LOOKING FOR AN EVOLUTION OF GRID SCHEDULING: META-BROKERING*

I. Rodero, F. Guim, and J. Corbalan
Barcelona Supercomputing Center (BSC) and Technical University of Catalonia (UPC)
{ivan.rodero, francesc.guim, julita.corbalan}@bsc.es

L.L. Fong and Y.G. Liu
IBM T.J. Watson Research Center
{llfong, ygliu}@us.ibm.com

S.M. Sadjadi
Florida International University
sadjadi@cs.fiu.edu

Abstract A Grid Resource Broker for a Grid domain, or also known as meta-scheduler, is a middleware component used for matching works to available Grid resources from one or more IT organizations. A Grid meta-scheduler usually has its own interfaces for the functionalities it provides and its own job scheduling objectives. This situation causes two main problems: the user uniform access to the Grid is lost, and the scheduling decisions are taken separately while they should be done in coordination. These problems have been observed in different efforts such as the HPC-Europa project but they are still open problems. In this paper we discuss the requirements to achieve a more uniform access to the Grids through a new approach to global brokering. As the results of these discussions on brokering requirements, we propose a meta-brokering design, so called meta-meta-scheduler design, and discuss how it can be realized as a centralized model for the HPC-Europa project, and as a distributed model for the LA Grid project.

Keywords: Grid resource management, meta-brokering/scheduling, interoperability

*This research has been supported in part by the Spanish Ministry of Science and Technology under contract TIN2004-07739-C02-01, the EU project HPC-Europa under contract 506079, the CoreGRID FP6 European Network of Excellence (Contract IST-2002-004265), NSF under grant OCI-0636031, and IBM.

1. Introduction

In a Grid environment, a resource broker, also called meta-scheduler[1], is usually used to manage user submitted jobs and the scheduling of jobs for execution to the available Grid resources from one or more IT organizations under a Grid domain. A Grid meta-scheduler has its own interfaces for the functionalities it provides and its own job scheduling objectives. However, there is not yet a standard on the interfaces of Grid meta-scheduler to support the interoperability of different meta-scheduling systems that bring us to the original idea of "The Grid", which promised an infrastructure to provide a uniform access to resources across different centers and institutions. This is an important issue because typically, a large Grid environment can be composed with different institutions or centers and each center would like to use its own scheduling system. The current Grid is managed by different meta-schedulers that manage a particular institution or virtual organization, and then the Grid becomes divided into several independent Grids without any interaction between them. In this context, the different Grid meta-schedulers are working independently, with different capabilities and using different languages for describing jobs, for submission, monitoring and so on. Different meta-scheduling projects can be found in literature, as detailed in Section 2. A meta-scheduling architecture can be based on different models, from a centralized to a distributed model as it is discussed in [26]. We also can find various approaches in scheduling policies such as economics [4], load balancing [25], or based on multi-criteria [16].

To solve the interoperability problem between different meta-scheduling systems, some initiatives have been developed. The HPC-Europa project is providing a solution to this problem through the development of a web portal to be used as a single point of access for different HPC centers in Europe [13]. In this approach, each center implements a plug-in with its own set of supported capabilities. Next, the user chooses manually the meta-scheduling system to submit their jobs. Finally, the job scheduling policies are evaluated inside the context of each center. Using the experience gained from this project, we have concluded that the portal approach is not enough to provide a transparent single point of access to Grid environments as the users are still involved with undesirable complexity and the scheduling results may less optimal. Therefore, we need to extend the model of HPC-Europa and support the mechanisms and policies on top of the meta-schedulers; basically, brokering the Grid meta-schedulers. In addition, the idea of the interoperability between different middleware and systems was studied in other projects such as in Grid Interoperability Project

[1]In this paper, we will use resource broker, broker, and meta-scheduler inter-changeably to refer to a grid resource broker. We will later define meta-broker as a broker on top of brokers, or a meta-meta-scheduler.

[3]. More specifically, this interoperability project tried to work on mechanisms to create Grids with a uniform access to both Globus and Unicore systems.

The idea of brokering on top of Grid meta-scheduler has been taken into account in other works such as in [15]. As a meta-broker, the scheduler on top of meta-scheduler should be called meta-meta-scheduler. Therefore, a meta-broker can be defined as the middleware component that selects the most appropriate meta-scheduler to submit a job following a particular policy. This point of view has influenced the way we are extending the approach in HPC-Europa as we propose in Section 5. Furthermore, the Open Grid Forum (OGF) is working on some recommendations regarding interoperability, but the work is still in working process. Therefore, we will consider the OGF approach only as a reference for our research activities.

