Multi-Tiered Storage with Xrootd at ATLAS Western Tier 2

Andrew Hanushevsky
Wei Yang

SLAC National Accelerator Laboratory
Why Divide Storage into Tiers?

- ALTAS production jobs stage input files to batch nodes, **BUT** analysis jobs read directly from Xrootd storage
- Need high performance storage to serve the random/sparse IO from analysis jobs
- Data becomes cold quickly

Questions:
- Do we really need to keep on buying high performance storage for all data?
- Can we divide storage into tiers and federate them?
Data is cooling down ...

Age: Last access time in days
Average over 17 identical Thumpers/Thors

Age: Last access time in days
Over 30 days
Design a Two-Tier Storage for ATLAS?

Top tier entrance:
- GridFTP data import (over WAN)
- Direct reading by analysis jobs
- All job outputs

All tier entrance sees all storage:
- SRM & Data management
- Data stage-in by production jobs
- GridFTP data export

Storage cluster with high performance configuration

Storage cluster configured toward maximizing capacity

Internal data flow

2012-05-22
Design details

• Uniform name space
• Data move between storage tiers automatically
  – Top moves cold data to back when space is needed
  – Reading at top triggers data stage-in from bottom
• Data movement is driven by the top tier
  – transparent to jobs, use Xrootd/FRM
• Validate checksum for all data movement
File Residency Manager

• FRM glues the two storage tiers together
  – It is Xrootd’s generic virtual remote storage interface.
  – What it does? Stage, Migration, Purge

  **Stage:** `frm.xfr.copycmd` in *your_stage-in_script* `$LFN` *other_args*
  **Migration:** `frm.xfr.copycmd` out *your_migrate_script* `$LFN` *other_args*
  **Purge:** `frm.purge.policy` *1024g 2048g hold 48h polprog*
    `frm.purge.polprog` *pfn | your_purge_script* *other_args*

  “stage” will put open() on hold until file is staged

• Simple to use.
  – Scripts. You have full control
  – Scripts’ return code and stdout will be checked

• Checking status:

  `frm_admin –c config_file` *query all* `Lf` `fn` `qwt`
Implementation at SLAC

• Top Tier storage
  – 17 Thumpers/Thors with Solaris/ZFS
  – 1TB /2 spindles, raid 1 (mirror), 7200rpm SATA
  – Total 748 spindling disks, ~340TB usable
  – Run xrootd/cmsd/FRM

• Bottom Tier storage
  – 5 hosts, each 6x 30-disk raid 6 + 1 hot spare
  – 1 host, 6x 15-disk raid 6 + 1 hot spare
  – 2TB/spindle, 5400rpm SATA
  – Total 990 spindling disks, ~1.6PB usable
  – Run xrootd/cmsd
Activity at Bottom Tier

Activity at two time spots 10 minutes apart: 15:50 16:00

Data out of the bottom tier
Data to the bottom tier
Cost (delay) of Stage-in

~400 running jobs
Other benefits of tiered storage

• Outage of a node in bottom tier has very little impact to the operation

• Allow scheduling top tier node outage without down time
  – Need to vacation the data, could be days

• Data constantly spread across all top tier nodes.
  – Natural avoids hot spots.
  – Bottom tier doesn’t have to have more space
Operational Experience

• Moving small files is inefficient
  – This is the case for all storage and data transfers
  – Xrootd’s look-up delay for new files is expensive for small file migration from top to bottom.
  • Bypass back tier redirector to improve performance

• Resource intensive operation when vacating a storage node.
  – Multiple steps, need to be careful
Future Developments

• Build 3 tier storage system
  – Top SSD (an array of 12 SSD’s is ready)
    • We know we can get 48Gbps with 250K IOPS
  – Middle 7,200rpm HD’s
    • First stop for new data
  – Bottom 5,400rpm HD’s

• Trick will be to properly manage the SSD’s
  – We suspect that as SSD’s become cheaper 2-tier setup will be more appealing