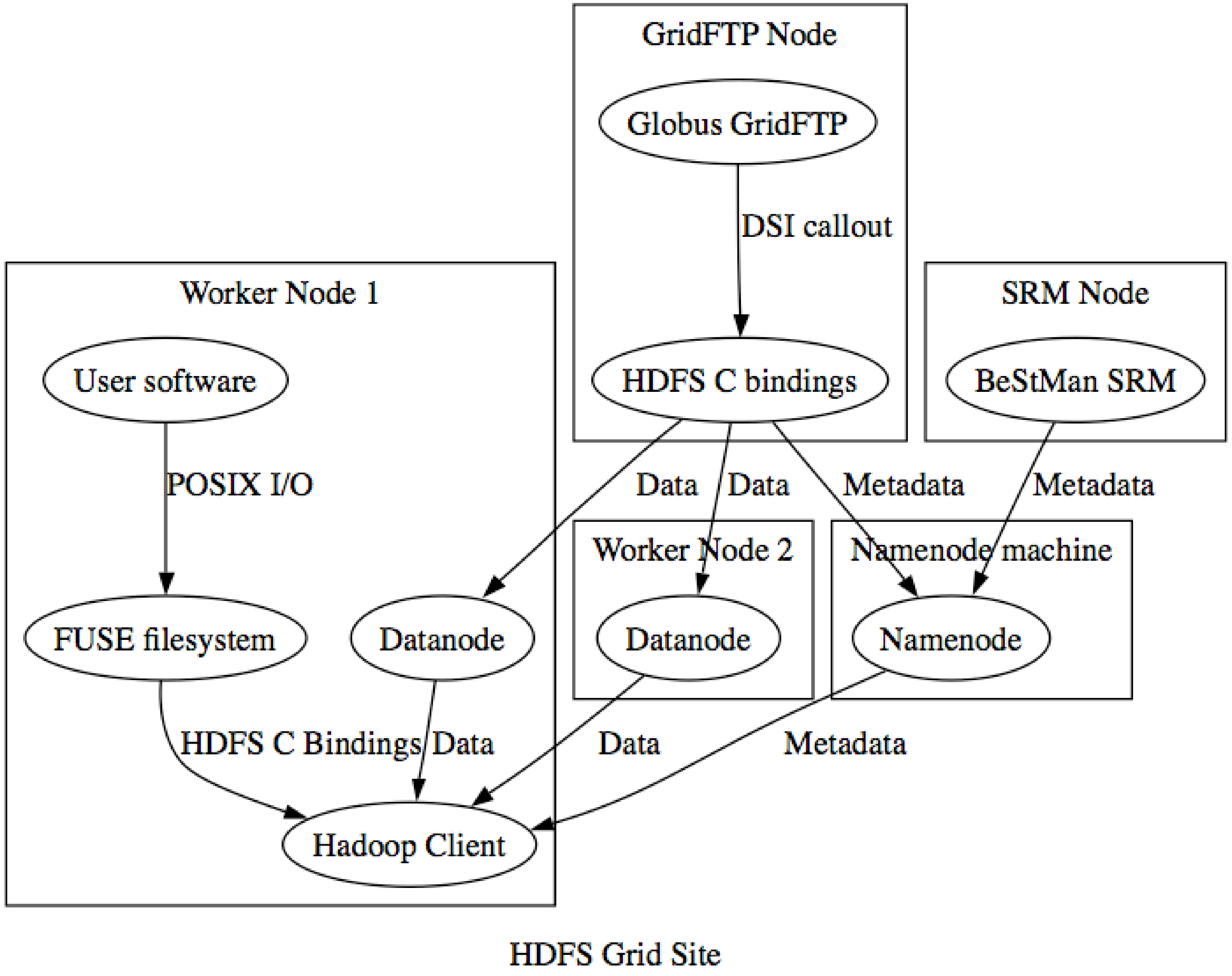
HDFS Replication

- HDFS replication allows client’s reads to survive:
  - Death of datanode currently reading from.
  - Death of namenode.

Grid-Enabling HDFS

- We combine HDFS with two grid components:
  - BestMan SRM server
  - Globus Gridftp
  - Both are well maintained & modular.
- And then we mount it on our WN for local file access.