In this paper, we will study the requirements for a meta-brokering system and we will define a set of requirements for common interfaces that allows accessing and managing different Grid meta-schedulers in a uniform way and provides users with single point of access. To achieve these goals we propose two approaches: (1) designing and implementing a centralized meta-brokering system on top of the different brokers; and (2) designing and implementing a distributed meta-brokering system by communicating the different brokers with a set of protocols and a certain agreement. For the first approach, we present a design in which we are working on the eNANOS [25] framework as an extension of HPC-Europa JRA2 activity, and for the second approach, we present a design based on the LA Grid meta-brokering project [17].

The rest of this paper is organized as follows. In Section 2, we present some of the current approaches in meta-brokering. In Section 3, we summarize the work done in the HPC-Europa and the lessons learned in the requirements of meta-brokering. In Section 4, we discuss the requirements and architectural elements of ideal meta-brokers for Grid environments. In Section 5, to provide the meta-brokering functions as described in Section 4, we first describe a design extension for the HPC-Europa as a centralized model; then we described the distributed design approach of LA Grid meta-brokering project; and finally we summarize the models in a table. In Section 6, we present some conclusions and some roadmap for our future works.

2. Related Work

Several projects regarding Grid meta-scheduling can be found in literature. Both specific and general-purpose initiatives have been developed during last years, and some of them are presented as follows. Condor-G [12] is based on the Condor approach for Grids that combines the inter-domain resource management protocols of the Globus Toolkit and the intra-domain resource and job management methods of Condor. AppLes [2] is a project targeted to

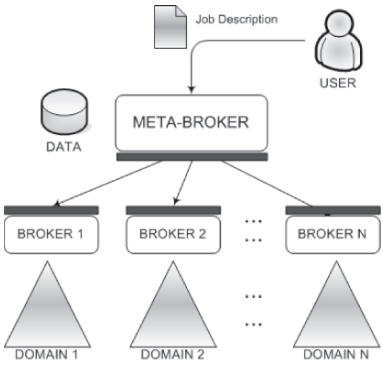

Figure 1. General meta-broker architecture [17]

the application level scheduling. GridBus [18] is an economy-capable Data Grid service broker for scheduling distributed data oriented applications across Grid resources. GRMS [10] is an open source meta-scheduling system, which allows developers to build and deploy resource management systems to large scale distributed computing infrastructures. The GridWay framework [11] is a component for meta-scheduling in the Grid Ecosystem intended for end users and Grid applications developers. The eNANOS project [25] is based on the idea of having a good low-level support for performing a good high-level HPC scheduling.

There are some initiatives regarding the interoperability of different meta-schedulers [3]. The Grid Interoperability Project (GRIP) aim was to realize the interoperability of Globus and Unicore combining the unique strength of each system and to work towards standards for interoperability of meta-scheduling in the OGF. This goal has been achieved in a real testbed and they have extended for different meta-scheduling systems, as in the HPC-Europa project that is described in Section 3.

In terms of meta-brokering, an abstract architecture has been proposed in [15] and the objectives are similar to those discussed in HPC-Europa [13] or in the interoperability project [3]. The architecture of these proposals is similar. They are based on a meta-broker model that receives the job submission, and manages the resource brokers with some data information, as it is shown in Figure 1. In this paper, we present new designs of meta-brokering that extends the work done in the Grid community and leverages our experiences from HPC-Europa.

As the Grid standardization organization, the OGF is working on scheduling architecture in the Grid Scheduling Architecture Research Group (GSA-RG) [20]. This group has worked on a scheduling hierarchy and the communication between scheduling instances. Recently the group has started working on interoperability and on a proposal regarding the interaction between Grid

schedulers. The OGSA Resource Selection Services Working Group (OGSA-RSS-WG) [19] will provide protocols and interface definitions for the resource selection services portion of the Execution Management Services (EMS) part of the Open Grid Services Architecture. The Resource Selection Services (RSS) consist of the Candidate Set Generator (CSG) and the Execution Planning System (EPS). The CSG can be used to generate a set of computational resources that are able to run a job in general, while the EPS uses this list to decide exactly what resources to run the job.

