

# Service-based Resource Brokering for Grid-Based Data Mining

**Valentin Kravtsov**

Israel Institute of Technology, Haifa, Israel  
svali\_ds@cs.technion.ac.il

**Vlado Stankovski**

University of Ljubljana, Ljubljana, Slovenia  
vlado.stankovski@fgg.uni-lj.si

**Thomas Niessen**

Fraunhofer Institute for Autonomous  
Intelligent Systems, Bonn, Germany  
thomas.niessen@ais.fraunhofer.de

**Assaf Schuster**

Israel Institute of Technology, Haifa, Israel  
assaf@cs.technion.ac.il

**Abstract** - *The shift towards intrinsically distributed complex problem solving environments is prompting a need for new systems, which utilize the virtually unlimited data and computational resources of the Grid and at the same time hide all the related complexity from the user. Currently, there is no coherent framework, which offers data miners, who are usually not Grid experts, the ability to easily construct data mining tasks and execute them on the Grid. Therefore, there is a need to assemble a complete system that includes: a) a user-friendly environment for defining complex data mining tasks and b) a Grid middleware that supports execution of such tasks, while utilizing mechanisms for managing data and computational resources as well as having sophisticated job-monitoring capabilities. This paper will focus on the high-level design of such a system, which currently is being developed in the DataMiningGrid project with emphasis on the design and implementation of the resource broker service. We show how different resources from various domains can be exploited, in order to give the data mining researchers the ability to access and utilize resources needed for modern, distributed and computationally intensive data mining algorithms.*

**Keywords:** Data Mining, Resource Broker, GridBus

## 1. Introduction

As Grid [1] and pervasive computing environments are becoming commonplace, the availability of data mining [2] operations, algorithms and services available within proprietary Grid environments and the visible Web is constantly increasing. Such

scenarios give rise to novel ways of building data mining applications that dynamically integrate distributed data mining operations or services into a (logical) single system or tool [3]. Currently, there is a considerable variety of distributed, computational and data intensive data-mining applications and algorithms that simply cannot be fully exploited without Grid infrastructure. Nevertheless, we could not identify any software that provides a complete, user-friendly solution for data mining researchers willing to perform long-running data mining tasks on massive data on the Grid. Generally, data mining researchers are not Grid experts. Therefore, any solution that will be eventually adopted by them has to focus on user-friendly interfaces, which have to hide all the Grid complexity from the user. The success of such Grid-based data mining frameworks is dependent on the implementation of an essential component that will make “intelligent” decisions regarding all the Grid aspects, such as location of job execution, destination of data movement, error handling, error propagation etc. In a distributed system, this component is the resource broker.

Resource brokering is not a new concept in Grid realm. Resource Brokers were in the past and continue to be developed by different commercial companies and academic research groups. Software packages like CSF [4] and GridWay [5] are available for download from Globus Alliance [6] Web page. The list of resource broker implementations includes also GridLab Resource Management System (GRMS) [7], Nimrod/G [8], GridBus Resource Broker [9] and others that are already in use in diverse scientific areas and projects. Each one of the above tools has different advantages and integration

constraints. However, at the time of the design release none of the above brokers had a WSRF-compliant implementation with the tested ability of submitting jobs to Grid Resource Allocation Manager 4 (WS-GRAM). The meaning and the importance of WSRF will be discussed in the next section.

## 2. OGSA approach

The vision shared by many ongoing research projects is that future Grid systems will be based on the Open Grid Service Architecture (OGSA) [10]. OGSA is a distributed interaction and computing architecture based around the Grid Computing service, assuring interoperability on heterogeneous systems so that different types of resources can communicate and share information. OGSA addressed a number of requirements, which led to the definition of stateful Web services. Namely, as Web services are static entities, they do not give enough flexibility for creating on-demand systems. Such systems require dynamic creation and destruction of entities that represent virtual resources.

In terms of Grid middleware, both Globus Toolkit 3 [11], which implements the Open Grid Standards Infrastructure (OGSI), and further, the Globus Toolkit 4 [12], which implements the Web Services Resource Framework (WSRF), follow the OGSA.

