R&D for new SE's: Hadoop

Michael Thomas, Dorian Kcira

California Institute of Technology

CMS Offline & Computing Week

San Diego, April 20-24th 2009

What is Hadoop

- Map-Reduce plus the HDFS filesystem implemented in java
- Map-Reduce is a highly parallelized distributed computing system
- HDFS is the distributed cluster filesystem
  - This is the feature that we are most interested in
- Open source project hosted by Apache
- Used by Yahoo for their search engine. Yahoo is a major contributor to the Apache Hadoop project.

HDFS

- Distributed Cluster filesystem
- Extremely scalable – Yahoo uses it for multi-PB storage
- Easy to manage – few services and little hardware overhead
- Files split into blocks and spread across multiple cluster datanodes
  - 64MB blocks default, configurable
  - Block-level decomposition avoids 'hot-file' access bottlenecks
  - Block-level decomposition means the loss of multiple data nodes will result in the loss of more files than file-level decomposition

HDFS Services

- ***Namenode*** **– manages the filesystem namespace operations**
  - File/directory creation/deletion
  - Block allocation/removal
  - Block locations
- ***Datanode*** **– stores file blocks on one or more disk partitions**
- ***Secondary Namenode*** **– helper service for merging namespace changes**
- **Services communicate through java RPC, with some functionality exposed through http interfaces**

Namenode (NN)

- Purpose is similar to dCache PNFS
- Keeps track of entire fs image
  - The entire filesystem directory structure
  - The file block → datanode mapping
  - Block replication level
  - ~1GB per 1e6 blocks recommended
- Entire namespace is stored in memory, but persisted to disk
  - Block locations not persisted to disk
  - All namespace requests served from memory
  - fsck across entire namespace is really fast

Namenode Journals

- NN fs image is read from disk only once at startup
- Any changes to the namespace (mkdir, rm) are written to one or more journal files (local disk, NFS, ...)
- Journal is periodically merged with the fs image
- Merging can temporarily require extra memory to store two copies of fs image at once

Secondary NN

- The name is misleading... this is NOT a backup namenode or hot spare namenode. It does NOT respond to namespace requests
- Optional checkpoint server for offloading the NN journal → fsimage merges
- Download fs image from namenode (once)
- Periodically download journal from namenode
- Merge journal and fs image
- Uploaded merged fs image back to namenode
- Contents of merged fsimage can be manually copied to NN in case of namenode corruption or failure

Datanode (DN)

- Purpose is similar to dCache pool
- Stores file block metadata and file block contents in one or more local disk partitions. Datanode scales well with # local partitions
  - Caltech is using one per local disk
  - Nebraska has 48 individual partitions on Sun Thumpers
- Sends heartbeat to namenode every 3 seconds
- Sends full block report to namenode every hour
- Namenode uses report + heartbeats to keep track of which block replicas are still accessible

Client File Access

- When a client requests a file, it first contacts the namenode for namespace information.
- The namenode looks up the block locations for the requested files, and returns the datanodes that contain the requested blocks
- The client contacts the datanodes directly to retrieve the file contents from the blocks on the datanodes

Native Client

- A native java client can be used to perform all file and management operations
- All operations use native Hadoop java APIs

File System in User Space (FUSE)

- Client that presents a posix-like interface to arbitrary backend storage systems (ntfs, lustre, ssh)
- HDFS fuse module provides posix interface to HDFS using the HDFS APIs. Allows standard filesystem commands on HDFS (rm, cp, mkdir,...)
- HDFS does not support non-sequential (random) writes
  - root TFile can't write directly to HDFS fuse, but not really necessary for CMS
  - but files can be read through fuse with CMSSW / TFile - eventually CMSSW can use the Hadoop API
- Random reads are ok

Gridftp/SRM Clients

- Gridftp could write to HDFS+FUSE with a single stream
- Multiple streams will fail due to non-sequential writes
- Brian at Nebraska developed a GridFTP dsi module to buffer multiple streams so that data can be written to HDFS sequentially
- Bestman SRM can perform namespace operations by using FUSE
    - srmrm, srmls, srmmkdir

