“Please make my analysis run faster”: a Tier3 case study with HDFS

Tom Crane
Simon George
Barry Green
Govind Songara

Royal Holloway
University of London

The Crisis

- Local ATLAS analysis group has a paper deadline looming
- Hammering the local batch farm and file server 24x7 to get results out
- It gets slower and slower
- Then it breaks.

- We fixed it but this got us thinking.

The Use Case

- Tier3 cluster: 48 nodes, 220 slots
- 2 NFS servers (one with 60TB mainly used)
- Network: 1Gb/s with 4x1 trunked between rack-top switches and to NFS server
- Data: various simulated data sets, ~7.5TB in total. Root files (ATLAS D3PD)
- Workflow: use natural subdivision of data sets, submit ~90 jobs, each job runs over data set reading file by file.

Some Options

- We want something that scales better AND is more resilient, so we could take a server offline and fsck a filesystem for example, without loss of service.
- NFS server – can get more out of it?
  - Performance: yes, probably a little, but not a lot.
  - Resilience: high availability?
    - Second server for failover protects against server problems
    - But still needs single backend storage
- Something more advanced, e.g. Lustre
  - Scalable
  - Widely acknowledged to be more complex to manage (kernels etc.)
    - Problem since we have so little manpower available for Tier3
  - Still not possible to take out a file server without losing access to (subset of) data
- Learn from ATLAS: copy data to WNs
  - Maybe even use some kind of local cache to avoid repeat copies of same file
  - Problem: only ~40GB free on local disk of WN
  - Upgraded some to test, but users were not excited about having to add this complexity to their job scripts
- Give up and use the Grid
  - Well, maybe. Our users do this for data (not simulation). It has been copied to ATLAS_LOCAL_GROUP_DISK at a UK site, and if their jobs don't queue for long they complete in about 4 hours.
- Something more experimental: hadoop?
  - Read on

Hadoop file system – what is it?

- "Apache Hadoop is a software framework that supports data-intensive distributed applications under a free license. It enables applications to work with thousands of computational independent computers and petabytes of data. Hadoop was derived from Google's MapReduce and Google File System (GFS) papers." – Wikipedia

- **HDFS is a distributed, scalable, and portable file system written in Java for the Hadoop framework**.
- Comprises data nodes and name node.
- File access through CLI, FUSE mount, ROOT i/o plugin

We like:

- Designed for use with old/cheap hardware that could fail.
- For N replicas, resilient to failure of N-1 data nodes: internal mechanism to automatically replicate up again. Replicas can be set per file.
- Data nodes act independently. New nodes automatically join in.
- Fail-over 2nd name node
- Web pages monitor state
- Management CLI

HDFS – our set up

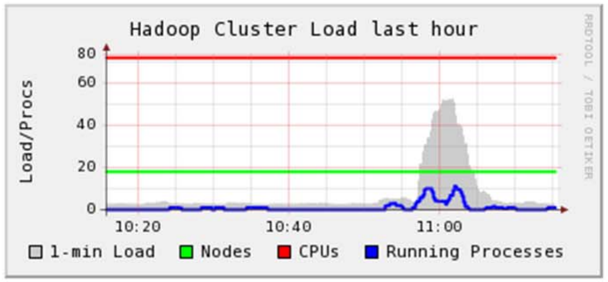
- RPMs from OSG
- 11 data nodes on old hardware:
  - 32 bit Intel Xeon 3GHz 2x2core
  - 2x PATA Maxtor 5A300J0 300GB 5.4k rpm disk
    - hdparm –t gives typically 40MB/s
  - Ext3 fs
  - 1Gb/s network
- A name node and backup name node, on similar hardware
- Added more data nodes during test
  - simply run `hadoop balancer` to spread data across them.
  - New nodes have 7k rpm 3TB disk
- 3 replicas per file (property of file, set default)

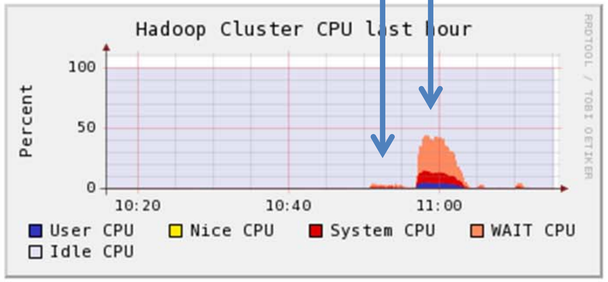
Aside: failed node, or orderly decommissioning, handled well. But major network problems put multiple nodes out of touch – very bad, data lost.