HDFS SE Diagram

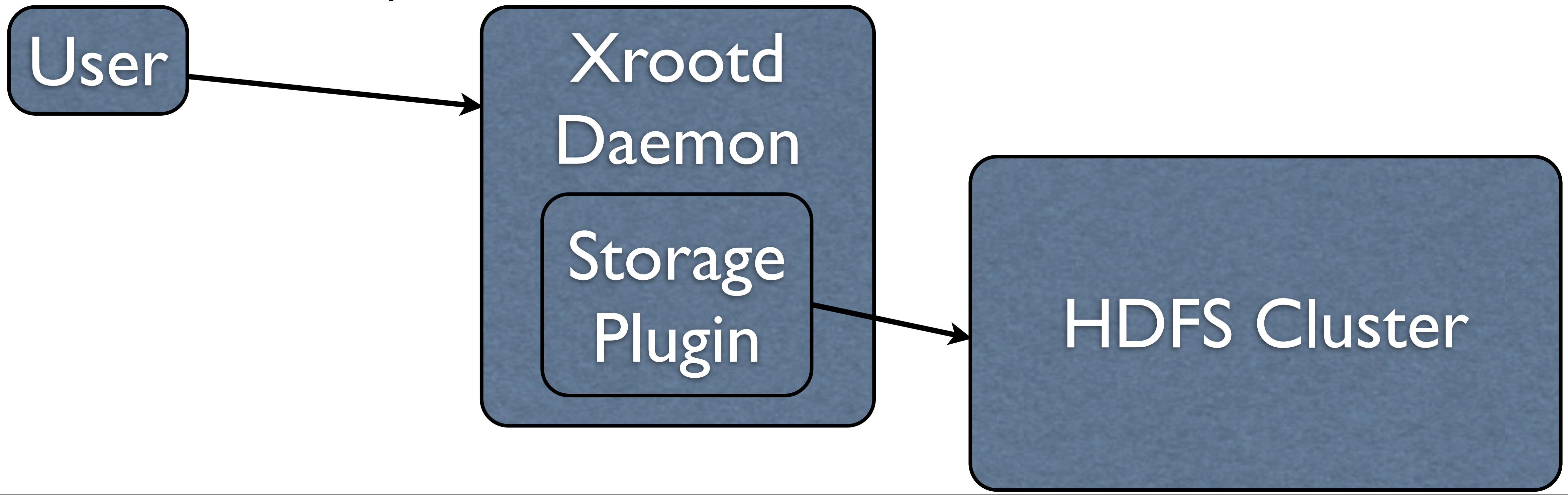


Extending Hadoop

- Because it has Java, C, and POSIX interfaces, it is easy to adopt HDFS to other protocols.
  - Using FUSE: SRM, HTTPS (with Apache)
  - Using C interfaces: GridFTP, Xrootd
  - Using Java interfaces: FDT (upcoming)
- Lesson: Good, stable APIs are more important than a specific protocol.

Xrootd/HDFS

- Get to benefit from all the nice, HEP-desired features Xrootd provides.
- Especially nice for "close" interactive access (maybe another campus accessing your cluster)



