Any Data, Any Time, Anywhere: Year 2

Brian Bockelman,
On behalf of the AAA project

AAA Overview

- "Any Data, Any Time, Anywhere" is a NSF-funded project to improve data accessibility in HEP.

- It is a three-year collaboration between UNL, UW, and UCSD. Award #s 1104549, 1104664, 1104447. Starting the second year.

- We thanks the NSF for their support of this project.

AAA Goals

- Increase the data accessibility for physics.
- Deliver tools to decrease the barriers between physicists and data.
- Increase the portability of the CMS environment.
- Remove the data locality requirement and increase the number of sites where a job can run.

Year 1 Summary

- AAA Federated Storage Infrastructure.
  - Covers all of USCMS, and increasing percentage of global CMS.
  - Tens of TB a day transferred.
  - Thousands of jobs using remote I/O a day.
  - CMS internal target is majority of sites in AAA by the end of the year.
- Two regional redirectors (UNL and Bari), and a global redirector (CERN).

Participating Sites

- US Redirector:
  - Nebraska
  - Purdue
  - Caltech
  - UCSD,
  - Florida
  - Vanderbilt
  - MIT
  - Wisconsin
  - FNAL (disk-only)
- EU Redirector:
  - Legnaro
  - Bari
  - Pisa
  - London / Imperial
  - DESY
  - CERN EOS

Use Cases:

- Use cases introduced last year:
  - **Fallback**: Automatically reading a file via remote when it is unavailable locally. CMS now tests to see if this is configured. Just Works.
  - **Interactive Access**: Increasing use by physicists.
  - **Diskless sites**: Pilot site is the Notre Dame CMS T3, where they didn't have enough experience/time to run their own storage.

Overflow

- A majority of transfer volume is due to *overflow* jobs.
  - These are jobs which have been in the queue for awhile, sent to a site without the data, under the assumption they will stream from a site with data.
- We monitor the success rates and the CPU efficiencies of these jobs - no significant issues observed.

Opportunistic Usage

- We worked with Parrot and CVMFS to provide a userspace-only version of CMSSW using the ptrace mechanism.
  - Required refactoring of CVMFS - there is now "libcvmfs".
  - Also provides a user separation layer - but without glexec!
- With Xrootd for data and Frontier for conditions, we can launch CMS jobs practically anywhere.
  - For example, we've run CMS jobs on the BNL Tier-1.
- Tested extensively, but not yet rolled out to production.

Monitoring

- See Matevz's talk for the details!
- General message is that things are working well, but we need to think about the deployment for the next level of scale.
- Having MonALISA (summary), Dashboard (detailed), and Gratia (detailed) allow us to cross-check the results of the monitoring.
  - Not 100% yet, but a great tool for debugging!
- We internally get daily reports and have our own Nagios tests - but will be better integrated with global monitoring in the future.
  - We can correlate jobs and data monitoring to figure out which data server caused job failures.

Integrating Regions

- Static redirect on ENOENT seems like a promising approach.
  - Does not work for CMS due to the infinite loop bug.
  - We are evaluating our options, but they are all likely fairly nasty workarounds.
  - REMINDER: It now takes >3 years to turn-over client software. Any bugs that exist today will exist in user jobs in 2015.
    - I.e., new Xrootd client will be used as early as 2015.

I/O Optimization

- Every improvement to CMSSW makes AAA a more pleasant infrastructure.
- Improved cache design - far fewer network round-trips for folks creating skims.
- With CERN IT, we've been getting the remaining bugs out of the separate prefetch thread. Will be available, but not default, in CMSSW_6_1_X.
- Have been working with DPM team to make sure application performance is similar on HTTP.

Improved Caching

- CMSSW makes two large improvements to ROOT's TTreeCache:
  - Improved startup - while the TTreeCache is in learning phase, it reads out all bytes for the first of 20 events. Prevents many round-trips during job startup.
  - On a cache miss, read out full event immediately. This is because one cache miss on one branch/event usually implies there will be more cache misses on the same event.

