

## ReSS: a Resource Selection Service for the Open Science Grid.

Gabriele Garzoglio<sup>1</sup>, Tanya Levshina<sup>1</sup>, Parag Mhashilkar<sup>1</sup>, Steve Timm<sup>1</sup>

<sup>1</sup> Fermi National Accelerator Laboratory  
Pine st. And Kirk Rd.  
60510 Batavia, IL, USA  
{garzogli, tlevshin, parag, timm}@fnal.gov

**Abstract.** The Open Science Grid offers access to hundreds of computing and storage resources via standard Grid interfaces. Before the deployment of an automated resource selection system, users had to submit jobs directly to these resources. They would manually select a resource and specify all relevant attributes in the job description prior to submitting the job. The necessity of a human intervention in resource selection and attribute specification hinders automated job management components from accessing OSG resources and it is inconvenient for the users.

The Resource Selection Service (ReSS) project addresses these shortcomings. The system integrates condor technology, for the core match making service, with the gLite CEMon component, for gathering and publishing resource information in the Glue Schema format. Each one of these components communicates over secure protocols via web services interfaces.

The system is currently used in production on OSG by the DZero Experiment, the Engagement Virtual Organization, and the Dark Energy. It is also the resource selection service for the Fermilab Campus Grid, FermiGrid. ReSS is considered a lightweight solution to push-based workload management.

This paper describes the architecture, performance, and typical usage of the system.

**Keywords:** Resource Selection, Large Distributed Computing, Open Science Grid

## 1 Introduction

The Open Science Grid (OSG) [1] is a consortium of US National Laboratories and Universities that provides a US-wide Data Grid to address the computing needs of scientific communities. OSG makes available to its collaborators hundreds of computing and storage resources. For such large distributed system, the selection of an appropriate set of resources to run user applications can become a complex problem. Even when such selection occurred, the Grid middleware and the application environment often need to be informed about the characteristics of the selected resource, in order to dispatch and run the application appropriately.

In September 2005, the Resource Selection Service (ReSS) Project [2] was charged with developing and integrating an end-to-end solution to these problems for the OSG. The project was sponsored by the DZero experiment [3] and the Fermilab Computing Division in collaboration with the Open Science Grid, FermiGrid [4], the CEMon gLite Project (PD-INFN), and the Glue Schema Group.

There were five main goals that the project set for itself:

- Implement a light-weight cluster selector for push-based job handling services
- Enable users to express requirements on the resources in the job description
- Enable users to refer to abstract characteristics of the resources in the job description
- Provide registration for clusters and mechanisms for automatic expiration of the registration
- Use the standard characterizations of the resources via the Glue Schema

Being the sponsoring Virtual Organization (VO), DZero provided most of the initial requirements. However, the resulting system is now adopted on the Open Science Grid as a general service and used by the DZero experiment, the Engagement VO, and the Dark Energy Survey. In addition, the ReSS system is the resource selector for the FermiGrid Campus Grid.

This paper presents the Architecture of ReSS, the Resource Selection model utilized, the current deployments of the system, and discusses the results of different studies and evaluations.

## 2 Architecture

The ReSS system is composed by two components:

1. an information provider and publisher service, deployed at the resources
2. an information repository and resource selector, deployed semi-centrally.

Figure 1 shows a diagram of the system architecture.

(1) Information providing and publishing is implemented by the CEMon service, a component of the gLite software. CEMon collects information at the resource by invoking local commands, logically grouped using the abstraction of "Sensors". The sensor developed for OSG invokes and parses the output of the Generic Information Providers (GIP), a suite of scripts that present resource information organized according to the Glue Schema [5]. For each sensor, the information can be presented in different formats, currently, LDIF, XML, new classad, and old classad. Each

format is implemented according to the specifications of Glue Schema mapping documents. The ReSS project has driven the definition of the Glue Schema to old classad format mapping. ReSS uses the old classad format, to use Condor software for resource selection.

CEMon can publish information synchronously or asynchronously. Clients can access information synchronously, by invoking web services interfaces directly, or they can subscribe to CEMon in order to receive periodic asynchronous events. Alternatively, administrators can configure CEMon with a pre-defined list of subscribers, so that specified clients can be periodically notified with asynchronous events.

