

The SAM-Grid / LCG Interoperability Project

Gabriele Garzoglio, Andrew Baranovski, Sinisa Veseli

February 04, 2005

Abstract

The goal of this project is making available to DZero the resources provided by the LHC Computing Grid (LCG) through the SAM-Grid system.

This document explores an implementation that provides a job forwarding mechanism from the SAM-Grid to the LCG systems. It discusses modifications to the current SAM-Grid system and tries to define a division of responsibility between the LCG and the SAM-Grid team.

Contents

1	Motivation	3
2	Limitations and Requirements	3
3	Highlights of the proposed architecture	3
4	Known and Potential problems	5
4.1	Accessibility of the services	5
4.2	Usability of the resources	5
4.3	Scalability of the services	6
4.4	Security configuration	6
5	Implementation	7
5.1	Prototype Test Bed	7
5.2	Development	8
5.3	Description of the typical job flow	12
6	Tasks and Responsibilities	15

1 Motivation

Goal of the project is making available to DZero the resources provided by the LHC Computing Grid (LCG) through the SAM-Grid system.

The project has also an educational component, as it will bring together the experience of different Grid communities to provide better quality of service for the current and future Grid systems.

2 Limitations and Requirements

Before proposing a software architecture, we try to define a few limitations/requirements that stand irrespectively of the architectural choices.

- Most of the LCG resources will not provide a SAM-Grid gateway node. The necessary VO-specific services will be instantiated "near" the LCG sites.
- The project should require minimal changes to the current infrastructures. In other words, this is an *integration* project that may require some minor development effort.

3 Highlights of the proposed architecture

We propose an architecture that provides a job forwarding mechanism from the SAM-Grid to the LCG. Submitting jobs from LCG to the SAM-Grid resources is not the focus of the project at this time. Considering a pre-existing deployment of SAM-Grid and LCG, the two main new types of components are the following:

- SAM-Grid/LCG forwarding nodes: architecturally, what is today the SAM-Grid gateway node of a site, becomes a gateway to the whole LCG grid. There will be one or more such nodes. Multiplicity of this node can help overcoming potential scalability limits.
- Remote SAM-Grid VO-specific services: these include data handling, monitoring, sandboxing. In the SAM-Grid model these services are instantiated at every "execution" site. In the integrated model, they will be provided remotely to multiple LCG sites at a time.

