ReSS: Resource Selection Service for National and Campus Grid Infrastructure

Parag Mhashilkar[1], Gabriele Garzoglio, Tanya Levshina, Steve Timm

Fermi National Accelerator Laboratory, P O Box 500, Batavia, IL – 60510. USA

{parag, garzogli, tlevshin, timm}@fnal.gov

Abstract. The Open Science Grid (OSG) offers access to around hundred Compute elements (CE) and storage elements (SE) via standard Grid interfaces. The Resource Selection Service (ReSS) is a push-based workload management system that is integrated with the OSG information systems and resources. ReSS integrates standard Grid tools such as Condor, as a brokering service and the gLite CEMon, for gathering and publishing resource information in GLUE Schema format. ReSS is used in OSG by Virtual Organizations (VO) such as Dark Energy Survey (DES), DZero and Engagement VO. ReSS is also used as a Resource Selection Service for Campus Grids, such as FermiGrid. VOs use ReSS to automate the resource selection in their workload management system to run jobs over the grid. In the past year, the system has been enhanced to enable publication and selection of storage resources and of any special software or software libraries (like MPI libraries) installed at computing resources. In this paper, we discuss the Resource Selection Service, its typical usage on the two scales of a National Cyber Infrastructure Grid, such as OSG, and of a campus Grid, such as FermiGrid.

1. Introduction

The Open Science Grid (OSG) [1] is a consortium of National Laboratories and Universities in the US and abroad, which address the computing and storage needs of scientific research communities. The Resource Selection Service (ReSS) [2] [10] project was started in September 2005 and sponsored by the DZero experiment [3] and the Fermilab Computing Division. The project was executed in collaboration with the Open Science Grid, FermiGrid [4], the CEMon gLite Project (PD-INFN) [12], and the GLUE Schema Group [5]. Initial requirements for the project were provided by the DZero experiment. ReSS is integrated with the OSG information systems and resources and is used in the OSG by VOs such as Dark Energy Survey (DES), DZero, and Engagement VO. ReSS is also used as a Resource Selection Service for Campus Grids, such as FermiGrid. VOs use ReSS to automate the resource selection in their workload management system to run jobs over the grid.

In the past year, the system has been enhanced to enable publication and selection of storage resources and of any special software or software libraries (like MPI libraries) installed at computing resources.

In this paper, we discuss ReSS, its architecture, its typical usage on the two scales of a National Cyber Infrastructure Grid, such as OSG, and of a campus grid, such as FermiGrid.

2. Architecture

ReSS is composed of three major components:

[1] To whom any correspondence should be addressed.

1. CEMon: an information gathering and publication service, deployed at the resources
2. Condor: an information repository and match making service, deployed semi-centrally
3. Information Gatherer (IG): an information gathering and forwarding service, deployed semi-centrally to relay resource information received from the publisher to the information repository.