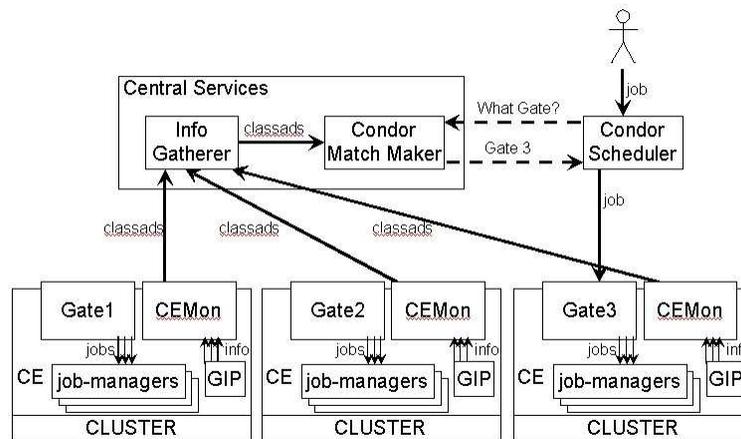


Fig. 1. The architectural diagram of the Resource Selection Service.

(2) ReSS uses asynchronous event notification to publish resource information to the central information repository. The Condor [6] Information Collector implements this repository, which is used by the Condor Match-Making service to implement resource selection.

Information is routed from CEMon to the Condor Collector by the Information Gatherer (IG) component. Effectively, IG acts as an interface adapter between the two services. In ReSS, IG is deployed as a central stateless service. The service can be configured to apply simple transformations to the incoming information. This feature is used to add attributes containing expressions that, when evaluated, validate the semantics of the attributes for the OSG use case. These expressions check, for example, the presence of "critical" attributes, the consistency of attribute values (e.g. the number of nodes in a cluster cannot be negative), etc.

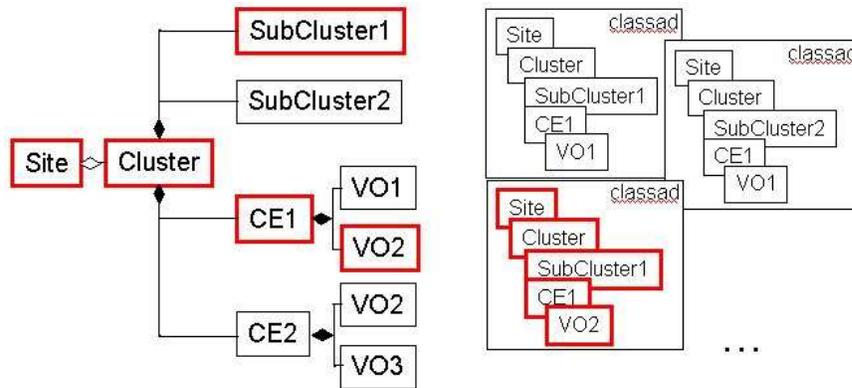
Since resource selection is implemented using Condor components, the infrastructure provides for a seamless integration of condor-based job scheduling services, like Condor-G. In addition, Condor provides interfaces to query the central information repository, in case users prefer to adopt ad-hoc algorithms for selecting resources.

## 2.1 Glue Schema to old classad mapping

The Glue Schema is a resource description model adopted by commercial companies and Grid organizations. Resources are described in terms of entities, such as Clusters or Storage Areas, and their logical relationships, like association or aggregation, are expressed using UML diagrams.

In order to use readily available match-making services for resource selection, ReSS describes resources in old classad format, an unstructured list of [attribute , value] pairs. The project therefore faced the problem of mapping the Glue Schema structure into a set of unstructured classads. The activity was conducted in collaboration with the Glue Schema group and resulted in a mapping document [7] and its implementation in CEMon.

In the Glue Schema, a computational resource is described by a "Cluster", which is part of a computing center or "Site". A Cluster is composed by one or more groups ("Subclusters") of homogeneous computing nodes ("Hosts"). Hosts are characterized by parameters related to the processor type, memory, operating system, etc. Access to the Cluster can be achieved by one or more gateways or queues ("Computing Elements" or CE). The Computing Element is described by informational parameters, such as the gateway address, state parameters, such as the number of running or idle jobs, policy parameters, such as the maximum wall clock time allowed by the local scheduler, etc. In addition to total values for these parameters, like the total number of running jobs at the CE, the model also allows for VO-specific values ("VOView"), like the number of running jobs for a specific VO.



