Using Hadoop as a Storage Element on the WLCG

Brian Bockelman

ISGC 2009

Hadoop

- Hadoop is a data processing system developed by the Apache Software Foundation that uses the Map-Reduce paradigm.
  - Primary contributor is Yahoo!
- It consists of two components - a job scheduling framework and a distributed file system, HDFS.
- Designs are taken from Google's MapReduce and Google FS.

Why HDFS?

- HDFS excels at the following:
  - Manageability.
  - Reliability / protection against data-loss.
  - Enabling commodity hardware.
  - Used without specialized clients (POSIX-like semantics).

Why HDFS?

- Another enabler for HDFS is that grid storage components are now sufficiently modular to layer on top of generic file systems.
- Indeed, others are exploring alternatives (Lustre/GPFS/xrootd) to the traditional grid SEs (Castor/dCache/DPM).

Why HDFS?

- Oh yeah - and scalability.
  - Putting hard drives in worker nodes enables us to hit a much higher Gbps #.
  - The namenode's unique design *guarantees* any read operation will never trigger disk I/O or
    - Yahoo! clocks the NN at 50k ops / sec.

HDFS SE Design

- Hadoop design: http://hadoop.apache.org/core/docs/r0.19.1/hdfs_design.html
- HDFS core is split into two parts: the namenode (NN) and the datanodes (DN).
- Familiar design to anyone running a SE.
    - Keeps files broken into large chunks on the system.
- The NN takes care of the namespace and file locations.
- The DNs store data chunks

