Status and Developments of the CREAM Computing Element Service

P Andreetto^1, S Bertocco^1, F Capannini^2, M Cecchi^2, A Dorigo^1, E Frizziero^1, A Gianelle^1, F Giacomini^2, M Mezzadri^3, S Monforte^4, F Prelz^3, E Molinari^3, D Rebatto^3, M Sgaravatto^1, L Zangrando^1

^{1} INFN Padova, Via Marzolo 8, I-35131 Padova, Italy
^{2} INFN-CNAF, Viale Berti Pichat 6/2, I-40127 Bologna Italy
^{3} INFN Milano, Via Celoria 16, I-20133 Milano, Italy
^{4} INFN Catania, Via Santa Sofia 64, I-95123, Italy

Abstract. The CREAM CE implements a Grid job management service available to end users and to other higher level Grid job submission services. It allows the submission, management and monitoring of computational jobs to local resource management systems. CREAM, which is part of the gLite Grid middleware, is available in the EGEE production Grid where it is used by several user communities in different job submission scenarios. In this paper, after a quick description of the CREAM CE architecture and functionality, we report on the status of this Grid service, focusing on the results, feedback and issues that had to be addressed. We also discuss about its integration with other job submission services, in particular the gLite Workload Management System. The planned future activities, concerning the maintenance and evolution of the CREAM CE, are reported as well.

1. Introduction

In Grid terminology, a Computing Element (CE) is the interface to a large farm of computing hosts managed by a Local Resource Management System (LRMS), such as Condor, LSF or Torque. A CE provides additional features to those of the underlying batch system, such as Grid-enabled user authentication and authorization, accounting, fault tolerance and improved performance and reliability.

The Computing Resource Execution and Management (CREAM) service is the CE implementation in the gLite [7] Grid middleware. CREAM is the job management component that can be used to submit, cancel, and monitor jobs for execution on suitable computational resources. It exposes an interface based on Web Services, which enables a high degree of interoperability with clients written in different programming languages.

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, such as astrophysics, biomedicine, computational chemistry, earth sciences, high energy physics, finance, fusion, geophysics, multimedia, etc.

In this paper we first of all briefly describe the functionality of CREAM. We then discuss how it is integrated with higher level job management components. After having summarized some recent new developments, we then report about the usage of CREAM by the Worldwide LHC

Computing Grid (WLCG) user community. Finally we report about some new functionality and developments foreseen for the future.

2. The CREAM service

CREAM is a job submission service: users can submit jobs, described via a classad-based Job Description Language (JDL) expression [12], to CREAM CEs. The JDL is a high-level notation based on Condor classified advertisements (classads) [10] for describing jobs and their requirements. CREAM supports the execution of batch (normal) and parallel (MPI) jobs. Normal jobs are single or multi-threaded applications requiring one CPU to be executed; MPI jobs are parallel applications which usually require a larger number of CPUs to be executed, and which make use of the MPI library for interprocess communication. CREAM is a Java application which runs as an Axis servlet inside the Tomcat application server.

The CREAM interface exposes a set of operations which can be classified in three groups (see [1] for details). The first group of operations (*Lease Management*) allows the user to define and manage leases associated with jobs. The lease mechanism has been implemented to ensure that all jobs get eventually managed, even if the CREAM service loses connection with the client application due to network partitioning. Each lease defines a time interval, and can be associated with a set of jobs. A lease can be renewed before its expiration; if a lease expires, all jobs associated with it are terminated and purged by CREAM.

The second group of operations (*Job Management*) is related with the core functionality of CREAM as a job management service. Operations are provided to create a new job, start execution of a job, suspend/resume or terminate a job. Moreover, the user can get the list of all owned jobs, and it is also possible to get the status of a set of jobs.

Finally, the third group of operations (*Service Management*) deals with the whole CREAM service. It consists of two operations, one for enabling/disabling new job submissions, and one for accessing general information about the service itself. Note that only users with administration privileges are allowed to enable/disable job submissions.

CREAM submits requests to the LRMS through Batch-system Local ASCII Helper (BLAH) [9], an abstraction layer providing a unified interface to the underlying LRMS. CREAM supports all the batch systems supported by BLAH which at the time of writing are LSF, PBS/Torque, Condor, SGE and BQS. CREAM relies on BLAH also for receiving status change notifications from the LRMS.

3. Integrating CREAM with other job management services

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. These command line tools are written in C++ using the gSOAP library [13].

CREAM can also be used through higher level job management services, such as the gLite Workload Management System (WMS) component [3] and Condor-G [8].

The WMS comprises a set of Grid middleware components responsible for the distribution and management of tasks across Grid resources, in such a way that applications are conveniently, efficiently and effectively executed.

The CREAM–WMS integration is realized with a specific module, called Interface to Cream Environment (ICE). ICE receives job submissions and other job management requests from the Workload Manager (WM) component of the WMS; it then uses the appropriate CREAM methods to perform the requested operations. Moreover, ICE is responsible for monitoring the state of submitted jobs and for taking the appropriate actions when the relevant job status changes are detected (e.g. the trigger of a possible resubmission if a Grid failure is detected).