We take into account the OGF recommendations as a reference. In our approach, we promote a more practical point of view as we develop a meta-brokering system in real environments with particular solutions.

3. Lessons learned from the HPC-Europa project

One major activity of HPC-Europa project is to build a portal that provides a uniform and intuitive user interface to access and use resources from different centers. As most of the HPC centers have already deployed their own site-specific HPC and Grid infrastructure; therefore, an important requirement is to keep the autonomy of HPC centers by allowing them to use their favorite middleware, local policies, and so on. For instance, there are currently five different systems that provide a job submission and basic monitoring functionality in the HPC-Europa infrastructure: eNANOS [25], GRIA middleware [9], Grid Resource Management System (GRMS) [10], Job Scheduling Hierarchically (JOSH) [14], and UNICORE [28]. Additionally, eNANOS, GRMS and JOSH use the Globus Toolkit to access underlying resources provided for the HPC-Europa infrastructure.

The Single Point of Access (SPA) effort of HPC-Europa provides two sets of interfaces to application users. Firstly, a generic interface set that can be used by all users for most of their batch applications. To this end, this uniform interfaces are provided for the most relevant Grid functionality identified from a requirements analysis of the centers. The key set of functionalities has been determined to be required for the realization of the SPA: job submission, job monitoring, resource information, accounting, authorization, and data management. Secondly, an application-specific set of portlets are being developed to allow users to manage more complex (e.g., interactive or requiring many specific input parameters) applications in a straightforward manner.

In order to provide end-users with transparent access to resources, we developed a mechanism responsible for the management of uniform interfaces to diverse Grid middleware. Using this mechanism the Single Point of Access enables dynamic loading of components that provide access to the functionality of specific Grid middleware through a single uniform interface. These components are called plug-ins in this context (see Figure 2). These uniform interfaces

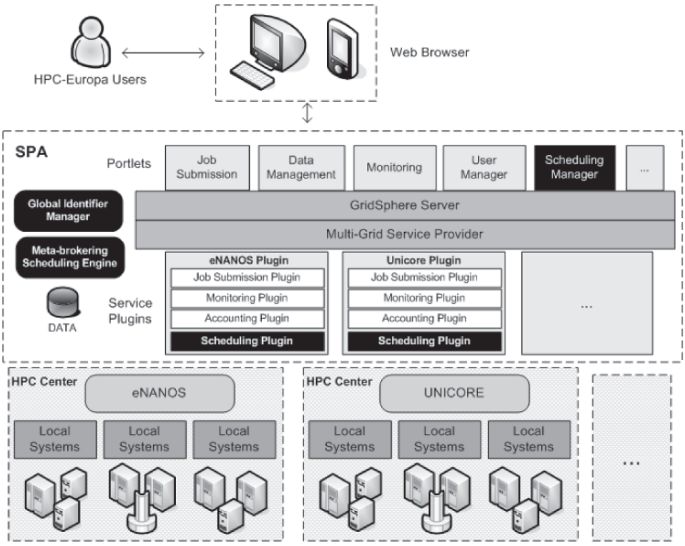

Figure 2. HPC-Europa architecture with the proposed extensions

are based on standards where possible (e.g., JSDL for job submission [21]) and functionalities provided by Grid middleware deployed in HPC centers.

From the end-user perspective, a uniform GUI is provided that is common for all systems deployed in the HPC-Europa infrastructure. This GUI can be dynamically adapted to particular systems and still keep the same look and feel. Only slight modifications such as disabling fields and limiting lists of values are allowed. When a user wants to submit a job, the user is required to choose the center to which the job has to be submitted and to specify its requirements. There is no global scheduling and the brokering is done manually by the user.

To this end, we have implemented the ability to check the functionality of every resource broker system by retrieving the capabilities of site-specific plug-ins. These descriptions of the implemented capabilities are returned in the form of the appropriately constrained general XML schema. A plug-in returns two descriptions: a description of the methods it supports and a description of data structures (e.g., job description). In [21], this mechanism is described in more detail using the Job Submission Portlet as an example.

4. Design of Meta-brokering

4.1 Requirements for Meta-brokering

From the experiences in HPC-Europa project we have observed some requirements that should be taken into account when developing a meta-brokering system:

1 Global Addressing Mechanism: We need a mechanism to address the different involved resources. In this case, the main resources of a meta-brokering system are brokers or their services, not the resources to execute the job.

