PRACTICAL APPROACHES TO GRID WORKLOAD AND RESOURCE MANAGEMENT IN THE EGEE PROJECT

P. Andreetto, S. Borgia, A. Dorigo, A. Gianelle, M. Mordacchini, M. Sgaravatto, L. Zangrando, INFN Sezione di Padova, Via Marzolo 8, I-35131 Padova, Italy
S. Andreozzi, V. Ciaschini, C. Di Giusto, F. Giacomini, V. Medici, E. Ronchieri, V. Venturi, INFN CNAF, Viale Berti Pichat 6/2 , I-40127 Bologna, Italy
G. Avellino, S. Beco, A. Maraschini, F. Pacini, DATAMAT S.p.A., Via Laurentina 760, I-00143 Roma, Italy
A. Guarise, G. Patania, INFN Sezione di Torino, Via P. Giuria 1, I-10125 Torino, Italy
D. Kouřil, A. Křenek, L. Matyska, M. Mulač, J. Pospíšil, M. Ruda. Z. Salvet, J. Sitera, J. Škrabal, M. Voců, CESNET z.s.p.o., Zikova 4, 160 00 Praha 6, Czech Republic
V. Martelli, M. Mezzadri, F. Prelz, D. Rebatto, INFN Sezione di Milano, Via Celoria, 16, I-20133 Milano, Italy
S. Monforte, M. Pappalardo, INFN Sezione di Catania, Via S. Sofia 64, I-95123 Catania, Italy

Abstract

Resource management and scheduling of distributed, data-driven applications in a Grid environment are challenging problems. Although significant results were achieved in the past few years, the development and the proper deployment of generic, reliable, standard components present issues that still need to be completely solved. Interested domains include workload management, resource discovery, resource matchmaking and brokering, accounting, authorization policies, resource access, reliability and dependability. The evolution towards a service-oriented architecture, supported by emerging standards, is another activity that will demand attention. All these issues are being tackled within the EU-funded EGEE project (Enabling Grids for E-science in Europe), whose primary goals are the provision of robust middleware components and the creation of a reliable and dependable Grid infrastructure to support e-Science applications. In this paper we present the plans and the preliminary activities aiming at providing adequate workload and resource management components, suitable to be deployed in a production-quality Grid.

INTRODUCTION

The aim of the EU funded EGEE (Enabling Grids for E-science in Europe) project [1] is to build on recent advances in Grid technology and develop a service Grid infrastructure in Europe, available to scientists. This also means providing robust middleware components, deployable on several platforms and operating systems, corresponding to a set of core Grid services.

Workload management is one of these key Grid services. In the past few years significant results have been achieved on the problem of scheduling and efficiently managing a big number of different data-intensive jobs to a Grid encompassing many and heterogeneous resources. The Workload Management System implemented within the DataGrid project [2, 3], the Condor system [4] and the EuroGrid-Unicore resource broker [5] are some examples that must be reported. However the problem of Grid scheduling and resource management can of course not be considered as completely solved yet since many areas still require attention.

Taking into account the previous experiences from other Grid projects, the feedback and requirements coming from the reference applications, the on-going standardization specifications, an architecture for the EGEE Workload Management System has been designed and it is being implemented. It is presented in the rest of this paper.

ARCHITECTURE OF THE EGEE WORKLOAD MANAGEMENT SYSTEM

The Workload Management System (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 specific kind of tasks that request computation are usually referred to as “jobs”. In the WMS, the scope of tasks needs to be broadened to take into account other kinds of resources, such as storage or network capacity. This change of definition is mainly due to the move from batch-like activity to applications with more demanding requirements in areas like data access or interactivity, both with the user and with other tasks. The WMS will broaden its scope accordingly.

Functionality

The core component of the Workload Management System is the Workload Manager (WM), whose purpose is to accept and satisfy requests for job management, expressed via a Job Description Language (JDL) based on ClassAd [6], coming from its clients. The other fundamen-

tal component is the Job Logging and Bookkeeping Service, which is described below.

For a computation job there are two main types of request: submission and cancellation (the status request is managed by the Logging and Bookkeeping Service).