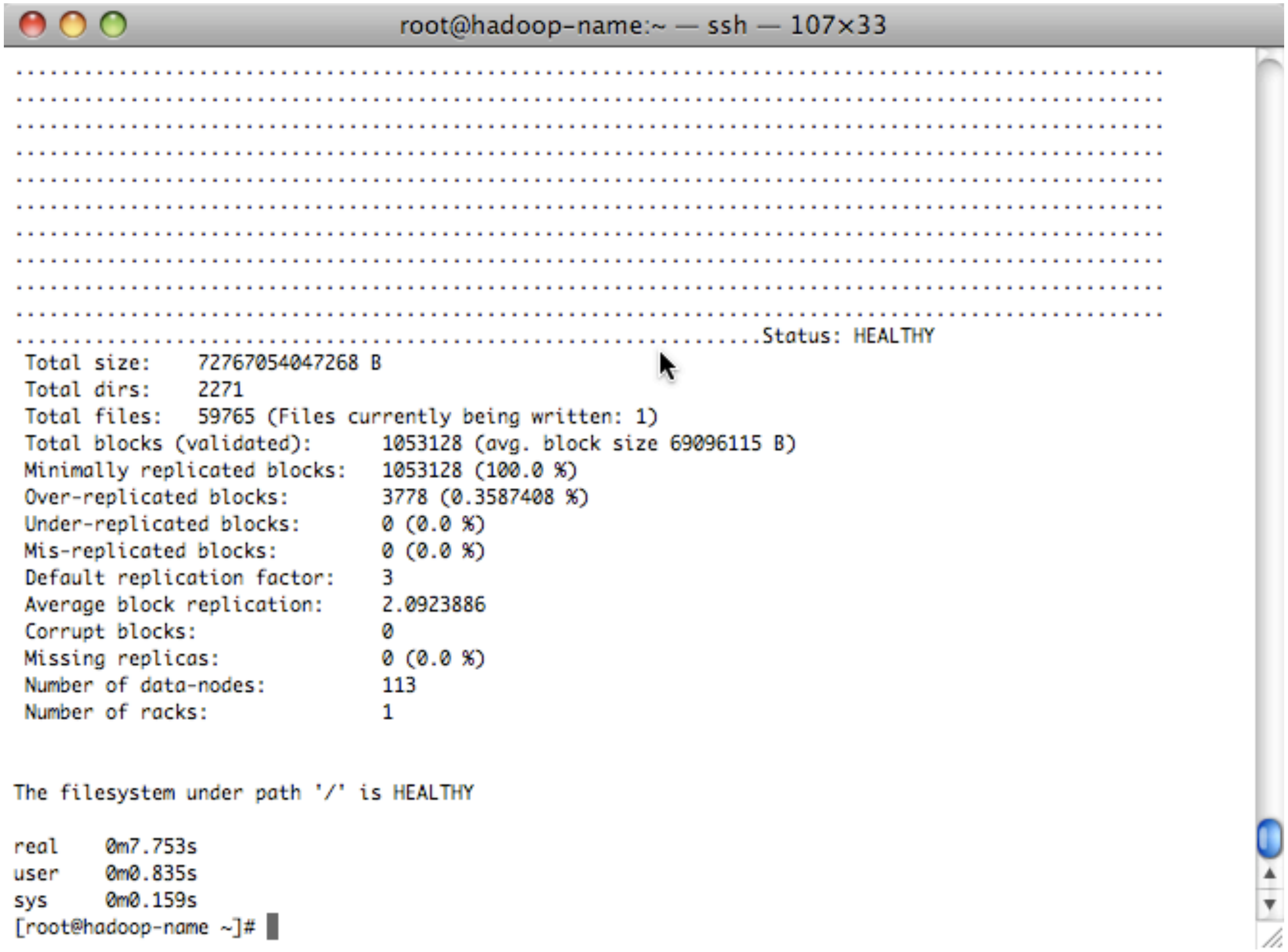
Advantages of HDFS

- In order, these are the primary drivers of our use of HDFS:
  - Manageability
  - Reliability
  - Usability
  - Scalability

Manageability

- The following tasks are trivial:
  - Integration of statistics with **Ganglia**.
  - **Decommissioning** hardware.
  - **Recovery** from hardware failure.
  - **Fsck**!
    - Checks the current knowledge of the filesystem and counts how many block replicas there are per file, and highlights any which are under-replicated.
  - **RPM** and Pacman-based install for the whole kit (including Grid components).
  - Many of our “well-known” problems are not possible.
    - **Don’t need a separate admin toolkit**! (one gremlin)
  - Setting **quotas**.
  - **Backups**.
  - **Balancer** is included.

FSCK example



```
........................................................................................................
........................................................................................................
........................................................................................................
........................................................................................................
........................................................................................................
........................................................................................................
........................................................................................................
........................................................................................................
........................................................................................................
........................................................................................................
...................................................................Status: HEALTHY
 Total size:    72767054047268 B
 Total dirs:    2271
 Total files:   59765 (Files currently being written: 1)
 Total blocks (validated):      1053128 (avg. block size 69096115 B)
 Minimally replicated blocks:   1053128 (100.0 %)
 Over-replicated blocks:        3778 (0.3587408 %)
 Under-replicated blocks:       0 (0.0 %)
 Mis-replicated blocks:         0 (0.0 %)
 Default replication factor:    3
 Average block replication:     2.0923886
 Corrupt blocks:                0
 Missing replicas:              0 (0.0 %)
 Number of data-nodes:          113
 Number of racks:               1


The filesystem under path '/' is HEALTHY

real    0m7.753s
user    0m0.835s
sys     0m0.159s
[root@hadoop-name ~]# 
```

FSCK Demo

- (SSH access permitting: demo of using ‘fsck’ utility on our live cluster)

Decommissioning Procedure

- To remove a node:
  - Add it to the hosts_exclude list.
  - Instruct HDFS to reload the list.
  - Verify the node is listed as "Decommissioning in Progress" on the webpage.
  - When done, node will be placed on the "dead list".

