PROCEEDINGS OF SCIENCE

CREAM Computing Element: status and new developments

Paolo ANDREETTO[a], Sara BERTOCCO[a], Alvise DORIGO[a], Eric FRIZZIERO[a], Alessio GIANELLE[a], Massimo MEZZADRI[b], Francesco PRELZ[b] David REBATTO[b], Massimo SGARAVATTO[a*] and Luigi ZANGRANDO[a].

a INFN Padova, Via Marzolo 8, I-35131 Padova, Italy
b INFN Milano, Via Celoria 16, I-20133 Milano, Italy
E-mail: paolo.andreetto@pd.infn.it, sara.bertocco@pd.infn.it, alvise.dorigo@pd.infn.it, eric.frizziero@pd.infn.it, alessio.gianelle@pd.infn.it, massimo.mezzadri@mi.infn.it, francesco.prelz@mi.infn.it, david.rebatto@mi.infn.it, massimo.sgaravatto@pd.infn.it, luigi.zangrando@pd.infn.it

The CREAM (Computing Resource Execution And Management) service, a service for job management operation at the Computing Element (CE) level, is one of the software products part of the EMI middleware distribution. It implements a Grid job management service which allows the submission, management and monitoring of computational jobs to local resource management systems. We present some new functionalities in this service introduced with the first EMI major release, and we discuss about some other new features being implemented.

EGI Community Forum 2012 / EMI Second Technical Conference,
26-30 March, 2012
Munich, Germany

*Speaker.

PoS(EGICF12-EMITC2)084

1. Introduction

The CREAM (Computing Resource Execution And Management) service [1] is the Computing Element (CE) implementation in the gLite Grid middleware stack. It implements a Grid job management service, allowing the submission, management and monitoring of computational jobs to local resource management systems. Users can specify jobs to be submitted via a Job Description Language (JDL) [2], which is a high-level notation based on Condor classified advertisements (classads) [3].

The CREAM business logic has been fully implemented in Java and its functionalities are all exposed by a Web Service interface which guarantees a high degree of interoperability with clients.

In fig. 1 a high level view of the architecture of CREAM is shown.

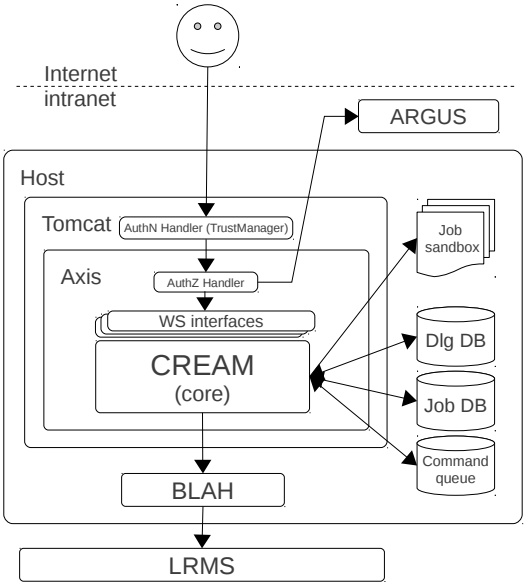

Figure 1: CREAM simplified architecture.

The CREAM service is executed inside the Axis container deployed in the Apache Tomcat application server.

Requests to CREAM traverse a pipeline of additional components which take care of security issues. One of this component is the *Trust Manager*. The Trust Manager is the component responsible for carrying out authentication operations. It is external to CREAM, and is an implementation of the J2EE security specifications. Authentication is based on PKI. The other security related component is the *Authorization handler*. The aim of the authorization process is to check whether an authenticated user has the rights to access services and resources and to perform certain tasks. The decision is taken on the basis of policies that can be either local or decided at the VO level. As described below in this paper, ARGUS can be used in such authorization process.

CREAM uses an external relational database (MySQL) to store its internal state. This improves fault tolerance as it guarantees that this information is preserved across restarts of CREAM. In particular information about jobs (*Job DB* in the figure) and delegated credentials (*Dlg DB* in the picture) are saved. Also the asyncronous commands still to be processed are stored in the database (this is shown in the picture as *Command Queue*).

Job input and output files are instead stored in the local file-system (job sandboxes) and accessible remotely via a gridFTP server.

CREAM submits requests to the Local Resource Management System (LRMS) through BLAH [4], an abstraction layer providing a unified interface to the underlying batch system. CREAM relies on BLAH also for receiving status change notifications from the LRMS.

