Back-end storage - use as a true archive

Dirk Duellmann, CERN IT

WLCG Data & Storage Management Jamboree,

Amsterdam

16th June 2010

Outline

- Back End Storage - some problems of current systems
  - System complexity -> Transparency (user), maintainability(service)
  - Scalability for analysis -> File access latency, client protocol & caches
  - Does the HSM model still help?
- Benefits of possible conceptual and technology changes
  - Conceptual changes
    - Support for experiment defined file-sets
    - Independent data access and archive storage
    - Independent storage and transfer components
    - Archive mode vs random access tape
  - Technology changes
    - Tape archive and/or disk archive
    - Filesystems - what's there? what's missing?
    - In memory meta data for storage pools

HSM - Do we still use Model?

- Hierarchical Storage Management (HSM) systems promise to hide the storage hierarchy from its users.
  - Users see a simple file level (posix) interface
  - Data movements (disk->tape, tape->disk) are always done transparently and are optimised in the large shared setup managed by the HSM system.
- Is the HSM model still used / useful?
  - Production
    - Experiment work-flow system insure (pre-stage) dataset on disk
    - Disk-only pools play an important role
  - Analysis - also here HSM seems of limited utility
    - analysis input data must be on-disk, available volume is managed by physics WGs
    - users often do not have access to tape at all (to insure tape resources for production)
- Over the last years we have largely given up on using the HSM mode.
  - we just use automatic archiving of new data to disk
- Direct access to disk cache and archive components by experiment work-flow systems would re-gain transparency.

Independent Storage and Transfer Components

- HSM approach has in some cases (eg CASTOR) resulted in a very tight coupling between
  - Archive related meta-data (tape namespace)
  - Disk cache related meta-data (disk name space)
  - Transfer meta-data (state changes of data in flight)

- This has led to difficulties to evolve the system
  - work-flow meta-data is "polluting" the critical disk cache meta-data
  - data schema changes affect several functional components

- Proposal:
  - focus on pure storage components (access & archive pools)
    - stat, put, get - plus posix access for the cache
  - transfer components maintain their work-flow meta-data internally
    - connect to storage pools via their external interface
    - would eg allow to create HSM pools from from basic storage pools

File Set Support

- Current storage systems provide a convenient filename space to experiments
- but do not really aid several of their main work-flow primitives
  - change disk/tape state for a complete set of files
  - check if a file set is complete on-disk/on-tape/at-a-site
- from the service perspective
  - file-set knowledge would help in more efficient dataset placement on disk & tape
  - garbage collection on disk

- File set concept would allow for more efficient support of production workflows

File set Support - How? Where?

- How to define file sets without breaking the end user API (posix)?
  - Experiment datasets are often collocated in the name space
    - eg the list of files in a given sub-directory, list of sub-dirs,
  - **Would a directory based property be sufficient to define file sets in a storage system?**
- File sets content often move together (eg archive->user disk, T0->T1)
  - Atomic file-set movement and meta-data registration would increase scalability and reliability significantly
  - Archive storage and transfer components may not require the knowledge about the internal structure of a file-set
  - File-set placement on disk storage could exploit maximum spread of member files across available disks to avoid contention for hot file-sets
- File sets get "closed" at some point after creation
  - individual file data and meta data stabilises
  - could be stored/recovered from self-describing archive media
- **Would the concept of closed file sets be acceptable for experiment users?**

Archive vs Access Cache

- Archive
  - few sequential, heavy streams
    - user file is not the best management entity
- Access Cache / Disk Pools
  - many user connections,
  - high open and stat rates
  - large random I/O component

- Low latency storage technologies for the mass market
  - Flash is there, phase-change memory is coming
- Gap between archive and cache storage may broaden
- Need to maintain independent archive and access components
  - allows to use new technologies once ready for production

Archive Media - Tape or Disk? Another Disk Layer before Tape?

- Starting point: long tape related access latencies
  - implies long, complex work-flow queue inside larger storage systems
    - fluctuation of request completion -> user transparency
    - often: a workflow DB -> operational effort, DBA support
- Proposal: investigate disk based archives
  - exploit parallel and direct access to archived data on disk?
    - goals: reduced latency, simplify work-flow
  - focus on power & budget efficiency rather than performance
    - very different corner of the phase space wrt user access pools
- disk archive demonstrator
  - economical feasibility (early model studies looked promising)
  - operational feasibility (test h/w lifecycle support, redundancy & recovery)
  - prove work-flow simplification (eg throttling instead of scheduling)

Filesystems - At least two Functional Roles

- 1) the "client protocol" used to access data (ideally as mount) on a WN
  - Should provide
    - support secure authentication (incl. X509, Kerberos)
    - client side data cache, support for vector reads
    - redirect clients in case one access path is unavailable
  - Examples: NFS4.1, XROOT/FUSE, AFS, {GPFS}

- 2) the software used to access/manage cluster storage
  - Should provide
    - high performance namespace, quota system
    - scalability in aggregate performance (eg file replication, striping)
    - support for online storage re-organisation
    - storage availability through media redundancy
  - Examples: GPFS, Lustre, AFS, XROOT
- **For the moment: no system can claim to implement both functional areas**
  - **but clustering storage is an attractive starting point for several T1 sites**

File Systems as Clustered Storage

- Used in different areas (-> site consolidation)
  - Core storage, software areas, home directories
- Behind core storage systems: dCache, Xroot, DPM, {STORM}
  - no direct client access
  - Lustre and GPFS are successfully used
    - both lack direct X.509 support
    - both lack traceability of user I/O bandwidth
    - Lustre lacks support for online storage reconfiguration
    - Qs: commercial viability / long term futures / vendor lock-in
- Questions to sites / storage development
  - How big is the gain in operational effort of using a clustered filesystem
    - if storage re-configurations require a service outage?
    - if an outside authentication/authorisation system is required?
  - How far do clustered file systems scale today in terms of volume and aggregate experiment throughput?
  - How big are the savings in not having to develop/maintain a robust and performant name space and clustering layer?

True Archive Mode

- Storage setups consisting of separate cache and archive components will only run effectively if
  - the active data to cache ratio is close to one
  - changes to the active set are covered by the available archive bandwidth
- Any temporary violation of the above means increased access latency
  - longer term violation means overload and unavailability
- Current systems can not reliably prevent the above to happen
  - Their shared nature makes it hard to isolate the origin of additional ingest rate or changes in the active set
  - Operational teams and experiment teams are largely blind wrt the impact of work-flow changes in another corner

How to get to Archive Mode

- There is no silver bullet to get closer to archive mode
  - need to jointly (sites + experiments) analyse key operational conditions
    - cache efficiency vs active data sets
    - top consumers of archive bandwidth
- Provide experiment management the necessary input to spot and throttle overcommitment at the source
  - Proactively manage the main data flows in the system
    - Agree, document and monitor their rates and latency to archive
  - Agree on max rate for lambda users (if any!)
- As long planning discussions are about global disk and tape volumes we will achieve stable conditions only via over-provisioning
  - This works too, but given a fixed budget I'd expect a genuine interest of experiments in this activity
- Need to close the loop between storage providers and experiments storage managers at least for the main data flows