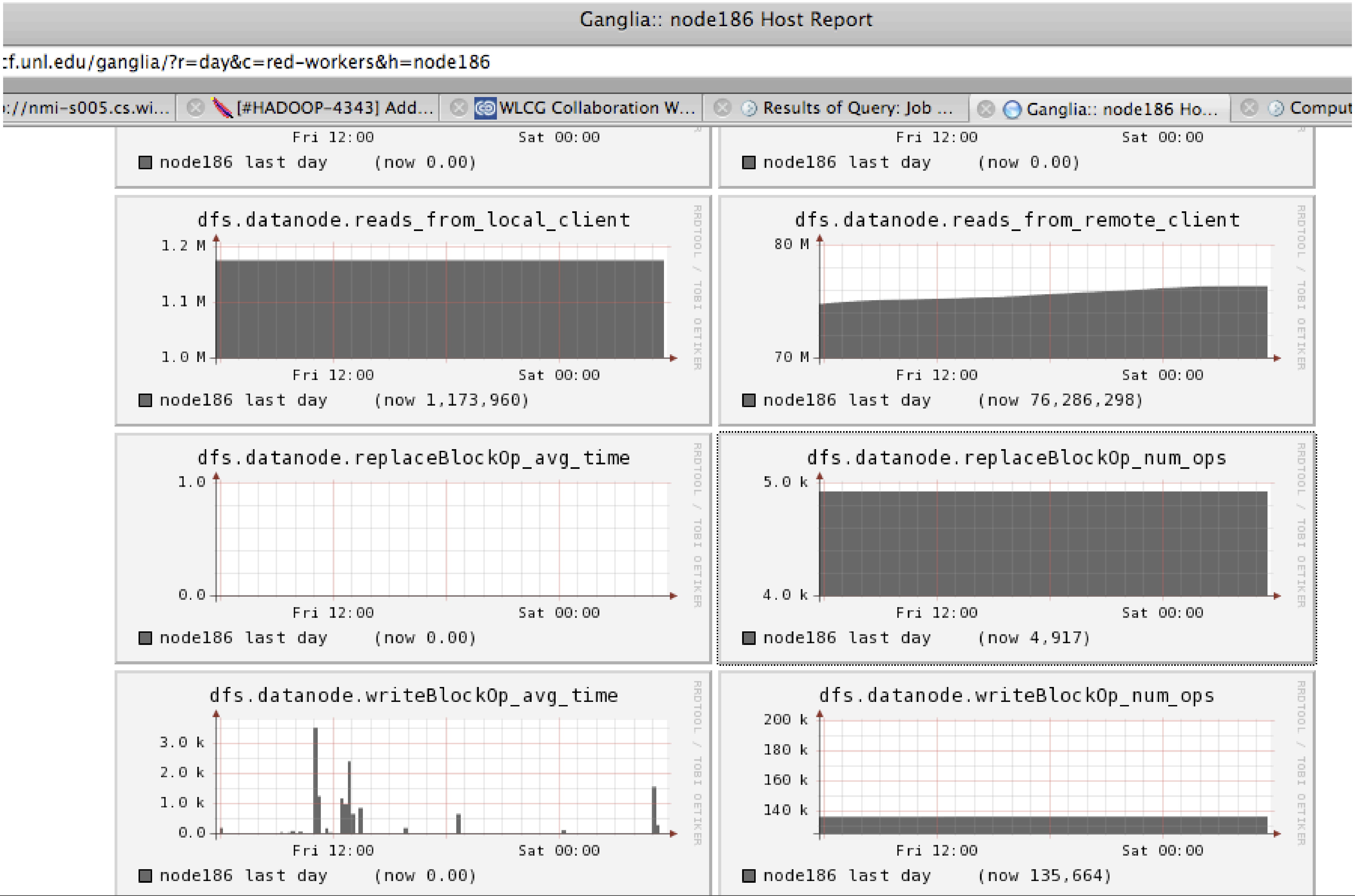
Decommissioning Demo

- (SSH access permitting: demonstration of removing a node from the cluster)

Backup Procedure

- To have automated backups created:
  - Edit /etc/sysconfig/hadoop to reflect the name of the backup server.
  - Start the daemon.
  - (Extra credit). ‘scp’ generated checkpoints offsite

Ganglia Graphs


Ganglia:: node186 Host Report
:f.unl.edu/ganglia/?r=day&c=red-workers&h=node186
://nmi-s005.cs.wi...
[#HADOOP-4343] Add...
WLCG Collaboration W...
Results of Query: Job ...
Ganglia:: node186 Ho...
Comput

Fri 12:00 Sat 00:00
■ node186 last day (now 0.00)

Fri 12:00 Sat 00:00
■ node186 last day (now 0.00)

dfs.datanode.reads_from_local_client

■ node186 last day (now 1,173,960)

dfs.datanode.reads_from_remote_client

■ node186 last day (now 76,286,298)

dfs.datanode.replaceBlockOp_avg_time

■ node186 last day (now 0.00)

dfs.datanode.replaceBlockOp_num_ops

■ node186 last day (now 4,917)

dfs.datanode.writeBlockOp_avg_time

■ node186 last day (now 0.00)

dfs.datanode.writeBlockOp_num_ops

■ node186 last day (now 135,664)

Reliability

- Replication works incredibly well.
- Client (CMSSW) reads live through restarts of any piece of HDFS.
- Writes are pipelined; guaranteed to have N copies on cluster when close() returns.
- Each datanode does a constant background checksum of all data & upon reads.