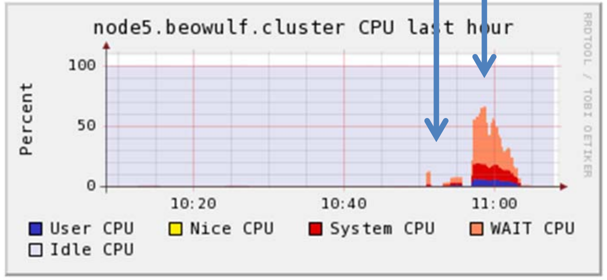
HDFS: copy and replicate files

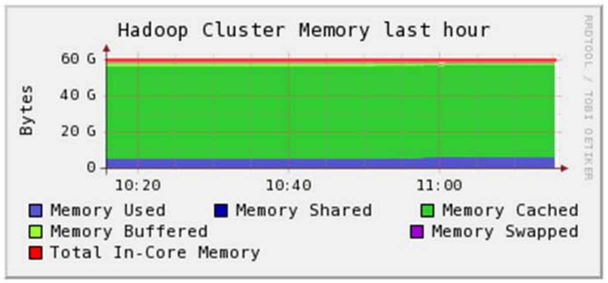
Extend to 16 replicas

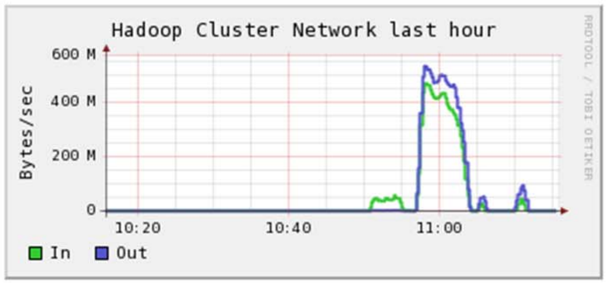
Copy 10GB file with default 3 replicas

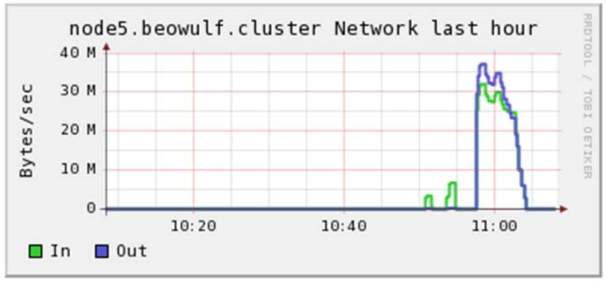










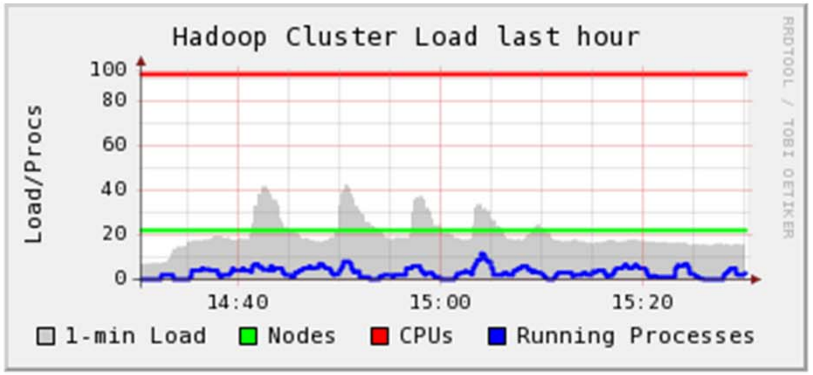


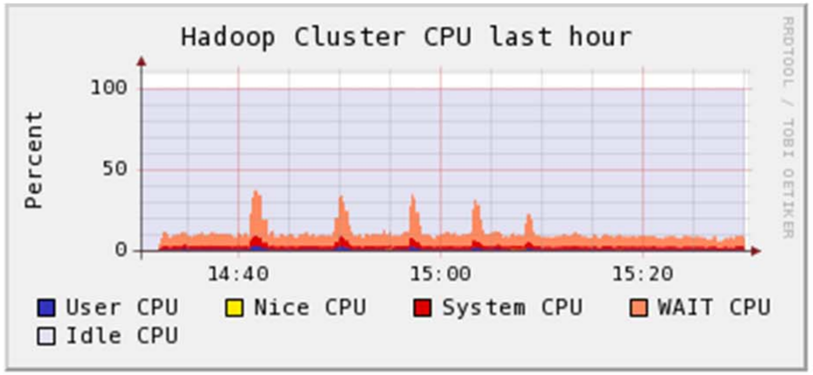
Cluster total

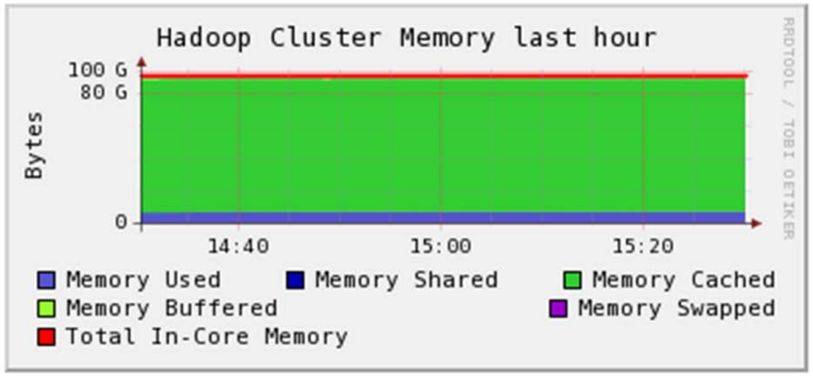
Example data node

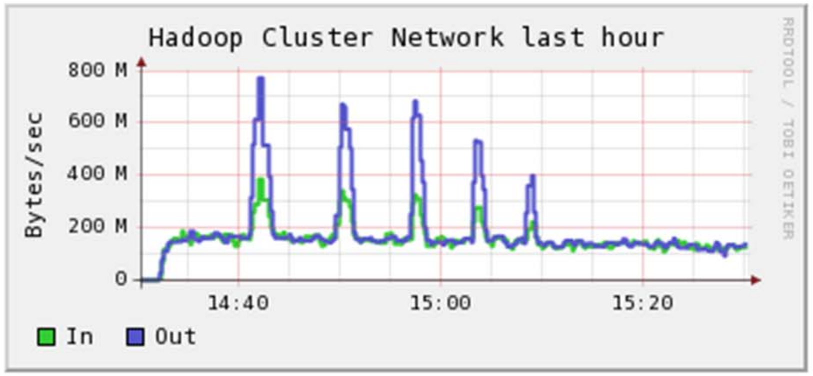
Writing 10 GB file via FUSE mount took ~4 minutes, i.e. 40 MB/s. NFS was similar.