Web Services Resource Framework (WSRF) is aimed at defining a generic framework for modeling and accessing persistent resources using Web services so that the definition and implementation of a service and the integration and management of multiple services is made easier [13]. WSRF narrowed the meaning of Grid services to those services that conform to the WSRF specification [14], although, a broader, more natural meaning of a Grid service is still in use. As defined in Foster's "Physiology of the Grid" [10], a Grid service is any service used in the Grid environment that conforms to specific interface conventions accepted as standard throughout this Grid.

However, the low-level fabric provided by WSRF is insufficient for developing complex-problem solving environments. Complex higher-level functionality is necessary to provide a seamless, transparent, manageable, reliable, efficient and secure common computing environment, in which a resource broker is an integral part.

Figure 1 illustrates the software components to be used and services to be developed in the DataMiningGrid project. The DataMiningGrid project focuses mainly on the development of the

components and services belonging to the higher-level of the OGSA architecture.

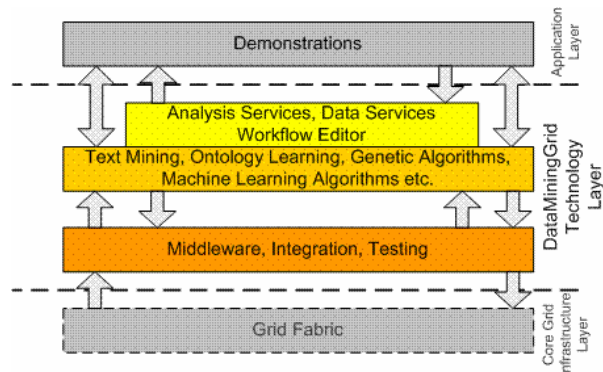


Figure 1: DataMiningGrid technologies

Analysis services and data services are also being developed, which make use of different data mining algorithms (located in the layer above them) in a distributed environment. The graphical workflow editor hides the complexity of the underlying Grid technology layer from the users, who are expected to be experts on data mining, but not on Grid computing. In doing so data miners can concentrate on defining data mining workflows appropriate to their problem, while all Grid related tasks (e.g. resource allocation, stage-in and stage-out of the required data and results) are carried out by the resource broker and underlying Grid system.

## 3. Generic use-case for the resource broker

The common usage scenario of the system is a combination of three main steps:

- Step 1: Workflow composition by the user
- Step 2: Execution of the workflow in the Grid
- Step 3: Retrieval and annotation of the results

*Step1:* Workflow generation is done by the user via a graphical user interface, namely the workflow editor. A data mining workflow may be large and complex, include conditional loops and branching as well as many other advanced features.

*Step2:* After the workflow has been created, it is submitted to the workflow manager for execution. The workflow manager interacts with the resource broker service, which is responsible for intelligent matching between the user's requests (jobs) and

available computation and data resources, as well as for later execution of jobs on the Grid. The resource broker service not only has to balance the load on the Grid nodes, but also has to make decisions sophisticated enough to satisfy a broad range of the requests. Many data mining algorithms pose very strict requirements regarding execution environments, data transfer policies, parallelisation constraints, free temporary and permanent storage necessities, etc. After the resource broker has successfully matched the jobs with the proper resources, the jobs are submitted to the selected machine or cluster.

*Step3:* After the execution has completed, the final results of the data mining operation are stored either in the Grid itself or on the user's client machine. These results may also be annotated with meta-data, e.g. for providing provenance information about the workflow by which these results were obtained. However, this step is independent of the implementation of the resource broker.

## 4. Requirements

Over the past years, data mining has been adopted by many diverse application areas, including biology, medicine, ecological modeling and customer relationship management. There are requirements that any system for performing data mining in the Grid has to satisfy in order to provide surplus value for data miners when compared to traditional data mining applications. The following list explains the key requirements:

- 1) Defining and performing data mining operations in a Grid must be user-friendly.

Professional data miners, while being experts in their own field, cannot be assumed to be experts on Grid technology. Due to the fact that Grid technology is just a new method for enabling data mining to them, as much Grid-related details as possible have to be hidden from these users.

- 2) Data mining operations must be executed on suitable machines in the Grid without direct user interaction.

