CMS Use of a Data Federation

Ken Bloom
for the CMS Collaboration
Thanks to AAA collaborators!
October 15, 2013

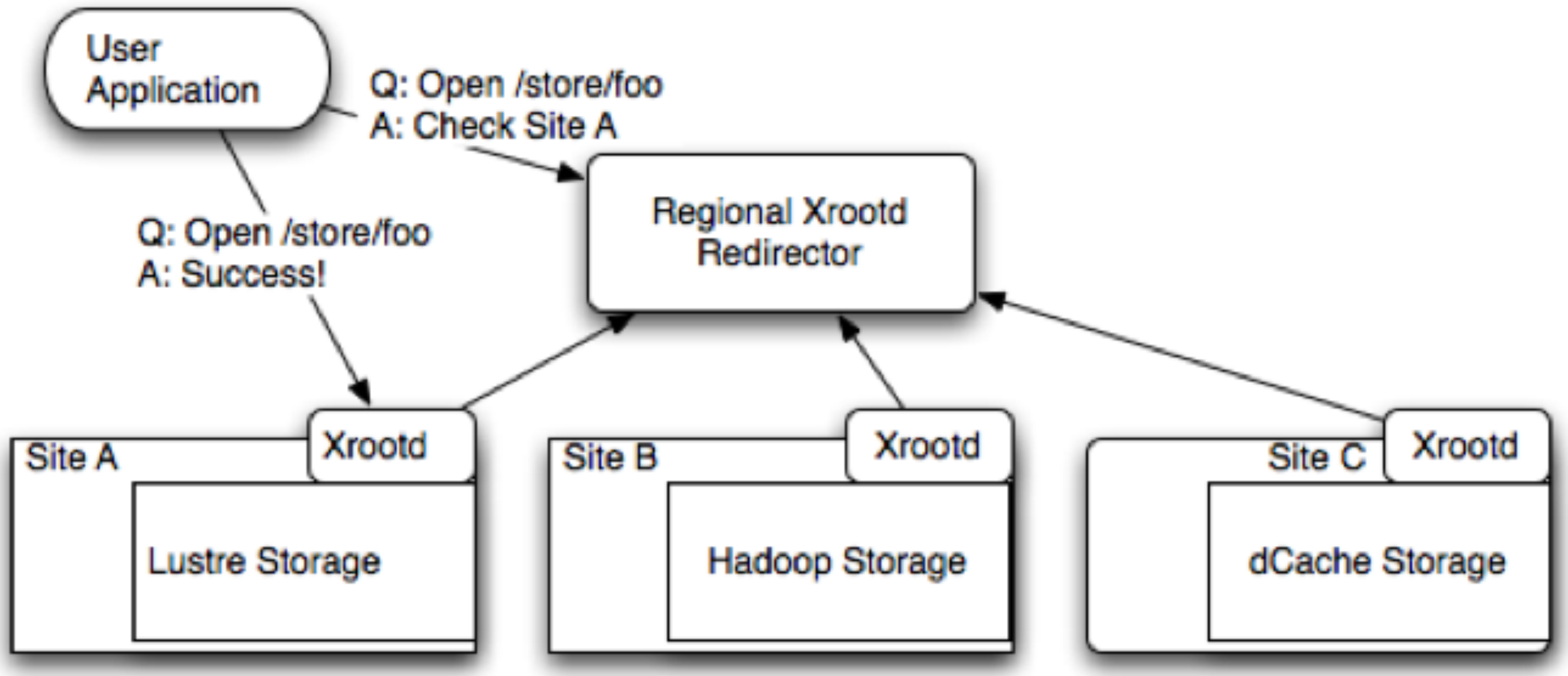
Grid paradigm at CMS so far

- CMS generally moves jobs to data so that the job runs in the same room as the data are stored
  - Design decision from 10 years ago, based on assumption that data transfers are slow and unreliable
- Very successful, but experience has indicated problems
  - Longer queuing times than needed when free CPU not near data
  - Hard to incorporate resources not dedicated to experiment
  - Larger storage requirements than affordable in the future
  - Users prefer to run locally if possible, but might not have the data
- All of this can be solved if we could provide access to any data, anytime, anywhere!

Any Data, Anytime, Anywhere (AAA)

- Goal: make all data even more straightforwardly available to any CMS physicist, anywhere
  - Reliably: no access failures
  - Transparently: never notice where the data actually reside
  - Easily: no operational burdens for physicists to have local access
  - Universally: fulfill the promise of opportunistic grid computing
- Technical solution is federated storage: a collection of disparate storage resources transparently accessible across a wide area via a common namespace
- NSF-funded US CMS effort based at Nebraska/UCSD/Wisconsin to achieve these goals and propagate to CMS as a whole

How it works



- Underlying technology is Xrootd
  - Uniform interface in front of heterogeneous storage systems
- Sites in data federation publish their data to a redirector which can then be queried by applications seeking to access data
  - If data absent in region, fall back to query for data elsewhere
- Access is authenticated