Decommissioning 4 nodes



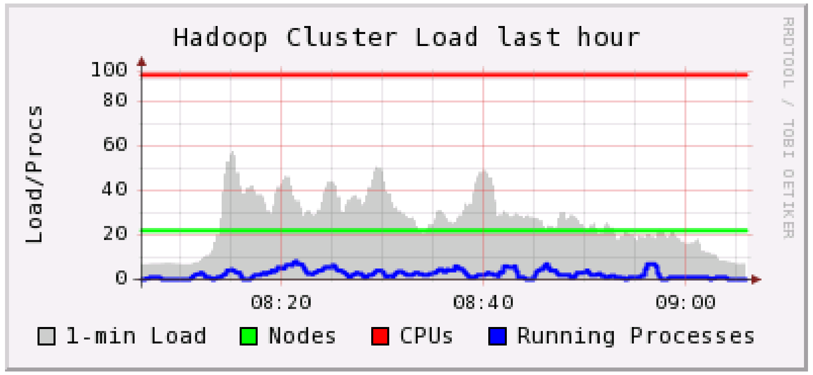


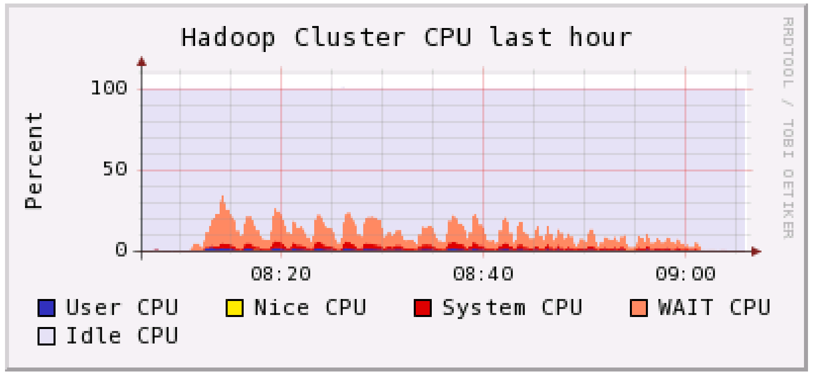


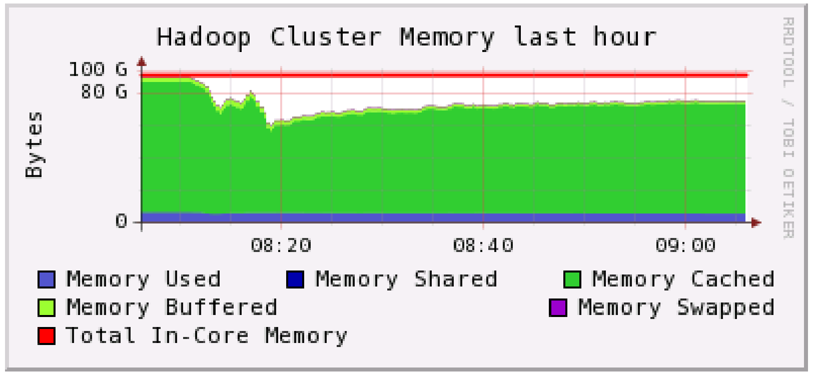


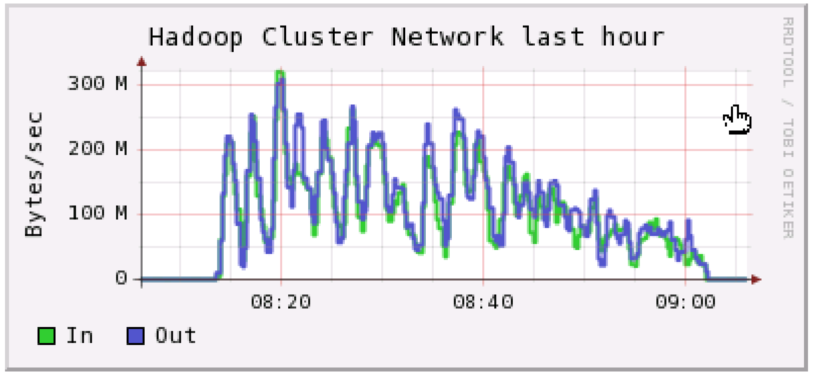
Time taken depends on how much data is on nodes.
All the data movement is automatic, to maintain replicas.

Recommissioning 12 nodes



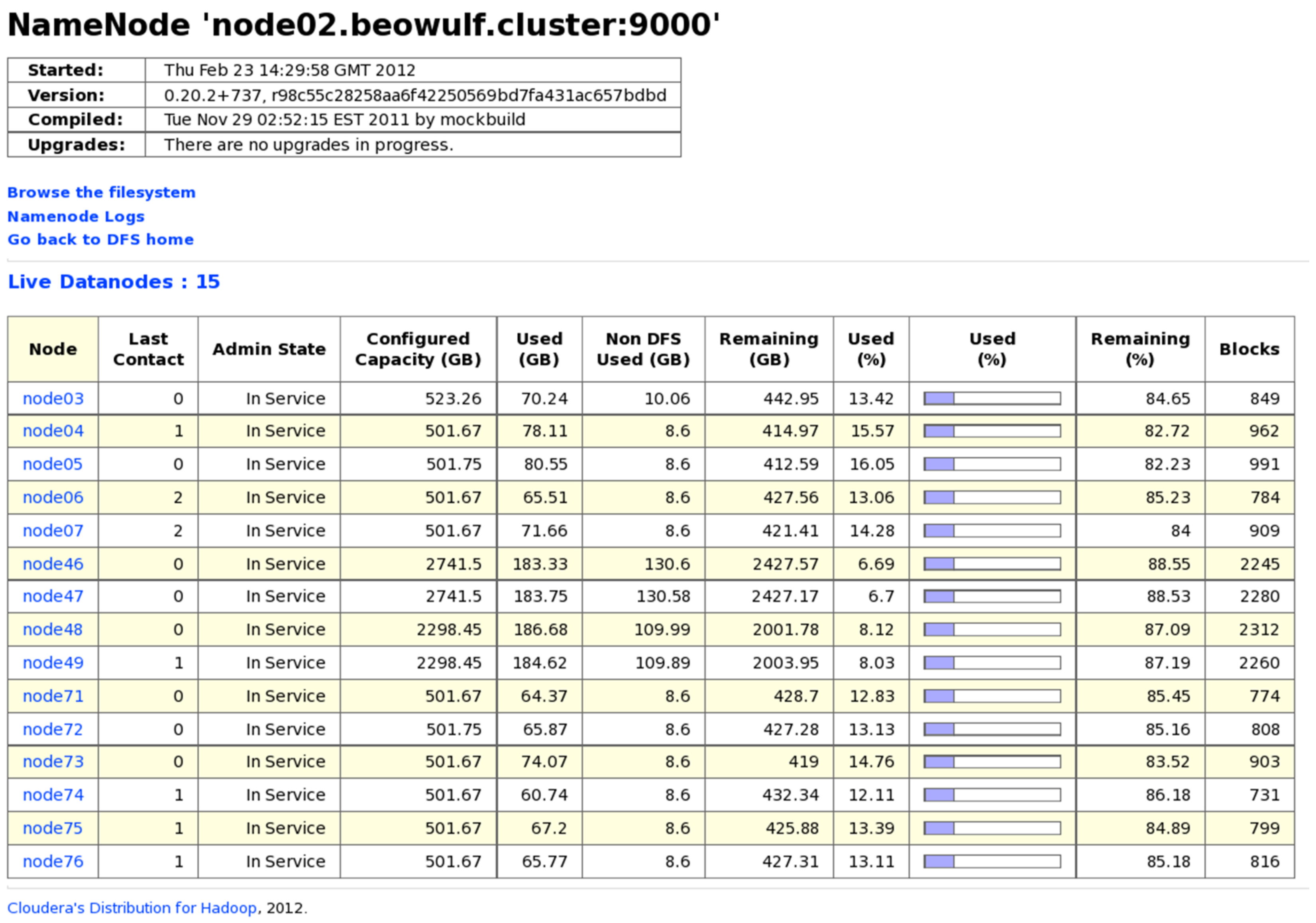






Balancing redistributes data
Increased replicas.

HDFS: after adding new data nodes & rebalancing



NameNode 'node02.beowulf.cluster:9000'

Started:	Thu Feb 23 14:29:58 GMT 2012
Version:	0.20.2+737, r98c55c28258aa6f42250569bd7fa431ac657bdbd
Compiled:	Tue Nov 29 02:52:15 EST 2011 by mockbuild
Upgrades:	There are no upgrades in progress.