Users can interact directly with the CREAM service by means of a set of command line utilities which can be used to manage jobs by directly invoking CREAM operations.

CREAM can also be used through higher level job management services. In particular the gLite WMS [5] component and Condor-G [6] both support the submission to CREAM CEs.

CREAM is currently deployed in several production Grids on a national, European and international level, and it is used in a wide variety of application scenarios. At the time of writing this article, in a top BDII of the EGI infrastructure almost 400 CREAM CE head nodes, spread in more than 230 sites, are published.

CREAM is currently being maintained and developed in the context of the European Middleware Initiative (EMI) project [7], which has the goal to bring together the major middleware providers in Europe (in particular ARC, gLite, UNICORE and dCache) to deliver a consolidated set of middleware components for deployment in EGI, PRACE and other Distributed Computing Infrastructures (DCIs).

EMI major releases are delivered once per year. The first one (code name *Kebnekaise*) was released on May 2011. The second major release (*Matterhorn*), under preparation at the time of writing, is scheduled for the end of April 2012, while the *Monte Bianco* release, the third one, will be released at the end of February 2013.

2. Recent developments

Several new features have recently been implemented in the CREAM CE.

In the first EMI release, the following new functionalities have been introduced:

- Integration with the ARGUS authorization service
- Support for gLite-CLUSTER
- Initial support for publication in Glue 2 format
- Support for execution on multicore environments
- Support for automatic output data registration
- Support for Grid Engine local resource management system [8]

In the second major release, being finalized at the time of writing, these other new functionalities will be provided:

- Support for new platforms (namely Scientific Linux 6 and possibly Debian 6)

- Finalization of Glue 2 information publishing
- First support for the EMI Execution Service
- Support for bulk submission in the CREAM User Interface

Some of the just mentioned new capabilities are discussed in the following subsections.

2.1 Integration with the ARGUS authorization service

In the CREAM CE implementation available before the EMI project, authorization was managed through gJAF. gJAF (Grid Java Authorization Framework) provides a way to invoke a chain of policy engines and get a decision result about the authorization of a user.

Local user mapping was instead managed via gLExec [9], a software component that implements the mapping between the Grid world and the Unix notion of users and groups. gLExec in turn makes use of the LCAS (Local Centre Authorization Service) and LCMAPS (Local Credential Mapping Service) software components.

The CREAM CE runs also a gridFTP server, used for different purposes (in particular to support client data push for input data and to allow the publication of the installed software runtime environments). Authorization for this gridFTP server was also implemented through LCAS and LCMAPS, but using different configuration files with respect to the ones used by gLExec.

When multiple authorization systems are used, as in this case, inconsistent authorization decisions can happen (e.g. a security component could grant authorization for a certain Grid user, while another one could deny the operation). The use of multiple components implementing the same or similar functionalities is also a concern in terms of operations of the system.

All these problems have been addressed in the CREAM CE released with EMI-1, where the integration with the ARGUS authorization service [10] was implemented.

ARGUS is a system meant to render consistent authorization decisions for distributed services. ARGUS is the only component used to manage authorization (i.e. to decide if a certain operation issued by a certain user is allowed) and defining the mapping between Grid and local users in the CREAM CE. Also the gridFTP server on the CREAM CE is integrated with ARGUS.

Using a unique authorization system (ARGUS), inconsistent authorization decisions can therefore not happen anymore.

2.2 Support for gLite-CLUSTER

In the first EMI release, support for gLite-CLUSTER was implemented.

glite-CLUSTER is a node type responsible to publish information about the physical resources available in a site, and about the software installed on the Worker Nodes. It can be referenced by any number of compute elements available in the site. Basically the gLite-CLUSTER contains a subset of the functionality incorporated in the CREAM node type: the publication of the GlueCluster and its dependent objects, the publication of the GlueService object for the GridFTP endpoint, and the directories which store the RunTimeEnvironment tags, together with the YAIM (the configuration tool used for the CREAM-CE) functions which configure them.

gLite-CLUSTER in particular eases the deployment in sites having multiple CREAM CE nodes and/or having multiple subclusters (i.e. disjoint sets of worker nodes, each set having sufficiently homogeneous properties).

The deployment of multiple CE headnodes is a quite typical scenario in particular for large sites, for redundancy or load-balancing. Without the use of gLite-CLUSTER, the information about physical resources would be published by all the resource BDIIs of all the CREAM CE nodes. Besides publishing multiple times the same information, this would overcount the installed capacity at the site, unless some manual workarounds are used.