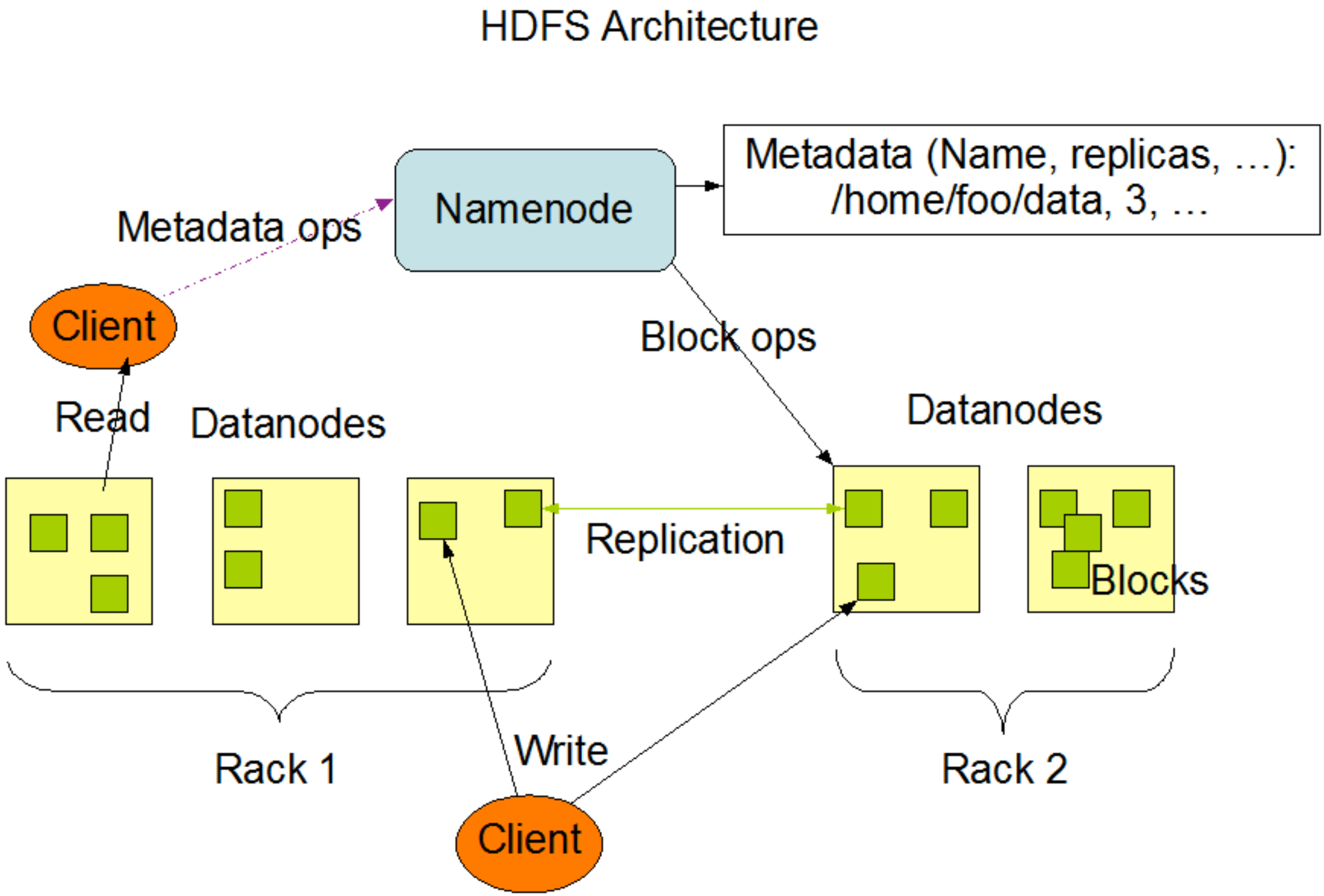


Image courtesy of Hadoop website

HDFS SE Design

- On top of HDFS core, we also run:
  - Mounted on worker nodes via FUSE.
  - Globus GridFTP for WAN transfers.
  - BestMan for SRM interactions.
  - HDFS checkpoint server (we do checkpoints at least hourly).
  - HDFS balancer (to keep % of disk utilized same on all the data nodes.)

HDFS SE Design

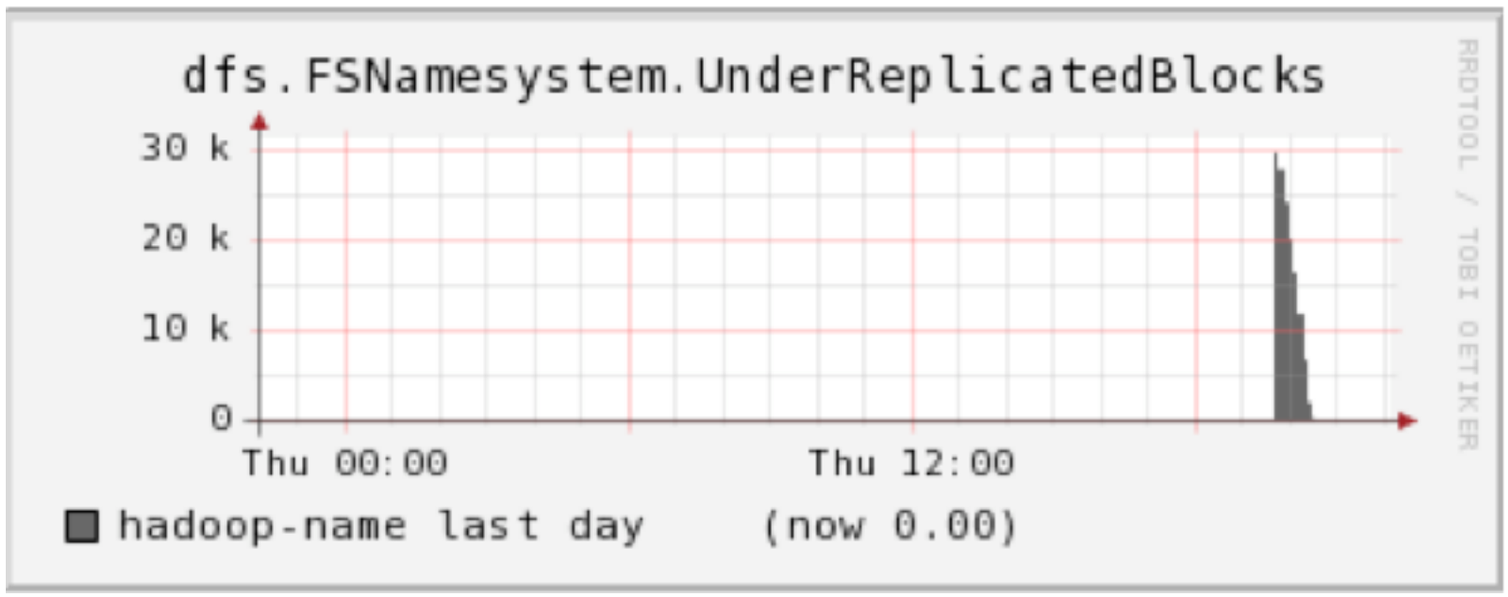
- I gave a thorough HDFS walkthrough of the system's highlights (in a scant 51 slides!) here:
  - https://twiki.grid.iu.edu/bin/view/Storage/HdfsWorkshop
  - Read through this at your leisure.