Browse the filesystem
Namenode Logs
Go back to DFS home

Live Datanodes : 15

Node	Last Contact	Admin State	Configured Capacity (GB)	Used (GB)	Non DFS Used (GB)	Remaining (GB)	Used (%)	Used (%)	Remaining (%)	Blocks
node03	0	In Service	523.26	70.24	10.06	442.95	13.42	▰▱▱▱▱	84.65	849
node04	1	In Service	501.67	78.11	8.6	414.97	15.57	▰▱▱▱▱	82.72	962
node05	0	In Service	501.75	80.55	8.6	412.59	16.05	▰▱▱▱▱	82.23	991
node06	2	In Service	501.67	65.51	8.6	427.56	13.06	▰▱▱▱▱	85.23	784
node07	2	In Service	501.67	71.66	8.6	421.41	14.28	▰▱▱▱▱	84	909
node46	0	In Service	2741.5	183.33	130.6	2427.57	6.69	▰▱▱▱▱	88.55	2245
node47	0	In Service	2741.5	183.75	130.58	2427.17	6.7	▰▱▱▱▱	88.53	2280
node48	0	In Service	2298.45	186.68	109.99	2001.78	8.12	▰▱▱▱▱	87.09	2312
node49	1	In Service	2298.45	184.62	109.89	2003.95	8.03	▰▱▱▱▱	87.19	2260
node71	0	In Service	501.67	64.37	8.6	428.7	12.83	▰▱▱▱▱	85.45	774
node72	0	In Service	501.75	65.87	8.6	427.28	13.13	▰▱▱▱▱	85.16	808
node73	0	In Service	501.67	74.07	8.6	419	14.76	▰▱▱▱▱	83.52	903
node74	1	In Service	501.67	60.74	8.6	432.34	12.11	▰▱▱▱▱	86.18	731
node75	1	In Service	501.67	67.2	8.6	425.88	13.39	▰▱▱▱▱	84.89	799
node76	1	In Service	501.67	65.77	8.6	427.31	13.11	▰▱▱▱▱	85.18	816

Cloudera's Distribution for Hadoop, 2012.

NFS server

- Disks: 40x 2TB WDC WD20EADS
- Controllers: 2x Areca ARC-1280, each with two RAID6 volumes of 10 disks
- LVM: 4x 14.5TB striped into single logical volume
- Filesystem: 58 TB XFS
- Network: 4x 1Gb/s NICs bonded, mode=0

Aside on disk reliability

- "Desktop" disks
- Returned 50% in 3 years
- Stability improved a lot by:
    - Set TLER to 7s on disk (where supported)
    - Areca firmware upgrade

HDFS vs NFS tests

Test protocol 1 – analysis-like file access

- 60 jobs
- Each job given 10 data files at random from list of ~4600
- Varying sizes, mean 163 MB – see later
- Directly read file from FUSE mount, using dd to /dev/null

Test protocol 2 – simply read one large file

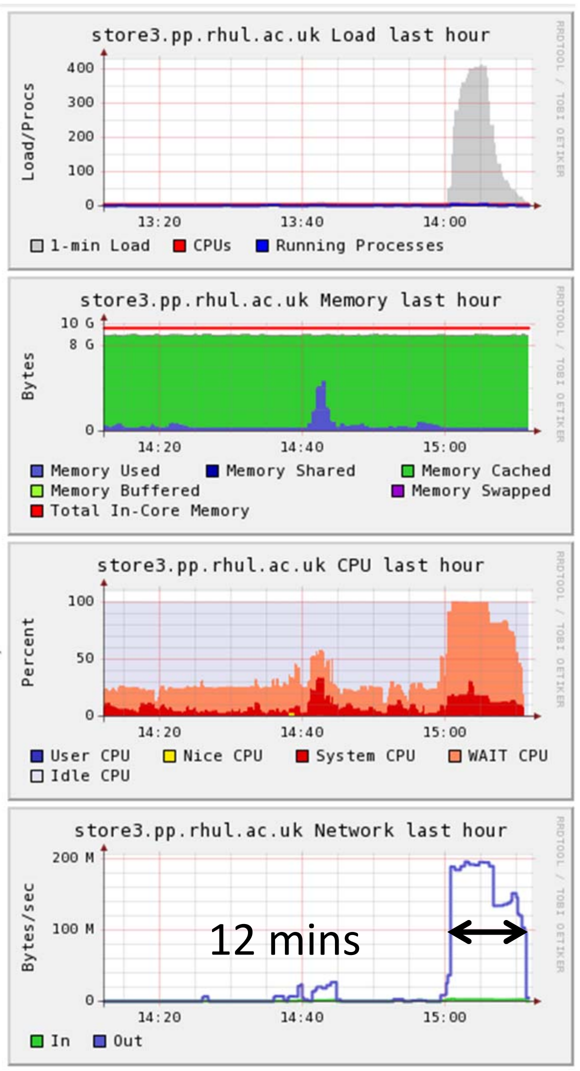
- 60 jobs
- Each job reads 10GB file (multiple replicas in HDFS)
- dd via NFS mount, dd via FUSE mount, or use hadoop CLI to cat, to /dev/null
- Measure time from start to finish of all jobs (using Ganglia)
- Speed = 60 x 10GB / elapsed time

General

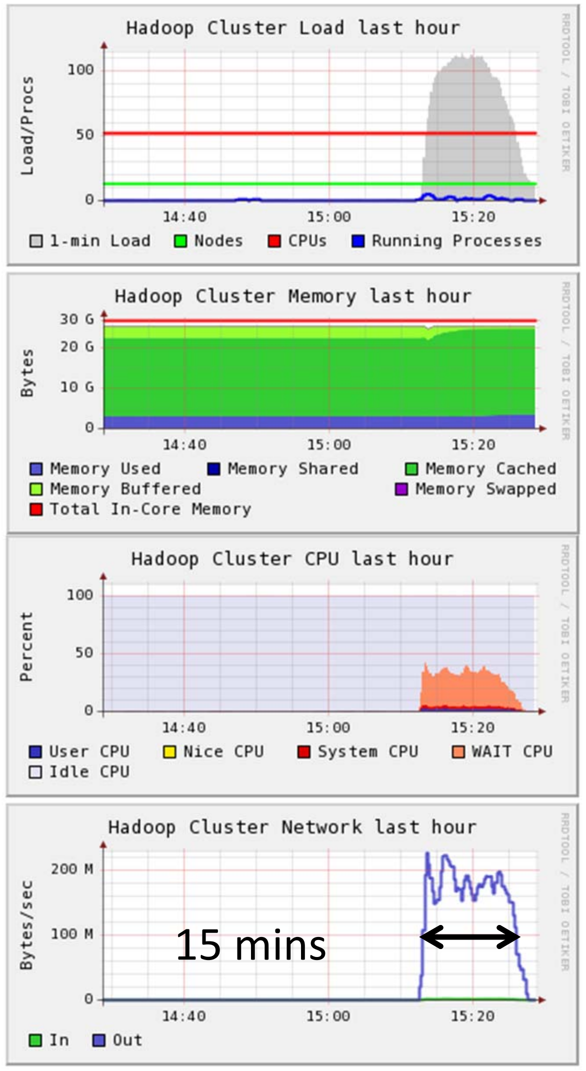
- Submit jobs to Tier3 cluster
- 220 slots on 48 nodes, 1 Gb/s
- Mainly used when idle, but few cpu-bound jobs should not interfere

