Storage at UW-Madison CMS Tier-2

Will Maier
wcmaier@hep.wisc.edu

University of Wisconsin - High Energy Physics

OSG Storage Forum, U. Chicago

- dCache 1.9.5-8 (Golden Release)
  - Configuration available online [1]
- dCache SRM/GridFTP/dcap doors
- Xrootd Demonstrator instance
- Scientific Linux 5.3

[1] http://code.hep.wisc.edu/dcache-cftree

- Four core dCache machines: PNFS, admin/misc, doors, hotspare
- 1 Gbps connections, SATA disks, 2x2 2.0 GHz Opterons, 16 GB RAM
- PNFS: RAID10 (10kRPM disks), 2x4 3.0 GHz Xeons
  - Current configuration ∼12 months old
  - Nearly saturated (again)...need more IO
- admin, doors, hotspare: commodity, nothing special
  - No observed scaling issues with shared SRM/dcap door
  - ∼32 GridFTP doors live on data nodes (easy to add more)

- ∼50% dedicated, ∼50% shared with compute resources
- Dedicated nodes: mix of RAID6, RAID0 with 1 - 2 TB SATA disks
- Shared nodes: no RAID, .5 - 2 TB SATA disks
  - Co-located with 4 - 16 slots and 2 - 48 GB RAM for computing
- ∼950 raw TB, ∼450 writable TB, 908(ish) pools, 210 nodes
  - One pool per disk, mostly
  - 100% increase over 2009
- Redundant 2x10 Gbps uplink (and several stacks of Cisco 3750s)
  - Managed by campus
  - Eliminated a few Etherchannels connecting racks
  - Upgraded rest to 10 Gbps bundles
- Burn in new hardware using stressapptest [2]

