BROKERING SOLUTIONS FOR GRID MIDDLEWARES

Attila Kertész
PhD student
Software Engineering Department
University of Szeged

Abstract

The Grid society has gone through some generations and there are some well known and widely used architectures. The latest middlewares are under careful development taking into account the advanced techniques. Brokering is a useful and even more necessary service of grids. Researchers have started to extend the previous middlewares with this function. The Globus Toolkit [8] is the most widespread middleware among grids, which still needs this improvement. GTbroker is a proper solution for this toolkit to provide automated job submission for the users. The EGEE project [4] is developing a new middleware to stand all demands of grid users, which has some common features with the GTbroker and Globus combination.

Key words: brokers, Globus grid, grid middlewares

1. Introduction

Several grid middleware solutions have been developed up till now. The goal of these efforts was to provide a new way for distributed computing, and to help the cooperation between scientists in a wider area or even through continents. Developers created environments, platforms for users to deploy applications to the grid, or just to execute individual or parallel tasks. The area of grids are getting more and more attention these days. Not only computer scientists, but also people from other scientific fields started to deal with this topic, because using the grid resources makes scientific development and research faster and produces better results. As the number of the users are growing and grid services are starting to become commercial, resource brokers are needed to free the users from the cumbrous work of job handling. Though most of the existing grid middlewares give the opportunity to choose the environment for the user's task to run, but originally they are lack of such a tool that automates the discovery and selection. Brokers meant to solve this problem. When developers realized this gap and they started to create this necessary component. This paper introduces a way how to fill such gaps and investigates how the latest middlewares handle this problem.

2. Related work

This area is well studied, several projects aimed at developing brokers. Some solutions just provided an easier method for submitting jobs over a grid, but in several cases other facilities were built in the broker such as graphical user interfaces, load balancing or grid monitoring. The Nimrod-G resource broker [3] is a broker for managing, steering and executing task farming applications on the grid based on deadline and computational economy. It is based on users' QoS requirements, dynamically leases services at runtime depending on their quality, cost and availability. It provides a graphical interface for resource discovery, resource trading and scheduling. Automatic resource discovery is not provided. It uses the main Globus services. The Gridbus broker [14] extends the Nimrod-G model to distributed data-oriented grids. Nimrod-G specializes in parameter-sweep computation and the scheduling approach aims at optimizing user-supplied parameters for computational jobs only. The Gridbus broker extends Nimrod-G parametric modelling language by supporting dynamic parameters, whose values are determined at runtime. It focuses on a resource scheduling strategy, and concentrates on adaptive scheduling algorithms and brokering for heterogeneous resources that are shared by multiple user jobs. It supports the deployment of distributed data-intensive applications. The AppLeS Project [2] supports deployment of parameter-sweep applications on computational grids, but its adaptive scheduling algorithm focuses on data-reuse. Users can identify common data files required by all jobs and the scheduling algorithm replicates these data files from the user node to computational nodes. It tries to reuse the replicated data to minimize the data transmission when multiple jobs are assigned to the same resource. The EZ-Grid project [7] is similar to my solution. It is built upon Globus, developed in Java and has a graphical user interface. It aims at efficient job execution and controlled resource sharing. The interaction with Globus is achieved through CoG Kit libraries. It consists of several components: an EZ-Grid Client, an EZ-Grid Server and a Register, which is an index server of the registered hosts in the grid.
These solutions may require other tools to run and the user usually needs to modify its configuration or even additional software needs to be installed to the grid middleware. This paper introduces GTbroker, a broker for the Globus Toolkit, which uses only the APIs and services provided by the toolkit, performs automatic resource discovery and job submission. The latest middlewares originally offer brokering services. The greatest project in this field in the European region is the EGEE project (Enabling Grids for E-sciencE). The middleware called gLite [5] has many similarities to the Globus Toolkit, and the way GTbroker performs job handling is close to its brokering methodology.