Protocol 1: NFS vs HDFS, analysis data

NFS – 1 server



HDFS (FUSE) – 12 data nodes



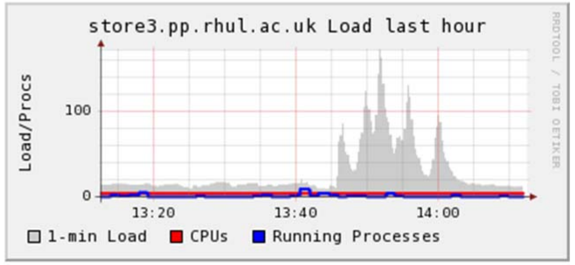
60 jobs read random files so not directly comparable but this typifies the observed trend.

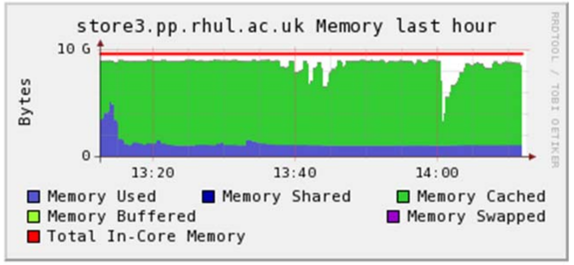
HDFS lower load, not i/o bound.

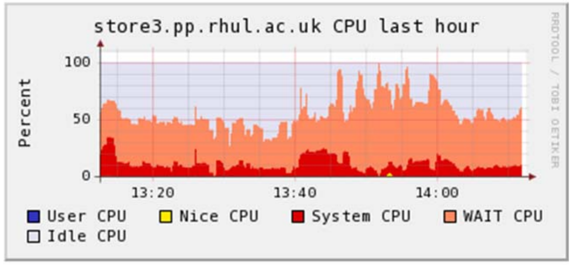
HDFS slightly higher peaks but erratic. NFS quicker.

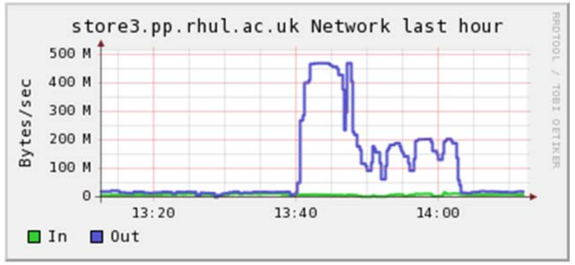
Protocol 2: NFS vs HDFS, 10GB file

NFS – 1 server









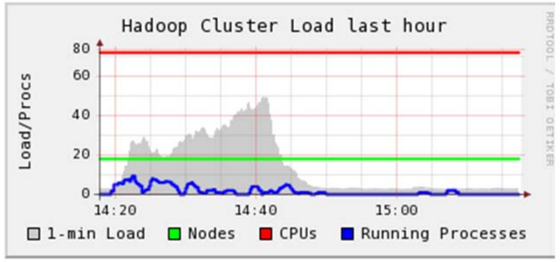
60 jobs read same 10 GB file (or its 16 replicas)

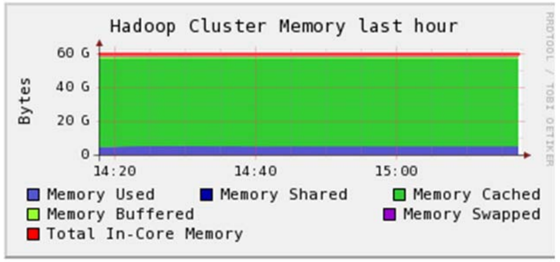
HDFS lower load, not i/o bound.

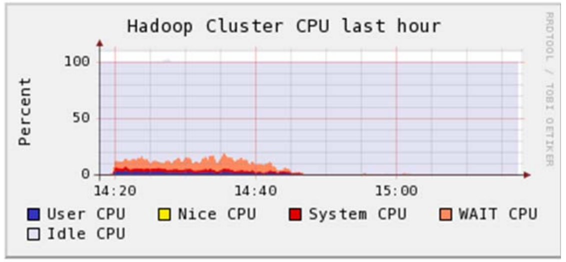
HDFS: jobs end gradually, no sudden stop

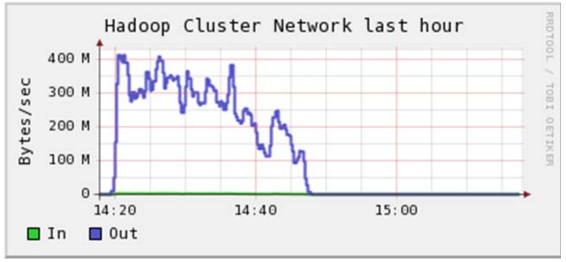
NFS no worse. Why?