Caltech Setup

- Current Tier2 cluster runs RHEL4 with dCache. We did not want to disturb this working setup
- Recently acquired 64 additional nodes, installed with Rocks5/RHEL5. This is set up as a separate cluster with its own CE and SE. Avoids interfering with working RHEL4 cluster
- Single PhEDEx instance runs on the RHEL4 cluster, but each SE has its own SRM server
- Clusters share the same private subnet

Caltech Setup

- Namenode runs on same system as Condor negotiator/collector
  - 8 cores, 16GB RAM
  - System is very over-provisioned. Load never exceeds 1.0, JVM never exceeds 200MB
  - Plenty of room for scaling to more blocks
- Secondary NN runs on same system as condor batch worker
- 64 data nodes, 170TB available space
  - Includes 2 Sun Thumpers running Solaris
  - Currently only 4.5TB used
  - All datanodes are also condor batch workers
- Single Bestman SRM server using FUSE for file ops
- Two gridftp-hdfs servers

Deployment History

T2_US_Nebraska first started investigating Hadoop last year. They performed a lot of R&D to get Hadoop to work in the CMS context

- Two SEs in SAM
- Gridftp-hdfs DSI module
- Use of Bestman SRM
- Many internal Hadoop bug fixes and improvements
- Presented this work to the USCMS T2 community in March

Tier2 Hadoop Workshop

- Held at UCSD in early March 2009
- Intended to help get interested USCMS Tier2 sites jump-start their hadoop installations
- Results:
  - ○ Caltech, UCSD expanded their hadoop installations
  - ○ Wisconsin delayed deployment due to facility problems
  - ○ Bestman, GridFTP servers deployed
  - ○ Initial SRM stress tests performed
  - ○ UCSD ⇔ Caltech load tests started
  - ○ Hadoop SEs added to SAM
  - ○ Improved RPM packaging
  - ○ Better online documentation for CMS
- https://twiki.grid.iu.edu/bin/view/Storage/HdfsWorkshop

Caltech Deployment

- Started using Hadoop in Feb. 2009 on a 4-node testbed
- Created RPMs to greatly simplify the deployment across an entire cluster
- Deployed Hadoop on new RHEL5 cluster of 64 nodes
- Basic functionality worked out of the box, but performance was poor.
- Attended a USCMS Tier2 hadoop workshop at UCSD in early March

Caltech Deployment

- Migrated OSG RSV tests to Hadoop in mid-march
- Migrated T1 → Caltech load tests to Hadoop in early April
- Attempted to move one /store/user/$USER directory to hadoop in early April, but failed due to TFC problems

Current Successes

- SAM tests passing
- T1 → Caltech load tests passing
- RPMs provide easy installs, reinstalls
- Bestman + GridFTP-HDFS have been stable
- Great inter-node transfer rates (2GB/s aggregate)
- Adequate WAN transfer rates (200MB/s)

Not without problems...

- OSG RSV tests required patch to remove “:” from filenames. This is not a valid character in hadoop filenames. (resolved)
- Bestman dropped VOMS FQAN for non-delegated proxies, caused improper user mappings and filesystem permission failures for SAM, PhEDEx (resolved)
- TFC not so “t” anymore*
- Datanode/Namenode version mismatches (improved)
- Initial performance was poor (400MB/s aggregate) due to cluster switch configuration (resolved)