3. Globus tools

The Globus Toolkit is the most widespread middleware among grids. Originally there is no resource broker given with this software. The toolkit provides several

services for job submission: GRAM (Globus Resource Allocation Manager) for job submission and management, GASS (Global Access to Secondary Storage) for remote file management, MDS (Monitoring and Discovery Service) for storing information from the resources, the underlying GSI (Grid Security Infrastructure) - a PKI authentication and encryption for security. Finally the RSL [13] (Resource Specification Language) for specifying job requirements. This provides a common interchange language to describe resources. The core of the RSL syntax is the relation. Relations associate an attribute name with a value (’executable = myjob’ means that the ’myjob’ executable file has to be run). Globus has two main commands for job submission: the globus-job-run and the globusrun. Using them one can submit jobs on grid, but exact arguments need to be added such as resource manager contact, job parameters, etc. Running these commands with special parameters one can get detailed information about the states of the resources connected to the grid and the submitted jobs. Most of the time the user needs to know up-to-date information about the grid to use these commands for job submission, which makes this kind of task execution cumbrous and time consuming. It is obvious that a resource broker could make this toolkit more usable to many users. GTbroker is a solution that makes anyone able to specify easily the environment needed for its job with RSL, and the broker uses these datas to find automatically a suitable host on the grid.

4. GTbroker and Globus

There is an abstract architecture for brokers, and every broker has about the same role [1] within grids. A broker has to keep in touch with the user, with an information service of the grid and with the environments, on which jobs can run. Brokers should provide a layer of abstraction, which enables the user to submit jobs on a grid easily; therefore one does not have to bother with the direct interaction with resources. Therefore this component creates a link between the user and the grid resources. The following figure shows the place of GTbroker in Globus:

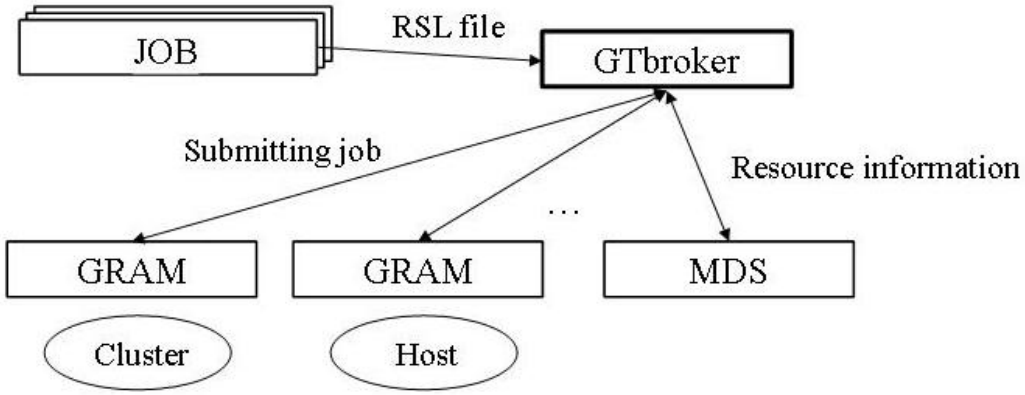

Fig. 1
GTbroker in Globus

GTbroker was developed in C++, it uses GT2 C API [9] functions to perform interaction with Globus and job submission. For determining the available hosts in the grid it queries the MDS with LDAP requests [11]. The job submission to resources is done through GRAM, and a GASS server is used to put the files needed for the job to the remote host and to get back the result files if there are any. Job requirements have to be specified in a simple RSL file. It must include at least the executable of the job. In addition to the main arguments a minimum hard disk requirement can also be specified for Gtbroker, in order to prevent improper job submissions. The main reason for developing this broker was to provide a way to submit up to hundreds of small jobs (taking between half a minute and a few minutes to run) as fast as possible, without any failures. Earlier the method for running these jobs was selecting a cluster and submitting all the jobs there. The makespan took much time, because the jobs that could not get a resource to run had to wait for the others to finish. GTbroker acts in the following way: First it reads the given RSL file to get the user requirements and job properties. Then it turns to the information service of the grid to get information form the resources. After getting the available nodes from the MDS the broker orders them by a predefined criteria, therefore the best hosts can get higher priority during the submission. This criteria can be modified: this method is used for scheduling. In the criteria one can use the following metrics: CPU speed, number of CPUs, free CPUs on the node, disk size and whether a node is a cluster. With these metrics the hosts can be ordered in a way that the ones having the best resources get higher priority than the others. Furthermore if needed, the broker can be extended to use all the information available in MDS. But using MDS in scheduling can cause some problems. Usually the refreshing of this information service is slower than it is needed depending on the load of the nodes (the free CPUs). Therefore the GTbroker uses an additional method to get more up-to-date information: it checks the actually selected cluster, whether there are enough resources for the job, and if there are not, it skips the cluster. To do this it uses the ’pbsnodes’ command of the PBS clusters. This provides extra information about the nodes in the cluster (such as the number of nodes and free CPUs on the nodes). This query is solved with a simple fork job sent directly to the cluster. The result file is staged back for processing. With these methods it automatically finds the best resources and submits only jobs that can actually run. Should a job fail or be pending for too long on a node, the broker cancels and resubmits it to another host. The actual state of the jobs is tracked by the broker, that’s why it is possible to cancel and resubmit jobs. Not only can the broker submit jobs, but also can give special information of the state of the grid and accepts additional user requirements during the execution. These functions can be activated by parameters. GTbroker was designed to handle parameters like globusrun and globus-job-run commands do. RSL parsing, dryrun, quiet mode and output-enabled execution also handled, furthermore GASS server usage and scheduling parameters can be defined in the latest version. Besides job submission, when a user does not want to execute jobs just to get information about the grid, the broker can list the whole or parts of the MDS. As we can see GTbroker enables

