The SAM-Grid / Runjob Integration

Gabriele Garzoglio, Andrew Baranovski, Sinisa Veseli

February 14, 2005

Abstract

The SAM-Grid is an integrated job, data, and information management infrastructure adopted by the Run II experiments to assist with their meta-computing needs. It is currently integrated with the mc_runjob software suite, in order to support the production of simulated events and the reconstruction of raw files for DZero. By design, mc_runjob is used to shield the grid infrastructure from application-specific knowledge, for example in the preparation of the job environment. In many practical cases, though, we feel that the separation of responsibilities is sometimes fuzzy.

The mc_runjob team has developed a new suite of software, called Runjob, which generalizes and extends the capabilities of mc_runjob. The integration between SAM-Grid and Runjob is necessary because the support for mc_runjob will naturally fade away. The integration, though, is also an opportunity to define a clear separation between the domains of the two middleware suites and redesign their interaction.

This document states the principles upon which the separation of responsibilities should be based. It lists a series of application-specific information that is currently handled by the SAM-Grid and that we believe should be handled by Runjob. The document also discusses features of the SAM-Grid that were found of value in our past experience, hence should be retained for the future. In the end, it describes the way the SAM-Grid currently uses mc_runjob.
Contents

1  SAM-Grid and Runjob: Separation of Responsibilities 3
2  Application-Specific Features Implemented in the SAM-Grid 3
3  How SAM-Grid Uses mc_runjob 5
1 SAM-Grid and Runjob: Separation of Responsibilities

The SAM-Grid is an integrated job, data, and information management system. It supports running applications on largely distributed resources. Ideally, such an infrastructure should be shielded by information that is application-specific.

In our experience, for high energy physics (HEP) applications, the creation of the job environment is a non-trivial task, which dramatically influences the behavior of the application itself. In addition, HEP applications requires a careful management of the job work flow, which often consist of chains of multiple executables.

To deal with the details of environment preparation and work flow management, the DZero community has development a suite of software called mc_runjob. The SAM-Grid uses mc_runjob to separate its middleware from the application layer. In other words, this design clearly separates between a grid management middleware and a application management middleware capable of interacting with various grid infrastructures.

In practice, we have seen that there are various "gray" areas in the definition of the responsibilities between these two entities. In addition, there are cases where historical needs have dictated the implementation of features on one middleware suite as opposed to another. Section 2 analyzes some of the SAM-Grid features that we believe should be best implemented in the application middleware layer.

2 Application-Specific Features Implemented in the SAM-Grid

This section lists a series of features that we believe could be migrated to Runjob.

- **Application wrappers**: the DZero applications that run today on the SAM-Grid are not completely grid-enabled. In other words, the applications do not know how to take advantage of some grid and fabric services, such as sandboxing, grid monitoring, or some services of data handling. The SAM-Grid has implemented wrappers that interact with
the grid on behalf of the application. Some of these wrappers contain application-specific knowledge, which we believe should be transferred to the Runjob framework. The following is the most relevant case.

The SAM-Grid provides facilities to import datasets from SAM in the job environment. This feature is used, for example, to dynamically install the DZero software before launching the application. This resulted to be very useful to guarantee the reproducibility of the job results. Some wrappers use this feature to import files, such as input data, minimum bias files, and montecarlo generated input files. On the other hand, for all these cases, the wrapper uses application-specific knowledge to prepare these files. For example, the wrapper imports only a specific number of minimum bias files, or a specific set of events from the generated input.

- **Number of application instances**: a grid system is responsible for managing resources according to the policies of the virtual organizations. A typical example of such policy is maximum time for job completion. These policies are generally enforced by optimizing metrics such load of grid services. A mechanism to implement these optimizations is splitting a job into multiple instances, each executing on different resources. This splitting mechanism must be transparent to the user, in order to ease job management. For example, if a user wants to cancel a job request, the system must provide a mechanism to automatically cancel all the jobs instances.

There are situations where the application may be able to recommend a certain degree of job multiplicity. For example, the DZero reconstruction application needs in average two days to process an input file. Splitting the job request into as many jobs as files in the input dataset is a reasonable policy declaration to minimize job completion.

Currently, these application-specific policies are declared and implemented by the SAM-Grid job management layer. We believe that the SAM-Grid should be responsible for **implementing** the policy, but not for **declaring** it. In other words, the application middleware should give SAM-Grid a recommendation on the job multiplicity, thus declaring the application-specific policy. The SAM-Grid should then be responsible to weight this recommendation with other constraints and implement a certain job splitting.
• **The Job Description Language**: the SAM-Grid JDL contains application-specific knowledge, such as the version of DZero code release, or the concept of minimum bias file. This knowledge should be transparent to the SAM-Grid JDL and interpreted by the Runjob layer.

• **User Analysis Preparation**: the SAM-Grid provides minimal support for the creation of the run time environment of user analysis code. We believe that the integration with Runjob can provide such facilities. This should be done preserving the capabilities of the SAM-Grid to dynamically install software releases.

### 3 How SAM-Grid Uses mc_runjob

SAM-Grid uses mc_runjob to run three types of applications: monte carlo chains, reconstruction, and file merging. In all cases, the job management layer (application wrappers) prepares the environment for mc_runjob at the worker node, then passes control to it. The mc_runjob code is dynamically installed at the node and it is delivered via SAM.

In the case of the reconstruction and merging applications, the wrapper prepares the mc_runjob macro from configuration parameters coming either from the user or from the execution site. In the case of monte carlo, the wrapper prepares the pre-processing macro, with information such as the montecarlo request id, then mc_runjob gathers the rest of the information from the SAM db server.

In general, we believe that the job management should provide generic primitives to prepare the job, such as file delivery, environment variable setting, application multiplicity. Runjob should take advantage of these primitives, shielding itself from the details of the fabric environment.