In particular the meaning of the submission request is to pass the responsibility of the job to the WM. The WM will then pass the job to an appropriate Computing Element (CE) for execution, taking into account the requirements and the preferences expressed in the job description. The decision of which resource should be used is the outcome of a *matchmaking* process between submission requests and available resources. The availability of resources for a particular task depends not only on the state of the resources, but also on the utilization policies that the resource administrators and/or the administrator of the Virtual Organization (VO) the user belongs to have put in place.

Scheduling Policies

A WM can adopt a more or less eager or lazy policy in order to schedule a job. At one extreme, eager scheduling dictates that a job is bound to a resource as soon as possible and, once the decision has been taken, the job is passed to the selected resource for execution, where, very likely, it will end up in a queue. At the other extreme, lazy scheduling foresees that the job is held by the WM until a resource becomes available, at which point that resource is matched against the submitted jobs and the job that fits best is passed to the resource for immediate execution. Varying degrees of eagerness (or laziness) are applicable.

At matchmaking level the main difference between the two extremes is that eager scheduling implies matching a job against multiple resources, whereas lazy scheduling implies matching a resource against multiple jobs.

The WM internal architecture will accommodate application of the different policies, implemented as easily interchangeable plugins, depending first of all on the requirements and preferences expressed in the job description, but also on the overall state of the Grid, according to appropriate heuristics. Such knowledge can only come from proper investigation (including the evaluation of relevant metrics, covering both resource utilization and user satisfaction), with the purpose to understand strengths and weaknesses of the different scheduling policies in different scenarios.

The Information Supermarket

The mechanism that allows the flexible application of different policies is the decoupling between the collection of information concerning resources and its use. The component that implements this mechanism is dubbed *Information Supermarket* (ISM) and represents one of the most notable improvements in the WM as inherited from the EU DataGrid (EDG) project.

The ISM basically consists of a repository of resource information that is available in read only mode to the matchmaking engine and whose update is the result of either the arrival of notifications or active polling of resources or some arbitrary combination of both. Moreover the ISM can be configured so that certain notifications can trigger the matchmaking engine. This functionality will not only improve the modularity of the software, but will also support the implementation of lazy scheduling policies.

The Task Queue

The second most notable improvement in the WM internal design is the possibility to keep a submission request for a while if no resources are immediately available that match the job requirements. This technique is used, among others, by the AliEn [7] and Condor [4] systems. Non-matching requests will be retried either periodically (in an eager scheduling approach) or as soon as notifications of available resources appear in the ISM (in a lazy scheduling approach). Alternatively such situations could only lead to an immediate abort of the job for lack of a matching resource.

The component that implements this feature is dubbed *Task Queue* (TQ) and, as for the ISM, provides a necessary mechanism for the support of lazy scheduling policies.

Job Logging and Bookkeeping

The Logging and Bookkeeping Service (L&B) tracks jobs in terms of *events*—important points of job life, e.g. submission, finding a matching CE, starting execution etc.—gathered from various WMS components as well as CEs. The events are passed to a physically close component of the L&B infrastructure (*locallogger*) in order to avoid network problems. This component stores them in a local disk file and takes over the responsibility to deliver them further.

The destination of an event is one of *bookkeeping servers* (assigned statically to a job upon its submission). The server processes the incoming events to give a higher level view on the job states (e.g. *Submitted, Running, Done*) which also contain various recorded attributes (e.g. JDL, destination CE name, job exit code, etc.). Retrieval of both job states and raw events is available via legacy and WS querying interfaces.

Besides querying for the job state actively, the user may also register for receiving notifications on particular job state changes (e.g. when a job terminates). The notifications are delivered using an appropriate infrastructure.