Figure 1 shows the system architecture and deployment model for ReSS

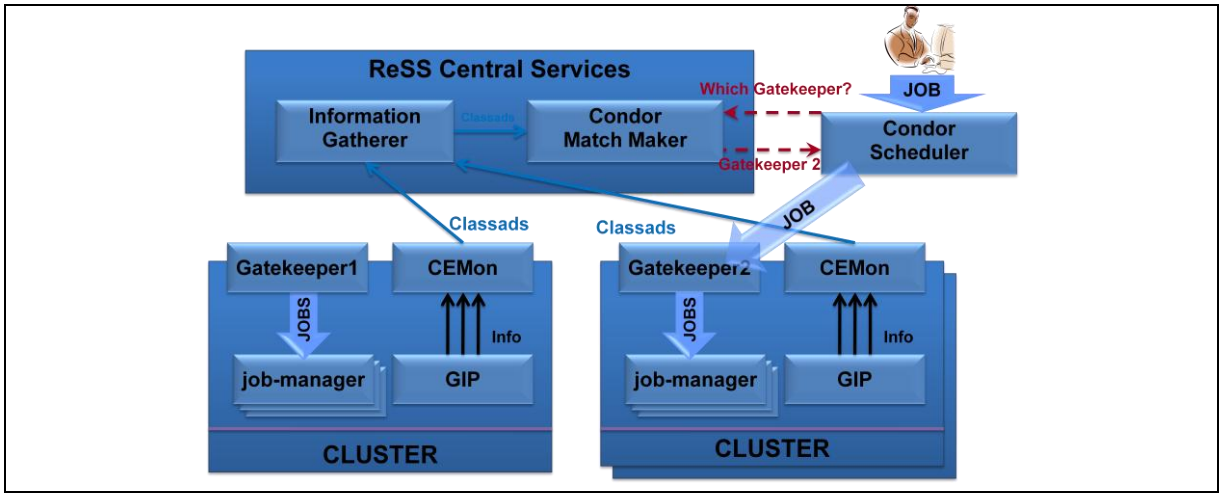

Figure 1: ReSS architectural diagram

The CEMon service, a component of the gLite software stack is capable of gathering and publishing information collected on a computing node. The CEMon "sensors" act as the producer of the information. The sensor developed for OSG invokes and parses the output of the Generic Information Provider (GIP) [13], a suite of scripts that present resource information organized according to the GLUE Schema. This information can be presented in different formats, such as LDIF, XML, new classad and old classad.

The publisher component of CEMon acts as the consumer of the information produced by the sensors. CEMon can publish the information produced by the sensors either synchronously or asynchronously. In ReSS, administrators configure CEMon with a pre-defined list of subscribers, to publish events to IG asynchronously. The OSG sensor uses old classad format [11] representation of the information. Information is routed from CEMon to Condor through IG. IG acts as a stateless interface adapter between the CEMon publisher and Condor collector while the Condor collector acts as the information repository for the classads. IG can be configured to apply simple transformations to the incoming information. This feature is used to add special classad attributes that contain expressions to check the semantics and consistency of the critical attributes in the classad used for the use case of OSG.

Since resource selection is implemented using Condor components, the infrastructure provides for a seamless integration of condor-based job scheduling services, like Condor-G [6]. 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 format 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 the UML diagrams.

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 of one or more groups ("Subclusters") of homogeneous computing nodes ("Hosts"). Hosts are characterized by 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). CE 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.

Recent, the focus of the work has been to advertise the Storage information in OSG via ReSS. A Storage Element (SE) is described in the GLUE Schema by the "StorageElement". Each SE can access multiple "Storage Areas" (SA). SA is described by informational parameters, such as the path to the SA, total size ("TotalOnlineSize"), used size ("UsedOnlineSize"), free size ("FreeOnlineSize"), etc. In addition, the model also allows for VO-specific ("VOInfo") values for these attributes which are specific to a given VO.

Figure 2 shows a schematic UML representation of a site. 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.

For CE, the mapping [7] from the GLUE Schema to a set of old classads is built considering all possible combinations of inter-related CE, cluster, and subcluster entities. In addition, if VO-specific parameters are available to characterize CE (VOView); these are used instead of the general CE attributes. Thus each resulting classad is a virtual homogeneous cluster with one access gateway, described from the point of view of the VO. Thus the number of classads considering the information related to CEs is -

$$N_{CE\text{-}classads} = N_{cluster} \times N_{subcluster} \times N_{CE} \times N_{VO}$$

For OSG resources, $N_{cluster}$ is always 1.

For the SE, the mapping from the GLUE Schema to a set of old classads is built considering all possible combinations of SEs, SAs and VOs supported by each SA. Thus the number of classads considering the information related to SEs is -

$$N_{SE\text{-}classads} = N_{SE} \times N_{SA} \times N_{VO}$$

The VO acts as a common link in binding together the classad information contributed by CEs and SEs. In other words, the information for CE and SE is joined together for a given VO to form a single classad. Thus in the system, total number of classads from a given site is

$$N_{Total\text{-}classads} = N_{CE\text{-}classads} + N_{SE\text{-}classads}$$

Being combinatorial, this algorithm produces a very large number of classads for each site. In OSG, this results in a simple site producing as few as 1 classad while the most complex site produces over 1000 classads.