brokering in Globus based grids providing a similar interface for job submission as the conventional commands.

4.1. Evaluation

The GTbroker was developed at the Johannes Kepler University, Linz. The evaluation of the broker was performed on nodes connected to the Austrian Grid. For the tests I could use 3 clusters with PBS (hydra, altix1, schafberg) and 2 hosts (hydra, altix1). One of the clusters had 64 nodes and the others had 16. The middleware is GT version 3.9.5, MDS version 2. The following figure shows the results:

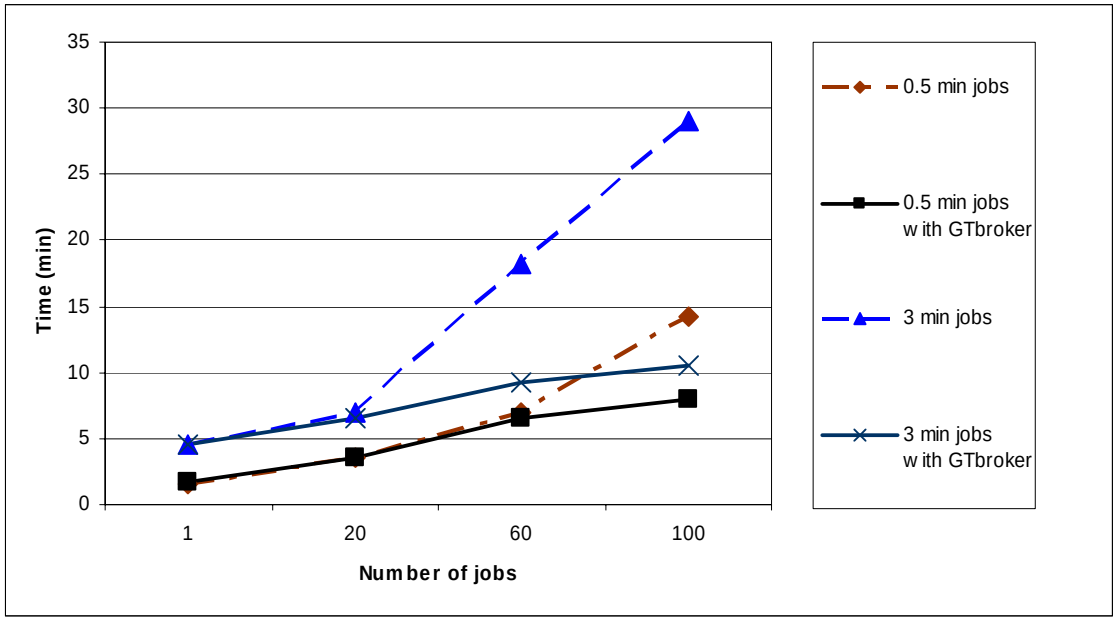

Fig. 2
Diagram of the evaluation

Running a few number of jobs with the broker does not give any speed up, because gathering extra information about the hosts and tracking the jobs take some time. But above a certain number of jobs (about 20) the makespan is shorter. The running time of the jobs also affects the performance: the longer the jobs the more time the broker can save (without brokering the pending jobs have to wait on the same cluster for the others to finish). So the speed up depends mainly on three factors: the number of the jobs, the running time of the jobs and the load on the grid. There was only minimal load on the nodes during evaluation. When plenty of jobs are running on the grid and the load is heavier, the broker could perform better, because more jobs could be pending or fail.