**Fig. 2.** Mapping the Glue Schema "tree" (left) into a set of "flat" classads (right): the algorithm navigates through all possible combinations of (Cluster, Subcluster, CE, VO)

Figure 2 shows a schematic UML representation of a computational resource. The mapping between this structure and a set of old classads is built considering all possible combinations of inter-related CE, Cluster, and Subcluster entities. In other words, each combination (classad) contains a single CE, Cluster, and Subcluster entity. In addition, if VO-specific parameters are available to characterize the CE (VO View entity), these are used instead of the general CE attributes.

Each resulting classad can be thought of as a virtual homogeneous cluster with one access gateway, described from the point of view of the VO. The ReSS system matches each job to one of these "virtual" computational resources.

Storage descriptions are "flattened" using similar rules and are added to their associated Cluster classad.

Being combinatorial, this algorithm could in principle produce a very large number of classads for each site. In practice, typical OSG installations consist of one storage resource and one cluster, considered homogeneous for simplicity (i.e. one Subcluster), with a few Computing elements and up to a couple dozen supported VOs. In the current system, complex sites advertise 48 classads, while simple sites advertise 1. With this multiplicity, we expect the system to scale up to a few hundreds sites before incurring into scalability limitations of the condor system. This limit is deemed acceptable considering that the current number of OSG sites is about 80 and the annual growth is typically of a few sites per year.

## 2.2 Typical Uses of the System

ReSS provides a direct and indirect mechanism to access resource information from the system.

The direct mechanism consists in querying the ReSS information repository. Queries are a set of constraints on the resource attributes (e.g. "List all clusters that support the DZero VO"). Filters can be applied to display specific information only. Resources that meet the specified criteria are returned to the user in bulk, appropriately formatted according to the given filters. The user can then run ad hoc algorithms to narrow down her selection and submit the job using standard Grid interfaces.

The indirect mechanism consists in accessing the ReSS information repository through the Condor-G system. Users specify their resource requirements and rank criteria using the condor job description language. Attributes can be dereferenced to enhance the job environment or to send directives the job handling middleware.

Fig 3 show an example of a simple job description and how it references attributes from a resource classad.

A popular way of using the ReSS system is a hybrid between the direct and indirect methods described above. A VO queries the ReSS information repository directly, getting an initial coarsely-selected list of resources. Each classad in this list is enhanced with VO-specific attributes. For example, the Engagement VO of OSG adds parameters from test jobs run periodically at resources. The enhanced list is then uploaded to a Condor matchmaker, controlled by the VO. Users of the VO configure the schedulers of their Condor-G deployments to point to the VO match maker. Thus, users can access the VO matchmaker indirectly, specifying resource and VO-specific attributes in the condor job description language.

The DZero experiment adopts a direct mechanism to submit jobs to OSG resources. DZero jobs are logically grouped in units of computation. In general, these units consist of several jobs. An example of such computation, called data processing, applies a data transformation algorithm to an input dataset, consisting of multiple files. Because of the typically long processing time, each file is input to a single job. The jobs that process a whole input dataset define the unit of computation.

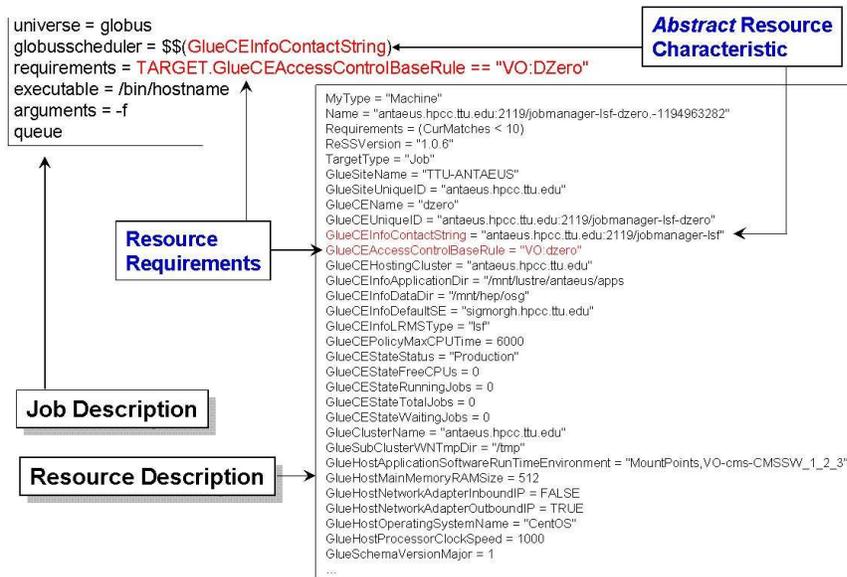


Fig. 3. Attributes in the resource description (right) are referenced in the job description (top-left).

