WMAgent (and the SLC6 transition)

Andrew Melo
Vanderbilt University
OSG All-Hands Meeting
March 19th, 2012

Outline

- Introduction
- Migration from ProdAgent to Wmagent
  - Motivation
  - Advantages
- Migration from SLC5 to SLC6
  - OSG Stack
  - CMS Software
- Conclutions

ProdAgent to WMAgent

ProdAgent: CMS' Old Production Framework

- Used for Production workflows
  - Monte Carlo simulation
  - Data processing/reprocessing
- Some of the codebase is used elsewhere (ProdCommon)
  - CRAB2
  - SAM Tests (now known as SUM?)
- Message passing architecture lead to scalability/ reliability issues
  - Lead to a lot of manual intervention to fix "missing" jobs
- More complicated workflows were difficult to do

ProdAgent Message Passing

- Proper function of ProdAgent depends on components transferring updates via messages
  - For instance, if a fails and needs to be retried, a message is sent to the JobSubmitter component
- In case of excessive load or hardware/ software fault, these messages can be dropped or lost
- If this happens, ProdAgent's idea of the state of the job can become confused, or worse, lost
  - This is the cause of a lot of issues with CRAB2 in server mode

WMAgent

- WMAgent replaces the message-passing architecture with a state machine backed by CouchDB
- Statuses of jobs and workflows are stored in Couch documents
  - Since the state is stored in a database, transient errors or component failures won't cause inconsistencies
  - CouchDB was chosen over MySQL or Oracle for its scalability, query language and ability to easily store arbitrary data
- CouchDB stores JSON documents + attachments
  - Lumi Masks (ACDC Component)
  - Python Configurations (ConfigCache Package)
  - FrameworkJobReports (FwkJobReport Package)
- CouchDB can execute complex programs within the database itself (CouchApps)
  - Database views are idempotent, so caching can be used

WMAgent

- Provides a generalized way to describe workflows made of a series of discrete steps
  - Adding a new type of processing workload involves simply combining existing building blocks together (i.e. CMSSW, StageOut, LogArchive, etc..)
  - Shared code means less, better tested code
  - Workloads are data-centric instead of job-centric

Current Status

- Approaching fully-featured
  - The last few percent seemingly take the longest
- From a site's perspective, should behave the same as before
  - Uses glidein/glite as the schedulers
  - Nearly the same runtime environment
- Some workflows are already run using WMAgent, others will follow

SLC6 Migration

SLC6 Migration

- RHEL5 was released in 2007
- Full support is ending at the end of 2012
  - The full life-cycle extends until 2020, but gradually only critical bugfixes will be released
- Libraries for RHEL5 are becoming outdated
  - Not necessarily a problem for CMS
  - A big problem for mixed-use clusters

SLC6 Migration

- Vanderbilt's cluster (ACCRE) is shared amongst various experiments and fields
  - ~80% of the worker nodes belong to non-CMS experiments
- Until recently, CMS didn't have its binaries validated on SLC6
- Other experiments really needed to upgrade the WNs for their own analysis
- Temporary solution: split the cluster

SLC6 Migration

- What was holding up a full migration?
  - CMSSW
  - OSG Worker Node Client

Migrating CMSSW

- Last December, CMS validated its SLC5 binaries on SLC6
  - Eventually, native SLC6 binaries will be made
- Appears to work great

OSG Worker Node Client

- Very recently, OSG provided SLC6-compatible worker node RPMs
- We tested them at Vanderbilt, everything appears to work “right”
  - Ran a few test CRAB jobs reading with FUSE and writing to the SE using the LCG SRM utils
  - Have a private WMAgent instance that we can test with soon
    - Need to finish writing the PBS interface
  - If things continue to check out, we’ll migrate the rest of the cluster over

Questions