HDFS Replication

- The namenode keeps track of the desired replicas of each block (user-parameter).
- As datanodes leave, the namenode will order new replicas created of the DN's data.
  - We can recover from data loss in hours.
- When writing, first replica goes to local DN, second goes to a different rack, third goes to yet another rack.
  - We can survive a whole rack disappearing!

Replication Example

- At no point when a HDD fails does a client fail!
- Example below: 1.5TB HDD failed; "danger zone" passed in ~ 1 hr.



HDFS Manageability

- HDFS is designed, top-to-bottom, for unreliable, failure-prone hardware.
- Common admin 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.
  - Setting quotas.

FSCK example

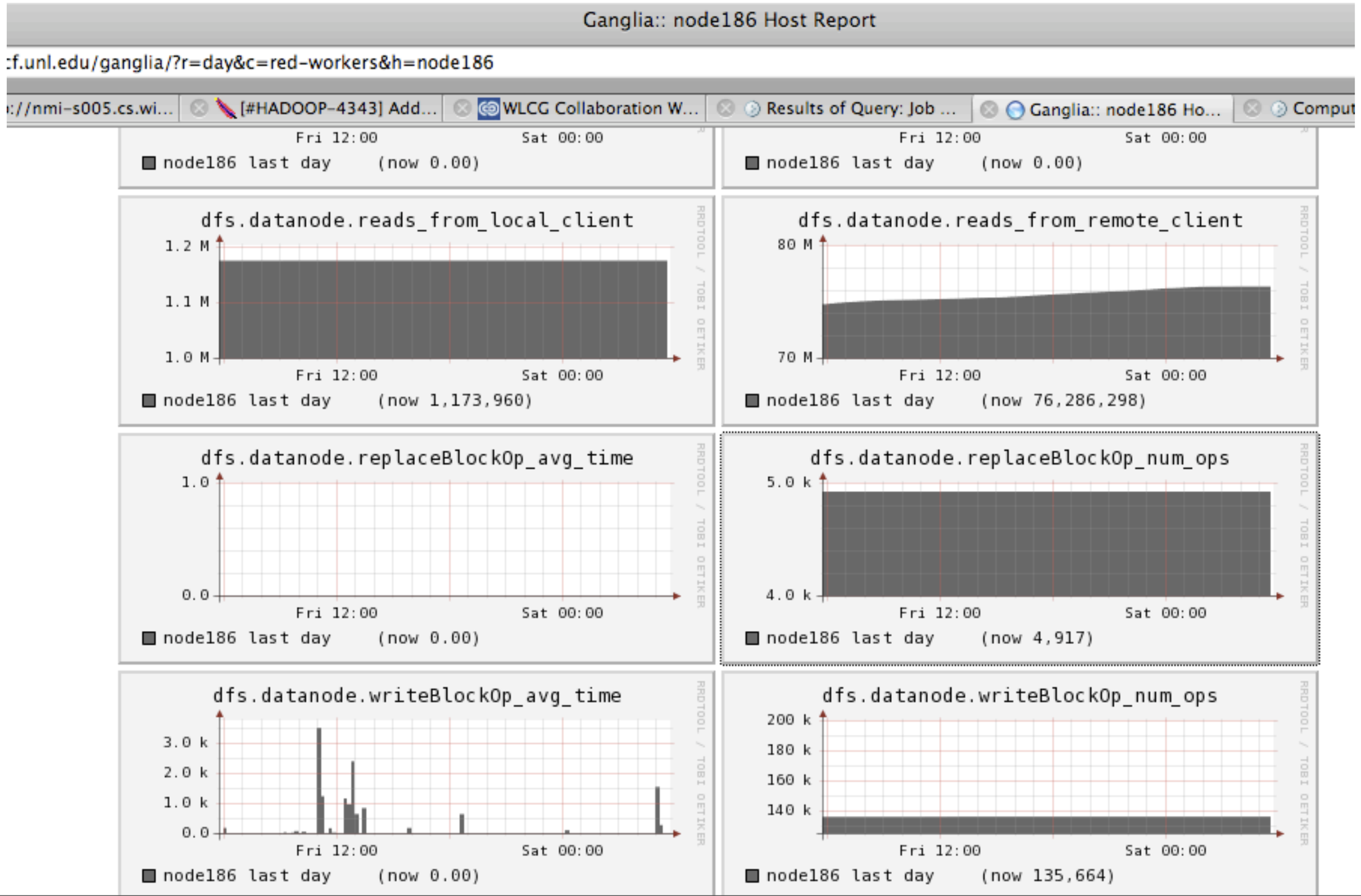
root@hadoop-name:~ — ssh — 107×33

```
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....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 ~]#
```