HDFS (FUSE) – 15 data nodes





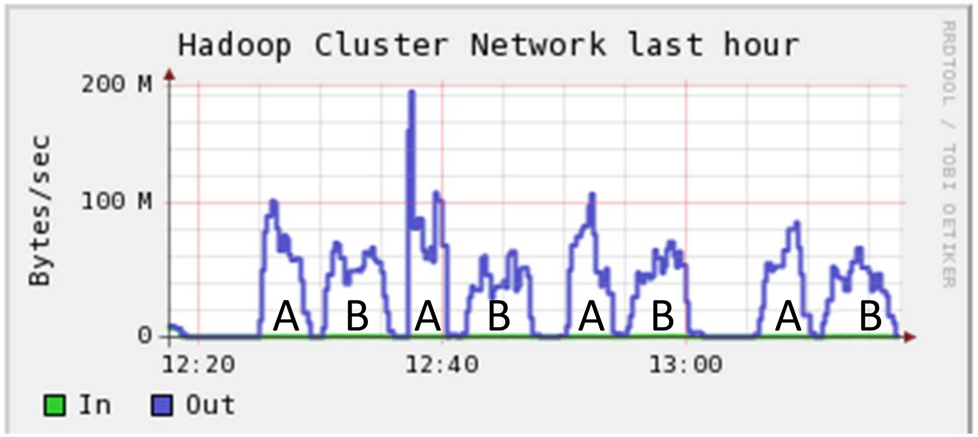




Hadoop CLI vs FUSE mount

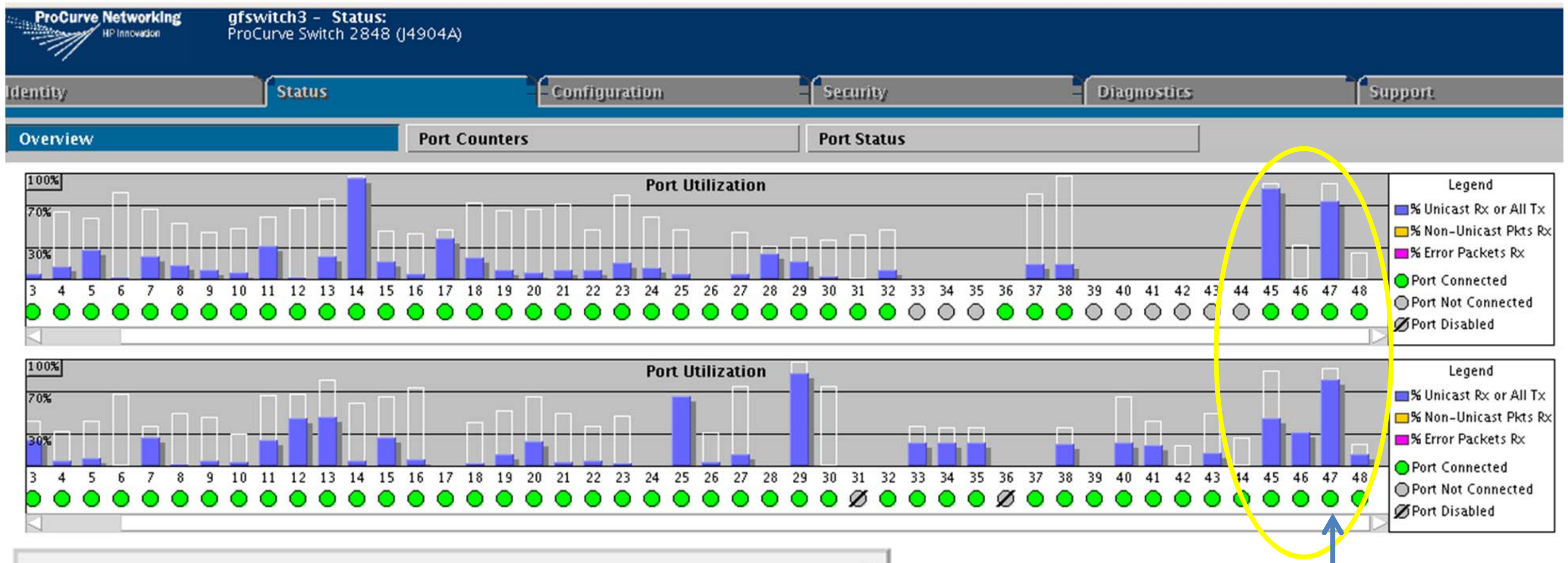
- Read file with 20 data nodes, 16 replicas.
- Run on one worker node:
  - `time hadoop fs -cat /george/testzero10G > /dev/null`
  - `time dd if=/mnt/hadoop/george/testzero10G of=/dev/null`
- Results

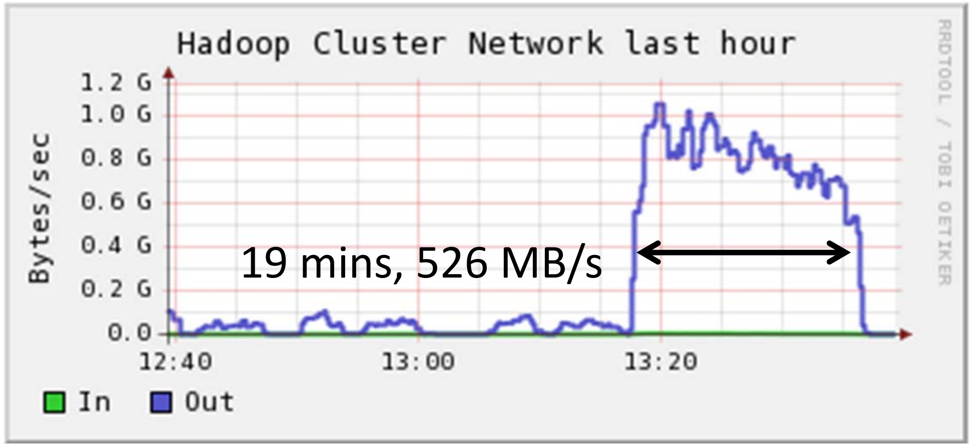
(A) Hadoop CLI	56 MB/s
(B) FUSE mount	38 MB/s



- Direct access with hadoop command line tools faster.
- 60 job test results even more striking.