The Overall Architecture

Figure 1 shows the overall architecture of the Workload Manager, together with the interactions with external entities. Among these the most coupled with the WM is the Logging and Bookkeeping Service, which keeps events generated by different components as a job traverses them. Such events contribute to the generation of the status of a

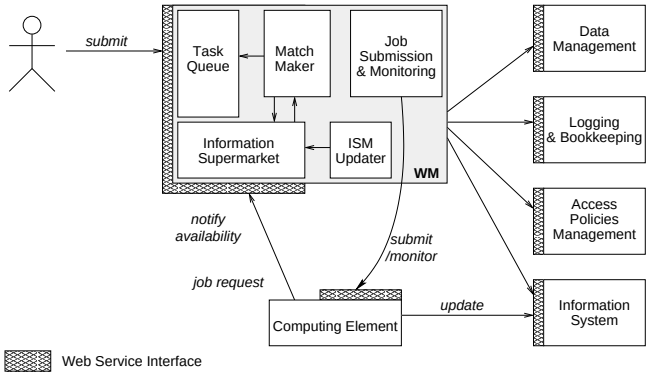

Figure 1: Internal architecture of the Workload Manager.

job. Other entities are the Information System, used, for example, to fill the Information Supermarket, the Data Management services, assisting the WM when the scheduling involves knowledge concerning location of data on the Grid and the Access Policies infrastructure.

Both the WM and the other services are expected to offer a Web Service interface.

COMPUTING ELEMENT

The Computing Element (CE) is the service representing a computing resource. Its main functionality is job management. In particular it must provide facilities:

- to run jobs (which includes also the staging of all the required files).
  Characteristics and requirements of jobs that must be executed are specified relying on the same Job Description Language (JDL), used within the whole Workload Management System;
- to get an assessment of the foreseen "quality of service" for a given job to be submitted. This reports, first of all, if there are resources matching the requirements and available according to the local policies. It then might provide an estimation of the local queue traversal time, that is the time elapsed since the job entered the queue until it starts execution;
- to cancel previously submitted jobs;
- to suspend and then resume jobs, if the local resource management system allows these operations;
- to get the status of some specified jobs, or of all the active jobs "belonging" to the user issuing the request;
- to be notified on job status, for example when a job changes its status or when a certain status is reached.

For job submission, the CE will be able to work in *push model* (where the job is pushed to a CE for its execution) or *pull model* (where the CE is asking the Workload Management Service for jobs).

When a job is pushed to a CE, it gets accepted only if there are resources matching the requirements specified by the user, and which are usable according to the local policies set by the local administrator. The jobs gets then dispatched to a worker node matching all these constraints.

In the pull model, instead, when a CE is willing to receive a job (according to policies specified by the local administrator, e.g. when the CE local queue is empty or it is getting empty) it requests a job from a known Workload Management Service. This notification request must include the characteristics and the policies applied to the available resources, so that this information can be used by the Workload Management Service to select a suitable job to be executed on the considered resource.

Various scheduling mechanisms within the pull model must be investigated to determine which provide the best performance in various situations when a CE willing to receive a job for execution, has to refer to multiple Workload Management Services. Possible mechanisms include:

- The CE requests a job from all known Workload Management Services. If two or more Workload Management Services offer a job, only the first one to arrive is accepted by the CE, while the others are refused.
- The CE requests a job from just one Workload Management Service. The CE then gets ready to accept a job from this Workload Management Service. If the contacted Workload Management Service has no job to offer within a certain time frame, another Workload Management Service is notified. Such a mechanism would allow supporting priorities on resource usage: a CE belonging to a certain VO would contact first a Workload Management Service referring to that VO, and only if it does not have jobs to be executed, the Workload Management Services of other VOs are notified, according to policies defined by the owner of the resource.

The CE, exposing a Web Service interface, may be used by a generic client: an end-user interacting directly with the Computing Element, or the Workload Manager, which submits a given job to an appropriate CE found by a matchmaking process.

The architecture of the Computing Element is represented in Figure 2.

The *Monitor* (MON) Service deals with notifications. It can be customized in particular to:

- asynchronously notify users on job status events, according to policies specified by users (e.g. when a job changes its status, when a job reaches a certain status, etc.).
- notify about the CE characteristics and status. In particular, for a CE working in pull mode, this service is used to request jobs to the Workload Management Service.