Fig. 1 shows a schematic view of CREAM integration with the kite WMS. Users can submit jobs through the gLite User Interface (UI), which transfers jobs to a WMS. Then the ICE component of the WMS interacts with CREAM via the legacy Web Service interface. Of course, users may also interact with CREAM directly (i.e. bypassing the gLite WMS), by using the CREAM command line tools.

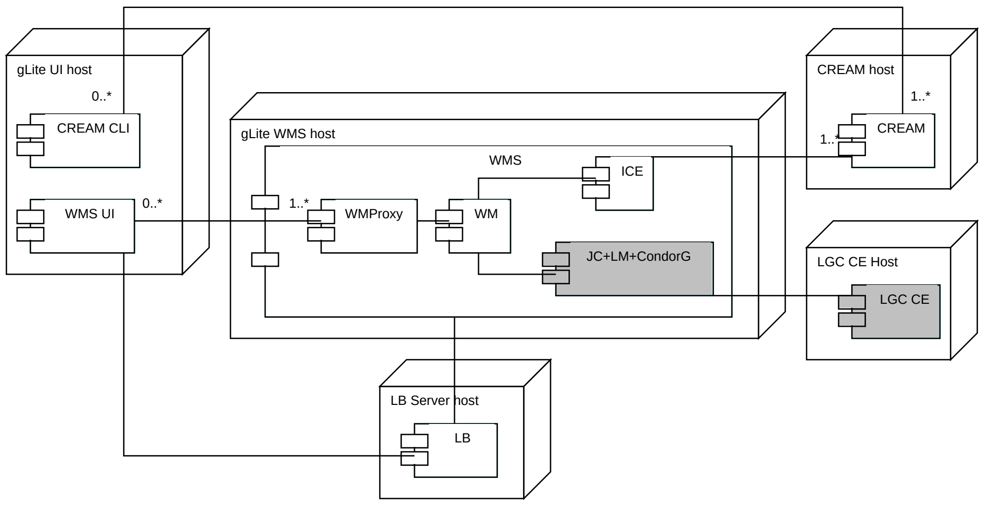

Figure 1. Job submission chain (simplified) in the gLite middleware. The JobController+CondorG+LogMonitor (JC+CondorG+LM) component is used for job submission to the legacy LCG-CE.

Condor-G is another higher level job management component. It can manage thousands of jobs destined to run at distributed sites. It provides job monitoring, logging, notification, policy enforcement, fault tolerance, credential management, and it can handle complex job-interdependencies.

Condor-G added functional support for CREAM in version 7.3.2, but important fixes and enhancements were introduced with version 7.5.3.

4. Recent developments

At time of writing, the latest available version of the CREAM CE is 1.6.3. Version 1.6 introduced several enhancements.

The first one is an improvement of job status changes detection by the ICE component of the WMS. In the previous versions, ICE obtained the state of a job in two different ways. The first one was by subscribing to a job status change notification service implemented by CEMonitor [2] (CREAM notified CEMonitor component about job state changes by using the shared, persistent CREAM backend; ICE subscribed to CEMonitor notifications, so it received all status changes whenever they occur). As a fallback mechanism, ICE could explicitly query the CREAM service to check the status of "active" jobs for which it did not receive any notification for a configurable period of time.

This mechanism proved to be not scalable: in scenarios of a WMS submitting many jobs to several CREAM CEs, big delays in detecting job status changes were observed. This problem was addressed introducing a new operation, called QueryEvent which basically just return status changes for jobs submitted by a certain user. This operation is now used by the ICE component: instead of using the CEMon notifications and/or explicitly polls to get the status for all "active" jobs, ICE simply queries the relevant CREAM CEs (the ones where it submitted jobs) to get just the job status changes, which is the only needed information.

Another improvement introduced with CREAM CE 1.6 concerns credential mapping. In previous releases, all operations to be done by the local account mapped to the relevant Grid user (job sandbox directory creation, submission to the batch system, etc.) were managed by glexec [6]. Each invocation of glexec results in an authorization decisions (a full LCAS/LCMAPS evaluation) which is not really needed and can contribute in degrading performance. After having discussed the issue in the context of the Middleware Security Group (MWSG) and Security Coordination Group (SCG) [11], it was decided to use *s*udo for such operations to be done on behalf of the local user, using the local id returned by glexec (which in this model is therefore used just once per job submission). This was implemented in CREAM v. 1.6

An other important change introduced in CREAM CE v. 1.6 concerns the BLAH component. In the previous implementations, BLAH parsed the LRMS log files to detect status changes. In the new implementation it is also possible to configure BLAH to poll the LRMS (therefore using the status/history batch system commands) instead of parsing the LRMS log files. Both models are currently supported (the site administrator can choose the preferred model at configuration time).

5. Deployment and usage of CREAM in the Worldwide LHC Computing Grid

The first version of the CREAM computing service was released in production in the WLCG production Grid on October 2008. At the time of writing there are about 160 instances of CREAM in such Grid infrastructure, and the number of installations keeps increasing.