For what concerns the management of multiple subsclusters, without the use of the gLite-CLUSTER the YAIM tool allows configuring a single cluster referring to a single subcluster per CREAM head node. gLite-CLUSTER instead allows the publication of multiple subclusters. Each batch system queue must then be configured to refer to the resources of a single subcluster.

The following deployment models are possible for a CREAM-CE:

- CREAM-CE can be configured without worrying about the glite-CLUSTER node. This can be useful for small sites having a very simple setup and which therefore don't want to worry about cluster/subcluster configurations. This is called *no cluster mode*.
- CREAM-CE can work on *cluster mode* using the glite-CLUSTER node type. The CREAM-CE can be in the same host or in a different host with respect to the glite-CLUSTER node.

2.3 Glue 2 support

The data published in the Information Service conforms to the GLUE (Grid Laboratory for a Uniform Environment) schema [11], which defines a common data model used for resources monitoring and discovery.

To address some shortcomings and problems found in the first version of the Glue specification, which has been used in production for many years, a new version (2.0) of the Glue specification was defined [12].

In the first EMI release a preliminary support for publishing in Glue 2.0 format was introduced. This will be completed in the EMI-2 release that at the time of writing is being finalized.

Both static and batch system dynamic information is provided in Glue 2.0 format. The former is provided by LDIF files that are created at configuration time. The latter is instead provided by dynamic GIP (Grid Information Provider) plugins, scripts that gather and publish fresh information, in particular for the batch system.

This Glue 2.0 publishing concerns both the CREAM service (supporting both cluster and no cluster modes) and the gLite-CLUSTER node. If a CREAM CE is configured in no cluster mode, all the Glue2 object classes are published by the resource BDII running on the CREAM CE. These objectclasses are the ComputingService, the ComputingEndPoint with the relevant AccessPolicies, the ComputingManager, the ComputingShares (which correspond to GlueVoViews in Glue 1.3) along with the MappingPolicies, the ExecutionEnvironments (one per subscluster), the Benchmarks, the ToStorageServices, the ApplicationEnvironments (one per installed software), the EndPoint for the ApplicationPublisher, the EndPoint for the CEMon service (published only if CEMon is deployed).

If a CREAM CE is instead configured in cluster mode, the resource BDII running on the CREAM CE publishes just the ComputingEndpoint objectclass (with the relevant AccessPolicies) and the EndPoint objectclass for CEMon (if CEMon is deployed). All the other objectclasses are published by the resource BDII running on the gLite-CLUSTER node.

More information about the Glue 2.0 suppport in the CREAM CE is available at [13]. It must be noted that the support for Glue 2.0 didn't stop the support for the first version of Glue: information is still published also in Glue 1.3 format.

2.4 EMI Execution Service

One of the objectives of the EMI project is to find an agreement in the consortium on which interfaces and protocols to use in order to enable computational job submission and management across technologies.

This goal was achieved by defining the specification of an EMI Execution Service [14] (or "EMI-ES" for short) targeted to Computing Elements (higher level services, such as workload managers, brokering services, or workflow systems, are instead out of scope). This specification covers the interfaces to create and manage activities (i.e. jobs), and the *Activity Description Language* (EMI-ADL), i.e. an XML dialect for describing the activity characteristics and its resource requirements. Data staging capabilities, activity and resource related information, delegation (needed to implement data staging) are also covered by this specification.

In the CREAM-CE that will be released in the EMI-2 version, a first support for EMI-ES will be provided: besides installing CREAM with the legacy interface, also the EMI-ES interface will therefore be optionally deployed.

Using the EMI-ES interface the following functionalities will be provided:

- job submission of single or multiple activities;
- job management operations: cancel, suspend, resume, list, wipe of activities;
- status and info activity related operations;
- proxy delegation operations: creation, renewal, and information about delegations;
- support for client data push (input data are staged from the client machine) and server data pull (input data are retrieved from one or more storage servers);
- support for client data pull (output data, when ready, are retrieved by the client) and server data push (output data are automatically stored in one or more storage servers);
- support for multiple sources for input files, and multiple targets for output files.

A EMI-ES Command Line Interface (CLI), installed in the User Interface, will be also released in the EMI second major release: this will allow the end user to use the functionality provided by any EMI-ES Computing Element (not only the CREAM EMI-ES implementation).