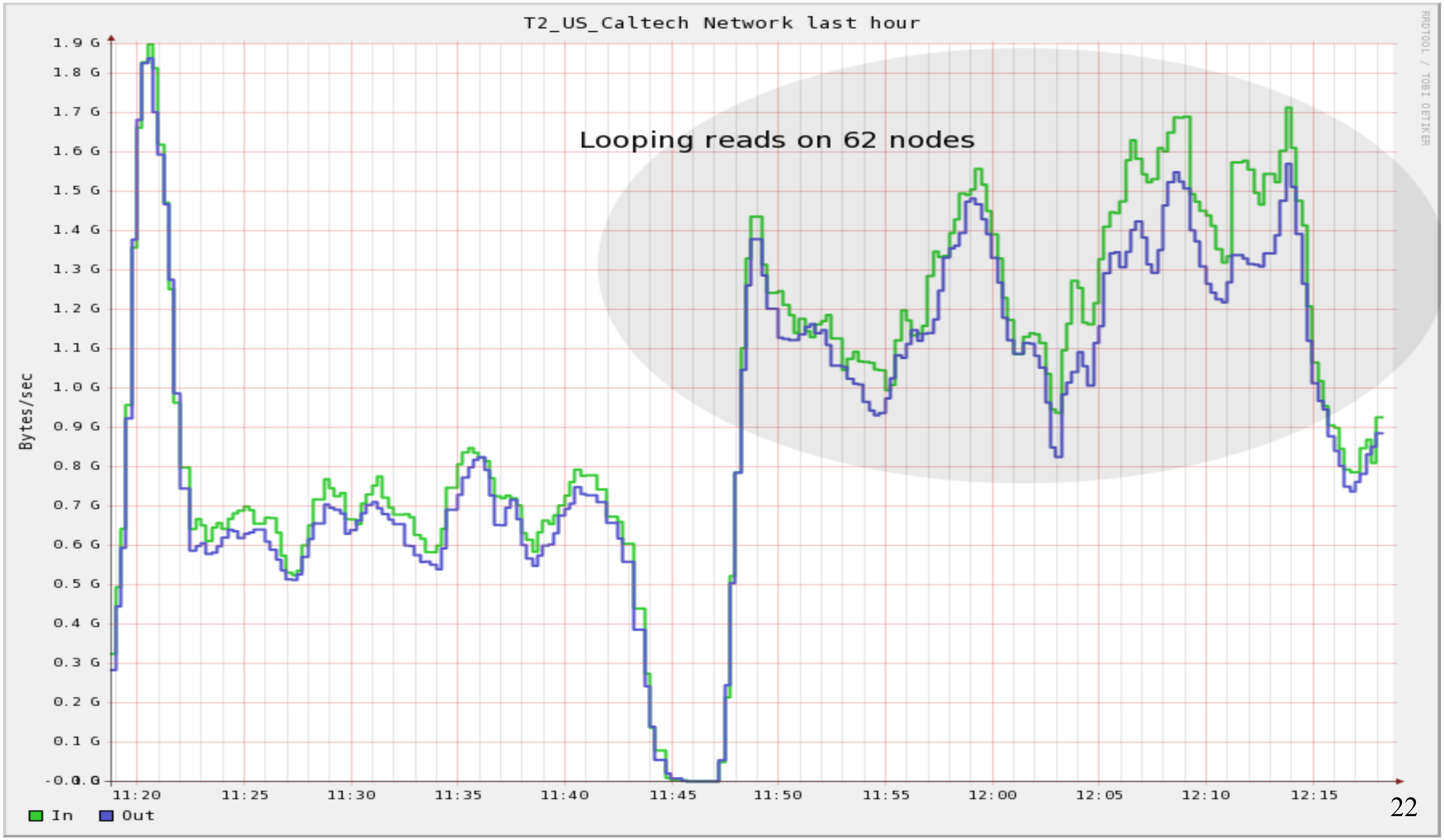
*) TFC = Trivial File Catalog

Not without more problems...

- FUSE was not so stable
  - Boundary condition error for files with a specific size crashed fuse (resolved)
  - df sometimes not showing fuse mount space (resolved)
  - Lazy java garbage collection resulted in hitting ulimit for open files (resolved with larger ulimit)
- Running two CEs and SEs requires extra care so that both CEs can access both SEs
  - Some private network configuration issues
  - Lots of TFC wrangling

