Requirements for the Grid job submission system for the Intensity Frontier experiments

Dennis Box , Joe Boyd , Rick Snider

V2, 1-Nov-2010

1. Overview

Grid-based computing utilizes resources distributed across some number of distinct, individually managed sites that provide those resources under a variety of managerial and operational agreements. Within the Open Science Grid (OSG), these resources are by design heterogeneous in terms of hardware, operating system, batch system, etc. While a common software infrastructure provided by the OSG allows these disparate sites to interoperate, the details of configuring and submitting jobs across the grid is still a non-trivial endeavor for end-users who are not computing experts. Beyond the simple task of submitting jobs, there may be resource limitations with which an experiment or system operators need to contend. While batch systems provide some tools for managing the priority of jobs and users, doing so across multiple grid sites is again non-trivial. In addition, there are essentially no provisions for managing or coordinating utilization of resources that are outside the control of a batch system, such as storage systems or networks.

The objective of the job submission system is to simplify the task of submitting jobs into this mildly hostile environment for the end-user by taking on the responsibility for understanding the job configuration needed to perform a specific computing task at any given site, while providing a set of tools to allow system operators and experiment leadership to manage utilization across heterogeneous assets and to define a uniform set of utilization policies independently of those enacted by the underlying grid sites. The purpose of this document is to describe the requirements such a system must have in support of the Intensity Frontier (IF) experiments.

The term job submission actually encompasses many layers of infrastructure, and includes at minimum a job submission client, an underlying job submission infrastructure, and a batch system configuration.

The submission client is component with which the end-user deals, and defines the feature set available to end-users and the interface for interacting with the batch systems. Since it must be able to submit jobs to a number of different resources

within a single interface, it must be independent of any particular batch system.

The job submission infrastructure talks to the submission client, prepares and configures the submission files according to the specifications set by the users, then performs the submission by direct communication with the batch system. This infrastructure may have multiple components operating on different machines, possibly at different locations, and may include parts that are submitted to the worker nodes along with the user application. The infrastructure will allow management of the computing resources by exploiting specific features of the underlying batch systems in combination with information from external sources, and by coordinating actions between other elements of the computing infrastructure.

The features of the batch system provide the primary mechanism for managing resource utilization. To the extent that the job submission system exploits these features to manage resources, the batch system configuration becomes an integral part of the job submission system.

We should note that the "batch system" under consideration here is not necessarily the native system in operation at a given site. Most modern job submission systems employ some sort of "overlay" technology, by which one batch system is installed on top of another. Use of this technology can effectively isolate the job submission batch system under consideration here from the underlying batch system at a grid site.

1.1. Basic Objectives

There are three distinct stakeholders in the job submission system, each with a different perspective on the system that lead to sometimes conflicting goals: end-users, experiment management, and computing system operators. The requirements of the job submission are derived from the following basic objectives with respect to these stakeholders:

- End-users
    - To provide access to the distributed computing resources available to the IF experiments, including both grid-based and non-grid computing elements.
    - To simplify the task of utilizing these resources to solve complex or large-scale computing problems.
- Experiment management
    - To allow experiments to manage the utilization of available computing resources to meet the physics goals of the experiment.
- Computing system operators:

- To provide mechanisms to manage utilization of the available resources in order to maximize computing throughput.
- To minimize the effort the required to manage computing resource utilization across multiple experiments.

The ultimate configuration of the system seeks a balance between these objectives.

2. Requirements

In developing the requirements, we have attempted to abstract the operational needs from the details of any specific batch system. Inevitably, some of the language in the requirements is borrowed from that of Condor, which is used extensively at Fermilab and throughout the grid.[1]

The requirements for the job submission system are listed below as sub-section headings. The discussion accompanying each requirement explains how it meets the objectives stated in Sec. 1, cites possible examples or use cases, and introduces related issues or considerations. Unless stated otherwise, none of the discussion points should be considered to be part of the requirement.