By policy, jobs belonging to the same unit of computation are executed at the same cluster. Resources are selected for the whole unit by querying directly the ReSS system. For each cluster, the resource-ranking algorithm computes the ratio of the number of Idle jobs (jobs queued at the cluster's local scheduler) over the number of Running jobs. The whole unit of computation is submitted to the resource with the lowest ranking value or, in other words, to the resource that has the least Idle jobs per Running job. The algorithm also strongly penalizes clusters with Idle jobs, but no Running jobs.

Variations of this simple algorithm are used in production by DZero to select OSG resources.

### 3 ReSS deployments

The ReSS system has been deployed at two major Grid infrastructures: the Open Science Grid and FermiGrid, the Fermilab campus Grid.

ReSS central services are deployed for OSG and FermiGrid on Xeon 3.2 GHz 4-CPU machines with 4 GB of RAM. These machines run the services at a very low load (<1).

On FermiGrid, the ReSS publishing services (CEMon) have been deployed on 8 of the campus clusters, advertising a total of almost 750 classads for a total of more than 12,000 job slots. The campus grid can be accessed through a single gateway via the

GRAM protocol. Jobs are locally queued up using Condor-G, before being routed to the resource that meets the job requirements.

On OSG, CEMon is deployed at about 64 sites, producing short of 2000 classads.

## 4 Validations and Evaluations

The ReSS project has undergone 2 independent evaluations processes:

1. a study of the resources utilized by CEMon when running on a typical OSG machine: this study compared CEMon to other popular information publishing services (MDS2) [8]

2. an evaluation of the use of ReSS to meet the Workload Management requirements of the US-CMS VO: this evaluation compared several workload management technologies [9].

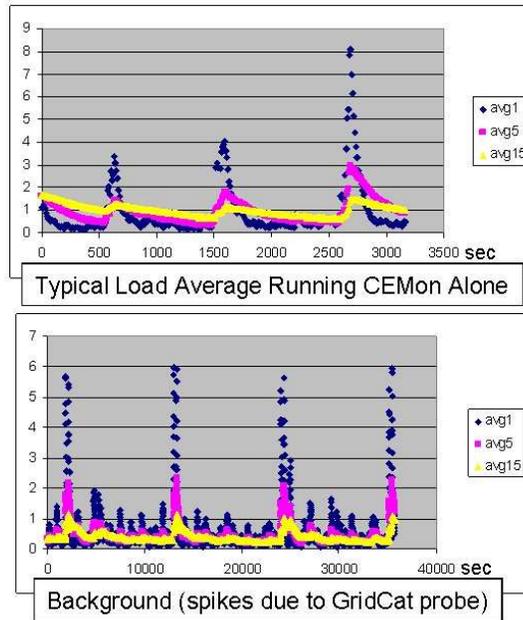
In (1), we studied the resource requirements of two information-publishing services, CEMon and GRIS/MDS2, on a typical OSG node under two different conditions. The first condition simulated a heavily loaded environment, where information on resources was required continuously by several external services. The idea was investigating the difference in resource utilization between the CEMon model, where information is pushed to a central collector, versus the MDS2 mode, where information is pulled directly from the site monitoring services by external services. The second condition compared the two publishing services queried at the same slow frequency, one every 10 minutes.

The machine characteristics studied were load, percentage of CPU utilized, and memory consumption. The conclusions of the study are

- running only CEMon does not generate more load than running only GRIS, or CEMon and GRIS together.
- CEMon uses less %CPU than a GRIS that is queried continuously, simulating a heavily loaded environment (0.8% vs. 24%). On the other hand, CEMon uses more memory (%4.7 vs. %0.5). This is not surprising because CEMon offers web services interfaces, a technology well known to be memory intensive.
- The average load to the machine is smaller when running CEMon alone (avg. 0.5) than when running a GRIS that is queried continuously (avg. 1.1). Both servers generate lesser load than when running the Generic Information Providers (GIP) scripts by hand continuously (avg. 1.8): this is expected because both servers cache data.

Fig 4 shows a typical load average profile, running only CEMon.