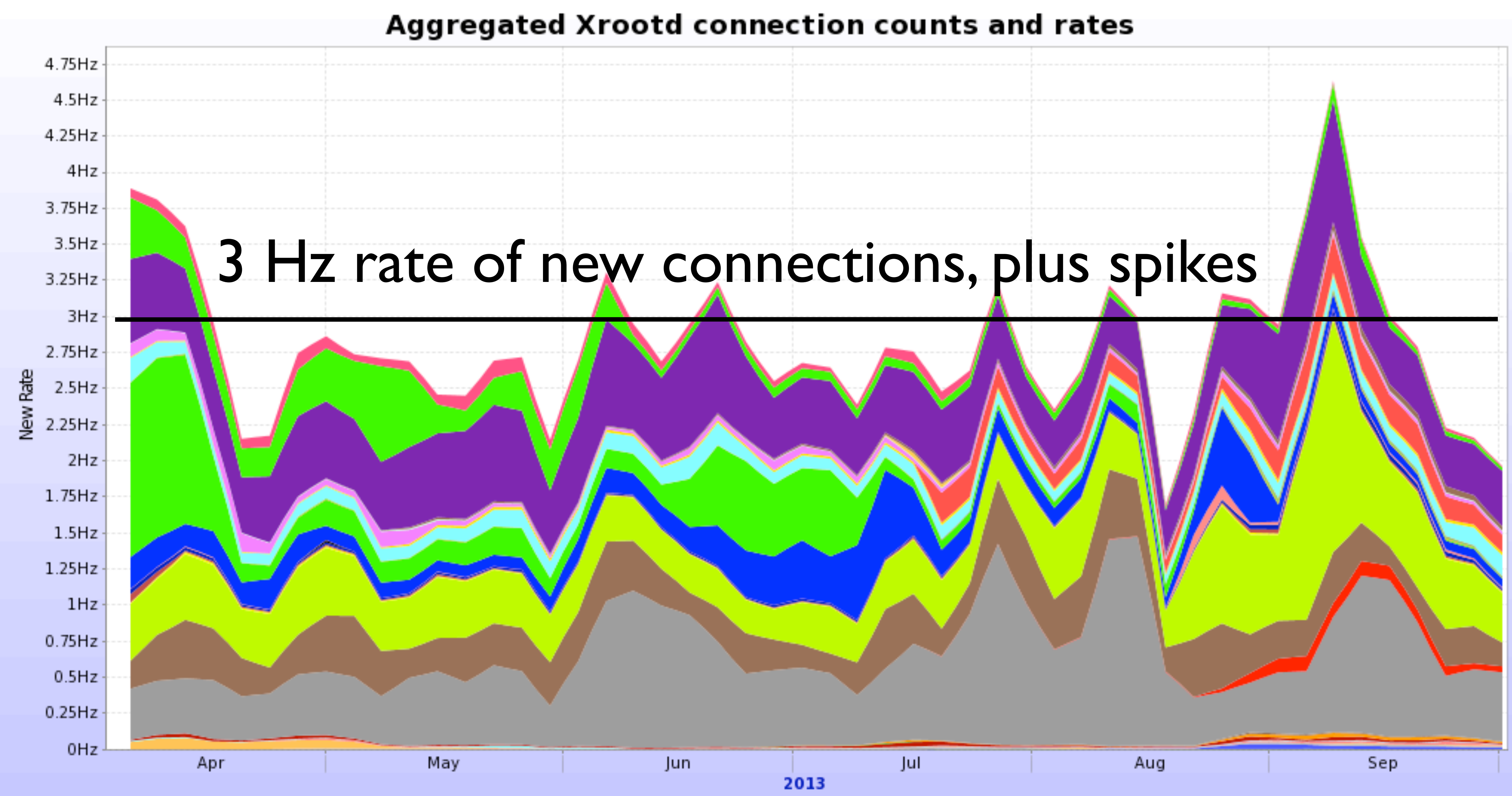
Why it works

- CMS has a globally consistent namespace
  - Physical filename = local prefix + logical (CMS) filename
- Much effort to optimize I/O stack to reduce read latencies
  - Carefully designed data formats to maximize potential for partial reads when data is filtered in analysis
- Wide-area networking has proven to be robust
- CPU efficiency still better for local reads
  - Local: 92% with 0.48 s/event, remote: 86% with 0.65 s/event
  - (based on analysis jobs at Tier-2 centers)
- but AAA technology allows us to make better use of expanding WAN bandwidth