Network limitations



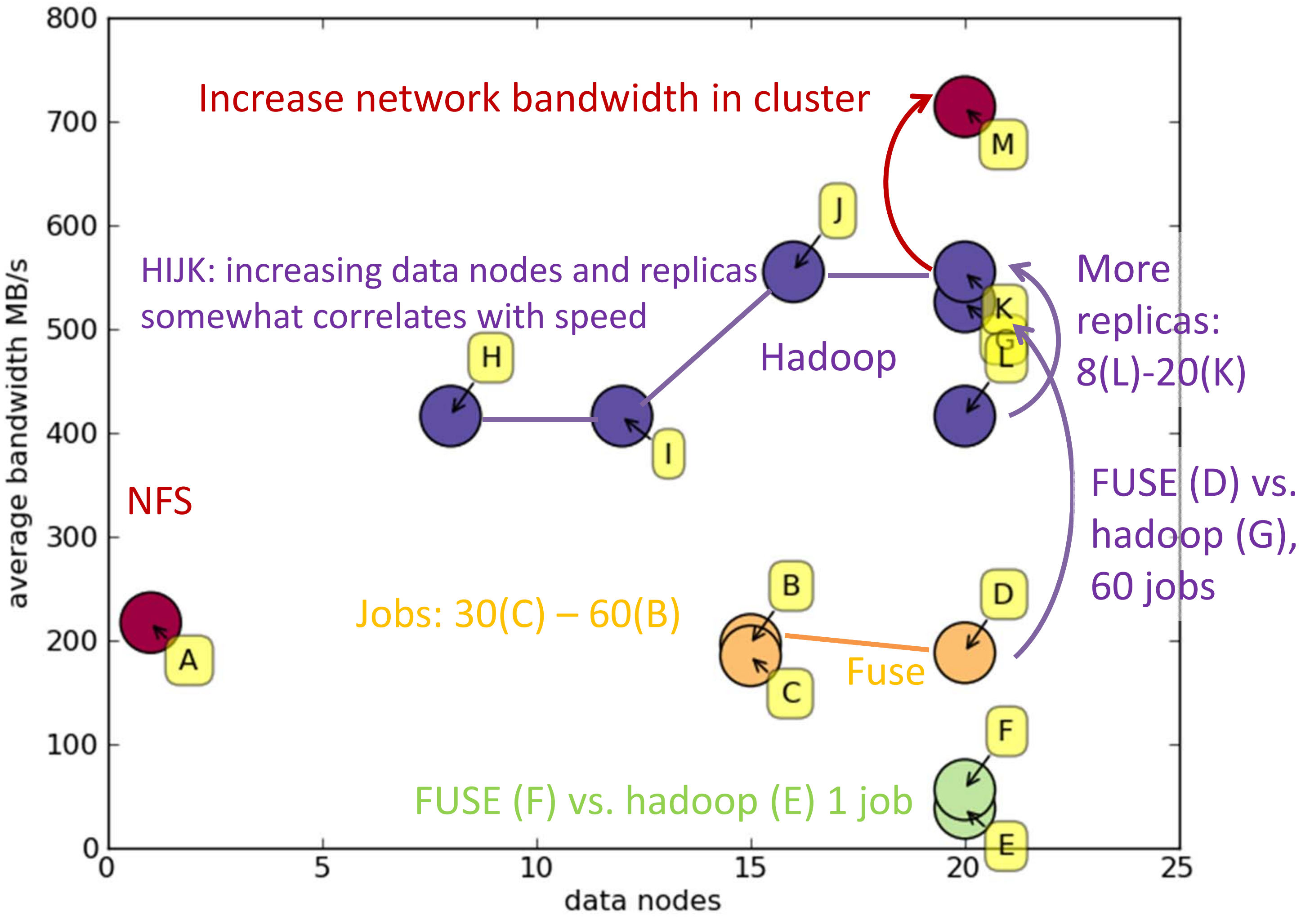


4x1 Gb/s trunk between switches – heavy load

Performance did not improve with 20 replicas. Added 1,2 more links, saw performance improve.

HDFS, hadoop CLI, 60 jobs, 20 data nodes, 16 replicas.

Test results



Max performance rarely limited by number of data nodes.
Caveats: large bandwidths not very accurate; reproducibility not thoroughly tested.

Test results - table

Test #	Speed (MB/s)	# data nodes	# file replicas	#jobs	File access	Comments
A	217	n/a	n/a	30	NFS	For reference: read from 1 NFS server
B	196	15	16	60	FUSE	More nodes/jobs doesn’t really change poor performance with Fuse mount
C	185	15	16	30	FUSE	
D	188	20	16	60	FUSE	
E	38	20	16	1	FUSE	Comparison of fuse mount vs hadoop CLI; the latter is faster
F	56	20	16	1	Hadoop	
G	526	20	16	60	Hadoop	
H	416	8	8	60	Hadoop	
I	416	12	12	60	Hadoop	Increasing nodes/replicas somewhat increases speed
J	555	16	16	60	Hadoop	
K	555	20	20	60	Hadoop	Faster with more replicas
L	416	20	8	60	Hadoop	
M	714	20	20	60	Hadoop	Increase inter-switch trunking: 4 Gb/s (K) to 5 (M), 6(N) Gb/s
N	588	20	20	60	Hadoop	

Conclusions on experience with HDFS

- **Positive**: easy to learn, easy to commission & decommission nodes, increase replicas, etc.
- **Negative**: can silently fail from user point of view (e.g. corrupt blocks, replication) – need to keep an eye on management web page.
  - Major trouble caused by network problem.
- Now happy with the read performance.
  - No sign of nice scaling initially, had to work for it.
  - Don’t use FUSE mount
  - Pay attention to balancing replicas, nodes and jobs
  - Look out for network bottlenecks
  - Write performance not great; don’t understand why, but less important
- Reliability
  - Is my data safe? I wouldn’t trust it (yet).
  - Use it for temp copies of heavily accessed data? Worth trying.
- Next: try real analysis, try ROOT with HDFS TFile plugin.