In (2), US CMS evaluated several Workload Management Technologies. The goal was measuring characteristics such as scalability and robustness, in order to select a technology that could meet the requirements of the VO. The study evaluated Condor-G and ReSS on a large test system, submitting thousands of jobs running the sleep command for a random time between 30 minutes and 6 hours. The test system consisted of 4 Grid sites that agreed to run the sleep jobs on the same nodes where "production" jobs were also running, virtually doubling their job slot capacity. Each cluster provided about 2000 slots for the sleep jobs.



**Fig. 4.** The load average of the machine vs. time; the measurements of the load resulting from running the CEMon process vs. time (top) are compared to the load when no user processes are running on the machine (bottom). The spikes on the bottom plot result from periodic processes being run for monitoring purposes.

Condor-G was evaluated submitting 40,000 jobs at a rate of about 8 Hz to 4 Condor-G schedulers. The system scaled well to this limit and resulted robust with a 0.5% failure rates. This rate could be in principle reduced by automatically resubmitting failed jobs. In some cases, it was observed crashes of the gateways. Gateway crashes resulted in discrepancies between the status of the jobs at the site and at the Condor-G scheduler.

Configuring the Condor-G scheduler to interact with the ReSS system did not change the scalability or robustness properties.

The conclusion of the study was that ReSS is a lightweight, scalable, and robust infrastructure. A criticism was that ReSS does not handle out-of-the-box user fair share among the VO, leaving this responsibility to site batch systems. Also, the ReSS model, which consists in pushing jobs to resources, does not allow for changes in user priorities after the job has been submitted. To overcome these limitations, US CMS decided to adopt the GlideIn Factory WMS technology and integrate it with ReSS for global-level resource selection.

## 5 Acknowledgements

We want to thank the developers of CEMon, in particular Massimo Sgaravatto and Luigi Zangrango, for their collaboration and promptness in addressing our concerns; the members of the OSG Integration Test Bed for their help in the validation of ReSS; the members of the Virtual Data Toolkit for their help in packaging CEMon; Igor Sfiligoi and Burt Holzman from US CMS for their evaluation of ReSS; University of Oklahoma, in particular Karthikeyan Arunachalam and Horst Severini, for spearheading the study on CEMon resource utilization; Marco Mambelli, UChicago, and MatsRynge, Renci, John Weigand, Fermilab, for their feedback.; the Glue Schema Group for their help with the Glue Schema to old classad document; FermiGrid for their interest in and contribution to the ReSS project. This paper was written at Fermilab, a US National Laboratory operated by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the United States Department of Energy.

## 6 Conclusions

The Resource Selection Service (ReSS) project provides cluster-level resource selection for the Open Science Grid and FermiGrid Campus Grid. The system uses the Glue Schema model to describe resources and the Condor classad format to publish information. ReSS integrates the Condor match making service, for resource selection, with gLite CEMon, for information gathering and publishing. The system naturally interfaces with the Condor-G scheduling system.

ReSS is a lightweight, scalable, and robust infrastructure for resource selection of push-based job handling middleware.

## References

1. The Open Science Grid home page: <http://www.opensciencegrid.org>
2. The Resource Selection home page:  
<https://twiki.grid.iu.edu/twiki/bin/view/ResourceSelection>
3. The D0 Collab., "The D0 Upgrade: The Detector and its Physics", Fermilab Pub-96/357-E.
4. D.R. Yocum et al.: "FermiGrid", FERMILAB-CONF-07-125-CD, May 2007. 5pp. Presented at TeraGrid '07: Broadening Participation in TeraGrid, Madison, Wisconsin, 4-8 Jun 2007.
5. The GLUE schema home page: <http://glueschema.forge.cnaf.infn.it>
6. J. Frey, T. Tannenbaum, M. Livny, I. Foster, and S. Tuecke, "Condor-G: A Computation Management Agent for Multi-Institutional Grids", in Proceedings of the 10th International Symposium on High Performance Distributed Computing (HPDC-10), IEEE CS Press, Aug. 2001.
7. The Glue Schema to Old Classad Mapping document:  
<http://glueschema.forge.cnaf.infn.it/SpecV13/OldClassAd>
8. K. Arunachalam, G. Garzoglio, "Performance Measurements of CEMon on an OSG Test Environment", OSG White Paper OSG-doc-521-v1

<https://twiki.grid.iu.edu/twiki/bin/view/ResourceSelection/CEMonPerformanceEvaluation>  
9 I. Sfiligoi, B. Holzman, "Evaluation of Workload Management Systems for OSG", Talk at  
the OSG council meeting on Mar 07  
<https://indico.fnal.gov/contributionDisplay.py?contribId=65&sessionId=13&confId=468>