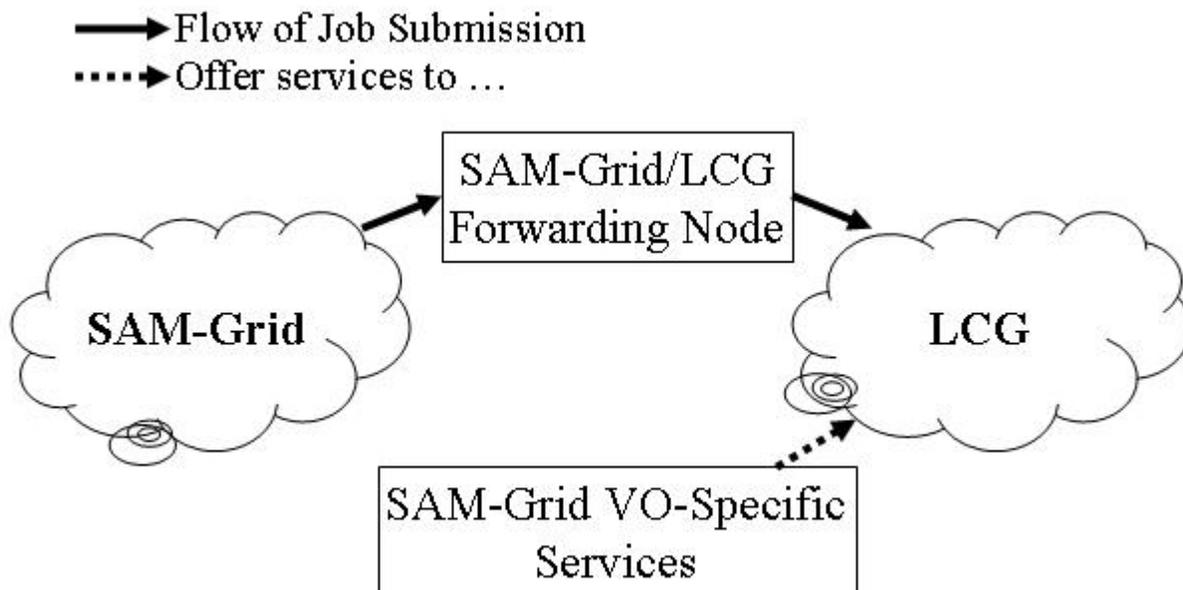


Figure 1: The proposed SAM-Grid/LCG integration architecture. Forwarding nodes will dispatch jobs from the SAM-Grid to LCG. SAM-Grid VO-specific services will be offered to LCG remotely.

Figure 1 shows a graphical representation of this concept. The advantages of such approach are

- a clear separation of responsibility between resource providers and grid infrastructures providers.
- the maintenance of the forwarding infrastructure and the VO-specific services is restricted to a limited numbers of nodes.

We propose to develop an initial prototype for the system with a single forwarding node and a single node for the SAM-Grid VO-specific services.

4 Known and Potential problems

4.1 Accessibility of the services

A service assumes certain configurations of the resources at a site. A typical example is network topology. *We foresee problems with the current implementation of the data handling services.* Some SAM station interfaces use call backs. In our experience these call backs may fail either because of firewall configurations at a site or because of the need to reach a client within a private network. Some of these issues are immaterial if the station is located within the site OR if the site firewall and network address translation allow incoming communication. On the other hand, the configuration that we assume consists of a SAM station external to LCG sites that do not allow inbound network traffic.

We have two proposals to address the accessibility problem: one takes about 1 man-month and is based on v5 the other takes about 4 man-month and is a partial migration of JIM to SAM v6 (see section 5 for details).

It should be noted that *this integration project requires computing sites that allow outgoing connectivity from the worker nodes.*

4.2 Usability of the resources

The SAM-Grid makes very little assumptions on the worker node environment. On the other hand these issues should be considered:

- The SAM-Grid software has been tested for production quality of service on Linux platforms only. We have no experience on different platforms.
- The worker nodes of a cluster need to know their fully qualified domain name. This requirement can be lifted with some development time (on the order of a couple of days).
- The clock of the worker nodes should be synchronized with the standard time. Failure in the clock synchronization leads to the Globus Security Infrastructure not working properly.

4.3 Scalability of the services

Local services and resources are configured having in mind a certain load. The load is directly related to parameters of the site, such as the size of the cluster, the policy to run jobs, etc. We understand these factors for jobs that are submitted to a cluster of computers, but we have no experience for jobs that are submitted to a set of grid sites. To overcome excessive load to the services, we need to design load balancing mechanisms. These can be done on a site by site basis or on a grid job by grid job basis. The appropriate strategy depends on the service.

For the prototyping phase, we will operate the system within the natural limits imposed by the load to the forwarding and the VO-service nodes.

To move beyond the prototyping phase, we need to address the following questions:

- how do we associate stations to LCG clusters/grid jobs? Is it better having stations associated with clusters to take advantage of caching mechanisms?
- how do we associate monitoring repositories to Grid jobs?
- how do we select storage services (durable locations)?

For the prototype, the instances of the services above will be hard-coded at the forwarding node. This is the association mechanism that is currently implemented in SAM-Grid, as it assumes the locality of the services.

4.4 Security configuration

The SAM-Grid security model is based on the Globus Security Infrastructure. We need to agree on an interoperable security configuration for authentication and authorization.

- Authentication infrastructure: we need to agree on a set of Certificate Authorities trusted by both systems. SAM-Grid users and services use mainly two CAs: DOEGrids and FNAL-KCA.
- Authorization infrastructure: SAM-Grid uses internal mechanisms to distribute authorization lists (grid-mapfiles). LCG should authorize DZero users for the use of its resources. This may require some work

on the LCG security service infrastructure, such as the creation or updating of the DZero VO. The certificate subjects of the DZero users can be accessed from the SAM database, using the samadmin or the database interfaces.