Deployment

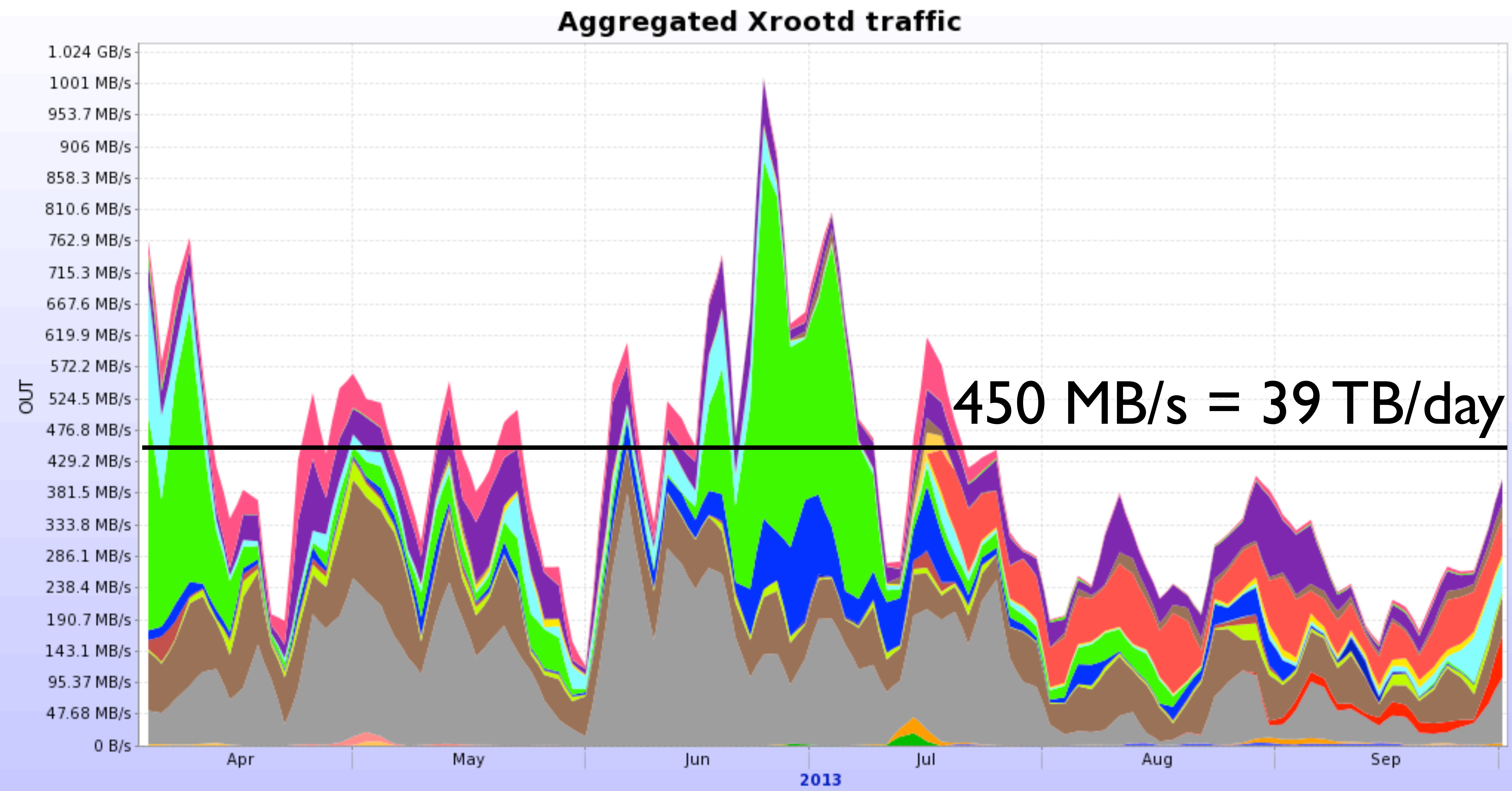
- Goal: all CMS T1 sites, as many CMS T2's as possible in data federation by the start of Run 2
- Currently 3/7 T1's (IT, UK, US) have placed all disk-resident data into the federation
  - Will grow as sites implement disk-tape separation
- 39/51 T2's are federated
  - Missing sites are typically smaller/less performant
  - > 95% of unique datasets resident at T2's are available in federation
- Works with a variety of storage technologies
  - dCache, Hadoop, DPM, StoRM, CASTOR....
- Status of infrastructure monitored through SAM, Nagios tests