BACKUP SLIDES

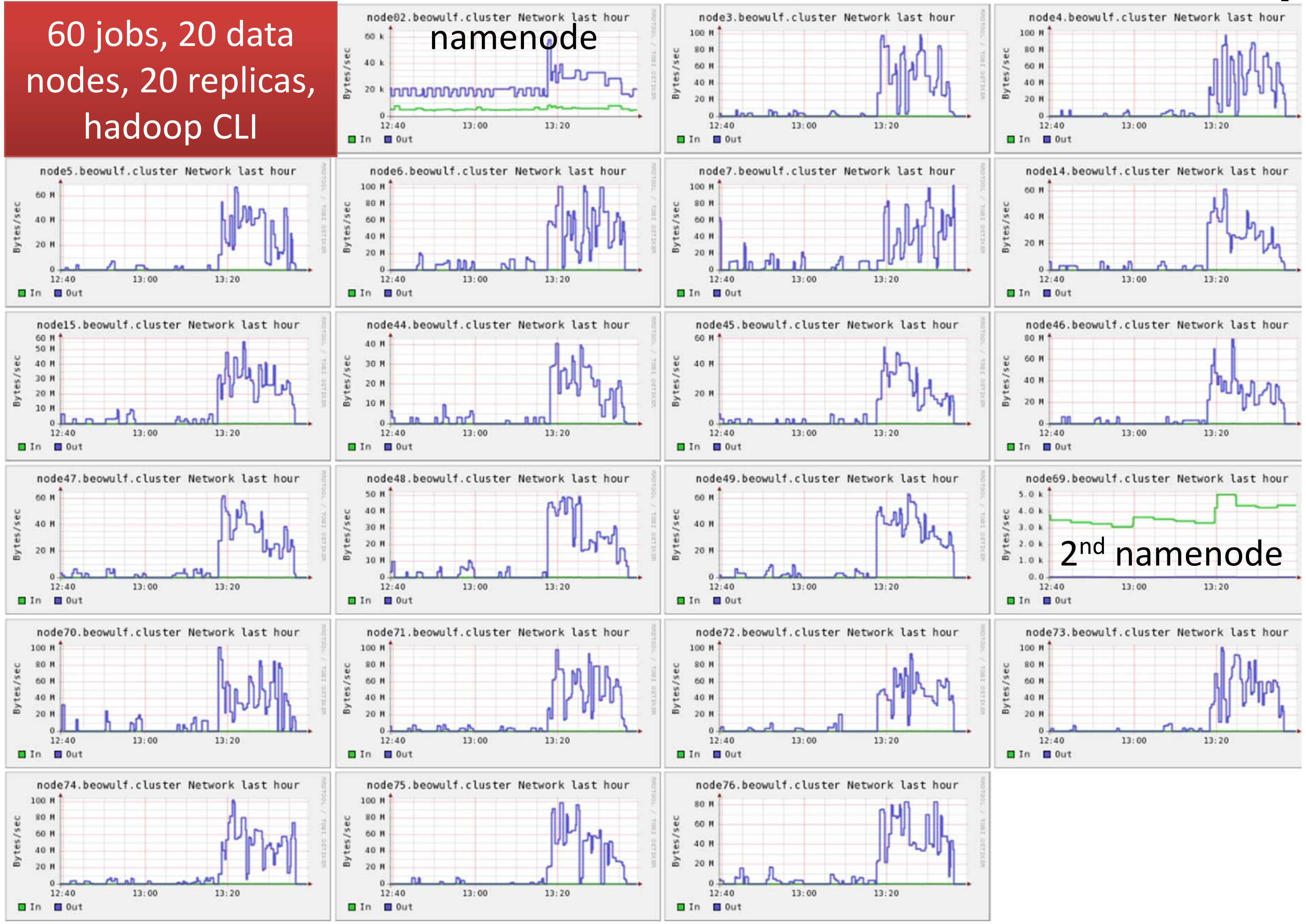
Fixing the NFS server

- XFS file system won't mount.
- Deleted journal and cleaned up, then mounted ro.
- We need to fsck and defrag the 60 TB xfs on the file server, which will take days.
- Built a temporary server (we had a spare machine, just put disks in and configured it) for the ~10TB of data that users could not live without for a few days. Copied data with parallel ftp.
- Bought us the time to sort out the troubled server.
- In case you were worrying, we downloaded the very latest xfs tools for this, and did some tests first. Bottom line, this is scratch space but people still would not like to use it.
- And it worked.

Aside: Areca controllers seem to cope well with failed disk, or a fee sector reallocations, but go badly wrong with a storm of sector reallocations.

See that data nodes are not that busy

60 jobs, 20 data nodes, 20 replicas, hadoop CLI

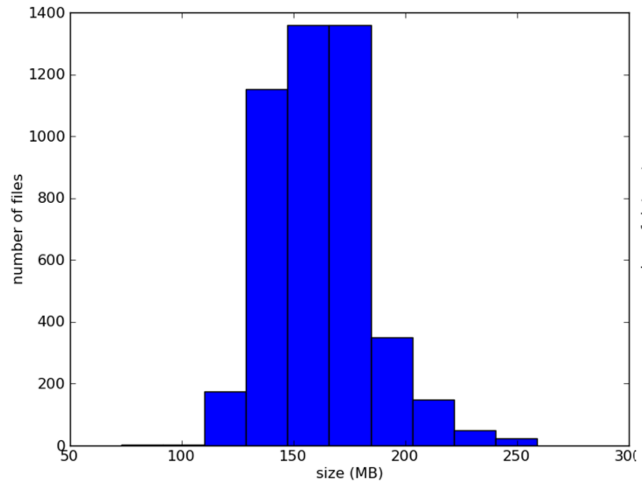


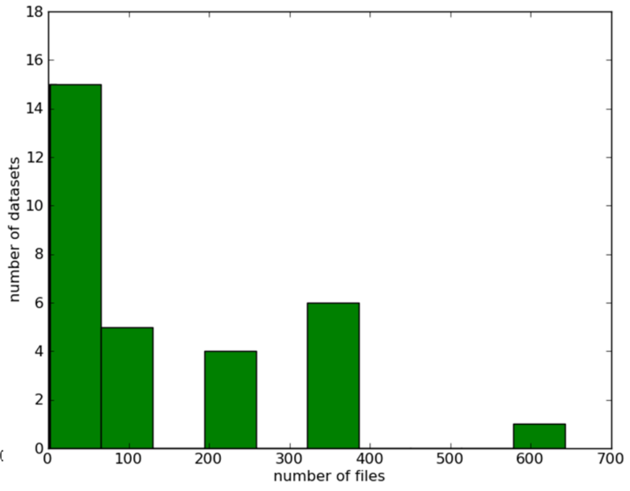
Analysis Jobs

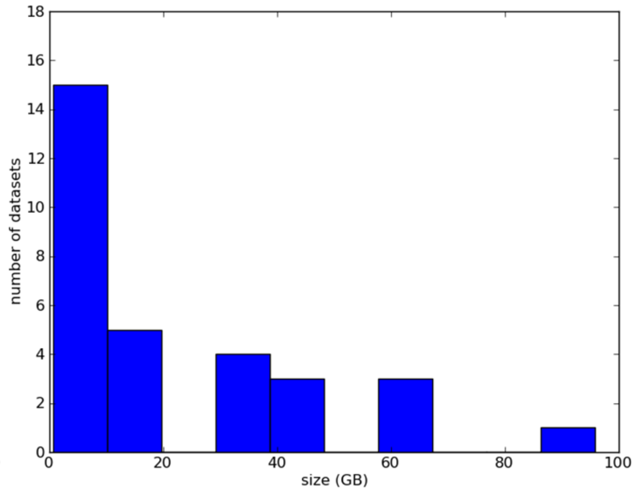
- A deeper look at how they work
- Top D3PD (root) simulated signal and background analysis
  - Read data, perform small amount of work, write histograms
- Normal job runs over ~7 TB of data and takes ~12 hours
  - For testing, made an example with ~0.7 TB.
- No claim this is typical analysis work, but maybe still interesting

- Jobs seen to alternative between CPU use and i/o: one at a time
- Uneven split of data between jobs: time to finish dominated by i/o speed of longest job, not max throughput
  - Addition of histogram merging to split more optimally was considered too much work
- Grid jobs
  - Splitting and merging automatic, therefore all data could be processed in 4 hours. Sometimes delayed waiting to run of course.
- Conclusion: real limiting factor was the way the jobs were organised. If we started again, this is the first place to work.
- Next: try HDFS root plugin, should be quicker than FUSE mount.

“Analysis” test data







- Total 0.72 TB
- 4624 files, mean 163 MB/file
- 31 datasets, mean 23.7 GB/dataset
- Observed that job swings rapidly (few seconds) between bursts of 100% CPU usage and complete i/o wait. Known feature.
- Time to finish longest jobs dominated by largest data sets and files, not parallel reading.