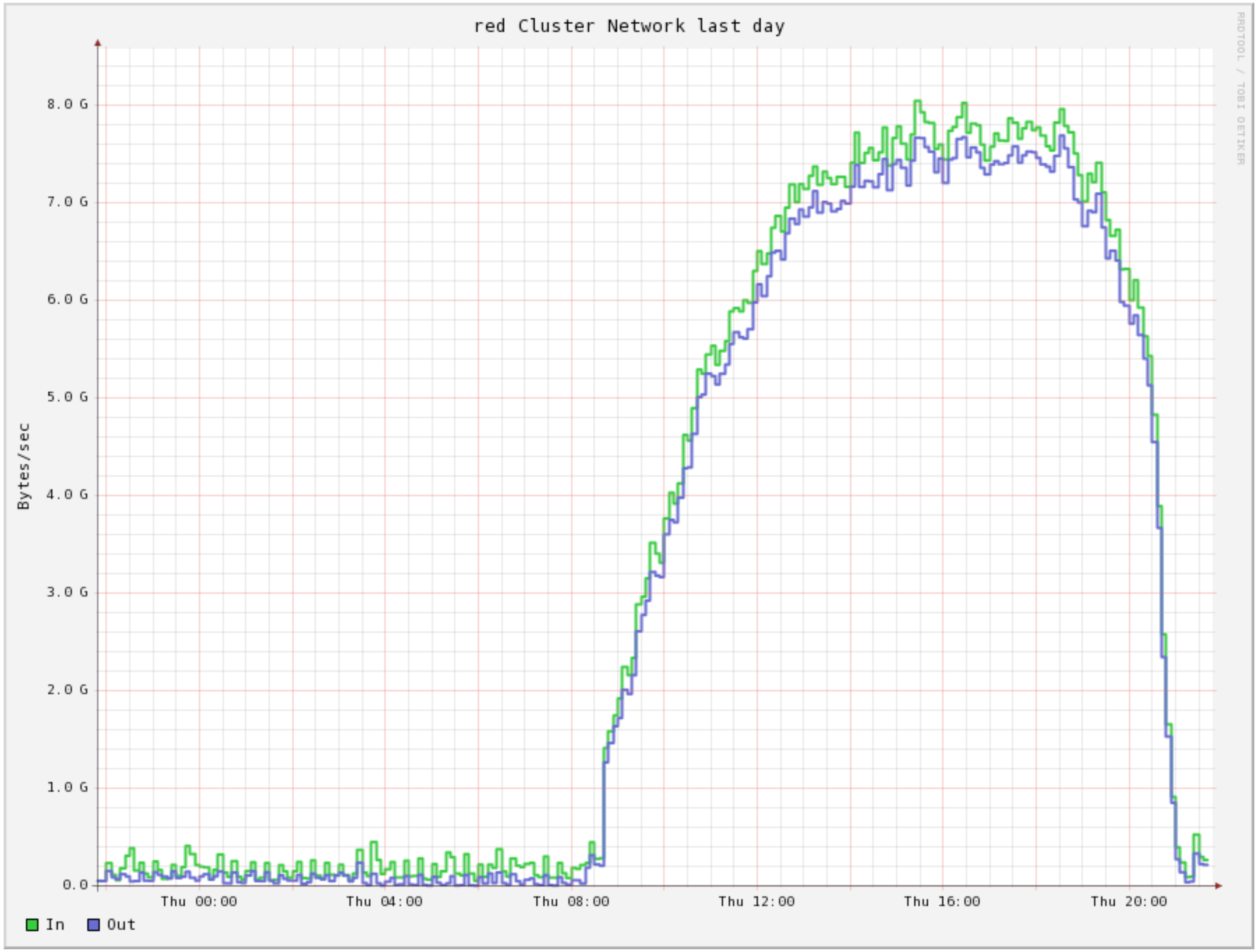
Checksums

- Can configure the cluster to checksum all data on all nodes every X days (default: X=14)
- Whenever a client reads data, the default is for it to compute checksums every Y KB (default=.5) and compare it against the datanode's record.