5 Implementation

5.1 Prototype Test Bed

For the prototype, we need to deploy a test bed for the forwarding node and for the SAM-Grid VO-specific services.

Forwarding Node: we propose to install the standard SAM-Grid execution site software suite. In other words, a job forwarding node will provide the following services:

- A Globus gatekeeper that can use the SAM-Grid job-managers.
- The JIM sandboxing service, including a gridftp server for data transport and an fcp server for load balancing .
- The Xindice XML Database, for configuration management and push-based monitoring. This service can be instantiated on a different machine, after the prototypical phase.
- A SAM "stager" registered with the SAM file storage server (FSS). Worker nodes buffer their output in the sandbox area of the forwarding node. The stager is used to transfer the output data to the final storage location. The routing strategy for the output data can be changed after the prototypical phase, lifting the need to instantiate a stager at the forwarding node.
- The SAM-Grid advertisement service, which registers the characteristics of the forwarding node with the SAM-Grid information collector. This information is used by the broker for the resource selection. This service can be run on a different machine.
- The LCG job submission client software.
- An LCG SAM Batch Adapter: this item needs to be developed (details later in this section).

VO Services: Considering that some services are instantiated for convenience at the forwarding node, the remaining VO services consist of a SAM Station and a durable disk storage.

The accessibility limits of the current implementation of the station (section 4) may prevent the deployment of a functional test bed. This may hinder our ability of developing the software for the forwarding node (sam batch adapter). To overcome this problem, we propose to instantiate the SAM Station within the boundaries of an LCG site. This site will be the initial test bed for job submission. With this configuration, also LCG sites that allow inbound network connectivity to their worker nodes can be part of the initial test bed.

5.2 Development

The main development consists of

1. migrating the SAM-Grid software suite to adopt station polling interfaces
2. the implementation of a SAM-Grid job management handler that uses the LCG client software to manage LCG jobs.

We are not aware of any other development. By following this proposal, the LCG handler will submit jobs as prepared by the sandboxing infrastructure of the SAM-Grid at the forwarding nodes. The job will consist of a self extracting bootstrapping archive, which includes a gridftp client, the user credentials, and a driving script. The driving script will be responsible to get the needed configuration and infrastructure i.e. the sam client and application specific wrappers. The wrappers will use SAM to get the appropriate *mc_runjob*, code releases, and auxiliary files (minimum bias, card files, etc.). There should be no need to do put extra effort in the development of *mc_runjob* for this project.

SAM Polling Interfaces

1. **Solution 1:** development of polling interfaces for the SAM station v5. This work will take about a month of Andrew Baranovski's time. The methods involved are the following:

- sam get next file: this interface is used to get a file from a sam project. The station does the bookkeeping of this process with a high-level of detail. the Project Master calls back the client once the file is available in the SAM Station cache.
- sam get dataset: this interface is used to get a set of files from a dataset. The station does the bookkeeping of this process with a low-level of detail. The Station Master calls back the client whenever a file is ready in the SAM Station cache.
- sam store: this interface is used to store a file. The File Storage Server (FSS) notifies the client upon completion of the storage.

2. **Solution 2:** partial migration of JIM to the SAM v6 software suite. This also requires the implementation of the polling interface for “sam get dataset” in SAM v6. The implementation will take about 4 man-months of SAM-Grid experts.

The idea is deploying both v5 and v6 sam client software to the worker nodes and use selectively one or the other depending on the situation. At a minimum, we need to modify the following components

- job management: After the migration, this component will use v6 of db server, station master and FSS. Migrating this component requires work on two main areas: (1) migration of the “sam get dataset” to sam v6. The “sam get dataset” command is currently implemented only for v5. This component is responsible for requesting files to SAM with a low-level of bookkeeping. It is used to get binary files, minimum bias files, etc. (2) Development of a clean switching mechanism between sam client v5 and v6 and relative environment. Currently the two versions are in conflict and cannot be used together. This mechanism will be used to prepare the environment for those sections of execution that require one suite or the other. For example, we plan to switch from sam client v6 to v5 when passing control to mc_runjob, after preparing the job environment.
- mc_runjob: after the migration, this component will use the v5 client and db server and the v6 FSS. We plan to change the SAM-StorageManager component only, so that the polling of the file storage status is done dumping the internal status of FSS, instead

of relying on a call back. This work around is necessary because it is difficult to adopt the v6 polling interfaces of FSS in the v5 sam client.