The rest of this section summarizes the experience with CREAM by the LHC experiments.

ALICE was the first LHC experiment trying and then using CREAM for production activities (the first tests were done even before the first official release of CREAM).

CREAM is now deployed in all Alice T1 and T2 sites, where it is used to run job agents. These agents, which are submitted directly to CREAM resources using the CREAM CLI (i.e. not relying on higher level tools such as the WMS or CondorG) are then responsible to retrieve from a central queue and eventually run analysis/reconstruction/simulation jobs suitable for the considered site.

Alice is still using LCG-CEs only at Cern, just because at Cern currently only 3 CREAM CEs installations are available vs about 20 LCG-CE instances.

Alice is reporting very good results with CREAM for what concerns deployment and operations.

The ATLAS computing model is also based on pilot jobs. In most of ATLAS sites, pilot jobs are submitted using CondorG.

The submission of pilot jobs to CREAM resources from CondorG was tested by ATLAS at the beginning of 2010. In these tests two major problems (one related to lease renewal and one related to the management of the sandbox files) were observed. In July 2010 a new version of Condor (v. 7.5.3) addressing these problems was released. Since that ATLAS is currently performing new tests: results so far look promising.

CMS software frameworks for production and analysis can refer to the gLite WMS or glideinWMS [14] for what concerns job submission.

In October and November 2009, CMS performed a test campaign in its T1 sites to test submission to CREAM through the WMS for productions. The outcome of these tests was that the inefficiencies due to CREAM were negligible. CMS therefore decided to use also CREAM

CEs for their activities. A bug in the ICE component of the WMS was then found: the problem was that in some cases jobs could be reported in a non final state even if finished. This problem was then fixed with WMS 3.2.14. Since that, no other particular issues concerning CREAM were reported by the CMS community.

For what concerns the submission to CREAM using glideinWMS (which refers to the submission to CREAM from Condor) this is still to be finalized. Some tests were already done, but referring to an old Condor implementation, affected by the same problems observed in the tests done by Atlas.

LHCB is also referring to a pilot based approach. In this case, however, pilots are submitted using the WMS. Since January 2010, about 40 % of pilots submitted by LHCB run on CREAM based resources.

LHCB is going to move towards direct submission model, submitting the jobs directly to the CREAM CE using the CREAM CLI (i.e. without using the WMS). A proof of concept Dirac agent referring to this model is already in place at INFN CNAF.

6. Future work

One of the new functionality foreseen for the CREAM service is the integration with the Argus authorization service [4]. In such scenario, Argus will be the only component used to manage authorization and defining the mapping between Grid and local users in the CREAM CE. In this way inconsistent authorization decisions could not happen anymore. Moreover this will help reducing the dependency footprint in the CREAM CE since glexec (which in turns depends on LCAS and LCMAPS) won't be needed anymore.

Another foreseen new development is the integration between CREAM and the Logging and Bookkeeping (LB) service [15]. In this way gLite users will be able to track jobs submitted to CREAM from any client (through the gLite WMS, directly using the CREAM CLI, etc.). This will also allow the integration between CREAM and higher level monitoring tools (e.g. dashboard systems), already integrated with LB.

7. 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, such as the gLite WMS and Condor-G.

More detailed information, including installation instructions, interface specification and usage manuals for CREAM can be found on the Web page [5].

8. 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] C. Aiftimiei et al, *Using CREAM and CEMON for job submission and management in the gLite middleware*, Proc. of CHEP'09 conference, Prague, Czech Republic, March 21 - 27 2009.

[3] P. Andreetto et al, *Practical approaches to grid workload and resource management in the EGEE project*, Proc. of CHEP'04 conference, Interlaken, Switzerland, Sept 27 - Oct 1 2004.

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

[5] CREAM home page, `http://grid.pd.infn.it/cream`

[6] GLExec home page, `https://www.nikhef.nl/pub/projects/grid/gridwiki/index.php/GLExec`

[7] gLite home page, `http://www.glite.org`

[8] 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.

[9] 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.

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

[11] Security Coordination Group home page, `http://egee-scg.web.cern.ch/egee-scg`

[12] M. Sgaravatto *CREAM Job Description Language Attributes Specification for the EGEE Middleware* Document Identifier *EGEE-JRA1-TEC-592336*, Available online at `https://edms.cern.ch/document/592336`.

[13] R. van Engelen, *gSOAP 2.7.6 User Guide'*, 29 Dec. 2005.

[14] I. Sfiligoi, D. C. Bradley, B. Holtzman, P. Mhashilkar, S. Padhi and F. Wuerthwein, *T*he Pilot Way to Grid Resources Using glideinWMS, Proceedings of Computer Science and Information Engineering, 2009 WRI World Congress on, pp. 428-432, March 2009, ISBN: 978-0-7695-3507-4, http://dx.doi.org/10.1109/CSIE.2009.950

[15] The Logging and Bookkeeping Subsystem, `http://egee.cesnet.cz/cs/JRA1/LB`