The ResourceInfo Port-type, used to get information about the CREAM CE resource in Glue 2.0 format, won't be implemented in the first EMI-2 release. It will instead be released with a

subsequent EMI-2 update. However information about the Computing Element in Glue 2 format is already available through the Resource BDII of the CREAM CE.

3. Future work

Among the foreseen new developments, the following ones are scheduled to be finalized by the end of the EMI project:

- Finalization of EMI Execution Service support.
- Provision of High Availability capabilities. This is discussed in the following subsection.
- Integration of the SLURM batch system [15].
- Support for job streams, that is the the possibility to constantly keep N jobs queued until a given condition is satisfied (e.g. no work to be done). This was requested by the WLCG Workload Management Technical Evaluation Group (TEG) [16] to make job pilot execution more efficient.

3.1 CREAM High Availability

One of the new functionality foreseen to be implemented after the second EMI major release in the CREAM CE is the ability to be continuously available for serving user requests even in case of planned and unplanned outages.

Like several popular Internet services, the idea is to achieve this goal relying on a cluster of commodity computers seen as a single service. This will allow not only to implement the High Availability (HA) goal, but also to improve the overall scalability and performance. The whole system complexity must of course be hidden to the user, who is not interested in distinguishing a single CREAM service from the clustered one.

In the context of CREAM clustering, we define a *CREAM node* as a separate CREAM instance running on its dedicated (virtual) machine, while a collection of such nodes is referred as *CREAM cluster*. In increased load conditions, the CREAM nodes must share the work load, while in the event of a (un)scheduled downtime, the involved nodes must be dropped from the cluster without shutting down the overall system; these nodes could also be replaced by backup nodes with a hot deploy.

In fig. 2 the foreseen high level CREAM clustered architecture is shown. It can be compared with the no-cluster architecture shown in fig. 1.

In this picture the logical centralization of information and all software components composing the cluster is highlighted, while for the sake of simplicity, replications are explicitly omitted.

The WEB server (e.g. apache) acts as gateway for incoming requests of authenticated users. These requests are delivered to the *load balancer* which redirects them to the proper CREAM nodes, basing its decisions on the selected scheduling algorithm (e.g. Round Robin, Weight based, etc). Moreover the load balancer can even provide fault tolerance capability, if appropriately configured. In turn the CREAM node applies the authorization rules and, if allowed, processes the requests exactly as in the no-clustered architecture.

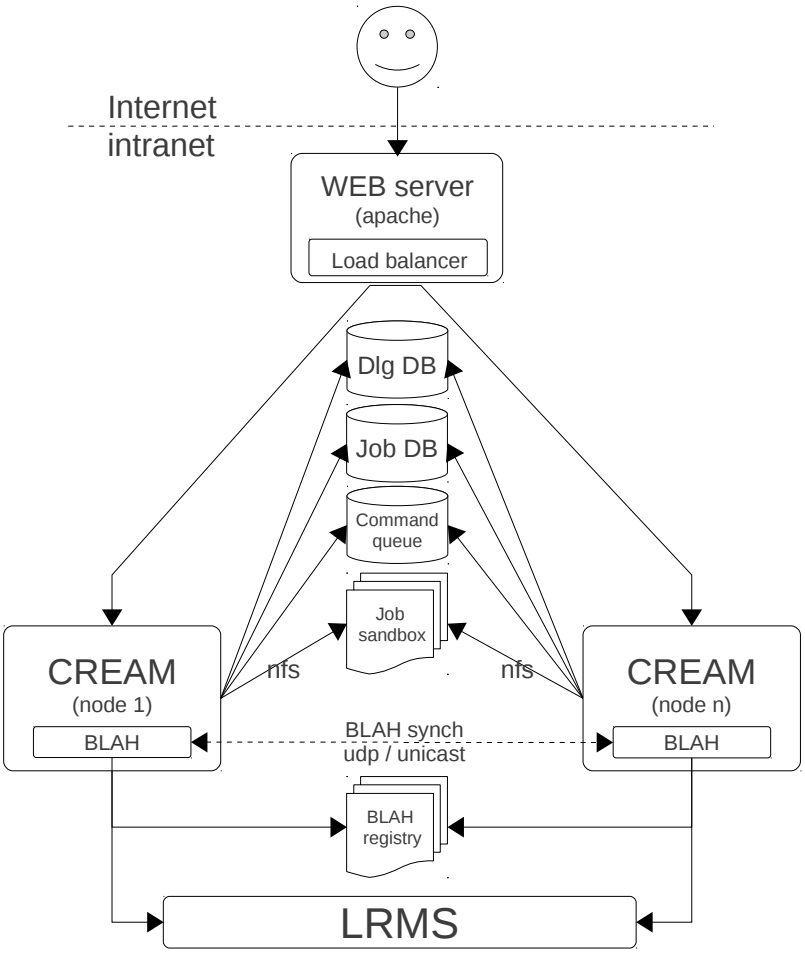

Figure 2: CREAM clustered architecture.