- merging application: after the migration, this component will use v6 of sam client, db server, station master and FSS. It requires full porting to the v6 client API.

LCG Job Management Handler: The handler is an executable that implements an interface to submit, lookup, and kill jobs. The implementation must be robust against failures of the LCG client software. This handler is used at the forwarding node by the job manager to manage jobs. The interaction between the handler and the job manager is mediated by the sam batch adapter. The responsibility of the sam batch adapter is providing a uniform interface to the underlying handler. The documentation is at http://d0db-prd.fnal.gov/sam_batch_adapter/

In detail, the handler will need to provide the following interfaces

- **job submit:** submits a single job to the LCG grid. This interface will be typically called multiple times for a given grid job: a grid job is split into multiple instances of LCG jobs at the forwarding nodes.

input parameters:

- executable: provided by the SAM-Grid framework at the time of job submission. It will consist of the self extracting bootstrapping archive
- project: a global job identifier that needs to be associated with the job. This information is used by the look up interface to return an aggregate status of all the jobs.
- stdout and stderr: the locations of the output and error streams at job completion. Each location is a full path on file system of the forwarding node. These parameters are used by the sandbox infrastructure to bundle the output sandbox.
- any other LCG specific input: this input can be configured at the forwarding node in the batch adapter configuration. They will be transparent to the SAM-Grid. An example of such parameter for PBS is the local queue name.

output parameters: a string containing the id of the local job. This is used for monitoring purposes. The string should be easy to parse.

example: `sam_lcg_handler.sh job_submit -project=%__USER_PROJECT__ -executable=%__USER_SCRIPT__ -stdout=%__USER_JOB_OUTPUT__ -stderr=%__USER_JOB_ERROR__`

- **global job lookup:** returns the aggregated status of all the job submitted with the same global job id. The signature of this interface should be the same as the local job lookup. The behavior should differ depending on what parameters are passed.

input parameters:

- project: a global job identifier. This was used to “name” the job during job submission.
- local-job-id: should be NULL

output parameters: A newline-separated list containing the local-job-id AND the status of each local job. Acceptable statuses are “pending”, “active”, “suspended”, “failed”, and empty string = “done”. The output should be easy to parse, e.g. a newline-separated list containing in each line `JobId=%__BATCH_JOB_ID__ Status=%__BATCH_JOB_STATUS__`

example: `sam_lcg_handler.sh job_lookup -project=%__USER_PROJECT__ -local-job-id=%__BATCH_JOB_ID__`

- **local job lookup:** returns the status of a given LCG job. The signature of this interface should be the same as the global job lookup. The behavior should differ depending on what parameters are passed.

input parameters:

- project: ignored if local-job-id specified
- local-job-id: the LCG identifier returned from the job submission

output parameters: A string containing the local-job-id AND the status of the job. It should be parsable in the same way as the output of the global lookup. Acceptable statuses are “pending”, “active”, “suspended”, “failed”, and empty string = “done”. The string should be easy to parse e.g. `JobId=%__BATCH_JOB_ID__ Status=%__BATCH_JOB_STATUS__`

example: `sam_lcg_handler.sh job.lookup -project=%__USER_PROJECT__
-local-job-id=%__BATCH_JOB_ID__`

- **global job kill:** kills all the job submitted with the same global job id. The signature of this interface should be the same as the local job kill. The behavior should differ depending on what parameters are passed.

input parameters:

- project: a global job identifier. This was used to “name” the job during job submission.
- local-job-id: should be NULL

output parameters: none

example: `sam_lcg_handler.sh job.kill -project=%__USER_PROJECT__
-local-job-id=%__BATCH_JOB_ID__`