2 Common Capability Description Language: Since each broker provides its own set of functionalities, it is required to have a Capability Description Language (CDL) to describe all the services capabilities (e.g., submission, monitoring, accounting, and control).

3 Common Job Description Language: To eliminate the complexity of each broker having its own job description language, it is required to have a common language for describing jobs, requirements and so on.

4 Global Job Identifiers: It is very important to have unique mapping of Grid jobs to different brokers and to the local resources. An implementation can be done using a single jobID provider for the meta-brokering system or just using each broker system to argument the job identifications. In any case jobIDs must be unique.

5 Unified Notifications Mechanism: It is required a common mechanism or protocol to notify events. The system can receive notifications from any broker and the notifications should be handled in the same way.

6 Unified Monitoring Mechanism: Since each broker has its own way to return the monitoring data, including mechanism, data type and schema, it is necessary to have a common mechanism and schema for monitoring.

7 Unified Accounting Mechanism: Usually the selection of resources is done using the accounting information especially when economic policies are applied. For meta-brokering the selection of brokers can be done in a similar way.

8 Unified Agreement Mechanism: A meta-brokering system needs a mechanism to make agreements between brokers. The agreement mechanism and an API are also required to establish protocols to communicate brokers.

9 Common Scheduling Description Language: We need a Scheduling Description Language (SDL) to describe the scheduling capabilities that a broker provides (e.g., depending on the user or the users center, a broker can offer a policy with more or less priority), and global meta-brokering policies.

In the Grid environments, this is especially important because there are interactions between several components and the different approaches can be

originated from different contexts. In HPC-Europa, we adopted the standards proposed by the OGF wherever it was possible. For the requirements listed above, we propose to use at least the following set of standards: JSDL 1.0 with some extensions for job description, WS-Agreement for the agreement protocol between brokers, and WS-Addressing for addressing resources in general.

Other APIs and schemas are yet to be defined to address the remainder requirements listed above. As an example, some schema such as the one used in HPC-Europa project and presented in [13] can be used for monitoring.

4.2 The Architecture

To meet the requirements listed in section 4.1, in this paper we propose two meta-brokering models. On one hand, we consider the centralized model which is suitable for a limited number of centers and institutions. This is the model considered in the HPC-Europa extension. On the other hand, we consider the distributed model for the LA Grid meta-scheduling project, which is more suitable for more dynamic environments with a higher number of centers and institutions (i.e., centers can be added to the infrastructure dynamically). Moreover, the distributed model is more appropriate for more heterogeneous environments [19] such as the case of the LA Grid, which can be composed of different kind of resources, from a collection of desktop PCs to supercomputing centers. These two different models are discussed later sections.

4.3 The Scheduling

In addition to the architecture model and the required interfaces, there is an important functionality such as the scheduling at the meta-broker level. We can implement different kind of scheduling policies depending on the kind of information we have at the meta-brokering level. On one hand, if the meta-broker has information about the details of the resources, it can implement the typical scheduling policies studied in the literature, such as in [1] or [4], but extending them to a larger amount of resources with the lower level brokers acting as job dispatchers and execution entities. On the other hand, if the meta-broker does not have any information about the resources and only has certain information about the brokers, we can implement other kind of policies. One of the possible meta-brokering policies can be based on the capabilities of the brokers. In this case, the selection of the appropriate brokering system can be done using a multi-criteria algorithm that can take into account the brokers capabilities or even dynamic information, prediction and so on. Another kind of policies can use accounting information to select the appropriate broker for a given job in a particular situation. This kind of policies maps directly to the economic paradigm.

Furthermore, a meta-brokering scheduling algorithm can be implemented many approaches. For example, in BSC we are currently working on policies that use the historical job/resource information to take its scheduling decisions. We are designing data mining techniques that uses the historical information of Grid resources usage, Grid workloads, and a minimum set of job requirements (e.g., executable, number of processors and input files) to estimate variables about: (1) the job requirements, which estimate how much processor, disk and memory a job will use; and (2) the future state of the different resources evolved in the Grid, which estimate, for instance, how much free space will be available in a given host, or what load it will have in a given future time.

We derive this information by correlating the past executions of similar jobs using similar resources, or with similar future load using similar prediction techniques that have been proposed in other works in terms of job performance prediction [5] [8] [12] [27] and resource usage [7] but using Grid workloads. In terms of meta-brokering scheduling, we are designing techniques that use this information for matching jobs to the resources that will better satisfy their requirements.