2.1. Common submission client for all experiments

The job submission client isolates the end-user from direct interaction with the underlying batch system. In doing so, it offers several useful functions. First, it provides a single interface for interacting with a possibly heterogeneous set of systems. Second, it allows instrumentation of job submissions in order to collect monitoring, debugging, or usage data that is not available from the underlying batch systems. (Note that instrumentation of user applications may substantially increase the value of data collected for some purposes, but is outside the scope of the job submission system.) Finally, and most importantly, the client layer creates the primary means by which complex job submissions are simplified by allowing automated generation of job submission configuration data.

2.2. Common submission infrastructure for all experiments

The motivation for using a single job submission infrastructure layer are essentially the same as those discussed for the submission client. In both cases, the use of a common code base will serve to reduce the effort required to support the software

1 While the exercise of abstracting the requirements from a specific system is important for the goal of arriving at the best possible specification for the system, we also note that a direct mapping from a set of conceptual requirements onto the features of an actual batch system is, obviously, extremely useful. The adoption of language pertaining to a specific batch system may therefore have benefits beyond simply communicating the requirements.

across many experiments. These gains are likely to be largest for the submission infrastructure layer since it is the more complex of the two.

2.3. Provides support for steering of jobs to specific resources

It is sometimes desirable to direct jobs to particular grid sites or to specific computing elements within a grid sites for reasons of testing or exploiting particular resources available at those locations. This feature is particularly important during times of operating system migration, which tend to occur over extended periods of time across the grid and on schedules that are beyond our control.

2.4. Supports the concept of "groups" for the purpose of setting priorities and accounting

Groups are aggregations of users to whom a set of specific privileges are granted. There are two levels of group definitions that are important for proper resource management. The first is membership in a particular experiment. Computing resources at a given site are often allocated according to experiment. Jobs may be granted priority access to a particular set of machines, or a certain number of job execution slots based upon the user's affiliation with a given experiment. A site might also want to allow access by opportunistic users only after exhausting demand from a the end-users for a given experiment or set of experiments. For these reasons, the system must know and track the experiment affiliation. Most of the functionality required for these decisions is handled via the Virtual Organization infrastructure resident at grid sites. The job submission infrastructure may need to interface to these systems.

The second level of important group definitions is that which is created by the experiment for the purpose of managing usage of its limited computing resources. These groups can be used, for instance, to define special groups for various types of data processing, such as service groups for centrally managed data and Monte Carlo production, high priority groups for jobs that need expedited processing, low priority groups for jobs that should run only opportunistically within the experiment, and so on. These and other group definitions can be used to make resource allocation decisions, such as limiting the number of execution slots available, automatic steering of jobs to particular resources, etc.

We would envision that users specify the group under which a particular job is run. A set of default groups would be provided to all users. Experiment management would determine high priority group membership.

2.5. Supports specification of external resources required by the job

Efficient management of limited resources under heavy demand requires knowledge of the specific resources required by a job prior to execution. Data input and output sources, for instance, can become overloaded with too many client applications, leading to large data access latencies and lost CPU cycles. A system aware of the potential load can mitigate the situation by throttling jobs, pre-staging data, or over-subscribing CPU slots. Other resource requests could lead to jobs being routed to specific sites. The job submission system must therefore support specification of required resources in sufficient detail to allow resource allocations to be optimized across contending jobs.

2.6. Supports job ordering dependencies

Job ordering requirements arise in a number of situations, such as that of a resource that requires extended job-specific provisioning, or that of a processing workflow consisting of multiple interdependent steps spread across independent job sections. For jobs that require data to be retrieved from tape, for instance, pre-staging of input data prior to the start of processing may be needed in order to ensure efficient use of CPU resources. Alternatively, constructing a Monte Carlo production workflow to support a repetitive sequence of event generation, simulation, and reconstruction steps may provide a significant reduction in effort to the end-users responsible for running the sequence. In all such cases, job submission maintainers should be involved with the configuration and deployment of the required workflow infrastructure.

2.7. Supports logging of job submission information not available via the batch system

A considerable amount of information regarding the type of processing for which a job is intended can be useful for monitoring operations, managing resources allocations to meet physics goals, or in planning resource procurements. Generally speaking, however, none of this information is available from the batch system monitoring infrastructure. The job submission system must therefore support specification and logging of this information.