Ganglia Graphs



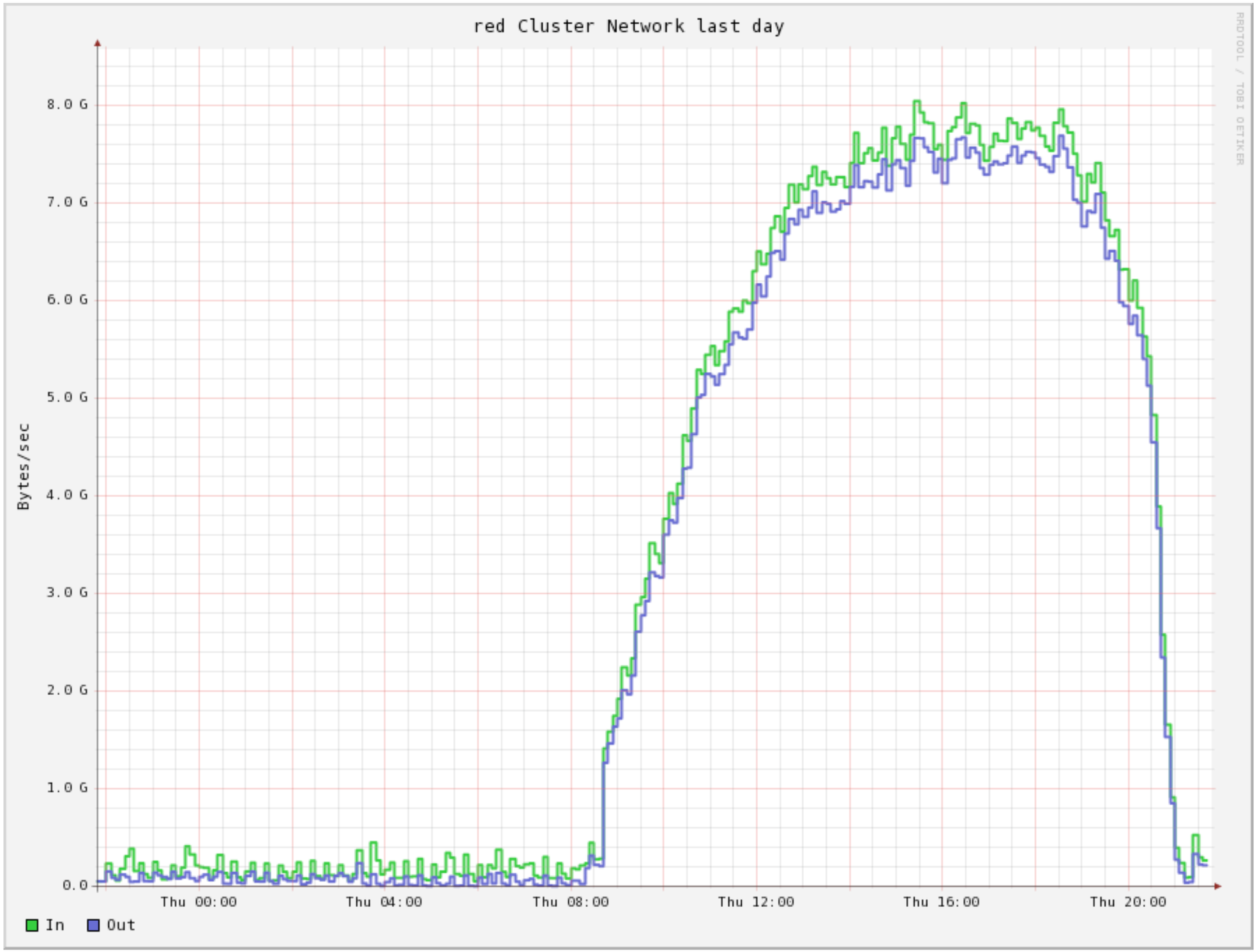
HDFS SE Limitations

- We go from being a large customer of a SE to a small customer of an open source SE.
  - But how much do we need this if the underlying system doesn't have issues?
  - Issues to date have been resolved promptly.
- HDFS does not have a strong-auth security.
  - It is coming! It can be locked down through smart firewall rules.
- Performance with ROOT (which has poor performance with many network-based storage systems) is not as well-known as with other systems.
  - But initial tests don't indicate we should be worried.