In this paper we discuss the importance of the scheduling for a meta-broker approach. However, we do not present any particular result because we do not dispose of a testbed or simulation environment for a meta-brokering system yet.

5. Meta-broker designs

In this section, we will provide the detailed designs of the centralized and distributed models of meta-brokers.

5.1 HPC-Europa Extension Proposal

In this proposal, we extend the solution in the HPC-Europa project by adding a meta-broker on top of the individual brokers in such a way that the scheduling decisions are taken depending on the capabilities of the individual brokers. Thereby, we add a functionality to make the selection of each broker or center in the meta-broker.

This extension should include new services to perform meta-brokering. As we have identified in the list of requirements, the HPC-Europa infrastructure should incorporate a new module for the broker scheduling, a scheduling plugin for each center, a language to describe the scheduling capability of brokers and global identifiers management. Moreover, the meta-broker can include some other services such as a predictor service or a historic data catalog to improve the scheduling techniques.

The idea is to design a system following the OGF standards such as the JSDL as a language for describing jobs. In Figure 2, we show the architecture of the

extended version of the HPC-Europa approach. The extension is distinguishable by the dark shading.

This meta-broker approach allows the incorporation of more scheduling functionality at the meta-broker level ("SPA" in Figure 2). For example, the meta-broker can store some useful information to improve the scheduling strategy ("DATA" in Figure 2). Some of this information can be the previous decisions regarding the selection of brokers as a historical data such as the achieved quality of service by the different brokers, the average waiting time, the reputation of brokers, or the level of availability and reliability of the resources under the broker domains.

In the new architecture, the meta-brokering scheduling engine is responsible for the broker selection and where to submit a job. At the portal level, it is evaluated one of the meta-brokering policies available in the framework and, associated to this component, a new portlet should be provided to configure policies and its main issues.

To map the scheduling performed in the meta-broker scheduling engine following the scheduling capabilities of the different brokers, we need a new plugin. This plugin implements the scheduling API and provides the functionality supported by each brokering system. Therefore, to define the scheduling capability of the different center (brokers), we need a scheduling description language. This language should include the scheduling policy capabilities, such as the quality of service, priorities, the support for co-allocations and for advanced reservations, economic issues, or the scheduling policy family. For example, in eNANOS we can offer as a scheduling capability the load-balancing of parallel applications in run-time or the support for MPI+OpenMP applications [25].

Finally, to allow the portal managing jobs coming from all the centers, we need global identifiers mechanisms. To obtain global identifiers, we need a new service (global identifier manager) that should be accessible from all the services and through the centers plugins. The global identifier service should not modify the rest of services; the different brokers should support this new functionality via the plugins. The Universally Unique Identifier (UUID) [29] seems to be a good candidate as identifier standard. It is used in software construction, standardized by the Open Software Foundation (OSF) as part of the Distributed Computing Environment (DCE). A UUID is essentially a 16-byte (128-bit) number.

5.2 Meta-brokering design in LA Grid

Latin American Grid initiative (LA Grid, pronounced "lah grid", [17]) is a multifaceted international academic and industry partnership designed to promote research, education and workforce development collaborations with major institutions in United States, Mexico, Argentina, Spain, and other locations

around the world. LA Grid has developed a global living laboratory where international researchers are empowered to build new research partnerships and explore the synergies of their research strengths in areas including transparent Grid enablement, autonomic resource management, meta-brokering, and job flow Management. The meta-scheduling project in LA Grid aims to support Grid applications with resources located and managed in different domains spanned over a Grid computing cyber infrastructure. This project addresses the architecture, design, implementation and deployment issues related to meta-brokering.

The meta-broker design in LA Grid is a distributed model with multiple meta-broker instances cooperate with each other to provide Grid functions. As illustrated in Figure 4, one instance of a meta-broker consists of three functional modules: resource module, scheduling module and job module. Resource module is responsible for resource discovery, monitoring and storage. Scheduling module is responsible for locate suitable resources or brokers for a job request. Job module is responsible for management of job lifecycle: submission, dispatching and execution monitoring. A meta-broker instance interacts with existent brokers within the resource domain.