While usually only a few computational resources in a Grid serve a specific purpose (e.g. data base servers, central registries), many machines simply share their computing power. However, the execution of any data mining operation on such a machine contains three subtasks:

- a) A list of resources needs to be compiled.

- b) The most suitable machine for executing the operation can be selected.

- c) The data and the data mining application have to be transferred to the selected machine for execution.

However, Grids are dynamic and possibly fast-changing distributed systems, based on virtualization of a large number of heterogeneous resources. Therefore, the user has no easy means of compiling a reasonably complete list of these resources. Yet, even if the user could provide such a resource list, it would probably contain only a small fraction of all available resources. This again may lead to reduced performance and an increase of idle time while waiting for the results of the operation. Furthermore, even if such a complete list were available, matching the individual machines' capabilities with the requirements of the data mining operation would be a difficult and error-prone task. Finally, transferring the required data and application to the selected machine is a tedious operation and depends on the knowledge of the transport mechanisms provided by the Grid middleware.

For the above reasons, a resource broker must address the following requirements:

- a) Automated compilation of a reasonably complete list of resources currently available
- b) Automated selection of the most suitable execution machine from this list
- c) Interaction with the Grid middleware in order to transfer the data and the application to the selected machine.

The broker has to carry out these operations without direct user interaction.

- 3) The Grid middleware must provide mechanisms for monitoring jobs.

One of the basic features of job execution in general, and on the Grid in particular – is the monitoring job state capability. Users must be able to retrieve information about the current job's execution status from the time it is submitted until the time when the user receives the computational results. This also includes as much additional information as possible (e.g. error logs, percentage of completion, etc). As a resource broker should perform all matching and submission activities in the system, it is clear the broker is the only component, which has all the information needed to provide sophisticated monitoring functionality.

- 4) Any resource broker must be extensible without forcing all users to install subsequent updates.

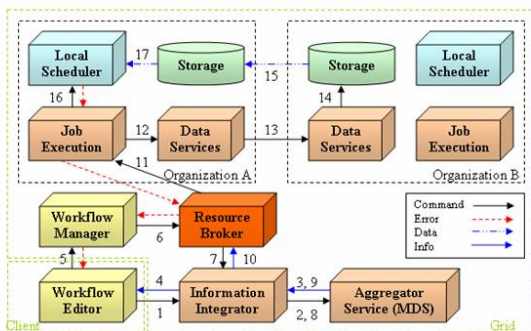
Resource brokering in a heterogeneous, fast-changing, distributed environment is a non-trivial task. Therefore, it is reasonable to assume that any implementation of a resource broker will pass through many development cycles. In the process, the broker's functionality and performance may be enhanced. Additionally, new emerging Grid technology and middleware may require revising the broker. However, acquiring and installing subsequent updates is a tedious task for the user to do and usually results in bad customer satisfaction.

- 5) All services must be compatible with existing Grid standards.

Even in the fast changing area of Grid technology adherence to standards and de-facto standards used by large parts of the community is crucial for building future-proof and accepted systems.

## 5. The job submission process

The system developed in the DataMiningGrid project is based on the most recent version of the Globus Toolkit. Figure 2 illustrates the interaction of services and components during composition of the workflow and job submission, which correspond to steps one and two mentioned in section 3. The figure does not describe step three, retrieval and annotation of results, as this step does not depend on the resource broker itself.



**Figure 2: Interaction of services during job submission in the DataMiningGrid system**

The process of job submission starts at the workflow editor, the only component to be installed on the client machine. This step of composing the workflow involves querying the Information Integrator service framework for available Data Services and Analysis Services (1-4). While the user composes the