[2] `http://code.google.com/p/stressapptest/`

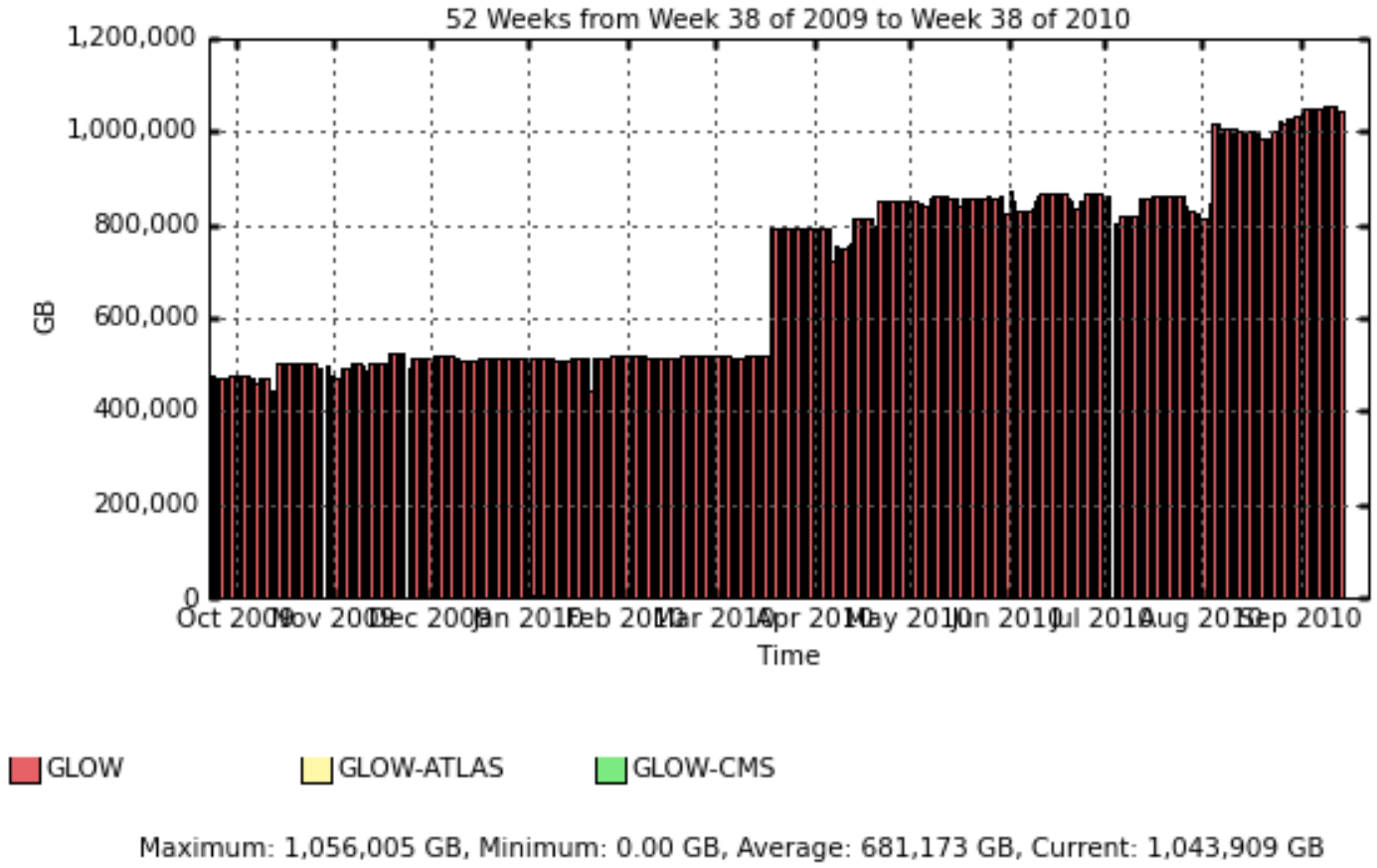

Figure: Total dCache (TB), 2009-2010

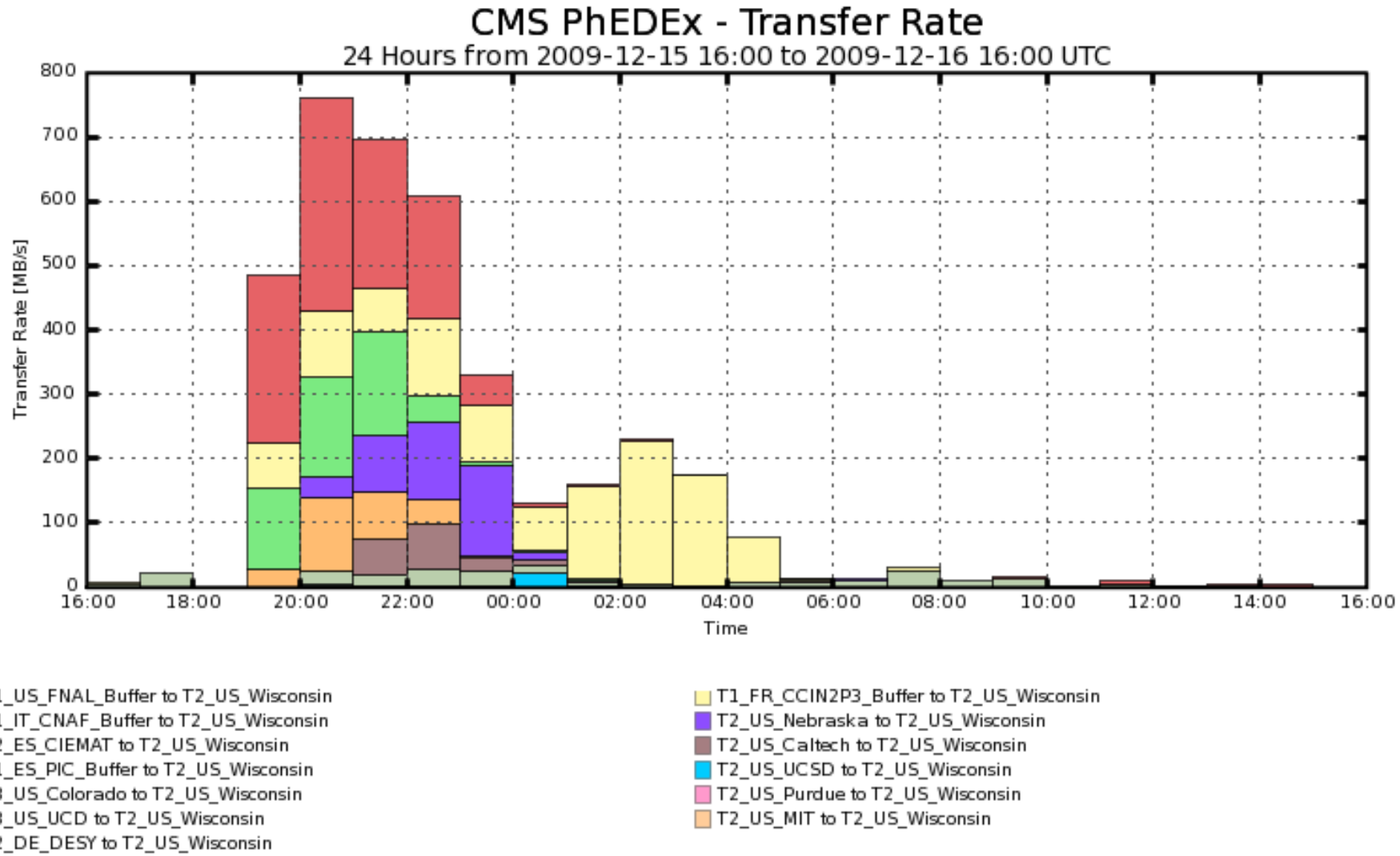

Figure: PhEDEx peak transfers, late 2009

- Install nodes with kickstart, manage live configuration with Cfengine
- Management scripts written with dcache-tools [3], based on CLI [4]
  - `dcache_absent`: List absent and offline pools
  - `dcache_billingrep`: Watch billing log and replicate new files
  - `dcache_clean`: Remove invalid/unlinked files directly from pools
  - `dcache_df`: `df(1)`-like overview of cluster storage
  - etc...
- Less web scraping, more dCache info provider
  - Previously, monitors/checks polled dCache web pages and scraped HTML
  - Info provider allows simple XML parsing (and exposes lots of detail)
  - Can request subsets of data (ie for `dcache_df`)

[3] `http://code.hep.wisc.edu/dcache-tools`
[4] `http://packages.python.org/pyCLI`

- Nagios checks for all servers in both the core and cluster
- Some Nagios protocol checks (GridFTP, SSH admin interface)
- Functional tests via cron jobs (SRM/dcap/GridFTP read/write)
- External monitoring (CMS SAM, JobRobot, Dashboard, SiteView, PhEDEx; OSG RSV, MyOSG, Gratia; . . .)
- Aggregate local monitoring in tsar [5]
  - Ingest time series data from simple collectors
  - Supports HTTP, other submission/query protocols to come
  - Data in memory for fast writes/reads (Redis [6] backend)
  - JavaScript, Python clients for visualization [7]

[5] http://code.hep.wisc.edu/tsar
[6] http://redis.io
[7] http://tsar.hep.wisc.edu/plots/dcache