5. The new middleware: gLite

The EU funded EGEE project has been established to build a service grid infrastructure in Europe based on the recent achievements in grid technology. The new middleware called gLite is under development within the frameworks of the EGEE project. The WMS [6] (Workload Manager Service) is responsible for accepting requests for job submission and management. They have broaden the meaning and the scope of the user tasks called jobs, because different kind of tasks require different environments and scheduling. The core component is the WM (Workload Manager), which handles brokering as GTbroker does in Globus grids. Job requirements are passed over in classad-based JDL (Job Description Language) files, whish is very similar to the RSL files. Dags can also described in the JDL and these dependent tasks are handled in a special way to observe the dependencies. They formed job-tracking into a service called LB (Logging and Bookkeeping). During submission a bookkeeping server is assigned to the job and processes the incoming events by logging the states of the jobs and some more information such as destination CE (Computing Element) name, job exit code, etc. All the services offer web service interface, which is an advanced communication method among grid services. For storing resource information the ISM (Information Supermarket) is used like MDS in Globus. The WM has a special improvement: if there is no resource for the job or the job has failed it still keeps the job for a while and store it in a TQ (Task Queue). Whenever the ISM or a CE signals to have free resources the job can be resubmitted. GTbroker also uses a method similar to this, but it does not have a queue for these jobs just resubmits them to the node with the highest priority. There is a new significant idea in job submission they use: the CEs have the ability to signal and ask for jobs when they have free capacity. Various scheduling mechanisms are available during the submissions based on these two new models: on the push and pull models. The push model is the one the average brokers use: they gathering information about the free resources, and push the jobs to the free ones. The pull model is a new approach, when a CE is asking the WMS for any or a special job to be received.

6. Conclusion

It is evident that brokers play an important role in the middlewares of grids. Previous versions need to be extended to provide brokering services to users, and the new ones need to be designed with this function. In later versions the greatest difficulty is that the brokers have to adopt the architecture of the middleware and find the communication points of the existing parts. In most cases the new guidelines and trends cannot be used. But these efforts are not done in vain, because there still exist such grid architectures and they are used by many users. We could see that using GTbroker for job submission in Globus grids is a working and effective solution.. The user does not need to know on which host will the jobs run; only simple RSL files have to be provided to the broker. It selects the best resources for the specified jobs, orders them by the user’s criteria and submits them to the best

available host handling job pending, failures and file staging. No additional software is needed to run the broker. gLite is a new middleware that was designed to handle brokering by taking into account the needs of the recent grid computing and developing trends.
The future work aims at extending the usability of GTbroker. The P-GRADE Grid Portal [12] is a workflow oriented portal, which provides a smart and easy way for the users for job submission by supporting several grid platforms. The cooperation of this portal and GTbroker can achieve greater performance in Globus grids, and could attract more users to use the grid services.

7. Acknowledgements

It was a pleasure working with the people in GUP, JKU, Linz. First of all I would like to thank Peter Praxmarer for his useful ideas and help during developing this broker. I am also grateful to my supervisor, Prof. Dr. Péter Kacsuk, for guiding me in the right way during my studies.

8. References

[1] AFGAN,E.: **Role of the Resource Broker in the Grid**. Proceedings of the 42nd annual Southeast regional conference, 2004.

[2] BERMAN,F., and WOLSKI,R.: **The AppLeS Project: A Status Report**. Berlin, Germany, Proceedings of the 8th NEC Research Symposium, 1997.

[3] BUYYA,R., ABRAMSON,D., GIDDY,J.: **Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid.** USA, The 4th International Conference on High Performance Computing in Asia-Pacific Region (HPC Asia 2000), IEEE Computer Society Press, 2000.

[4] **EGEE Project homepage,** http://www.eu-egee.org

[5] **EGEE Middleware Architecture,** https://edms.cern.ch/document/476451

[6] **EGEE User's Guide, WMS Service,** https://edms.cern.ch/document/572489

[7] **EZ-Grid Project**: resource brokerage for multi-site computing, http://www.cs.uh.edu/~ezgrid

[8] **Globus homepage,** http://www.globus.org/toolkit

[9] **Globus API Pages**, http://www-fp.globus.org/api.html

[10] IBM REDBOOKS: **Enabling Applications for Grid Computing with Globus.** IBM Corp., 2003.

[11] **OpenLDAP API,** http://www.openldap.org

[12] **P-GRADE Graphical Parallel Program Development Environment,** http://www.lpds.sztaki.hu/pgrade

[13] **RSL Specification 1.0,** http://www-fp.globus.org/gram/rsl_spec1.html

[14] VENUGOPAL,S., BUYYA,R., WINTON,L.: **A Grid Service Broker for Scheduling Distributed Data-Oriented Applications on Global Grids**. Proceedings of the 2nd workshop on Middleware for grid computing**,** 2004.