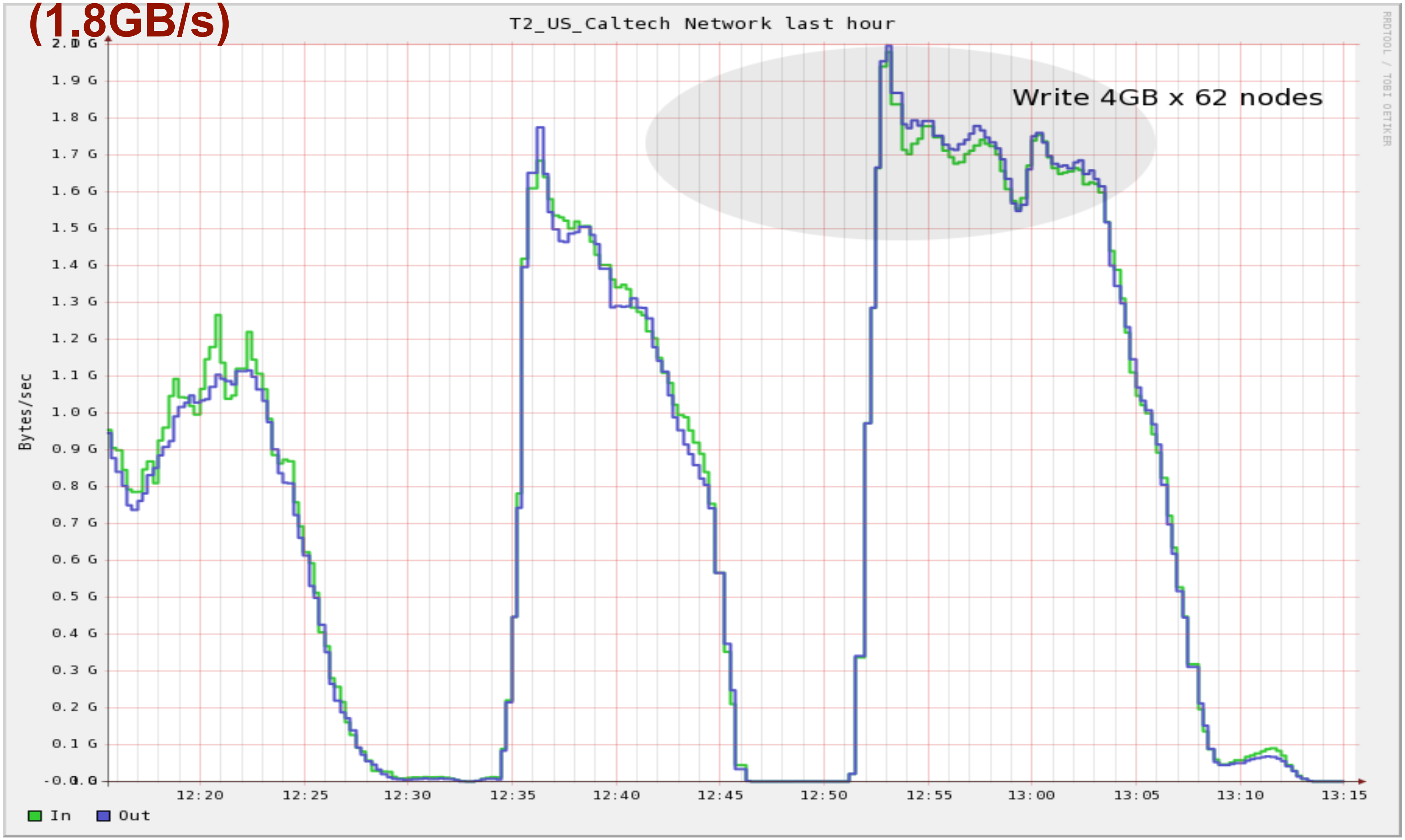
Many Read Processes

Looping reads on 62 machines, one read per machine



