BigPanDA and CMS Analysis Integration

Alden Stradling
(Univ. of Texas at Arlington)

16 June 2013

Milestones

- The groundwork is progressing well:
  - PanDA core is in process of factoring out ATLAS-only components (WP1)
  - Pilot and server modifications nearing completion
  - CMS sites are online and functioning

Items of interest

(the first two via Ian)

- **Discuss the coupling and integration of PandaMon and PandaDB**.
  - The idea it to investigate if it's possible to use PandaMon to provide a consistent monitoring view also for jobs that would be scheduled with Condor.
- The current functionality in the Panda system for **scheduling** and **authorization**.
  - **Scheduling**
  - **Authorization**
  - **Ways that groups**
    - **Prioritize**
    - **Authorize groups of users at specific sites**
- **How PanDA enforces global priority for groups and activities** are:
  - **Defined**
  - **Updated**
  - **Monitored** should be discussed.
- Development models that fully include CMS as a code contributor.
- ProdSys II

UT ARLINGTON

Core PanDA

- The central principles of PanDA derive from principles of:
  - Central management of system information
  - Distributed (“pull”-based) management of workloads based on system measurements

- Centralization strongly informs monitoring, authentication, and quota management

UT ARLINGTON™

PandaMon and PandaDB

"Investigate if it's possible to use PandaMon to provide a consistent monitoring view for jobs that would be scheduled with Condor."

- Central
- The PanDA Monitor is (and will continue to be) an AJAX overlay to the DB itself, without intermediaries
  - Tremendous advantage in latencies and accuracy (since checking consistency is not necessary)
  - Allows for tight integration
  - Schema evolution is also coordinated (monitor is written to accept changes gracefully)

UT ARLINGTON

Integration of CMS

- Proposed plan at the moment is that:
  - Condor communicates with the PanDA DB in essentially the same way that a pilot uses:
    - HTTP/S RESTful interface (curl + proxy + python)
  - Updates job status asynchronously.
  - Monitoring handles the information transparently.

Scheduling

- PanDA approach is to distribute jobs based on a constellation of parameters:
  - I/O or Tape requirements
  - CPU requirements
  - Software installation status
  - Disk availability on worker node or storage element
  - Site occupancy/downtime
  - Data locality/transfer costs

UT ARLINGTON

Scheduling

(continued)

- Multi-stage scheduling mitigates most bottlenecks, especially for managed production.
  - ➡ Task Brokerage
    - ➡ Site Selection
      - ➡ Job Dispatch
- Late binding prevents site uptime issues from bottlenecking production.
- File availability can still be an issue if prestaging was limited. Transfer cost weighed against site downtime.

UT ARLINGTON

Scheduling

(continued)

- Management of these smaller variations (usually with the more chaotic user analysis) is done with rebrokerage.
  - Jobs with long waits at sites (offline or just busy) are redefined and passed to other qualifying sites for execution.
  - 24h wait before rebrokerage to prevent excessive hopping.
  - Rare to get complaints about delays specific to rebrokerage.

UT ARLINGTON

Authorization

- Primary authorization is, of course, VOMS-based and central, with CRLs distributed in a pull-based system.
- Roles attached to the certificate permit PanDA to authorize access to different quotas and storage categories.
  - This allows, for example, a group MC coördinator to use the group quota from an individual certificate.

UT ARLINGTON

Authorization

(continued)

- Site usages and quotas can be enforced in a couple of ways:
  - The puserinfo mechanism allows DN-based access to a site.
  - Tunable -- users can be managed, allowing them priority for a period of time without blocking others from the site.
  - A new queue can also be added exclusively for local users, guaranteeing local access to the PanDA queue.

UT ARLINGTON

Authorization

(continued)

- The "pledge" tracking system allows sites to specify how much their installed systems exceed MoU.
  - Disk and CPU are separately tracked
  - The difference between pledged and actual capacity can be used at higher priority by users whose credentials match the site's parameters.
  - Updated by site admins in AGIS

Global Prioritization

- User priorities are connected to the user's DN in the PanDA central DB:
  - Tracked centrally.
  - Updated in real time based on actual CPU usage.
  - Recovers according to a consistent algorithm.*
- Similar mechanisms apply to group- and activity-specific priorities.
  - The same database handles separate single-number quantities for groups/activities.

* See backup slides

ProdSys II

- Now that we've built a successful system, we're rebuilding it in place to add flexibility and expand task-based job building.
  - Allows dynamic "workflow" definitions which incorporate multiple tasks, allow dependencies and rules.
  - Allows more granular and flexible recovery.
  - Formalizes some of the "organic" coding that brought the system about.
  - Brings task and meta-task tools to user analysis.

UT ARLINGTON

ProdSys II

(continued)

- Getting there:
  - Extending the DBs to hold the state of this more complete machinery.
  - Creating the ProdSys job definition code (JEDI).
  - Implementing a task definition database (DEFT) to handle task creation, manipulation, approval and tracking
    - Allows run to start for partially-defined tasks
    - Provides means to extend or abbreviate tasks in process.

UT ARLINGTON

ProdSys II

(continued)

- Mechanism for file-level and even event-level processing within tasks
- Extension via plugins for various kinds of processing
- Greater accuracy:
  - Scout jobs determine the real scale of submitted tasks, allow for better resource estimation and matchmaking.
- Offloading central processing requirements:
  - Heavy job definition tasks (LFC lookup for event-level tasks) runs on a WN somewhere other than the busy central server.

UT ARLINGTON

Development Models

- PanDA started as an ATLAS-centric system
  - Smooth sailing for ATLAS
  - We don't feel the bumps and ridges of incorporation
- Interested in feedback on how to best open the doors to real code collaboration
  - Move from CERN SVN to another provider? Perhaps with git?
  - Move from Savannah to an open bug tracker?
  - Any of the classic FOSS-community moves?

UT ARLINGTON

Backup slides

Job Priorities

Job priorities are calculated for each user by using the following formula. When a user submits a job which is composed of *M* subJobs,

$$Priority(n) = 1000 + F - \frac{T+n}{5} - W \times (U - Q) \times H(x = (U - Q))$$

where:

Priority(n): Priority for *n*-th subJob ($0 \leq n < M$)

F: Priority offset

T: The total number of the user's subJobs existing in the whole queue. (existing implies jobstatus = {defined,activated,running})

W: Weight

U: CPU usage for last 24 hours (in *kSI2kday*)

Q: CPU quota

H(x): Heaviside step function (0: $x \leq 0$, 1: $x > 0$)

For example, if a fresh user submits a job composed of 100 subJobs, the first 5 subJobs have Priority=1000 while the last 5 subJobs have Priority=981. The idea of this gradual decrease is to prevent large jobs (large = composed of many subJobs) from occupying the whole CPU slots. When another fresh user submits a job with 10 subJobs, these subJobs have Priority=1000,999 so that they will be executed as soon as CPU becomes available even if other users have already queued many subJobs. Priorities for waiting jobs in the queue are recalculated every 20 minutes. Even if some subjobs have very low priorities at the submission time, their priorities are increased periodically so that they are executed before they expire.