- **local job kill:** kills a single LCG job. The signature of this interface should be the same as the global job kill. The behavior should differ depending on what parameters are passed.

input parameters:

- project: should be NULL
- local-job-id: the LCG identifier returned from the job submission

output parameters: none

example: `sam_lcg_handler.sh job.kill -project=%__USER_PROJECT__
-local-job-id=%__BATCH_JOB_ID__`

5.3 Description of the typical job flow

This is a description of the typical job flow from user job submission to job completion. It focuses on the steps of the middleware at the forwarding node (“Grid job submission” and “Grid job polling and termination”) and at the worker node (“Batch job activity”)

Grid job submission

- The user submit a job to the SAM-Grid.

- The SAM-Grid system chooses the forwarding node as the gateway to the computational resources (selection criteria can be programmed in the SAM-Grid broker; this is not the focus of the present document; at a minimum, the user will be able to specify the name of the forwarding node in its job submission request).
- The grid job enters the forwarding node via the gatekeeper and it is passed to the SAM-Grid job-manager.
- The job-manager prepares the job sandbox area and a bootstrapping script that will be executed at the worker node. The sandbox area includes the sam client software, configuration files, application specific wrapper scripts. The wrapper script interact with the grid on behalf of the application. This is necessary because the D0 applications that we have encountered so far are not grid-aware.
- The job-manager uses the sam batch adapter layer to learn how to submit a job to the computational resources.
- The job-manager executes the command returned by the sam batch adapter and intercept the output. *This command will be the handler.*
- The job-manager asks the batch adapter how to parse the output in order to extract the local job id, for monitoring purposes: the information is recorded in the local XML database.
- The job-manager submits as many instances of the job as required by the framework (depends on the application)
- Grid job submission is finished.

Grid job polling and termination

- The SAM-Grid periodically polls the status of the grid job querying the job manager.
- The job manager uses the batch adapter to retrieve such status. The information is logged to the XML DB, aggregated, and returned to the grid

- If the grid job is finished, log files, output streams, and small output from the application are packaged and sent back to the SAM-Grid. Some clean up is performed.
- The user can download the output from the SAM-Grid web site.
- The grid job is finished.

Batch job activity

- The local job starts up at a worker node of an LCG cluster. The LCG infrastructure is responsible for selecting the site and transporting the bootstrapping executable at the site.
- The local job executes the self-extracting bootstrapping script. This contains the gridftp client, the user credentials, a driver script. Control is passed to the driver script.
- The driver downloads from the forwarding node the input sandbox. This contains the user input sandbox, the sam client, configuration files, an application specific wrapper script. Control is passed to the wrapper script.
- For d0 montecarlo and reconstruction applications, the wrapper asks sam to retrieve the d0 software, mc_runjob and other files (minimum bias files, etc.). The wrapper copies these files from the sam station via fcp/gridftp.
- The wrapper untars the D0 code and passes control to mc_runjob, which controls the flow of execution. mc_runjob is also responsible for the storage of output file via sam. Output files are staged to the sandbox area at the forwarding node. From here, a stager working with the station provides for robust storage.
- The wrapper interacts with the XML database to notify internal relevant events.
- The wrapper packages the small output from the application and sends it back to the forwarding node.
- The LCG infrastructure returns the output stream of the job to the forwarding node.

- Local job is terminated.

6 Tasks and Responsibilities

Referring to the section on implementation (section 5), we propose the following tasks and division of responsibilities

- Preparation of a test bed: for the forwarding node, the SAM-Grid group should help with the installation and configuration of the SAM-Grid gateway software. The LCG group should install the LCG client software. For VO-specific services, such as the SAM station, we should install the software within the boundaries of a test LCG site. This can be avoided if there are sites that allow worker node inbound connectivity. This restriction will be lifted as soon as the development on SAM is complete.
- The development of the handler/batch adapter: the LCG group should develop the handler. The SAM-Grid group should configure the batch adapter to use it.
- The development of the SAM polling interfaces: the SAM-Grid team will be responsible for this work. It requires one month of Andrew Baranowski's time.
- Testing: both groups should be involved.