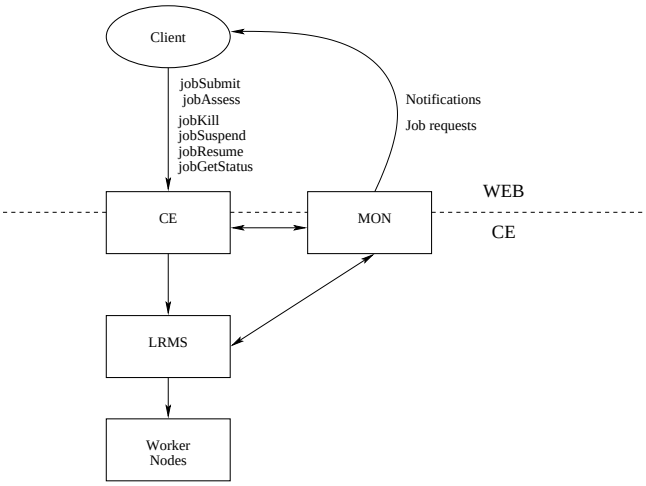

Figure 2: Architecture of the Computing Element.

OTHER SERVICES

Job Provenance (JP) is another Workload Management System service. The purpose of the Job Provenance Service is keeping track of the definition of submitted jobs, execution conditions and environment, and important points of the job life cycle for a long period (months to years). Those data can be used for debugging, post-mortem analysis, comparison of job execution within an evolving environment, as well as assisted re-execution of jobs. Only data of completed (either successful or failed) jobs are handled; tracking jobs during their active life is the task of L&B.

In general, gathered data are stored (i.e. copied) within the JP storage in order to really conserve a partial snapshot of the Grid environment when the job was executed, independently of changes of other Grid services. Obviously there are practical limitations of the extent to which it is feasible to record the entire job execution environment (in the ideal case this would encompass a snapshot of the entire Grid!). We restrict the recorded data to those that are processed or somehow affect processing of the Workload Management and Computing Element services.

Another service relevant with the Workload Management System is the Accounting Service. The Accounting Service accumulates information about the usage of Grid resources by the users and by groups of users, including Virtual Organizations as groups of users. This information allows preparation of statistical reports, to track resource usage for individual users, to discover abuses and to help avoid them. Accounting information could be used to charge users for the Grid resources they have utilized. The information available from the Accounting Service can also be used to implement submission policies based on user quotas or on resource usage (fair share). In principle it also allows the creation of a real exchange market for the Grid resources and services. The subsequent economic competition should result in market equilibrium, thereby promoting load balancing on the Grid.

More details on these services can be found in [8].

CONCLUSIONS

In this article the planned architecture for the EGEE Workload Management System has been presented. It consists of several services (Workload Manager, Logging and Bookkeeping, Computing Element, etc.) interacting among them and also with other services outside the Workload Management System.

The architecture is now being implemented by integrating and revising existing Grid software components. In particular the WMS implemented in the framework of the DataGrid project, which is currently deployed and used in the LCG Grid, is being revised in order to comply with the EGEE midlleware architecture.

ACKNOWLEDGMENTS

EGEE is a project funded by the European Union under contract INFSO-RI-508833. We also acknowledge the national funding agencies participating in EGEE for their support of this work

REFERENCES

[1] Home page of the EGEE project, http://www.eu-egee.org

[2] C. Anglano et al., “Integrating Grid tools to build a Computing Resource Broker: activities of DataGrid WP1”, in Proceedings of the 2001 Computing in High Energy and Nuclear Physics Conference (CHEP01), Beijing, China, September 2001

[3] G. Avellino et al., “The first deployment of workload management services on the EU DataGrid Testbed: feedback on design and implementation”, in Proceedings of the 2003 Computing in High Energy and Nuclear Physics Conference (CHEP03), La Jolla, Ca, USA, March 2003.

[4] M. Litzkow, M. Livny and M. Mutka, “Condor - A Hunter of Idle Workstations”, in Proceedings of the 8th International Conference of Distributed Computing Systems.

[5] M. Romberg, “The UNICORE Grid infrastructure”, in “Scientific Programming”, volume 10, pages 149–157, 2002.

[6] R. Raman, M. Livny and M. Solomon, “Matchmaking: Distributed Resource Management for High Throughput Computing”, in Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing (HPDC7).

[7] P. Buncic, A.J. Peters and P. Saiz, “The AliEn system, status and perspectives”, in Proceedings of the 2003 Conference for Computing in High-Energy and Nuclear Physics (CHEP 03).

[8] EGEE Design Team, “EGEE Middleware Architecture”, https://edms.cern.ch/document/476451.