EGEE User Forum

Contribution ID: **87** Type: **Oral contribution**

The gLite Workload Management System

Thursday 2 March 2006 14:30 (30 minutes)

The Workload Management System (WMS) is a collection of components providing a service responsible for the distribution and management of tasks across resources available on a Grid, in such a way that applications are conveniently, efficiently and effectively executed.

The main purpose of the WMS as a whole is then to accept a request of execution of a job from a client, find appropriate resources to satisfy it and follow it until completion, possibly rescheduling it, totally or in part, if an infrastructure failure occurs. A job is always associated to the credentials of the user who submitted it. All the operations performed by the WMS in order to complete the job are done on behalf of the owning user. A mechanism exists to renew credentials automatically and safely for long-running jobs.

The different aspects of job management are accomplished by different WMS components, usually implemented as different processes communicating via data structures persistently stored on disk to avoid as much as possible data losses in case of failure.

Recent releases of the WMS come with a Web Service interface that has replaced the custom interface previously adopted. Moving to formal or de-facto standards will continue in the future.

In order to track a job during its lifetime, relevant events (such as submission, resource matching, running, completion) are gathered from various WMS components as well as from Grid resources (typically Computing Elements), which are properly instrumented. Events are kept persistently by the Logging and Bookkeeping Service (LB) and indexed by a unique, URL-like job identifier. The LB offers also a query interface both for the logged raw events and for higher-level task state. Multiple LBs may exist, but a job is statically assigned to one of them. Being the LB designed, implemented and deployed so that the service is highly reliable and available, the WMS heavily relies on it as the authoritative source for job information.

The types of job currently supported by the WMS are diverse: batch-like, simple workflow in the form of Directed Acyclic Graphs (DAGs), collection, parametric, interactive, MPI, partitionable, checkpointable. The characteristics of a job are expressed using a flexible language called Job Description Language (JDL). The JDL also allows the specification of constraints and preferences on the resources that can be used to execute the job. Moreover some attributes exist that are useful for the management of the job itself, for example how much to insist with a job in case of repeated failures or lack of resources.

Of the above job types, the parametric jobs, the collections, and the workflows have recently received special attention.

A parametric job allows the submission of a large number of almost identical jobs simply specifying a parameterized description and the list of values for the parameter.

A collection allows the submission of a number of jobs as a single entity. An interesting feature in this case is the possibility to specify a shared input sandbox. The input sandbox is a group of files that the user wishes to be available on the computer where the job runs. Sharing a sandbox allows some significant optimization in network traffic and, for example, can greatly reduce the submission time.

Support for workflows in the gLite WMS is currently limited to Directed Acyclic Graphs (DAGs), consisting of a set of jobs and a set of dependencies between them. Dependencies represent time constraints: a child cannot start before all parents have successfully completed. In general jobs are independently scheduled and the choice of the computing resource where to execute a job is done as late as possible. A recently added feature allows to collocate the jobs on the same resource. Future improvements will mainly concern error handling and integration with data management.

Parametric jobs, collections and workflows have their own job identifier, so that all the jobs belonging to them can be controlled either independently or as a single entity.

Future developments of the WMS will follow three main lines: stronger integration with other services, software cleanup, and scalability.

The WMS already interacts with many external services, such as Logging and Bookkeeping, Computing Elements, Storage Elements, Service Discovery, Information System, Replica Catalog, Virtual Organization Membership Service (VOMS). Integration with a policy engine (G-PBox) and an accounting system (DGAS) is progressing; this will ease the enforcement of local and global policies regulating the execution of tasks over the Grid, giving fine control on how the available resources can be used. Designing and implementing a WMS that relies on external services for the above functionality is certainly more difficult than providing a monolithic system, but in fact doing so favors a generic solution that is not application specific and can be deployed in a variety of environments.

The cleanup will affect not only the existing code base, but will also aim at improving the software usability and at simplifying service deployment and management. This effort will require the evaluation and possibly the re-organization of the current components, yet keeping the interface.

Last but not least, considerable effort needs to be spent on the scalability of the service. The functionality currently offered already allows many kinds of applications to port their computing model onto the Grid. But additionally some of those applications have demanding requirements on the amount of resources, such as computing, storage, network, and data, they need to access in order to accomplish their goal. The WMS is already designed and implemented to operate in an environment with multiple running instances not communicating with each other and seeing the same resources. This certainly helps in case the available WMSs get overloaded: it is almost as simple as starting another instance. Unfortunately this approach cannot be extended much further because it would cause too much contention on the available resources. Hence the short term objective is to make a single WMS instance able to manage 100000 jobs per day. In the longer term it will be possible to deploy a cluster of instances sharing the same state.

Primary author: GIACOMINI, Francesco (Istituto Nazionale di Fisica Nucleare (INFN))

Presenter: GIACOMINI, Francesco (Istituto Nazionale di Fisica Nucleare (INFN))

Session Classification: 2a: Workload management and Workflows

Track Classification: Workload management and Workflows