Requests for job management will still be stored in the persistent command queue which, in contrast to the no-clustered architecture, will be logically centralized and accessed concurrently by all the CREAM nodes. So, every CREAM instance can process requests independently of whoever was queuing them. Given that by design CREAM is a stateless Web Service which treats each request as independent, no explicit session replication is required.

Static and dynamic job information, including delegation proxies will be stored in the (logically) centralized database (MySQL) while the job sandboxes, which contain input and output data files accessed and/or produced during the job's life cycles, will be stored in a shared file-system (e.g. NFS).

Since the database is a potential Single Point Of Failure (SPOF), it must itself be clustered.

4. Conclusions

In this paper we described the status and new developments for the CREAM service, a Java-based Grid CE service, which can be used directly by the end-user or can be integrated in higher level job management tools.

More detailed information can be found on the CREAM web page [17].

5. Acknowledgments

The research results of the EMI project are co-funded by the European Commission under the FP7 Collaborative Projects Grant Agreement Nr. 261611.

References

[1] C. Aiftimiei et al, *Design and Implementation of the gLite CREAM Job Management Service*, Future Generation Computer Systems, Volume 26, Issue 4, April 2010, pp. 654-667, doi: 10.1016/j.future.2009.12.006.

[2] *CREAM Job Description Language Attributes Specification*, available online at `https://wiki.italiangrid.it/twiki/bin/view/CREAM/JdlGuide`.

[3] R. Raman, *Matchmaking Frameworks for Distributed Resource Management*, Ph.D. thesis, University of Wisconsin-Madison, 2001.

[4] E. Molinari et al, *A local batch system abstraction layer for global use*, Proc. of XV International Conference on Computing in High Energy and Nuclear Physics (CHEP'06), Feb 13–17 2006, Mumbay, India.

[5] P. Andreetto et al., *Evolution of Grid meta-scheduling towards the EMI era*, in Proc. of CHEP'10, 18th International Conference on Computing in High Energy and Nuclear Physics, 18-22 October 2010, Taipei (Taiwan)

[6] J. Frey, T. Tannenbaum, M. Livny, I. Foster, S. Tuecke, *Condor-G: A Computation Management Agent for Mulled-Institutional Grids* Proc. of of HPDC 2001.

[7] European Middleware Initiative (EMI) project home page, `http://www.eu-emi.eu/home`

[8] Grid Engine Community home page, `http://gridengine.org`

[9] D. Groep, O. Koeroo and G Venekamp, *gLExec: gluing grid computing to the Unix world*, Journal of Physics: Conference Series, volume 119, number 6, year 2008

[10] ARGUS home page, `https://twiki.cern.ch/twiki/bin/view/EGEE/AuthorizationFramework`

[11] OGF GLUE Working Group, `http://forge.ogf.org/sf/sfmain/do/viewProject/projects.glue-wg`

[12] S. Andreozzi, S. Burke, F. Ehm, L. Field, G. Galang, B. Konya, M. Litmaath, P. Millar, JP Navarro, *GLUE Specification v. 2.0*, available online at `www.ogf.org/documents/GFD.147.pdf`

[13] Glue 2.0 support in the CREAM CE, `https://wiki.italiangrid.it/twiki/bin/view/CREAM/CreamGlue2`

[14] B. Schuller (editor), B. Konya, O. Smirnova, A. Konstantinov, M. Skou Andersen, M. Riedel, S. Memon, S. Memon, L. Zangrando, M. Sgaravatto, E. Frizziero, *EMI Execution Service Specification*, available online at `https://twiki.cern.ch/twiki/bin/view/EMI/EmiExecutionService`

[15] SLURM: A Highly Scalable Resource Manager, `https://computing.llnl.gov/linux/slurm/slurm.html`

[16] WLCG Workload Management Technical Evolution Group, https://twiki.cern.ch/twiki/bin/view/LCG/WorkloadManagementTechnicalEvolution

[17] CREAM home page, https://wiki.italiangrid.it/CREAM