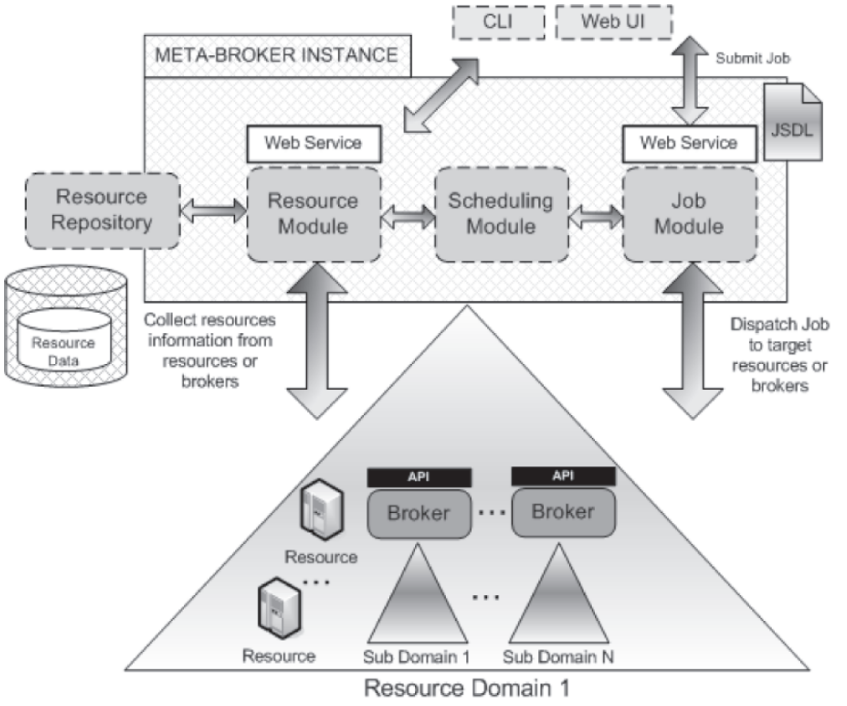

Figure 3. Meta-broker in LA Grid

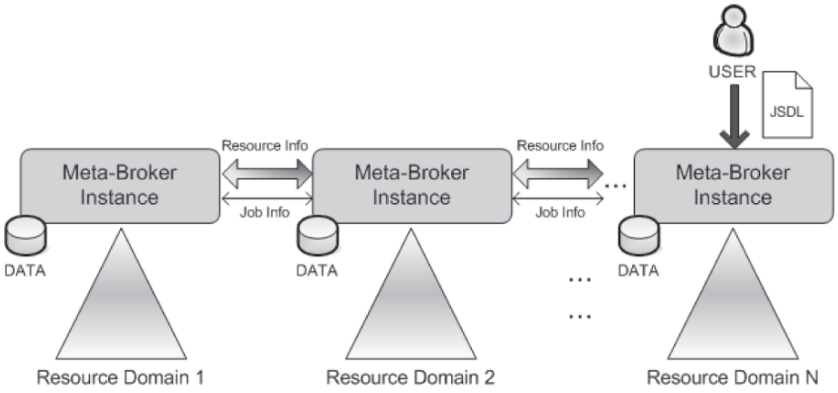

Figure 4. Meta-broker communication model in LA Grid

In LA Grid, resources of different institutions belong to their respective resource domains and each resource domain has a meta-broker instance (see Figure 5). Inside a domain, a meta-broker instance controls resources directly and/or indirectly through other brokers in the domain. In the former case, each resource reports its information directly to the meta-broker instance using resource modules web services. In the latter case, an existent broker is responsible for reporting the information of the resources under its control. How the broker collects the resource information is within its own implementation details. In this way, once an existent broker is able to use web services to report resource information to a meta-broker instance, different existent brokers can participate in the LA Grid cyber infrastructure.

Each meta-broker instance collects resource information from its neighbors and save the information in its resource repository or in-core memory. The resource information is distributed in the Grid and each meta-broker instance will have a view of all resources. The resources information is in aggregated forms to save storage space and communication bandwidth. Example of aggregated resource information on a set of servers in a domain is: type=CPU; speed={1G,2G}; OS=Win; quantity=3.

As shown in Figure 5, a job request is described in JSDL and can be submitted to any known meta-broker instance using web services. In general, a job request is submitted to the meta-broker instance in the same resource domain whose address is well-known in the domain. When a job request arrives, a meta-broker instance matches the job to a domain with the appropriate set of resources. The matching algorithm is influenced by multiple factors. One of the factors is the location of resources such that the preference will be given to the local domain at which the job is submitted.