Global picture



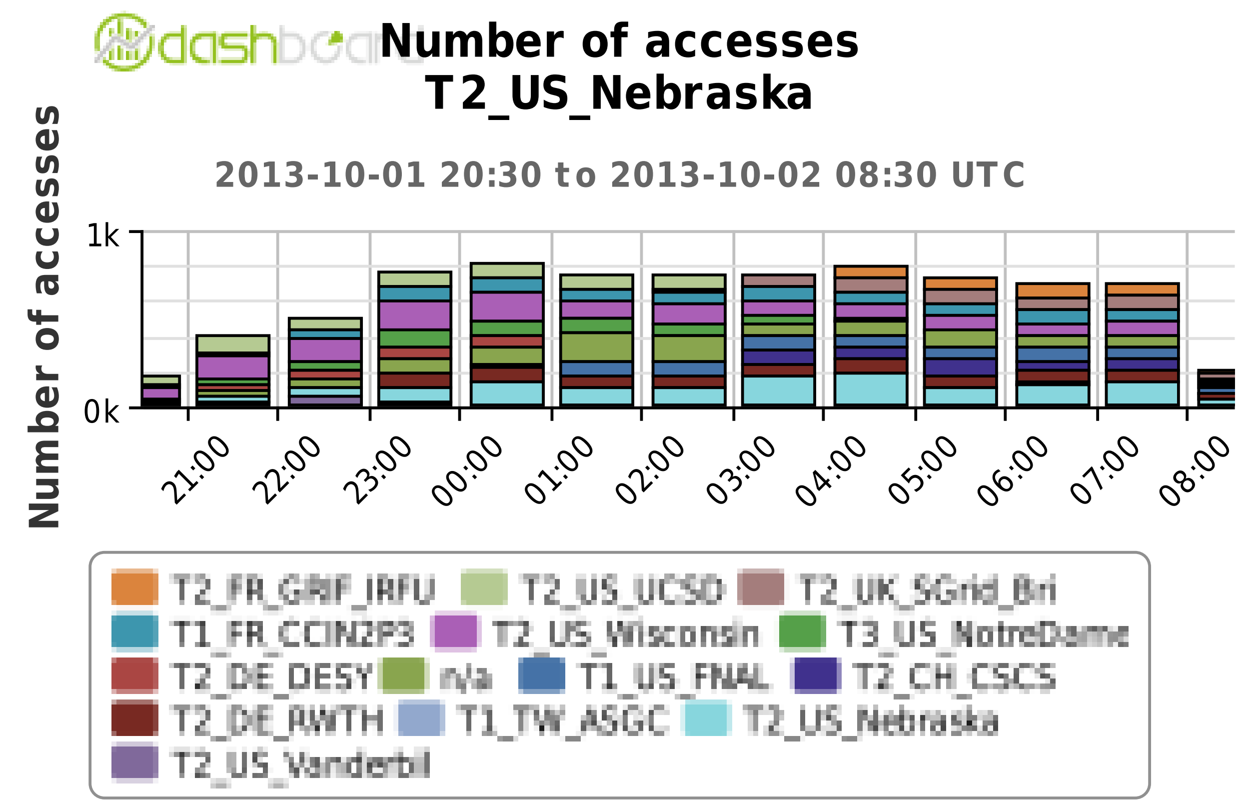
http://xrootd.t2.ucsd.edu

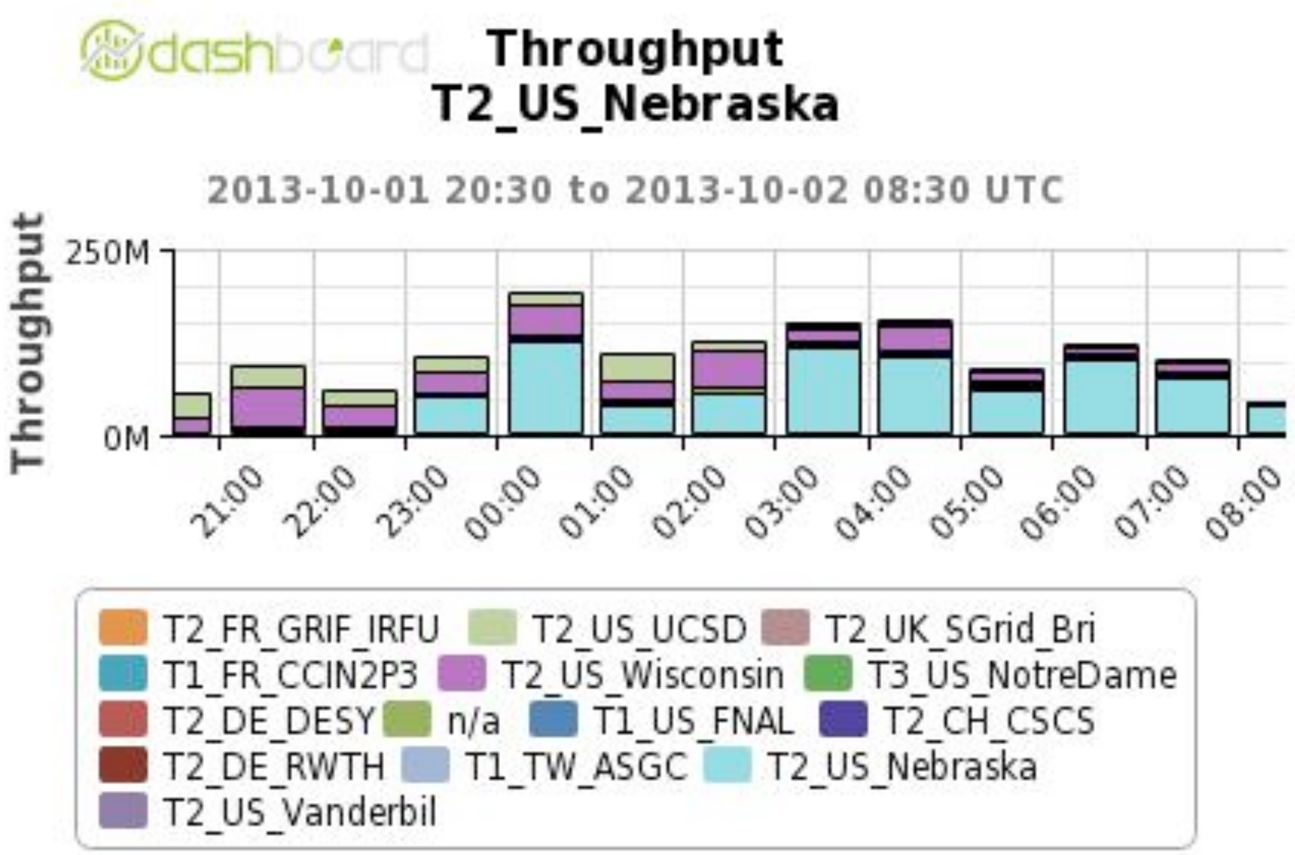
Global picture



For comparison: transfers via subscriptions = 81 TB/day