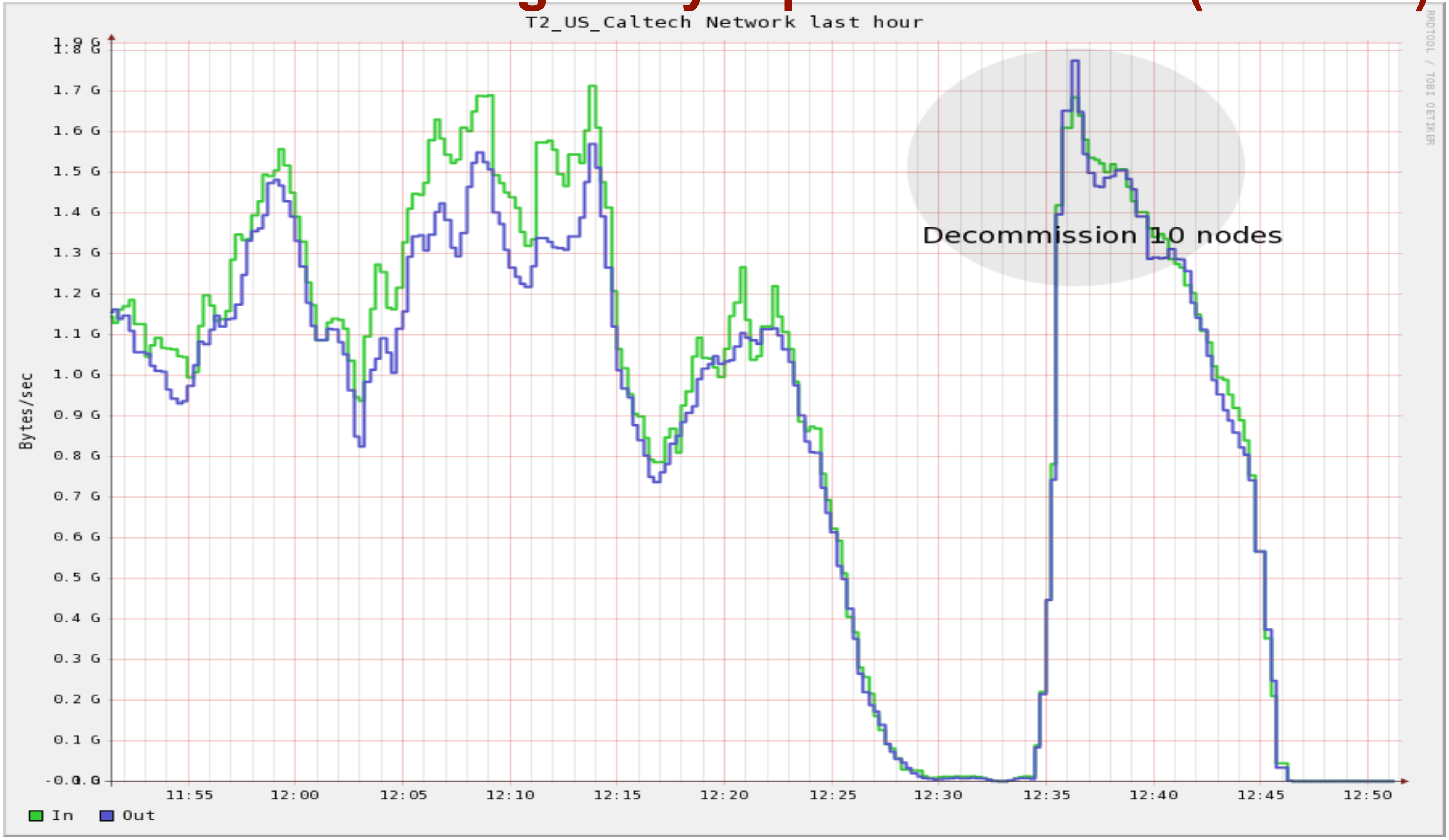
Many Parallel Writes with FUSE

Write 4GB file on 62 machines (dd+fuse) with 2x replication (1.8GB/s)