If the matched resources are outside of the domain, the job is dispatched to the meta-broker instance in another domain. The existent broker or meta-broker instance that the job is dispatched to has the resources required by the job and will be responsible to dispatch the job again if necessary. The resource, existent broker or meta-broker instance that the job is dispatched to will report the job status back to the original meta-broker instance.

In summary, we define a set of web services provided by meta-broker and make the incorporation of meta-brokers easy. We store a view of global resources in each meta-broker instance to provide speedy resource matching. Thus, users can experience short response time. Though storing a view of global resource locally costs storage space and communication bandwidth, we apply multiply technologies to reduce the overhead and to keep a similar performance with a incomplete view of global resources. Due to the space limitation, we shall report our efforts in these perspectives in a separate paper.

5.3 Summary of meta-brokering designs

In Figure 5, we summarize the main functionalities and some details for the discussed approaches: the original HPC-Europa approach, the HPC-Europa extension, and the LA Grid project.

	HPC-Europa	Extended HPC-Europa	LA Grid
Architecture	Centralized	Centralized	Distributed + P2P
Addressing mechanism	Plugins	Plugins	WS-addressing
Capabilities mechanism	Plugins + MultiGrid Service Provider	Plugins + MultiGrid Service Provider	Service Provider
Capability description lang.	API + XML	API + XML	API + XML
Job Submission	JSDL schema + ext.	JSDL schema + ext.	JSDL schema + ext.
Monitoring	XML schema	XML schema	XML schema
Accounting	NO	XML schema	To be defined
Scheduling	NO	XML schema (future JSDL)	XML schema (as a resource type)
Agreement	NO	WS-Agreement extension	WS-Agreement ext.
Job Description Language	JSDL 1.0 + ext.	JSDL 1.0 + ext.	JSDL 1.0 + ext.
Global identifiers	NO	YES	YES
Notification mechanism	NO	Callbacks	Callbacks
Agreement mechanism	NO	Centralized	P2P
Security	X.509 Certificates	X.509 Certificates	To be defined

Figure 5. Summary of the main functionalities of the discussed approaches

6. Conclusions and Future Work

In this paper we have discussed the convenience of the meta-brokering approach to achieve a single point of access to the Grid and to access to more

resources. We also have seen how some initiatives have been carried out as the HPC-Europa project. However, they need to be extended to achieve a real global scheduling on top of different brokering systems.

One important issue to take into account is the additional overhead added to the infrastructure with another layer. We can argue that in each layer of the infrastructure the scheduling and management system is oriented to a particular target. Moreover, incorporating another layer is acceptable because of the Grid environment characteristics (e.g., timeouts are longer).

Furthermore, from the experience obtained from the HPC-Europa project, we have presented the main issues to be taken into account to develop a meta-broker. We also have proposed to use some standards to implement such an extension. We have observed a lack of a consensus in the definition of the terms used in this area (meta-scheduler, broker, meta-broker, and so on). We miss a formal definition to avoid confusions.

As for the future work we have presented two ways to achieve the meta-brokering approach. On one hand we have presented a proposal for extending the solution developed in the HPC-Europa project providing more autonomy and performance. On the other hand we have presented an overview of the distributed meta-brokering system being done in LA Grid project. Finally, we think the main line of future work regarding the meta-brokering research is toward further investigation for scheduling policies that will allow the new Grids to become transparent, autonomous and efficient.

References

[1] S. Banen, A.I.D. Bucur, and D.H.J. Epema, “A Measurement-Based Simulation Study of Processor Co-Allocation in Multicluster Systems”, 9th Workshop on Job Scheduling Strategies for Parallel Processing, LNCS 2862, Seattle, 2003, pp. 105-128.

[2] F. Berman, R. Wolski, S. Figueira, J. Schopf, G. Shao, “Application-Level Scheduling on Distributed Heterogeneous Networks”, Supercomputing’96, Pittsburgh, PA, November 17-22, 1996.

[3] J. Brooke, D. Fellows, K. Garwood, C. Goble, “Semantic Matching of Grid Resource Descriptions”, LNCS 3165, January 2004, pp. 240-249.

[4] R. Buyya, “Economic-based Distributed Resource Management and Scheduling for Grid Computing”, Ph.D. Thesis, Monash University, Melbourne, Australia, April 12, 2002.