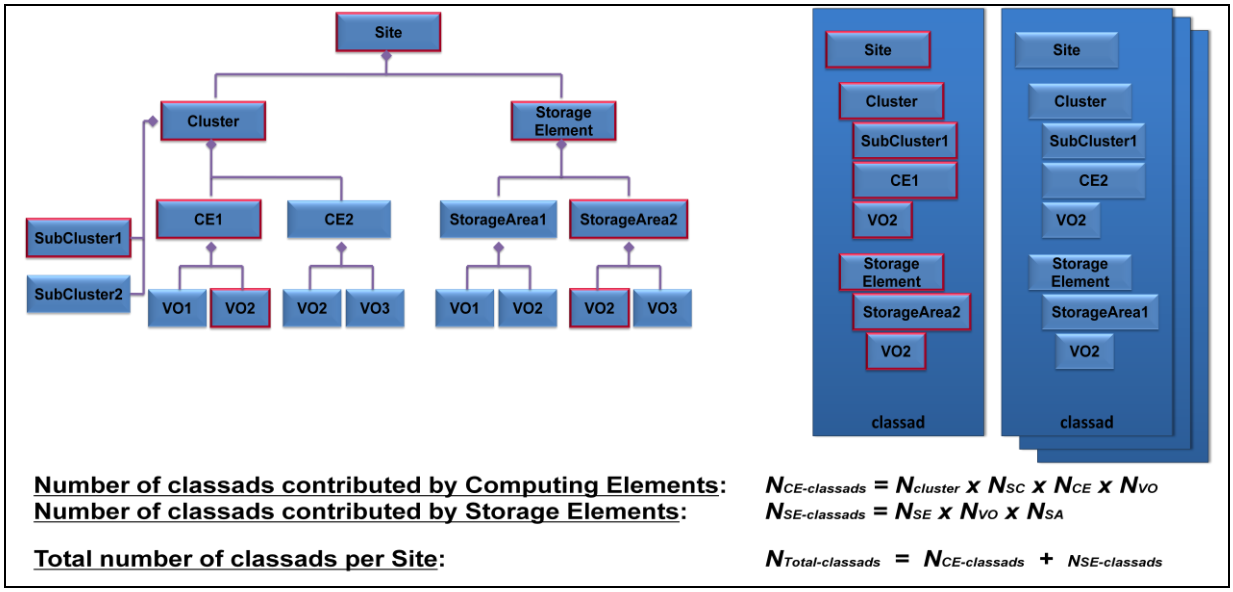

Figure 2: GLUE Schema to old classad mapping

3. ReSS Deployment Models and Use Cases

ReSS has been deployed at two major Grid infrastructures:

1. National Grid Infrastructure: The Open Science Grid
2. Campus Grid Infrastructure: FermiGrid (the Fermilab campus grid)

3.1. ReSS deployment in National Cyber Infrastructure Grid: The Open Science Grid

ReSS central services are deployed for OSG on Xeon 3.2 GHz 2-CPU machines with 4 GB of RAM. These machines run the services at a very low load (<1). CEMon is deployed at about 75 sites producing over 7000 classads and publishing the information to ReSS central services. Details about the validation and evaluation of ReSS before being adopted by the OSG can be found in two independent publications [8, 9].

ReSS information can be accessed directly via querying the repository using constraints to filter the list of resources. For example, the user can query the repository to list all the Grid resources that have cluster nodes with memory more than 2 GB. The user can then apply custom algorithm to this list to narrow down the selection to a single site. Alternatively, a user can query the repository indirectly via the Condor-G system. The user specifies the constraints in a job description language and submits the job to the Condor-G queue. Match-making middleware like condor negotiator can dereference these constraints during the match making cycle and submit the job to a Grid resource that matches the constraints.