Detailed monitoring

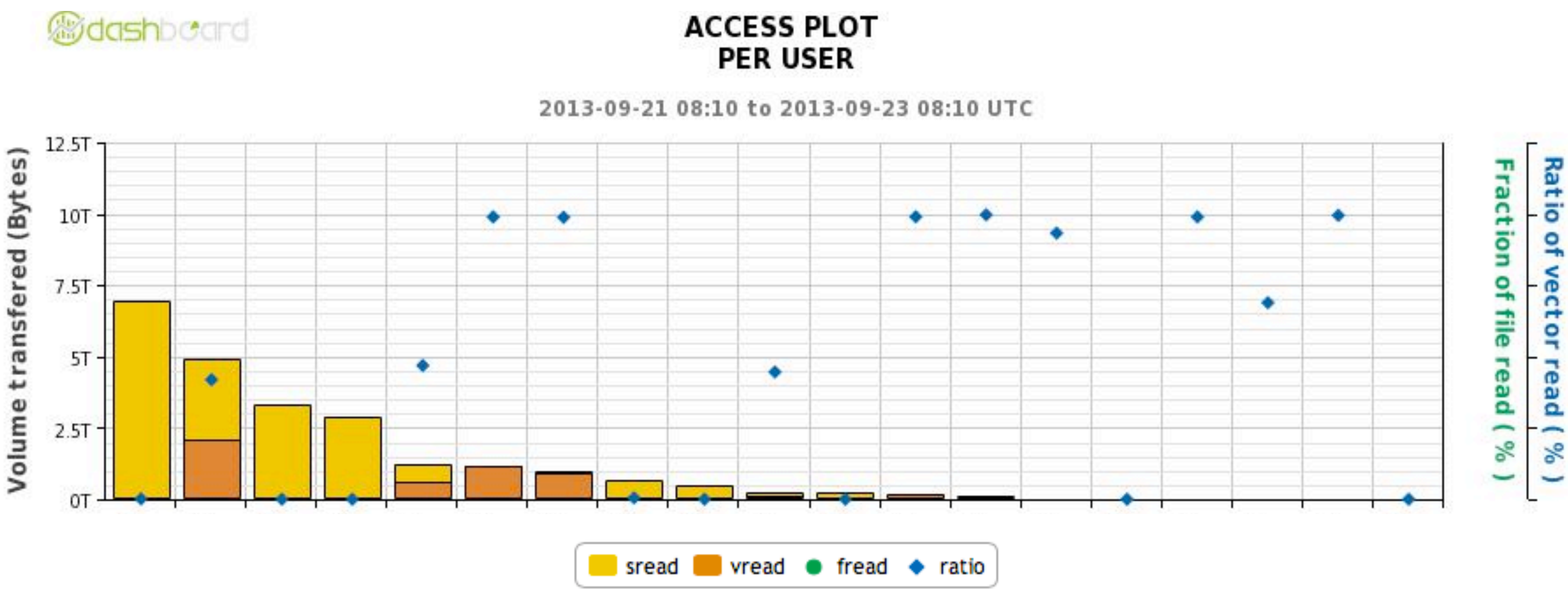




- Additional monitoring provides detailed information about activity in the system at the site level...