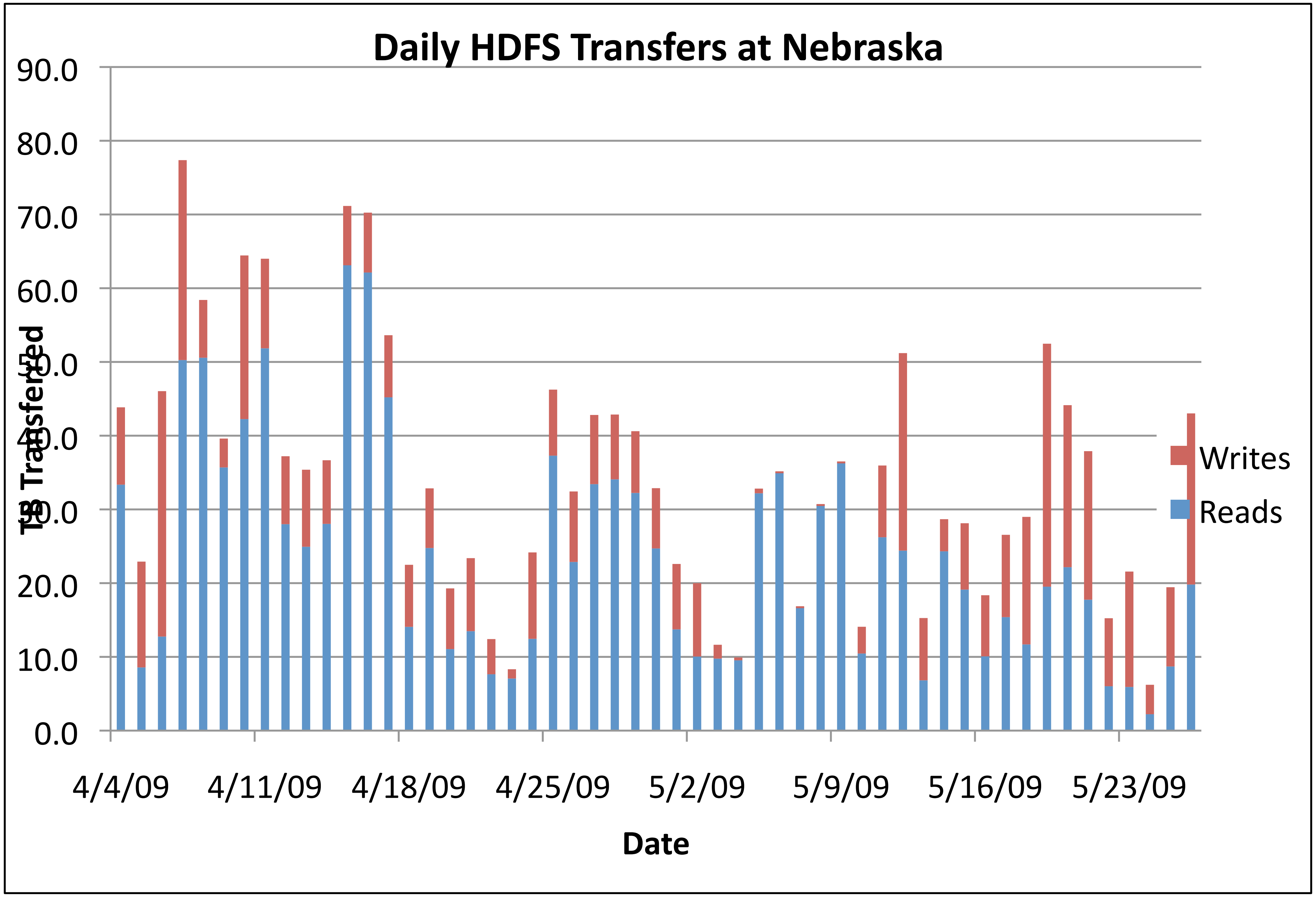
Usability

- POSIX works*; this opens a lot of doors to communities who are put off by recompiling their software.
  - * = writes are append-only
- Users no longer have to know about your FS-specific tools.
  - Although there are some nifty additions the tools provide.