In OSG, ReSS is used in a hybrid way. Work load management systems deployed by OSG VOs query the ReSS information repository directly to fetch list of resources that authorize the VO users to run their jobs. Each classad in the list is augmented with VO specific attributes and republished to the VO's information repository and match making system. This setup is used by the Engagement VO. Users of the Engagement VO configure their Condor-G job queues to communicate with the VO's match maker. Thus users access the VO matchmaker indirectly by specifying resource and VO-specific attributes as constraints in the job description language. Figure 3 (left) shows the use of ReSS by Engagement VO

The DZero experiment adopts a direct mechanism to submit jobs to OSG resources as shown in Figure 3 (right). 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. 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 ReSS. 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.

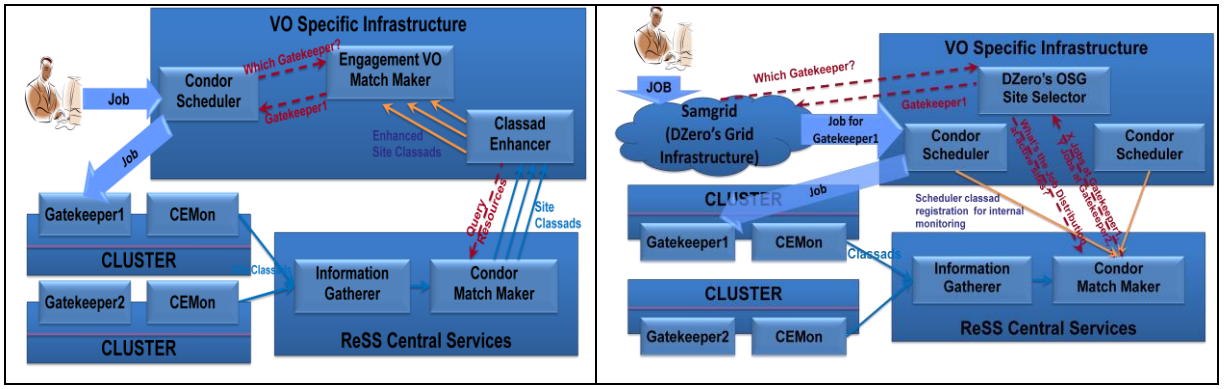

Figure 3: ReSS deployment in OSG for Engagement VO (left) and DZero VO (right)

3.2. ReSS deployment in Campus Grid

Figure 4 shows ReSS deployment in a Campus Grid Infrastructure such as FermiGrid.

