Job submission implementation plan

Dennis Box, Joe Boyd, Rick Snider
on behalf of REX

...

NuComp meeting
Fermilab
Nov. 17, 2010

Basic objectives of job submission system

- For end-users
  - To provide access to distributed (grid) computing resources
    - “local” resources in this context = one instance from a set of grid resources
  - To simplify the task of utilizing these resources to solve complex or large-scale computing problems
- For experiment management
  - To allow experiments to manage utilization of the available resources to meet physics objectives
- For computing system operators
  - To provide mechanisms to manage utilization of the available resources in order to maximize computing throughput
  - To minimize the effort required to do so across multiple experiments

The underlying assumption: limited computing resources available

Job submission system

- “Job submission” cuts through several layers
  - Submission client
    - What end users see: includes the feature set and user interface
    - Independent of underlying batch system(s)
      - Can provide a uniform way to access all available resources
  - Job submission and management infrastructure
    - Talks to the submission client and the batch system
    - May include pieces that live on several machines + pieces submitted with the job
    - Exploits features specific to a particular batch system
  - Batch system configuration
    - Provides features to support management of resource utilization

  Note that the relevant “batch system” may not be the one in operation at a given site.

    - Can “overlay” one batch system on another

Resources

- "Local batch" (via General Physics Computing Facility)
- "GP grid" (aka "local grid" or "Fermigrid")
- Other resources on Fermigrid (on-site resources in OSG)
- Grid at large (off-site resources, primarily those in OSG)
  - At least two flavors:
    - Collaborating institutions with priority access rights
    - Non-collaborating sites with only opportunistic / pre-arranged access

Job submission requirements

(in no particular order)

1) Common submission client for all IF experiments
2) Common submission infrastructure for all IF experiments
3) Provides support for steering of jobs to specific resources
4) Supports the concept of “groups” for setting priorities and accounting
5) Supports specification of external resources required by the job
6) Supports job ordering dependencies
7) Supports logging of submission information not available via the batch system
8) Meets operational requirements (TBD)
9) Provides extensible and maintainable code base
10) Returns error messages that users can understand and act upon
11) Provides tools to assist with tarball creation
12) Provides sensible defaults for any given user / experiment

Major considerations at this time

- Prefer adopting / adapting over writing from scratch
  - Most operating experiments have dealt with this problem in some fashion
    - Expect varying amount of work to make one work for IF experiments
  - Need to be a recognized stakeholder / co-author of adopted solution
    - Critical for any product that is part of core infrastructure
- Highly configurable; extensible via sub-classing
- Simple transition for users
- Deliver highest value-added features first

Proposed elements of a solution

(ie, not the entire story yet...)

- Retain, extend existing command-line interface
  - Can be done regardless of implementation
  - Easiest transition for users
- Adopt glideinWMS as the basic workload management system
  - Support based at Fermilab
  - Used by many experiments at Fermilab (CDF, CMS, D0, MINOS), elsewhere
    - Lots of code / features to steal, copy, study
  - Provides a basic wrapper for user jobs
    - Resource discovery
    - Creation and configuration of environment

Short term goals

- Introduce use of DAGs based upon submission options
  - Allow throttling of jobs at submission point
  - Support job ordering dependencies
    - Experiment-based workflows to be implemented first
    - Staging functions later, if needed
- Return sensible error messages
  - Will cover job submission failures + as many infrastructure issues as possible

    Note: Fixing this problem is intimately tied to job monitoring. More generally, we need to have good job monitoring to support job submission and management.

    Deploying improved job monitoring is out of scope here, but is a high priority short term goal for REX.
- Define groups to help manage priorities, resource utilization

Short term goals

- Support logging of information not available from batch system
  - First use for information needed by monitoring tools to be deployed for IF
  - Later include additional information needed for operations, planning
- Provide automatic tarball creation when submitting to grid
  - Making this useful will require some investigation into how to ensure that the unwound tarball will actually run
- Make submission system responsible for constructing submission files
  - Mainly a benefit for first time users or more complex submissions, should they be needed
- Automate creation of robot certs
  - Can steal existing code

Proposed short term priorities

- Implement DAGs for experiment-defined workflows
- Introduce groups to manage priorities
- Improve error messages
- ...
- Automate tarball creation
- Steering of job by resource requirements

Expect monitoring improvements to be deployed in parallel in the near future

Expectations

- Everything in short goals is relatively straight-forward
  - Have working examples elsewhere, expect most can be implemented quickly
  - Will not require any additional people to make it happen
- Need to work with experiments to develop more detailed plan and schedule
  - Are the short term goals and priorities OK?
  - What should we do next?