Year 2 Plans

- WARNING! The rest of this presentation contains forward-looking statements.
  - Take them with a grain of salt...
- We'll be focussing on a mixture of hardening Year 1 services, plus developing a few new capabilities.

Hardening and Deploying

- Our monitoring application is beginning to be run by others.
- CMS central operations will begin to take up the operation of the infrastructure.
  - For example, we add Xrootd to the SUM tests. This should verify fallback at all sites, and whether sites are working in the global redirector.
- Will start running Parrot/CVMFS as a part of our overflow job system.

gWMS Goes to the Cloud!

- GlideinWMS v3 has the ability to submit pilots to EC2-like IaaS. We use the same setup presented in our 2010 CHEP paper: use gWMS (jobs), CVMFS (software), Frontier (conditions) and AAA (data).
- In Year 2, we are studying the costs of such a setup.
  - Only small-scale tests, but still surprising. Sending a random subset of T2 jobs to Amazon and buying cycles on the spot pricing market.
    - Does spot pricing scale? If so, the costs are surprisingly close. Big assumption!
  - Will have to attend this talk next year to find out the conclusions!

Improved Routing

- We have a patch series for the redirector which allows plugins to express preferences about which servers to query for a client.
  - If not used, the code overhead is basically zero. Requires no client-side changes.
  - A global redirector could preferentially query US sites for a US-based client, and only query the rest of the world when nothing is found.
  - Several possible ways to determine locality - students are exploring a few approaches. Right now, matching based on TLD.
  - Solves the problem of "which redirector to start with," and maybe solves our issues with static ENOENT.
- Improving the routing allows us to avoid user confusion about which endpoint they should use!

File Caching

- We have aspirations to lower total hardware cost by trading reliability (2 replicas) with more WAN access.
- Looking at taking the Xrootd proxy and adding a caching layer to it.
  - Will allow for better performance than FRM approach, but will only work with sourcing files from Xrootd federations.
  - Hoping to fix the performance of the existing proxy and OSS layers by exposing the readv calls to the storage.
  - Requires an ABI break, so will need to wait for a major release. Prototype exists, might finish as soon as this calendar year.

Data-aware Scheduling

- Condor's matchmaking scales poorly if you try to have it match jobs and worker nodes based on files.
  - This is due to the fact each job becomes unique and must be treated individually.
  - Matchmaking would have to repeatedly call out to an external file location catalog, to an extent that most catalogs couldn't handle quickly.
  - CMS currently pre-compute the sites which host the data, then submits to Condor. Not very effective if file locations are dynamic with respect to job lifetimes.
- We are looking at two complementary approaches:
  - Have Condor periodically re-compute site locations, reducing the number of unique jobs without relying on static information.
  - Populate a database with this information and use more standardized queries to perform matchmaking in bulk.
- These will both be evaluated throughout the next year. Both rely on using the xrootd client to determine file location.

Things Still Missing

- While our most immediate needs have been filled in the last year, things we're looking forward to (or would like the following things):
  - Third-party-copy, especially integrated into FTS.
  - New client.
  - IPv6.
  - cmsd-only operation with static redirects (for improved dCache support).
  - Quite interested in what additional monitoring will come out of dCache.

Coordinating Deploys

- The ATLAS and CMS Xrootd approaches are very similar. To add support:
  - Run a second cmsd/xrootd instance.
  - Point at a different endpoint.
  - Swap out a few plugins (at minimum, N2N).
- However, we have two completely different sets of documentation. Can we make this more of a WLCG service?
  - Sounds like this should be straightforward, but we have no existence proofs yet.
- As far as I can tell, there's no way for a site to simply integrate ALICE on the same node.

Conclusions

- We are 1/3 through the AAA project, and are remarkably close to schedule.
- This is a time-limited project, so we are looking to do *contributions* to upstream projects and starting operations that will be maintained by others.
    - In this, I think we are doing well. This year we contributed to Condor, glideinWMS, Gratia, Xrootd, Parrot, CMSSW, and CVMFS.
- Year 2 will be the last year of large developments - then we start focusing on increasing component quality.