Replicate by Decommision

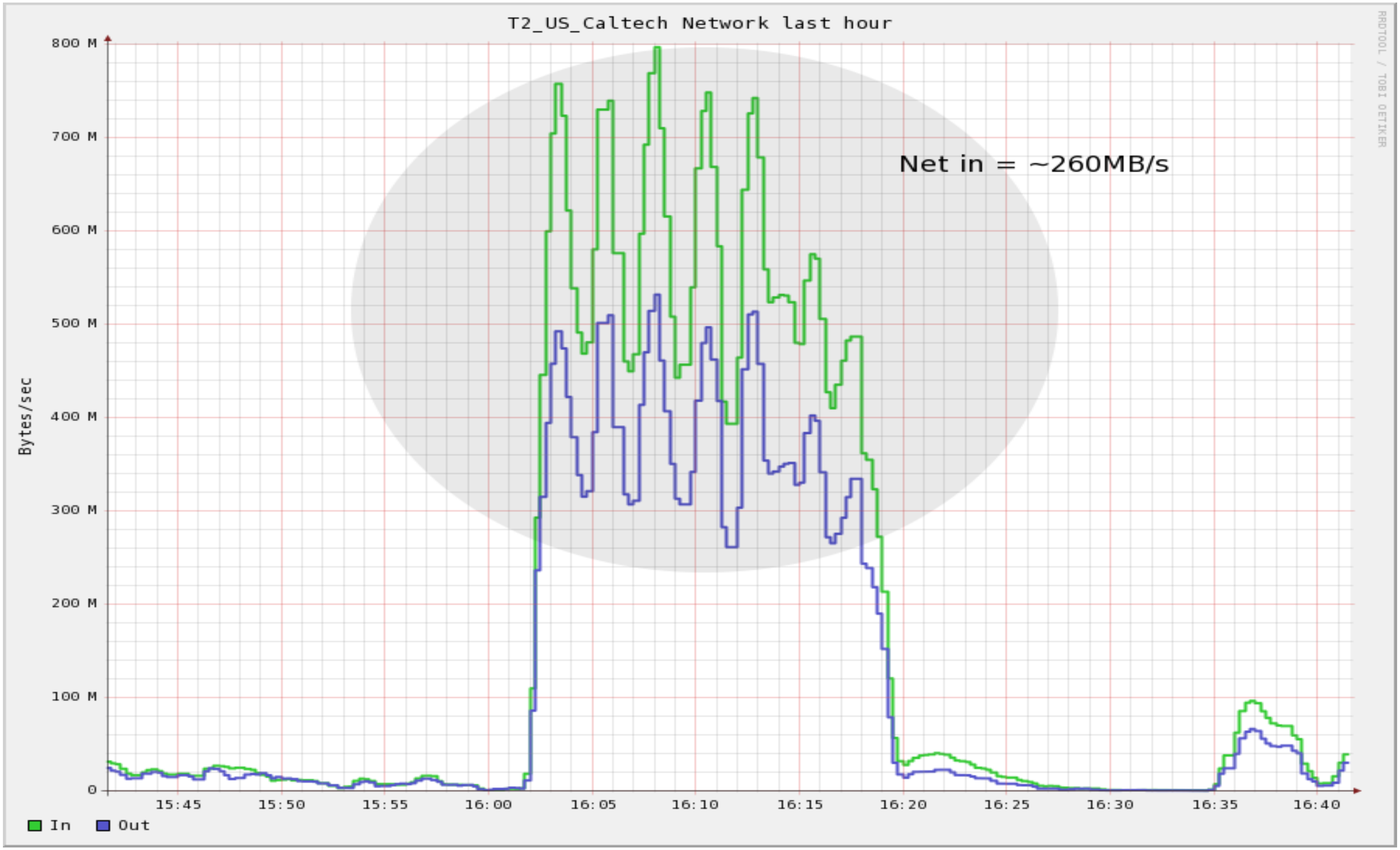
Decommission 10 machines at once, resulting in the namenode issuing many replication tasks (1.7GB/s)


T2_US_Caltech Network last hour

Decommission 10 nodes

UCSD →Caltech Load Tests

2 x 10GbE GridFTP servers, 260MB/s



Net in = ~260MB/s

Next Steps

- Make another attempt to move /store/user to HDFS
- More benchmarks to show that HDFS satisfies the CMS SE technology requirements
- Finish validation that both CEs can access data from both SEs
- More WAN transfer tests and tuning
  - FDT + HDFS integration starting soon
- Migrate additional data to Hadoop
  - All of /store/user
  - /store/unmerged
  - Non-CMS storage areas

Overall Impressions

- Management of HDFS is simple relative to other SE options
- Performance has been more than adequate
- Scaled from 4 nodes to 64 nodes with no problems
- ~50% of our initial problems were related to Hadoop, the other 50% were Bestman, TFC, PhEDEx agent, or caused by running multiple SEs
- We currently plan to continue using Hadoop and expand it moving forward