workflow, the system automatically creates a job description in the background according to the user's choices regarding the data and applications he wishes to use. Thereby, the physical location of the data and applications incorporated in the workflow is entered automatically into the job description without direct interaction from the user. During this process the user may specify hundreds of simultaneous executions (e.g. parameter sweep). However, this job description does not specify the machines to execute the jobs on. After the workflow has been compiled it is submitted to the workflow manager for execution (5). The workflow manager resides on a dedicated machine in the Grid and is responsible for executing the workflow in the correct order. During execution of the workflow, the workflow manager automatically passes all job descriptions to the resource broker (6). The broker chooses one or multiple optimal execution machines based on the information about application requirements, data sizes, data permissions and workload on individual execution machines in the Grid. It retrieves this information partially from the job descriptions themselves and partially from the Information Integrator framework (7-10). According to the chosen execution machine the broker completes the job description regarding the stage-in and stage-out of input and output files. It finally passes the descriptions on to the job execution service from Globus running on the selected execution machine (11). The job execution service contacts the data services on the respective machine, which use GridFTP [15] to transfer the files before and after execution and executes the actual application (12-17). This also includes the respective application executables themselves. If a selected machine is residing in a cluster, the scheduler from that cluster is contacted, which eventually schedules the execution. During the whole process of composing the workflow and executing it in the Grid, both the physical location of the data and applications incorporated in the workflow, as well as the operations performed for resource discovery and scheduling are transparent to the user.

## 6. The resource broker

### The GridBus resource broker

Unfortunately, the Globus Toolkit does not provide a resource broker that is capable of scheduling jobs over sites from different organizations (A and B), as displayed in Figure 2. We selected the GridBus Grid

Service Broker and Scheduler v.2.4 [16] to fulfill this task for the following reasons:

- The GridBus resource broker is capable of submitting jobs to Globus' execution subsystem as well as to many other computational resources (e.g. Alchemi, Unicore, XGrid, and others). To achieve this purpose the broker translates the job description it receives into the format of the specific middleware controlling the selected resource. The translated job description also contains information for the Grid middleware regarding transfer of the required data and application. The transfer operation itself is carried out without any direct user interaction (requirement 2).
- The GridBus broker's architecture is clearly structured and well designed from a software engineering point of view.
- Unlike many other resource brokers, this one does not require any particular information or security system. It is designed as a stand-alone piece of software, which can be integrated with various existent components.

However, despite being the most reasonable choice, a careful investigation of the resource broker also revealed several drawbacks in its design, which prevented us from using the broker in its original state.

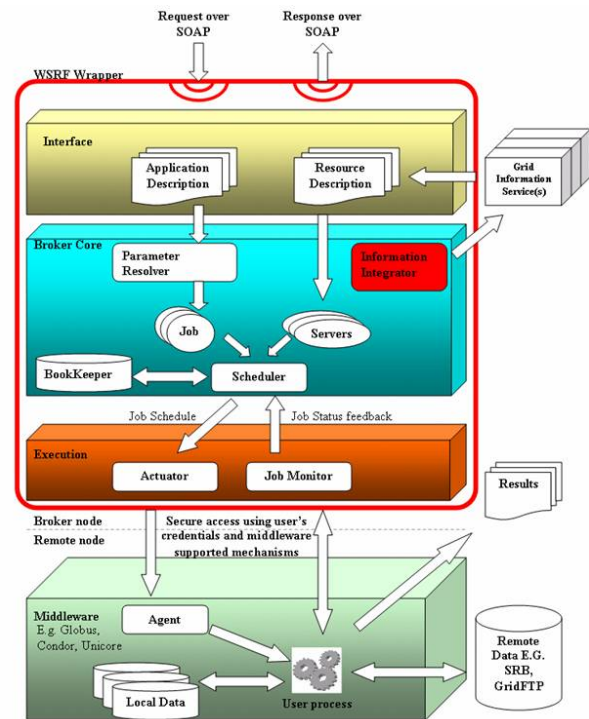
1. The broker was not service based, but needed to be installed on every client machine.
2. The broker did not provide mechanisms for automated compilation of a list of the computational resources currently present in the Grid, but required the user to do so. This list should be compiled from the Monitoring and Discovery System 4 [17] (MDS4), which is the implementation of an information system in the Globus Toolkit.
3. The broker was unable to query MDS4 for obtaining the status of the resources contained in the list.

### Internal design of the resource broker

Figure 3 illustrates the internal design of the resource broker including the modifications (in red) described in this paper. The design of the broker is composed of three main sub-systems:

- The application interface sub-system

- The core-sub-system
- The execution sub-system



**Figure 3: Enhanced GridBus resource broker architecture**

The interface layer presents the inputs of the broker. There are two main inputs:

- A) An application description list, which corresponds with the job description created during composition of the workflow, and represents the user request (job) that needs to be executed.
- B) A resource description list, which represents the available Grid resources for executing the job.