Detailed monitoring

http://dashb-cms-xrootd-transfers.cern.ch



Users (names redacted)

- ...and at the user and file level, too
- Only 22 sites are publishing detailed monitoring information, but growing quickly with WLCG release of plugin for dCache systems

Scale and scale tests

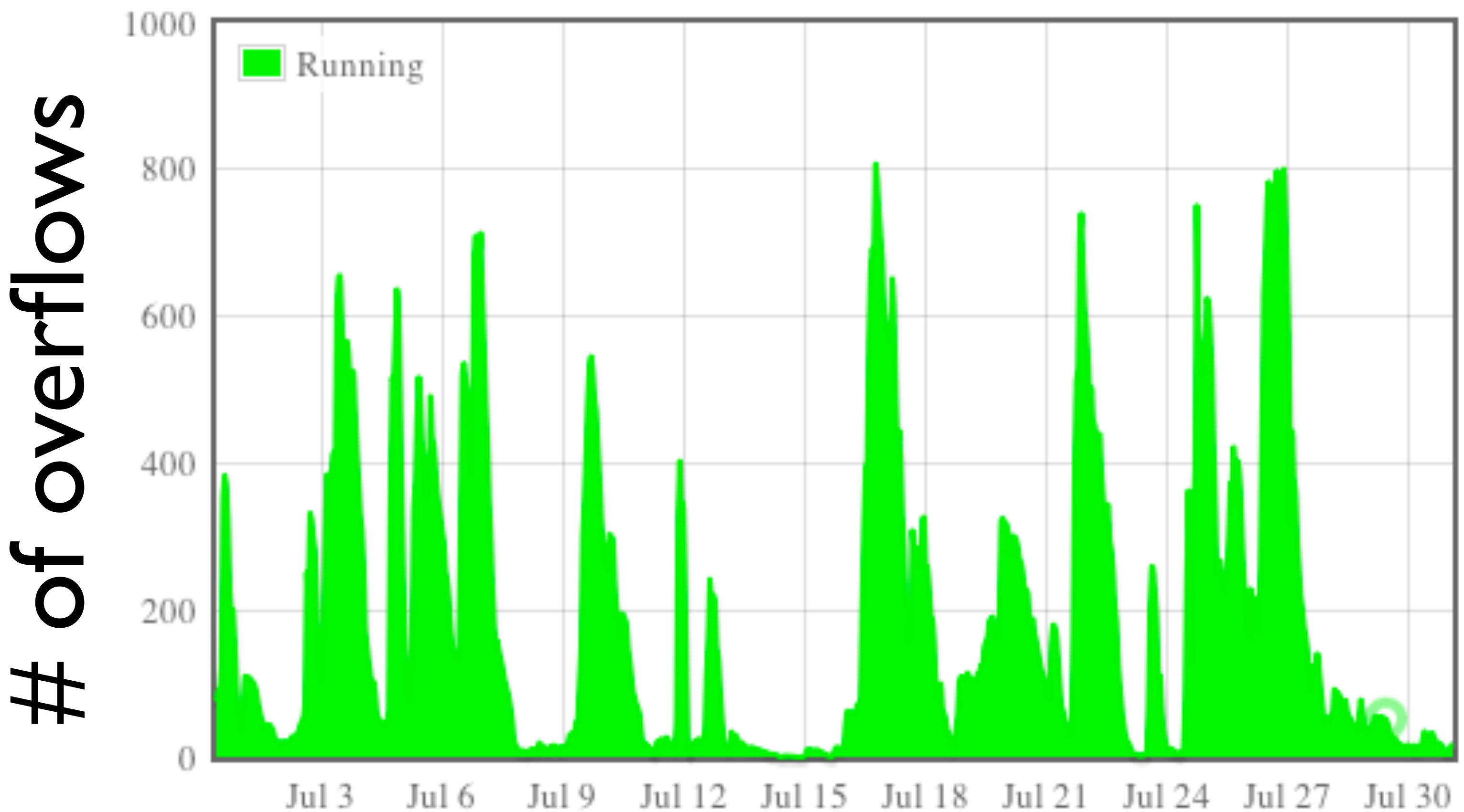
- Daily usage:
  - ~40 TB accessed
  - 5-15 different destination networks
  - 20-30 unique users (changing each day)
  - ~5-10% of user analysis jobs at T2
- Anticipate more growth before start of Run 2 in 2015
- Scale tests underway of redirectors, Xrootd servers, etc.
  - Have demonstrated that redirector can handle 10 Hz rate of file-open requests, OK for steady-state load

Applications: system robustness

- Usually, if a job fails to open an input file, it crashes
- AAA cures this via the "fallback mechanism"
  - On file-open failure, CMSSW asks redirector to find file elsewhere
  - Job then reads remote file, user never notices
- More throughput for users, less CPU time wasted on failed jobs
- Makes entire system more robust against single-site storage issues

Applications: efficiency for users

- Sites with popular datasets can have very long batch queues
- Re-direct jobs to another site with free job slots, read data via AAA
  - Smaller CPU efficiency, but jobs can start sooner
  - Achieved by changing scheduling policies in glideinWMS layer, regulate number of jobs to match WAN bandwidth
- So far, only small scale -- overflow amongst four sites in the US, ~O(1K) simultaneous jobs -- but eager to expand soon