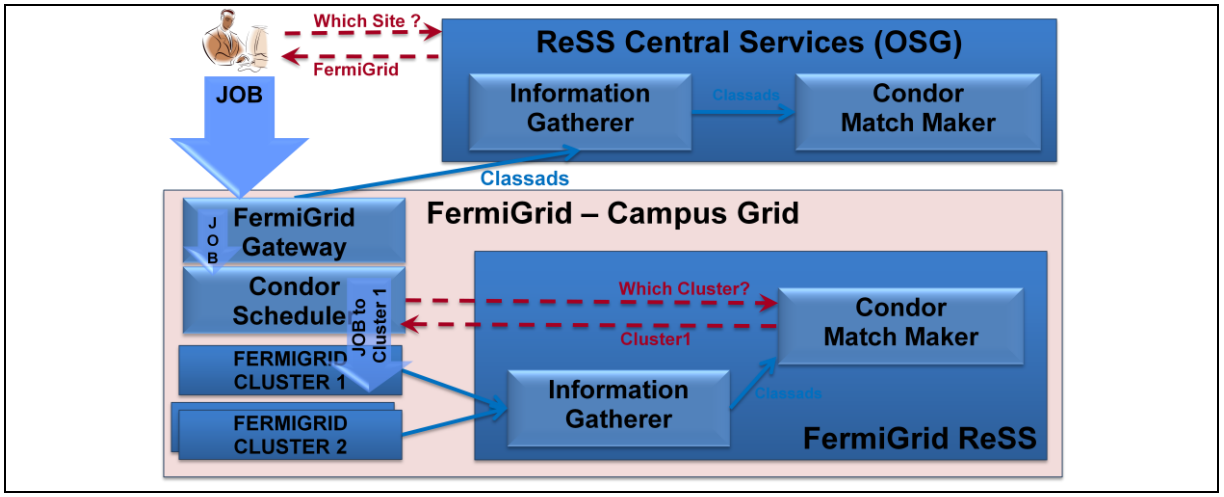

Figure 4: ReSS deployment in the FermiGrid, Campus Grid

FermiGrid is responsible for supporting the Grid infrastructure for the VOs within Fermilab. FermiGrid has 7 computing clusters publishing the information to the FermiGrid specific deployment of ReSS. These clusters also publish information to the OSG ReSS to be used by other VO users on an opportunistic basis. Thus FermiGrid has the most complex deployment model and use case for ReSS of all the other sites. CEMon is deployed on 7 campus clusters, advertising around 3000 classads for a total of more than 17,250 job slots to OSG. IG runs on virtual machine with 3GB of RAM and 4 CPUs. The condor matchmaker runs on a virtual machine with 2 CPUs and 2GB of RAM.

Users within Fermilab can submit jobs to the FermiGrid resources by configuring their Condor-G to point to FermiGrid's ReSS. Users outside Fermilab have access to the FermiGrid resources indirectly by talking to OSG ReSS. This setup enables large organizations with several computing resources to manage and support the user base within the organization as well as make the resources available for opportunistic usage to the users outside the organization for a larger collaboration. Success of the FermiGrid setup has encouraged few other Universities with large number of computing resources to evaluate this deployment model.

4. Support for MPI Resources in ReSS

In previous sections we described how ReSS acts as information repository for the resource classads and how users can submit jobs to the resources using ReSS. However, publishing MPI support over the Grid poses some challenge. Firstly, the information published by sites groups worker nodes that form homogenous part of a subcluster into a single classad. Secondly, there is no direct support in the GLUE Schema for advertising MPI related information. The first challenge can be overcome by pushing the responsibility of MPI job scheduling to the local batch system at the site. In order to get around the shortcoming of the GLUE scheme, ReSS uses the Software entity introduced in the GLUE Schema v1.3 to advertise MPI related information in the classads. Figure 5 shows section of the GLUE Schema with Software entity and its relation to the Cluster.

