Storage at UW-Madison CMS Tier-2

Will Maier
wcmaier@hep.wisc.edu

University of Wisconsin - High Energy Physics

OSG Storage Forum, FNAL

- ~500TB of writable disk, 520 pools, 190 nodes
  - Nodes host 2-4 pools each
- Most of our network architecture is provided by campus, but we manage the switches
- We emphasize reliability and performance on central servers
  - Reliability first: performance means nothing if the systems aren't up...
  - ...but work doesn't get done if clients are waiting to perform lookups on the central services
- Make use of any and all available machines as cluster nodes
  - Dedicated servers with large, local filesystems
  - Batch nodes
  - Retired test systems

- 10Gbps fiber uplink to campus, world
- 1Gbps Ethernet to all nodes and central servers
- Three stacks of Cisco 3750 switches
  - Main stack bridged to others via 4Gbps Etherchannel
  - Most connections use the stack backplanes; some node-to-node and node-to-WAN connections use the Etherchannels
- ∼300 TB and 90 nodes on one side; ∼200 TB and 96 nodes on the other

- Fast RAID for namespace database
  - Need speed: database journals require lots of writes
  - Need reliability: lots of pain if the database dies or becomes corrupted
  - At Wisconsin: LSI RAID10 (4x250 GB 10k RPM SATA disks)
- Databases and dCache daemons are happy to make use of available memory and cores
  - At least 2GB/core; most servers have 16GB for 4 cores
- All central servers on UPS; can survive short outages or shut down gracefully
  - Filesystem corruption hurts
- Otherwise, commodity hardware
  - Fewer configuration profiles to manage
  - Standard 7200 RPM SATA disks sufficient; no RAID (only namespace needs to persist)

- Majority of storage on whitebox, dual-purpose batch and storage nodes
- Nodes stay in the cluster until it's too expensive to keep them running
  - With five year warranties on disks, nodes can last a long time
  - RAM upgrades or motherboard failures aren't worth the trouble if the node is out of warranty (standard three years)

Generation	Disk (TB)	RAM (GB)	Cores	Count
1	1	2	2	32
2	1-1.5	4-16	4	69
3	3	16	8	32
4	4	16	8	32

Table: Wisconsin Dual-Purpose Cluster Nodes, 2005-2009

- Dedicated storage
  - Apple Xserve RAID with fiber channel to commodity controller node
  - Whitebox with 24 local SATA disks, LSI RAID6

Generation	Disk (TB)	RAM (GB)	Cores	Count
1	9	2	2	9
2	24	16	8	10

Table: Wisconsin Dedicated Storage Cluster Nodes, 2005-2009

- Originally six central servers, with one service on each machine
- Now, dedicated nodes for PNFS, 'admin' services, SRM/dcap
- Hotspare system running and ready to cover for any of the above
- PNFS
  - companion database
  - PFM replication (for fast namespace lookups)
- admin
  - http monitor, admin interface, companion and billing databases
  - Admin interface configured with SSH keys
  - `billingrep` live replicator (for access to the billing log)
- SRM/dcap
  - Only one door for each, and they live on the same machine
  - Haven't observed performance problems
- Hotspare
  - `/opt/d-cache` already present
  - Ready for quick redeployment of a failed central service

- 30 GridFTP doors scattered across nodes
- Configure dCache JVMs so that 1.5GB RAM/core remains
  - In most cases, four pool daemons, each with 400M JVMs
- Most storage nodes also run Condor (one batch slot per core)
  - Jobs are almost always running and fetching data
  - dCache pushing files
  - No bottlenecks (yet) on the nodes, but the Etherchannels are troublesome

- `/opt/d-cache` versioned by Mercurial, synced from AFS by CFEngine
  - Clean upstream branch; local branch with changes
- Configuration and installation automated by CFEngine
- CFEngine also installs extra RPMs, mounts PNFS, etc
- CFEngine handles upgrades, too:
  - Merge new `/opt/d-cache` with local
  - Turn off services
  - Push updates to all nodes and run `install.sh` (CFEngine)
  - Start services; revert to old `/opt/d-cache` if necessary
  - To roll back, revert to last known good `/opt/d-cache` and server RPMs

- Since we store data on commodity hardware (with no RAID), we make copies at the cluster level
- dCache's Replica Manager couldn't keep up with the flood of pool messages
- PFM performance slows with lots of pools and files
- `billingrep` for low-latency, first order replication
  - Watches billing log for file creations
  - `pp get file` to a random pool; replicated in seconds
  - Not aware of pool cost/availability; doesn't recover if replicas disappear
  - `http://code.hep.wisc.edu/dcache-tools`
- `pfm` for accurate policy enforcement
  - Walks PNFS namespace (∼10 minutes for 300k files), talks to each pool (20 minutes for ∼200 active pools)
  - Finds available replicas for each file and adds or removes replicas depending on policy
  - Policy defined by regular expressions matching logical file names
  - At Wisconsin: No more or less than two replicas for each file

- Nagios
- SAM, RSV
- root-owned files/directories
  - `find /pnfs/hep.wisc.edu/store/ /pnfs/hep.wisc.edu/osg/* -user root 2>/dev/null`
- dCache Health Check
  - Transfer from each GridFTP, dcap and SRM door
  - Write new files into dCache; read them out and compare checksums
  - Test transfers to and from FNAL via SRM
- Stuck transfers
  - Scan active transfers page for transfers with no significant activity
  - Often indicates unavailable files or broken pools
- Per-directory disk usage
  - Walk PNFS and report disk usage (including replication) for the top directories
- Absent pool report

- Very few files lost (thanks to replication)
- Good performance in LoadTest
- Without fast disks on PNFS node, transfer pileups
- Merging lots of small files hurts
- Highly efficient analysis of large files (relatively small overhead)

- Most local workflows involve extended analysis of large files or merging many small files
- Analysis:
  - dCache works well without much modification
  - Small tranfers overhead for small number of transfers, dcap provides fast access
  - Relatively few outputs for each job
- Merge:
  - More common in SLHC workflows (and increasingly common in the future?)
  - Most local approaches merge numerous small files in serial
  - Unavailable files wait for timeout; during peak usage, time to fetch available files exceeds reasonable timeouts
  - Improving PNFS performance helps, but this workflow is still inefficient on dCache
  - Parallelizing merge process helps, too

- Expand UPS coverage
- Local test stand/verify upgrades
- Add dcap doors
- Improve switch port efficiency so all nodes communicate across the same 16Gbps backplane
- Point `billingrep` at database, not log
- `pgpool` replication of PostgreSQL databases
- Centralize databases on high-performance server or provide faster disks on all central servers