2.8. Operational requirements

No operational requirements have yet been specified, although there are several that may merit consideration, such as up-time requirements, non-disruptive upgrade requirements, etc.

2.9. Provides extensible and maintainable code base

The following extensibility features should be considered as requirements of the job submission system. First, implementation of experiment-specific customizations should not require modification of the core job submission infrastructure. Configuration files and sub-classing from core classes provide two mechanisms by which to accomplish this requirement. Second, the job submission configuration must be adaptable from the command line. This feature is needed for rapid testing and adaptation to changes in the underlying batch system or other components that are beyond the control of the system operators.

2.10. Returns error messages that users can understand and can link to appropriate corrective actions

Ideally, error messages will identify an underlying root cause or a small set of possible root causes with well-defined prescriptions for corrective actions. These actions should be identified in the message or from an easily accessible secondary source. If the latter, the location of the secondary information must be broadly known within the end-user community.

2.11. Provides tools to assist with tarball creation

In general, worker nodes are isolated from the software environment and native data handling systems of an experiment. Working on the grid therefore inevitably involves transporting to the worker nodes not only the application program, but also a set of associated scripts and libraries needed to run the program and provide necessary services within that hostile environment. Part of preparing a grid job for submission typically involves creating one or more tarballs that contain all of these components. The task of packaging the job for the grid is a new step in the workflow for most IF experiment end-users, since the current IF computing model assumes that both the full software environment and native data handling systems are accessible at all times. The job submission system should provide a set of tools that assist end-users in identifying the components that need to be transported and in creating the necessary tarballs.

2.12. Provides sensible defaults so that the most simple submission command is correct for the largest possible fraction of users

The defaults of the job submission system should be configured in such a way that the most simple command possible, “submit <myjob> <to this resource>”, or the equivalent, is meaningful and adequate for the largest possible fraction of jobs and end-users. A user-defined override mechanism should be provided.

3. Short-term strategy

There currently exists a set of IF experiment-specific job submission scripts with names `minos_jobsub`, `minerva_jobsub`, etc. Although the feature set available with these scripts is limited, they have the advantage that they are already understood by users. One approach to meeting the job submission requirements in the short term is to consolidate the experiment-specific scripts into a single script with the experiment selected by local configuration or command line argument. The feature set could then be extended in order to provide the most important of the required functionality. Implementing this solution will require re-writing the infrastructure underlying these scripts. Part of this work has already been completed in a separate project with the goal of reducing the maintenance burden of supporting the individual submission scripts. Adding the capability to submit to grid sites is relatively simple, although much of the job configuration work still needs to be performed by the user.

4. Long-term strategy

In the long term, a single submission infrastructure must be developed or adopted to provide the back-end implementation for a job submission command common to the IF experiments. The solution must be extensible via sub-classing to provide experiment-specific behavior that cannot be accommodated via configuration. Providing an easy transition for end-users and a low maintenance burden for system operators will be important goals in evaluating solutions. If an existing non-commercial system is adopted, then CD/REX will seek formal recognition as a product stakeholder, possibly to the extent of accepting co-development responsibilities. Given the breadth of IF experiments to be supported, such an arrangement is required to ensure that operational issues, new feature requests, etc., are dealt with in an expeditious manner.

There are currently at least two existing systems with the potential to provide the required feature set, a significant sub-set of required features, or some models for good solutions. These are the CafSubmit / CafExe / CafMon infrastructure in use by CDF, and the PANDA system in use by ATLAS.

5. Summary

The requirements for the IF job submission system described in this document should afford the IF experiments with sufficient tools and flexibility to manage utilization of available grid computing resources to meet the physics priorities of the experiments.

At the same time, the system will allow system operators to maximize efficiency and throughput given the management strategy of the experiment. Beyond these technical goals, the overriding objective in adopting a job submission solution will be to simplify for end-users the task of submitting and managing large-scale grid computing projects.