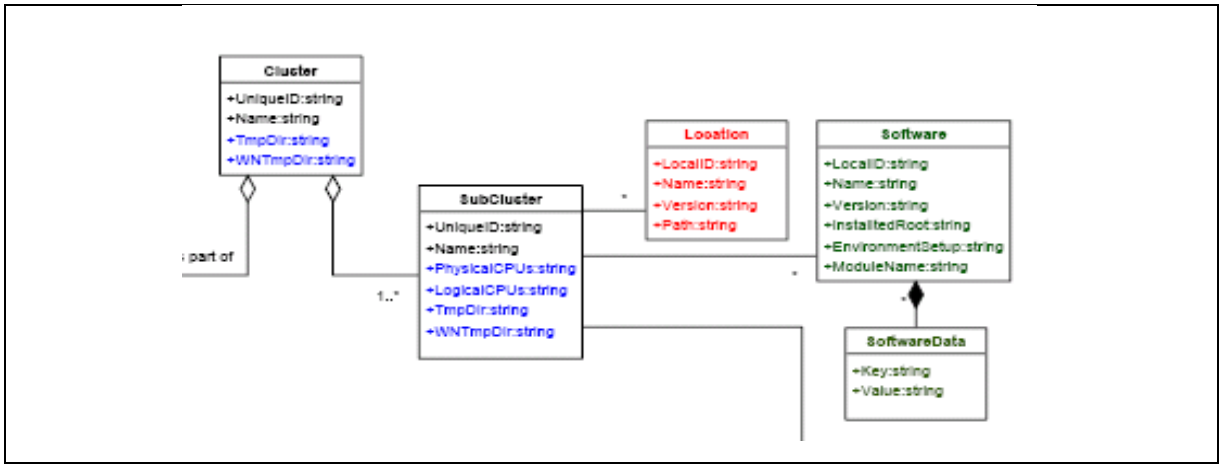

Figure 5: Using Software entity to advertise MPI information

Users are interested in running MPI jobs on worker nodes that have a specific compiler, from a specific vendor, with the compiler module of their interest installed on the worker nodes where the jobs run. Site administrators can configure their GIP to advertise information such as SoftwareName, SoftwareVersion, SoftwareModuleName, etc as shown in figure 6 to describe the information required to match MPI jobs. The Software entity in the GLUE Schema has a direct relation of 0:N with the SubCluster. Thus Software entity becomes an ideal choice for advertising information for MPI. Administrators can also configure the SoftwareData entity with additional information specific to the MPI software installed. This gives user flexibility to filter the sites supporting MPI, as per their requirements in the job description language. A typical requirement string in the job description for the example in figure 6 would look like –

Requirements = (…)&&(GlueSoftwareName==“MPICH2”)&&(GlueSoftwareVersion==“1.0.7_gcc”)

```
dn:GlueSoftwareLocalID=MPICH2_1.0.7_gcc,GlueSubClusterUniqueID=lepton.rcac.purdue.edu,
    GlueClusterUniqueID=lepton.rcac.purdue.edu,mds-vo-name=Purdue-Steele,o=grid
objectClass: GlueClusterTop
objectClass: GlueSoftware
objectClass: GlueKey
objectClass: GlueSchemaVersion
GlueSoftwareLocalID: MPICH2_1.0.7_gcc
GlueSoftwareName: MPICH2
GlueSoftwareVersion: 1.0.7_gcc
GlueSoftwareInstalledRoot: /apps/steele/mpich2-1.0.7/64/ssm-gcc-4.3.0
GlueSoftwareModuleName: mpich2-gcc
GlueChunkKey: GlueSubClusterUniqueID=lepton.rcac.purdue.edu
GlueSchemaVersionMajor: 1
GlueSchemaVersionMinor: 3
```

Figure 6: GIP output to advertise MPI information

5. Conclusion

The Resource Selection Service (ReSS) provides cluster-level resource selection for the Open Science Grid and Fermilab Campus Grid. The system uses the GLUE Schema v1.3 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.

6. Acknowledgments

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 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, University of Chicago, and Mats Rynge, 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 ReSS project; members from Purdue University, in particular Andrew Howard and Preston Smith for their help in testing MPI support. 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.

References

[1] The Open Science Grid home page (accessed on May 13, 2009): http://www.opensciencegrid.org

[2] The Resource Selection home page (accessed on May 13, 2009): 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] Yocum D et al. 2007 “FermiGrid”, FERMILAB-CONF-07-125-CD, TeraGrid '07: Broadening Participation (TeraGrid, Madison, Wisconsin, 4-8 Jun 2007) 5pp.

[5] The GLUE schema home page (accessed on May 13, 2009): http://glueschema.forge.cnaf.infn.it

[6] Frey J, Tannenbaum T, Livny M, Foster I, and Tuecke S 2001 “Condor-G: A Computation

Management Agent for Multi-Institutional Grids": 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 (accessed on May 13, 2009): http://glueschema.forge.cnaf.infn.it/SpecV13/OldClassAd
[8] Arunachalam K, Garzoglio G 2006 "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
[10] Garzoglio G, Levshina T, Mhashilkar P, Timm S 2007 "ReSS: a Resource Selection Service for the Open Science Grid.": Proceedings of the International Symposium of Grid Computing (ISGC07) (Taipei, Taiwan, March 2007)
[11] Condor classads (accessed on May 13, 2009): http://www.cs.wisc.edu/condor/classad
[12] CEMon homepage (accessed on May 13, 2009): http://grid.pd.infn.it/cemon
[13] GIP homepage (accessed on May 13, 2009): https://twiki.grid.iu.edu/bin/view/InformationServices/WebHome