CMS Input for Data Management TEG

Brian Bockelman

Forewarning

- Given the number of questions and the amount of time to present the information, I can't cover everything.
  - Consider https://twiki.cern.ch/twiki/bin/view/LCG/AnswersCMS1 authoritative, especially if there are disagreements between this presentation and the twiki page.
  - I've attempted to organize these answers into a (shorter) and more presentation-friendly format.
  - All omissions and errors introduced not in twiki link are mine!

Background

- CMS's Data Management is underpinned by the dataset placement service (PhEDEx) and the dataset bookkeeping service (DBS).
  - PhEDEx handles the interaction with the transfer layer, block location information, and dataset/block subscription.
    - Notice that file location is only tracked for in-flight blocks.
  - DBS handles physics metadata information.
  - PhEDEx has no knowledge of physics; DBS has no knowledge of

Background (Philosophy)

- CMS attempts to not micro-manage sites. Examples:
  - Site may use any space management technique they'd like.
  - No quota systems required. Sites "data managers" must approve all data transfers before they happen.
  - This has downsides too – we require active, engaged sites. Can't really "plug it in and walk away", although the operational cost drops considerably after initial setup.
- We feel the existing system is functional, and cost/ complexity is *about* right. Elements of SRM and FTS have excess complexity.
  - Realize we don't use LFC, of course.

The Big Issues (Storage)

- *Disk management in HSM*. What's available on disk, and what has made it to tape?
- *Efficient data access with increasing volumes and fewer spindles*. (Spindles)/(Cores) is going down. Adopt applications or die!
- *WAN access to storage*. Increase our ability to access storage at small scale over WAN.
- *Reliability*. Easy to overlook, but crucial. Less errors, and clear&clean error messages when necessary.
- *Intra-VO authorization*. We have a steady-state operational model, but it's not elegant.

The Big Issues (Continued)

- The greatest future challenge is the large data volumes.
  - If we cannot handle the data volume at the T0, we cannot run T1s. If we cannot handle at the T1s, we cannot run T2s.
    - So, while we start to think about the "end-user" and usability, handling data volume is still #1.
- We are working hard on making data pre-placement more dynamic (using data popularity).
  - Think of it as a partial adoption of ATLAS's methods.
- We are still worried about the various "transaction limits" in dealing with the SEs.
  - I.e., files per second, constraints on file size.

On file catalogs

- LFC is not used.
  - Our file catalog has about 300k HTTP requests / day with 600ms average response time.
  - Total request count is far reduced by the fact our transfer system uses direct Oracle access.
- Everything is in a custom catalog (PhEDEx) and tracked at the “block”, not the file level.
  - Accessible via a RESTful JSON-based service.

On Namespaces

- CMS has a global namespace. Each file has one, unique LFN.
  - Sites provide a set of mapping rules; these are used when jobs start, or right before the transfer is sent to FTS, to translate to a PFN.
    - It is a touch tricky communicating between sites – the remote site needs to a priori tell you the PFN, or you need to find its mapping rules.
  - CMS has no concept of a SURL, TURL, GUID, etc. Just LFN and PFN.
    - We do not track SURLs in our catalogs. The blocks we track are associated with an opaque string (usually a hostname) associated with the site.
- Observation: In our xrootd work, we've found it very convenient to have the WAN protocol speak "LFNs" instead of "PFNs". This way, your knowledge of the remote site only needs to be the endpoint.
  - It would be nice if such a thing crossed over to other protocols. Why does SRM or GridFTP need to expose site internals?

CMS Site Storage Requirements

- Stripped to the minimum:
  - Site must support wide area transfers via FTS.
  - Site must support one of the stock ROOT local protocols *or* provide an implementation in CMSSW.
  - Site must use one of our plugins for file stageout or provide one of their own.
- Note: a T2 site with a powerful gridftp server and a cluster file system would probably be just fine.

Cloud File Systems

- Would probably work for some of our workflows.
  - Requires some minor modifications to the software.
  - Give us a site with an efficient cloud file system, and we'll likely be able to use it.

Space Management

- CMS delegates space management to the sites. They make the decision of whether to take on more data, and police free space.
  - If the best way to do that is via SRM space tokens, sites can do that. In practice, zero-to-little use of SRM for space management.
  - CMS provides tools to discover discrepancies with the catalog and perform checksums.
    - Moderate-to-good uptake at the T1s.
    - Relatively little uptake at the T2s. (Perhaps this isn't necessary at T2s if data federations are widely available?)
    - This is being further expanded to cover files outside the centrally-controlled namespace (user files, for example).

Space Management

- We do use file families heavily.
- We must know when files are successfully on tape.
- We must know when files are on disk.