[5] A.B. Downey, “Using queue time predictions for processor allocation”, 3rd Workshop on Job Scheduling Strategies for Parallel Processing, LNCS 1291, 1997, pp.35-57.

[6] P.A. Dinda, “The statistical properties of host load”, Scientific Programming, 1999.

[7] R. Gibbons, “A historical application profiler for use by parallel schedulers”, Workshop on Job Scheduling Strategies for Parallel Processing, 1997.

[8] GRIA project Web Site. http://www.gria.org

[9] Grid Resource Management System (GRMS) Web Site. http://www.gridlab.org/grms

[10] GridWay Web Site. http://www.gridway.org

[11] F. Guim, J. Corbalan, J. Labarta, "Analyzing loadleveler historical information for performance prediction", Jornadas de Paralelismo, Granada, Spain, September 13-16, 2005.

[12] F. Guim, I. Rodero, J. Corbalan, J. Labarta, A. Oleksiak, J. Nabrzyski, "Uniform job monitoring using the HPC-Europa single point of access", Intl. Workshop on Grid Testbeds, in conjunction with CCGrid2006, Singapore, May 16-19, 2006.

[13] J. Frey, T. Tannenbaum, M. Livny, I. Foster, S. Tuecke, "Condor-G: A Computation Management Agent for Multi-Institutional Grids", 10th IEEE International Symposium on High Performance Distributed Computing, San Francisco, August, 2001.

[14] HPC-Europa Web Site. http://www.hpc-europa.org

[15] JOSH. http://gridengine.sunsource.net/josh.html

[16] A. Kertész, P. Kacsuk, "Grid Meta-Broker Architecture: Towards an Interoperable Grid Resource Brokering Service", CoreGRID Workshop on Grid Middleware, August, 2006.

[17] K. Kurowski, J. Nabrzyski, A. Oleksiak, J. Weglarz, "Multicriteria Aspects of Grid Resource Management", Grid Resource Management, J. Nabrzyski, J. Schopf, J. Weglarz (Eds), Kluwer Academic Publishers, Boston/Dordrecht/London, 2003.

[18] LA Grid Web Site. http://www.lagrid.fiu.edu

[19] C. Mastroianni, D. Talia, O. Verta, "A Super-Peer Model for Building Resource Discovery Services in Grids: Design and Simulation Analysis", European Grid Conference 2005, LNCS 3470, Amsterdam, The Netherlands, 14-16 February, 2005, pp. 132-143.

[20] K. Nadiminti, S. Venugopal, H. Gibbins, R. Buyya, "The Gridbus Broker Manual (v.2.0)", http://www.gridbus.org/broker

[21] OGF OGSA Resource Selection Services WG Web Site. https://forge.gridforum.org/sf/projects/ogsa-rss-wg

[22] OGF Grid Scheduling Architecture RG Web Site. https://forge.gridforum.org/sf/projects/gsa-rg

[23] A. Oleksiak, A. Tullo, P. Graham, T. Kuczynski, J. Nabrzyski, D. Szejnfeld, T. Sloan, "HPC-Europa: Towards Uniform Access to European HPC Infrastructures", 6th IEEE/ACM International Workshop on Grid Computing, Seattle, USA, 2005.

[24] I. Rodero, J. Corbalan, R. M. Badia, J. Labarta, "eNANOS Grid Resource Broker", EGC 2005, LNCS 3470, Amsterdam, The Netherlands, 14-16 February, 2005, pp. 111-121.

[25] I. Rodero, F. Guim, J. Corbalan, J. Labarta, "eNANOS: Coordinated Scheduling in Grid Environments", Parallel Computing, Malaga, Spain, 12-16 September, 2005, pp. 81-88.

[26] V. Subramani, R. Kettimuthu, S. Srinivasan, P. Sadayappan, "Distributed Job Scheduling on Computational Grids Using Multiple Simultaneous Requests", 11th Symposium on High Performance Distributed Computing, Edinburg, Scotland, 24-26 July, 2002, p. 369.

[27] D. Talby, D. Tsafrir, Z. Goldberg, D.G. Feitelson, "Session-based and estimation-less runtime prediction algorithms for parallel and grid scheduling", Technical Report, School of Computer Science and Engineering and the Hebrew University of Jerusalem, 2006.

[28] UNICORE Web Site. http://www.unicore.org

[29] Universally Unique Identifier. RFC 4122. http://www.ietf.org/rfc/rfc4122.txt