The application description list (A) together with the delegated user's credentials is received from the client through the Web service interface of the broker. The list containing resource descriptions (B) is compiled for each execution, based on the data from the Information System. The second list represents the current state of the available computational resources in the system.

In the broker core layer the above inputs are converted into "jobs" and "servers" – where "job" is the abstraction for a unit of work that will be assigned to some computational node, abstracted by "server". Once the jobs are prepared and the servers are discovered, the scheduler is started. The scheduler maps jobs (i.e. submits jobs using the

actuator component in the execution sub-system) to suitable servers based on its algorithm. The actuator is a middleware specific component, which dispatches the job to the remote Grid node. On receiving a job submission, each server uses its associated server-manager to actuate the middle-ware specific job-submitters (also known as Agents). The job-monitor updates the book-keeper by periodically monitoring the jobs using the services of the execution sub-system. As the job gets completed, the agent takes care of cleaning up and gathering the output of the jobs. The scheduler stops after all the jobs have been scheduled. The scheduling policy determines whether failed jobs are restarted or ignored [16].

### **WSRF wrapping**

The original version of the GridBus resource broker (2.4) comes as a stand-alone application that has to be installed on every client that wishes to use its functionality. While this original design may be feasible if interfacing with Grid middleware not based on a service-based architecture, it is highly undesirable in service-based systems. In order to fit it into a service-oriented architecture we wrapped it up as a WSRF-compliant service, exposing its main features through a predefined service interface. This modification provides three key advantages:

1. The broker's service interface adheres to the WSRF specifications. WSRF is currently the de-facto standard in the field of Grid services. Although its specifications have been submitted to the standard bodies of OASIS, it is not yet an adopted standard. Nevertheless, WSRF has been widely adopted by the Grid community already. For these reasons, the WSRF wrapper raises the acceptance of the resource broker and makes it more future-proof (requirement 5).
2. It is now possible to change the broker's internals and to introduce new functionality such as more sophisticated schedulers without forcing each user to download and install a new version of the broker as long as the service interface does not change. This facilitates short development cycles with many possible updates, without resulting in discomfort for the users (requirement 4).
3. As the broker is implemented in Java, it already provides a certain degree of platform independence. However, wrapping it as a service also provides language independency, thus enabling any application to use the broker,

regardless of the application's programming language, as long as it is capable of interfacing with Web services. This is a crucial advantage for later exploitation of the individual packages resulting from the DataMiningGrid project.

4. According to the development team from GridBus a WSRF-compliant version of its broker, which will provide the full set of features through a Web interface is scheduled. Thus, exposing the broker's main functions such as job scheduling and monitoring through our own service at this time reduces the risk of incompatibilities of future versions of the broker with the rest of the Grid system that performs the job submission process explained in section 5.

### **Introduction of the Information Integrator Service Framework**

In the context of the resource broker the role of the Information Integrator service framework developed in the DataMiningGrid project, is to provide the following information about computational resources:

- All computational resources, currently available in the Grid, from which the scheduler can choose.
- The capabilities of these resources regarding number of CPUs, main memory, etc.
- The current load of these resources.

The broker requires these pieces of information for choosing a suitable resource for executing the job described in the job description without any interaction by the user.

The framework automatically queries the information services provided by the Grid middleware (MDS4) and compiles a list of the resource entries maintained there. This approach results in two key advantages:

1. No user interaction is required for discovering available resources. By querying MDS4, which contains entries for each computational resource, we automate this process (requirement 2). As a result, the overall system becomes more user-friendly (requirement 1).
2. The original design of the broker allows to include different kinds of schedulers (e.g. cost-based, round robin, own developments, etc.). However, in some scenarios requirements may

arise that are independent of the actual scheduler, for example certain data may only be processed on machines belonging to certain organizations. For this purpose, it is reasonable to exclude all other resources independent of the applied scheduler. As the Information Integrator framework automatically compiles a list of available resources, which the scheduler may choose, it can perform a preliminary step of filtering the list of resources before the actual scheduler is called (requirement 2). Thus, any subsequent scheduler applied may remain as is and can be exchanged with a different one without any modifications.