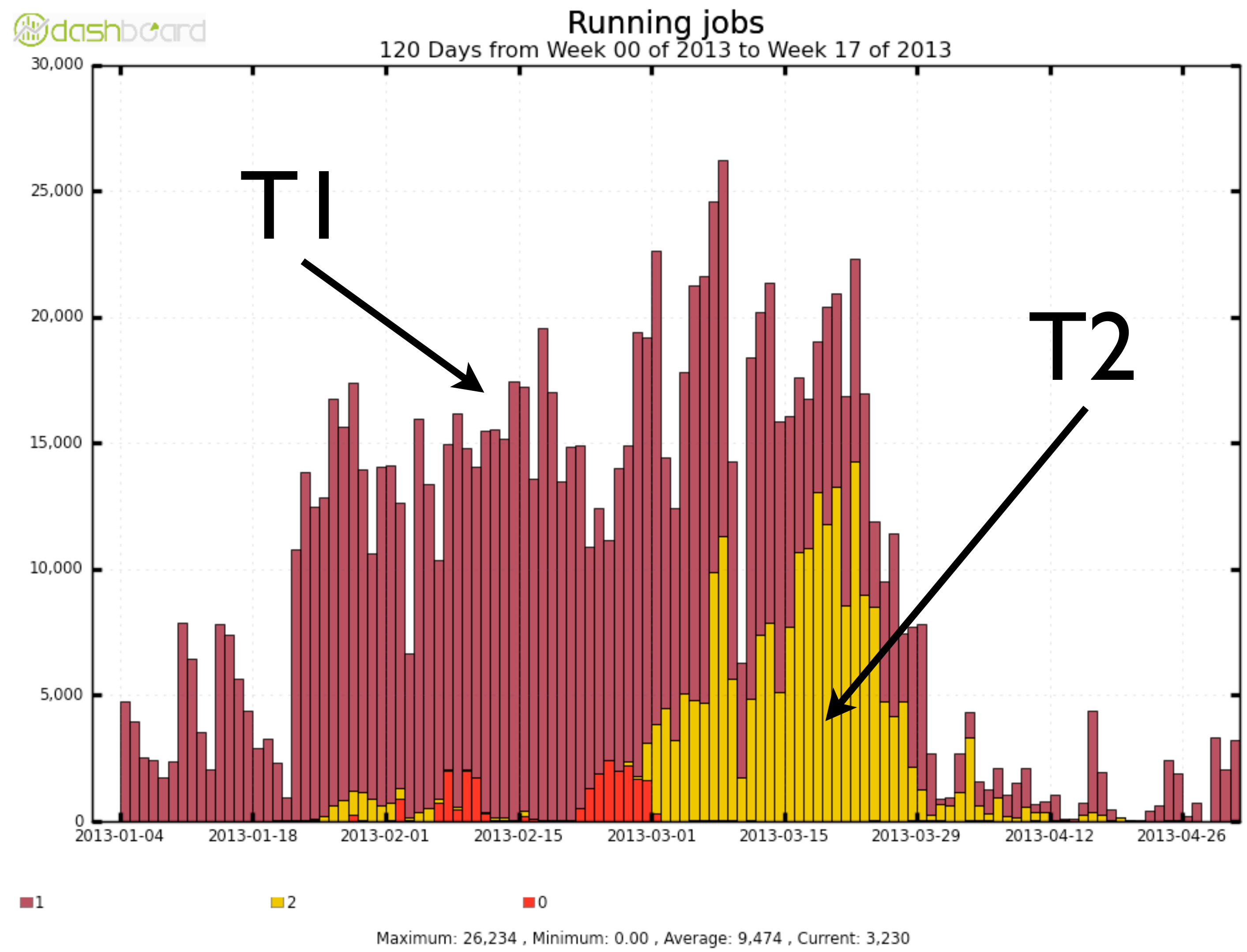


Applications: sites without data

- Some T3 sites are completing entire data analyses through AAA
  - Observed ~800 simultaneous jobs, 2-3 Gb/s WAN input sustained for a week, 99% success rate
  - Much satisfaction with local control over processing resources
  - "At this point, I basically don't pay attention to where the data is and just assume that jobs will find the data and run."
- Exploring possibility of diskless T2 sites at well-networked centers
- Sites that temporarily lose their data due to storage downtime (planned or unplanned) can continue to operate as normal through the fallback mechanism
  - Allows the continuity of processing capacity, system-wide

Applications: production with remote data

- “Legacy” reprocessing of 2012 data and associated simulation samples
- Inputs resident at T1 sites
- T1’s ran on data locally
- T2’s ran on simulations read via AAA
- Whole job done faster