Authorization Management

- Authorization management is handled by the sites.
  - We provide guidance on what roles/users should be able to access what, and the sites implement it.
    - And bug the sites until they are mostly in compliance.
  - Our DM tools prevent users from causing problems – *if* they are using our tools.
  - No strong, verifiable means to know that a site is protected against a malicious user (a-la ALICE security tokens).
  - Number one concern is keeping CMS data private from other organizations/ the world.
- Observation: we would benefit if intra-VO authorizations could be taken from VOMS, even if only for grid-based write mechanisms (SRM/GridFTP).
  - It would probably simplify things for sites, as the storage would need to be configured at the VO-level, not the sub-VO-level.
  - This is not a priority, however.

The SRM and GridFTP Question

- CMS's heaviest use of SRM is to load-balance GridFTP.
  - We can work with space tokens for the sites that desire SRM space management.
- Can CMS live without SRM? Yes!
  - However, experience with Bestman2 shows SRM can be made "small enough" to not be a problem
  - Replacing SRM would mean an appropriate replacement needs to be found…

HTTP vs GridFTP

- HTTP and GridFTP can both load-balance themselves, but may require more network expertise at sites.
- Many have observed that GridFTP can be replaced by HTTP.
  - Just need to figure out third-party-copy.
  - Oh, and delegation.
  - Oh, and standardize a mechanism for integrity checks without encryption
  - At this point, is HTTP a "standardized protocol" or "building a new SRM"?
- This would be an interesting investigation (HTTP is *much* more flexible and widespread than GridFTP), but aren't ready to jump ship yet.
- CMSSW *can* speak HTTP, but it hasn't been thoroughly tested. Assume the performance is poor!

Overall:

- We can get rid of SRM/GridFTP,
  - But we need a workable replacement.
    - Maybe the effort is better spent in simplifying and hardening existing services.
  - There is no burning desire to do this.
    - The big “cost” here is how much time developers are spending to support features we don’t use.

The HSM Question

- AKA, “should we hard partition archive and disk so HSM is no longer needed?”
- We don’t want to drop HSM right now, but find ourselves increasingly manage the split between archive and disk.
  - We still need the equivalent of “file families”, and strongly need to track file status.

On Change

- When discussing changes like dropping SRM/ GridFTP/HSM, we feel the experiment is fairly flexible and could do many of them. There is a clean-cut between “experiment layer” and “middleware layer”. Common-sense caveats:
  - Each change needs to be discussed thoroughly and plotted in advance.
  - Functionality we use needs to have working equivalents in $NEXT_GREAT_THING

Data Federations

- What is a data federation?
  - Hopefully to be better defined at the related workshop next week!
  - Loosely, it's the ability for users to access data across multiple source sites uniformly from a single endpoint.
    - Note that access implies reads, not writes.
    - Some differences between sites are hidden by the federation; user never has to do a location lookup, know local namespaces, or know site endpoints a priori.

Data Federations in CMS

- Each “network region” in CMS is building a data federation.
  - Best coverage is in US, where all T2s and T1 are participating.
- Provides:
  - Fallback access: if files are missing or damaged, applications transparently failover.
  - Interactive access: a mechanism for users to directly interact with the storage over WAN for small tasks (purposely has scaling limitations).
  - Overflow access: if extra CPU is available, send jobs there regardless of whether the necessary data is present (a small optimization on our data distribution).
  - Data for T3s. A slightly larger-scale “interactive access”.

Data Federations and Caching

- You can have a federation without a cache, and a cache without a federation.
- We do not currently see a future where the whole experiment is cache based.
  - Therefore, any use of caching will be alongside the existing system. Does this cut complexity?
  - Possibly useful in limited form – ie, T3s and data sharing.
- It's possible to recast the current system as more "cache-like".
  - For example, our systems assume that when they place data somewhere, it stays there indefinitely.
  - Not a safe assumption for T3s (unfortunately not safe for some T2s also!).
  - Can we take the current system, and start asking ourselves what needs to be changed if "cache eviction" was a normal, not exceptional, event.

Dynamic Preplacement – Better than Caching?

- Caching becomes less problematic if we instead did dynamic pre-placement.
- Can do T3s!

Failures

- FTS hasn't been as dynamic as originally promised – but we still want this (promised for FTS3)!
- No common equivalent to PhEDEx or DBS – but we're OK with that.
- Checksums don't seem to be delivered – we would like this still. Would be nice to have a better coordinated strategy.

Complexity Kills

- From the MB: Where should the complexity and intelligence lie - the experiments or the infrastructure? How do you view the balance now, and how would you like to see this change (if at all)?
  - Our gut is that we want to see sites be “simple and reliable”, and have the complexity managed by the experiments.
  - However – it strikes us that this opinion does not do much to cut “complexity and cost”.
- Not sure how to resolve this, hopefully the TEG has some ideas!

Cost and Complexity of "The Solution"

- We feel the CMS computing model is actually not-too-complex and not-too-costly for what is accomplished.
  - The system is well-spec'd and functional.
- However, there are significant parts of the WLCG stack we do not use.
  - Effort going to pieces/functionality we don't use could be spent on things we do!