## 7. Summary and outlook

This paper discussed parts of the high-level architecture of a service-based grid system for compute and data intensive tasks, with a detailed description of the implementation of the resource broker. The architecture as it is presented in this paper is tailored to meet the requirements arising from diverse data mining scenarios. In particular, the integration of the broker with the Information Integrator service framework, developed in the DataMiningGrid project, provides added value regarding user-friendliness by automatic discovery of available resources as well as automated collection of the information about the job requirements, needed by the broker. Furthermore, wrapping the broker as a WSRF-compliant service ensures its compatibility with the most recent de-facto standard in grid computing, while enabling developers to add more functionality without forcing users to download, and to install any updates. It also eases the integration on sophisticated monitoring mechanisms. The system is currently being tested in the DataMiningGrid test-bed [18] and a pre-release of the DataMiningGrid software is due by the end of August 2006.

## 8. References

- [1] I. Foster, C. Kesselman, *The GRID*, Morgan Kaufmann Publishers, Inc., San Francisco, 1999.
- [2] C. Shearer, “*The CRISP-DM Model: The New Blueprint for Data Mining*”, in *Journal of Data Warehousing*, 13-22, Vol. 5(4), Fall 2000.
- [3] V. Stankovski, M. May, J. Franke, A. Schuster, D. McCourt, and W. Dubitzky, “*A Service-Centric Perspective for Data Mining in Complex Problem Solving Environments*”, in PDPTA, Las Vegas, USA, 2004.
- [4] Community Scheduling Framework, online at [http://www.globus.org/Grid\\_software/computation/csf.php](http://www.globus.org/Grid_software/computation/csf.php)
- [5] GridWay open source meta-scheduler, online at [http://www.globus.org/Grid\\_software/computation/Gridway.php](http://www.globus.org/Grid_software/computation/Gridway.php)
- [6] Globus Alliance, online at <http://www.globus.org/>
- [7] GridLab Resource Management System, online at <http://www.Gridlab.org/WorkPackages/wp-9/>
- [8] Nimrod/G online at <http://www.csse.monash.edu.au/~davida/nimrod/nimrodg.htm>
- [9] GridBus Service Broker, a Grid scheduler for computational and data Grids, online at <http://www.Gridbus.org/broker/>
- [10] The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. I. Foster, C. Kesselman, J. Nick, S. Tuecke, 2002, online at <http://www.globus.org/alliance/publications/papers/ogsa.pdf>
- [11] Globus Toolkit 3, online documentation at <http://www-unix.globus.org/toolkit/docs/3.2/>
- [12] I. Foster, Globus Toolkit Version 4: Software for Service-Oriented Systems, IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779, pp 2-13, 2005
- [13] Tim Banks, “*Web Services Resource Framework – Primer*”, OASIS committee Draft 01 – December, 7 2005
- [14] K. Czajkowski, D. F. Ferguson, I. Foster, J. Frey, S. Graham, I. Sedukhin, D. Snelling, S. Tuecke, W. Vambenepe. “*The WS-Resource Framework*” March 5, 2004.
- [15] B. Allcock, J. Bester, J. Bresnahan, A. L. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnal, S. Tuecke, Data Management and Transfer in High Performance Computational Grid Environments *Parallel Computing Journal*, Vol. 28 (5), pp. 749-771, 2002
- [16] K. Nadiminti, S. Venugopal, H. Gibbins, T. Ma and R. Buyya, “*The GridBus Grid Service Broker and Scheduler*”, online at <http://www.Gridbus.org/broker/2.4/manualv2.4.pdf>
- [17] GT 4.0: Information Services, online at <http://www.globus.org/toolkit/docs/4.0/info/>
- [18] DataMiningGrid Consortium, “*Data Mining Tools and Services for Grid Computing Environments (DataMiningGrid)*”, research grant proposal submitted under EU IST FP6, IST- 004475 - Specific Targeted Research Projects, October 2003. (<http://www.dataminingGrid.org>)