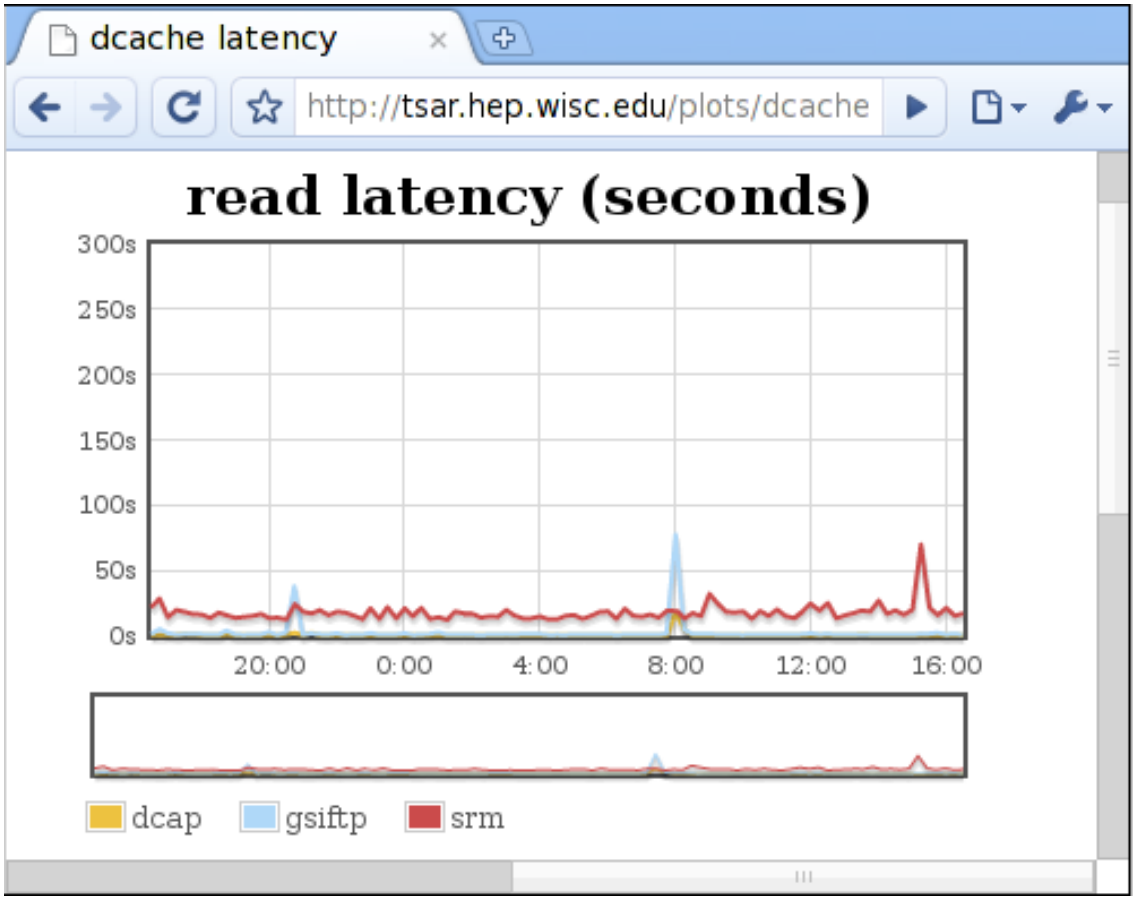

Figure: 24 hours of dCache latency checks in tsar, 2010.09.22

- Close network collaboration with University
- Co-locating storage and compute services on commodity hardware
- dCache (stability, throughput, OSG/FNAL support)
- Increasingly centralized monitoring and reporting

- Disks
- Monitoring fragmentation
  - Takes Firefox nearly two minutes to load them all on a netbook
- dCache (hotspots, replication, commodity data servers)
  - Model mismatch: small precious storage with large tape backend vs precious storage (no tape)
  - We've written thousands of lines of code to make running dCache at our T2 livable

- Centralize dCache logging
- Scale tsar
- Transition to alerting based on trends (instead of binary checks)
- Explore HDFS w/ xrootd frontend (to support jobs on local opportunistic resources)
- PNFS $\rightarrow$ Chimera? dCache $\rightarrow$ HDFS?
- Cfengine 2 $\rightarrow$ Cfengine 3/Puppet/Chef/...?
- Expand use of RPMs (for improved atomicity?)