(CMSSW) Performance

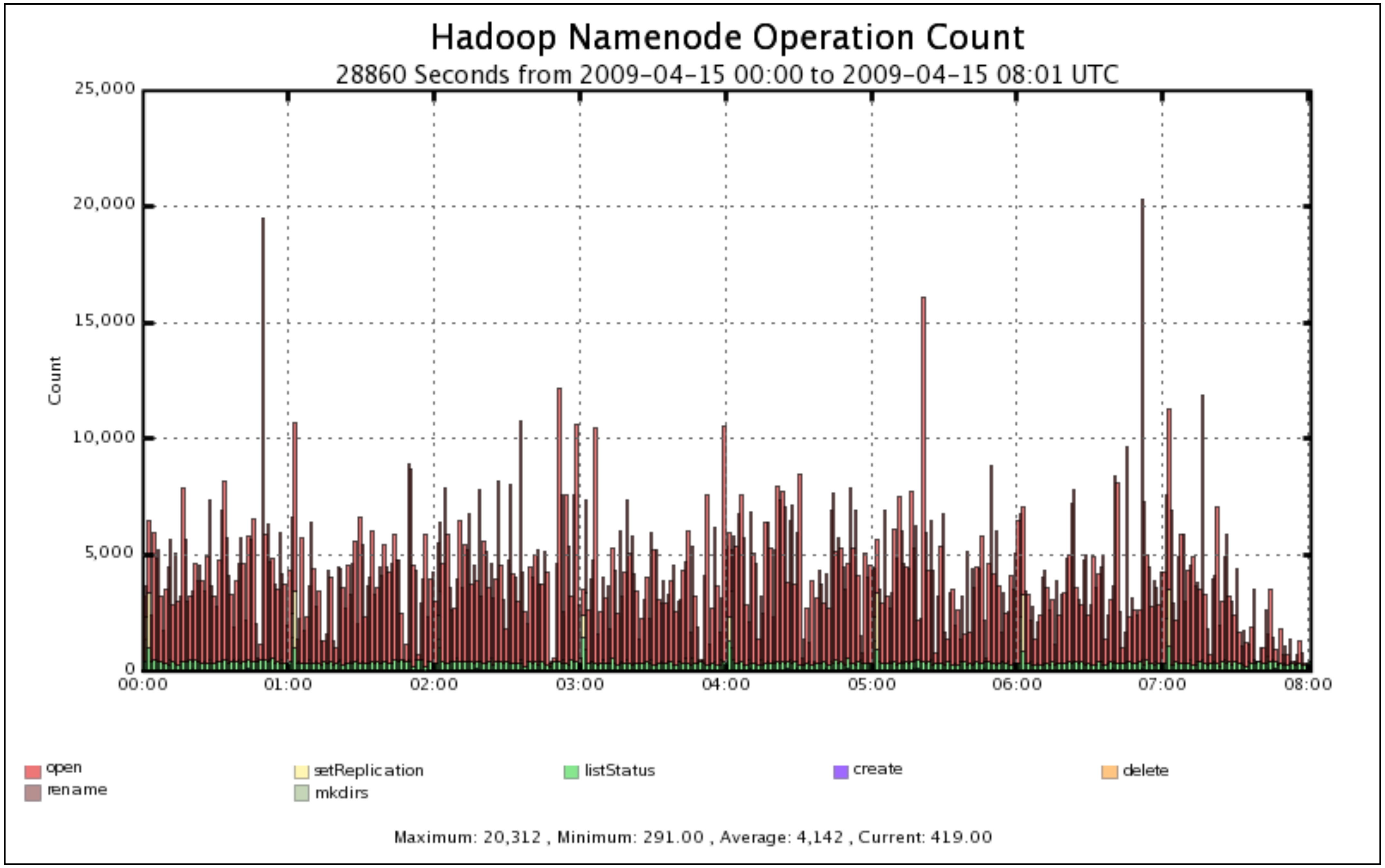


TB moved / day

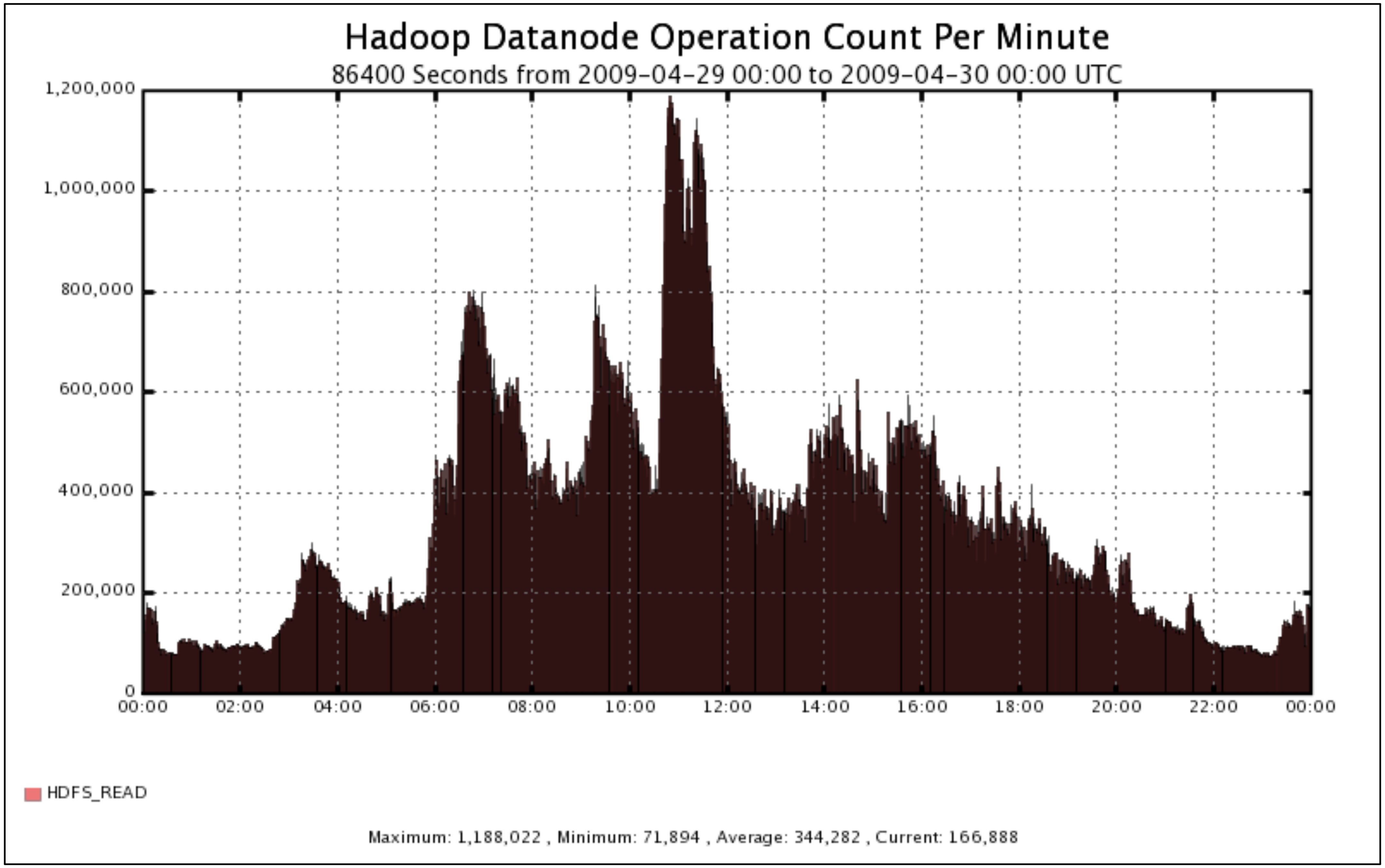


Daily HDFS Transfers at Nebraska
TB Transferred
90.0
80.0
70.0
60.0
50.0
40.0
30.0
20.0
10.0
0.0
4/4/09
4/11/09
4/18/09
4/25/09
5/2/09
5/9/09
5/16/09
5/23/09
Date
Writes
Reads

Namenode Ops



I/O Ops per Second



Performance Stats

- We've clocked:
  - The filesystem at 80Gbps.
  - 23 Gbps for 300 CMSSW processes analyzing a *single file* @ 2 replicas (we picked a fake workflow to pump up the per-job rate).
  - SRM endpoints at ~50Hz (these SRMs are stateless; load-balancing is trivial). Done using GUMS auth.
  - fsck takes <10s.
  - Decommissioning a pool <1hr.
  - Namenode restart in about 60s.
  - WAN transfers peak at 9Gbps, sustain 5Gbps.
  - 18,400 metadata ops / sec from the namenode.

Hadoop Pros vs Cons

- Pros:
  - Very good reliability
  - Very good manageability
  - Designed for commodity hardware (and you own commodity hardware)
- Cons:
  - New filesystem for HEP
  - Designed for commodity hardware (and you own high-end hardware)

Upcoming Work

- Upgrade to 0.20.0.
  - Integration with Yahoo! and Cloudera builds.
- Battle-hardening RPMs for GridFTP & BestMan.
- “Settling in”: getting a golden version for WLCG.

Conclusions

- Hadoop gives us significant improvements in manageability of storage => lowers cost of maintenance.
- Performance scalability benefits by co-locating storage and WNs
  - Reliability during disk failures => less failures seen by users.
- Allows us to use commodity hardware => lowers cost of hardware.