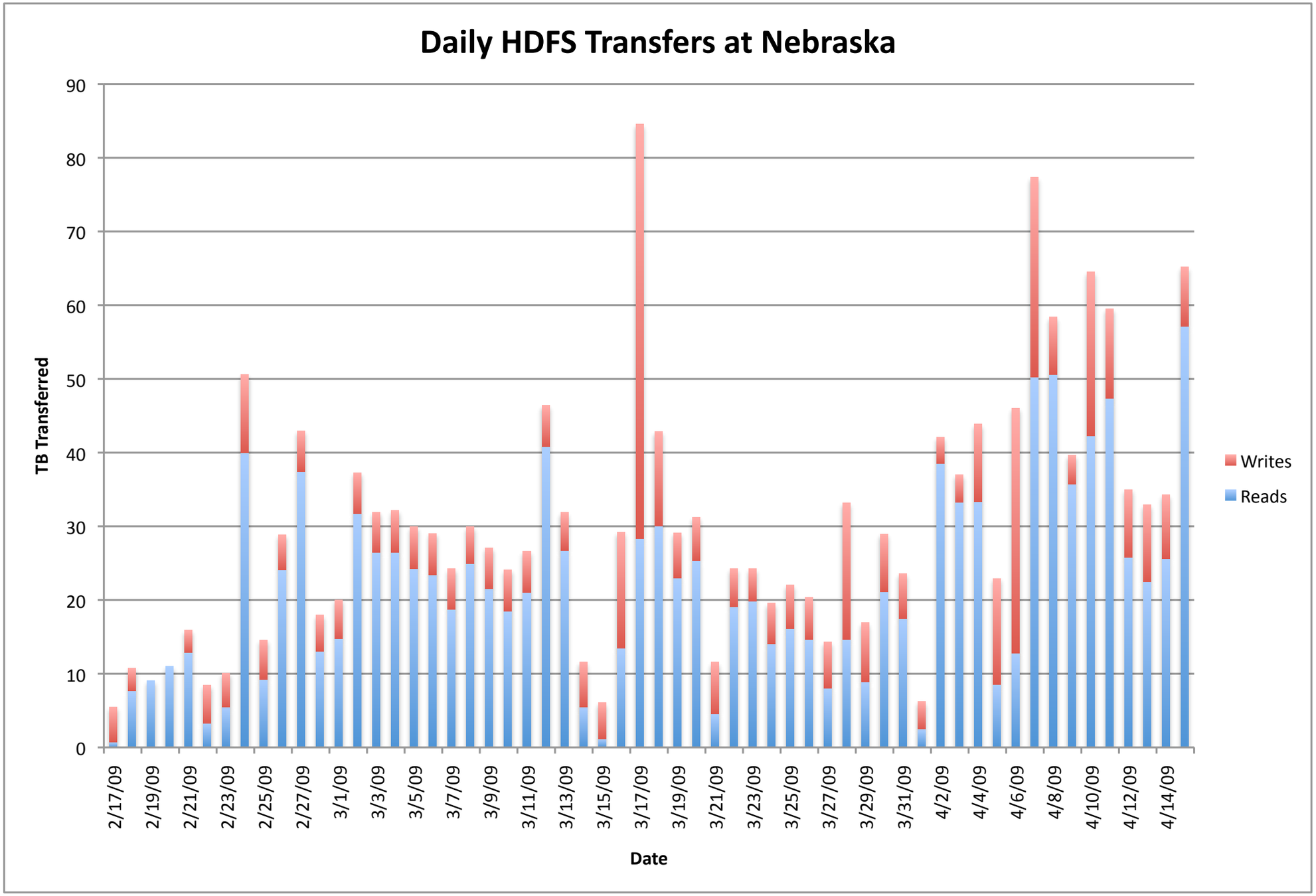
HDFS SE Performance Limits

- Because of the memory requirements of the NN, HDFS with more than a few 10's million files is not currently done in practice.
  - CMS worldwide currently has about 4.5 million files in PhEDEx -- we're not too worried at a T2.
- The centralized NN design limits the file system to about 50,000 opens / second.
  - *Nowhere* near this limit in production.

(CMSSW) Performance



TB moved / day


Daily HDFS Transfers at Nebraska
TB Transferred
90
80
70
60
50
40
30
20
10
0
2/17/09
2/19/09
2/21/09
2/23/09
2/25/09
2/27/09
3/1/09
3/3/09
3/5/09
3/7/09
3/9/09
3/11/09
3/13/09
3/15/09
3/17/09
3/19/09
3/21/09
3/23/09
3/25/09
3/27/09
3/29/09
3/31/09
4/2/09
4/4/09
4/6/09
4/8/09
4/10/09
4/12/09
4/14/09
Date
Writes
Reads

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 37Hz (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.