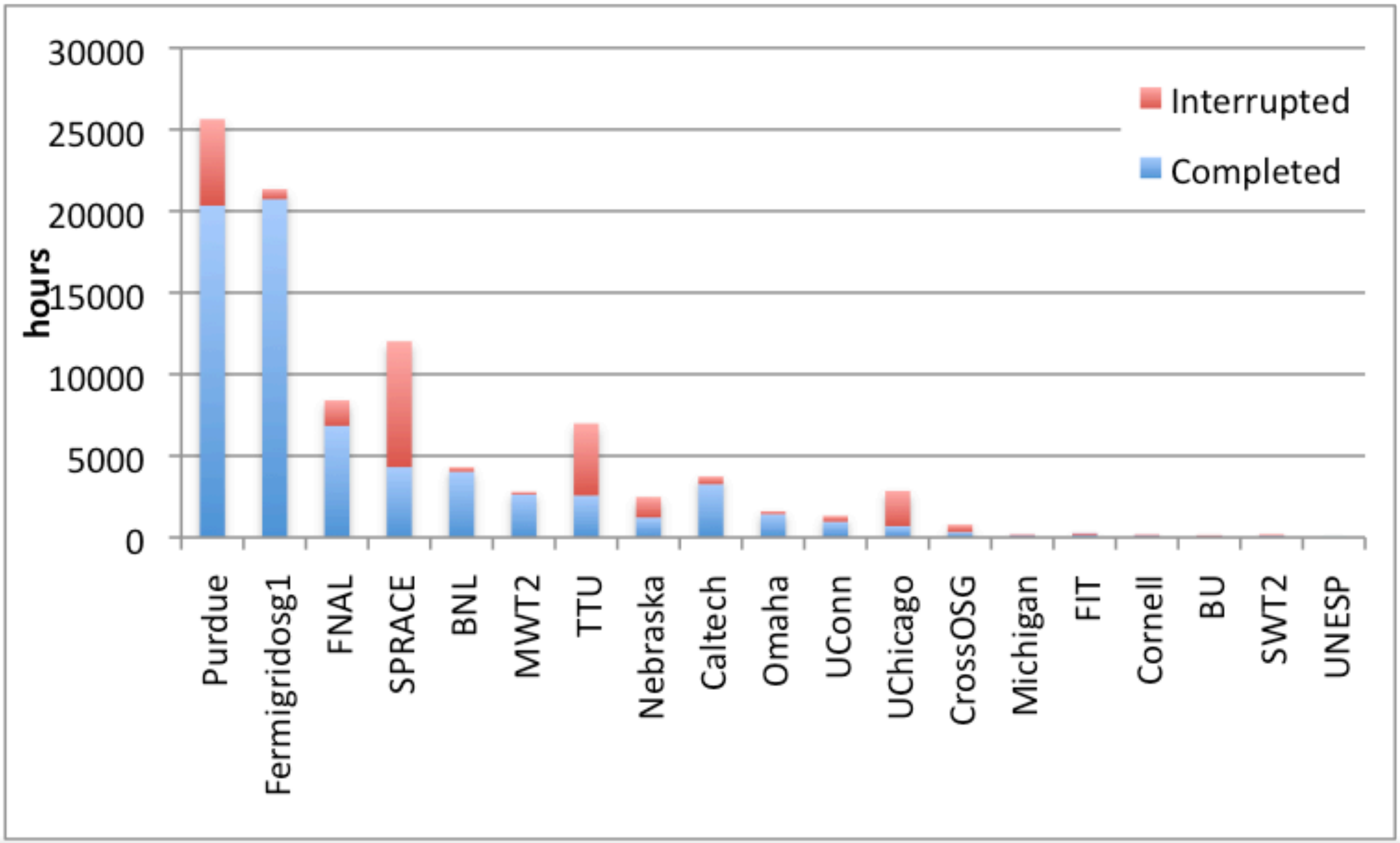


Applications: opportunistic usage

- Any data, anywhere means any computer, not just CMS-owned
  - For software, use Parrot and CVMFS for download on demand, brings in 500 MB of files rather than 17 GB
  - Then, read data through AAA fallback mechanism
  - Typical jobs only 2% slower than those running on CMS sites
- Opens the door to any opportunistic resource, e.g. clouds
  - Successful demonstration on Amazon cloud
  - Much CMS development work underway [CHEP presentation]

Applications: opportunistic usage

- Successful use of opportunistic resources on OSG: have run 2K simultaneous jobs across 15 sites, starting at a rate of 3 Hz
  - Total usage so far 1.2M CPU hours, some taken from ATLAS sites



Developments in progress

- Healing broken local files
  - If bad block is encountered mid-file, read and copy a working version from elsewhere; HDFS prototype exists
- Totally dynamic caching
  - When a site requests a file not available locally, the remote file is not just read, but copied into a dynamic cache
  - Opens the possibility of cache-only storage at sites
- Network-aware routing for initial redirection, then file streaming
- Data-aware scheduling
- Determine data popularity from monitoring, drive automated data placement and deletion off that

Outlook

- By deploying a data federation, CMS has made virtually all of its data available to all of its users, anytime, anywhere
- Widespread deployment at CMS, moving towards widespread use and acceptance
- Applications such as fallback, overflow, and diskless centers have already allowed more efficient use of existing resources, and open the door to greater use of opportunistic resources
- Coming developments will lead to more stable operations and even greater efficiency of resource use
- Empowers users to access the data when/where/how they want
- All of this supports the ultimate goal of CMS computing: let us get the physics out fast -- and first

Any Data, Anytime, Anywhere!