GES 2007 German e-Science

Available online at http://www.ges2007.de

Reservation-based Resource-Brokering for Grid Computing

Janin Jeske, André Luckow and Bettina Schnor

Operating Systems and Distributed Systems
Institute of Computer Science
August-Bebel-Strasse 89, 14482 Potsdam
email: {jeske, luckow, schnor}@cs.uni-potsdam.de
phone: (+49) 331 977-3120, *fax:* (+49) 331 977-3122

Abstract

In this paper we present the design and implementation of the Migol brokering framework. Migol [10, 7] is a Grid middleware, which addresses the fault-tolerance of long-running and compute-intensive applications. The framework supports e. g. the automatic and transparent recovery respectively the migration of applications. Another core feature of Migol is the discovery, selection, and allocation of resources using advance reservation.

Grid broker systems can significantly benefit from advance reservation. With advance reservation brokers and users can obtain execution guarantees from local resource management systems (LRM) without requiring detailed knowledge of current and future workloads or of the resource owner's policies.

Migol's Advance Reservation Service (ARS) provides an adapter layer for reservation capabilities of different LRMs, which is currently not provided by existing Grid middleware platforms. Further, we propose a shortest expected delay (SED) strategy for scheduling of advance reservations within the Job Broker Service. SED needs information about the earliest start time of an application. This is currently not supported by LRMs. We added this feature for PBSPro.

Migol depends on Globus and its security infrastructure. Our performance experiments show the substantial overhead of this service-oriented approach.

1 Introduction

Contrary to cluster systems, a Grid consists of many independent, heterogeneous resources, which are part of different organizations. Resources in a Grid are managed by local resource management systems (LRMs) like PBS or LSF and on Grid-level by Grid brokers. A LRM enforces the access policies of the resource owner at a single site while a Grid broker manages resources across different organizations. Often, Grid brokers and LRMs have conflicting objectives: Grid brokers desire an optimal performance for an application while LRMs aim for an optimal throughput.

Due to site autonomy a Grid broker has no control of Grid resources, i. e. a broker has no influence on the actual start time of a job. Without local control and global knowledge about the Grid a Grid broker can only provide best effort services. Advance reservation enables users and Grid brokers to obtain execution guarantees from local resources. No detailed knowledge of future workloads or of the resource owner's policies is required.

Migol significantly benefits from the availability of advance reservation. With advance reservation fail allocations in a Grid can be minimized and unnecessary file transfers in case of a migration can be avoided. In addition, advance reservation is necessary to implement important Grid use cases: Co-allocation requires the co-reservation of all involved resources. The predictability of Grid workflows can be significantly enhanced using advance reservation.

To enable advance reservation in a Grid, this paper makes the following contributions:

- Common Grid toolkits do not provide a common access layer to advance reservation capabilities of local resource management systems. The Migol Advanced Reservation Service offers an adapter layer to access reservation features of different LRMs in a secure way.
- The Migol Job Broker Service was extended to provide an advance reservation-based meta-scheduling using a shortest expected delay (SED) metric.
- Existing LRMs lack support for querying possible earliest start times for an application. Migol provides a PBSPro plugin, which demonstrates how an existing LRM can be enhanced by this feature.
- In a first performance analysis the response times of the WS GRAM and the Migol services were evaluated. Detected bottlenecks are described and analyzed.

This paper is structured as follows: After a brief discussion of related work in section 2, the brokering architecture of Migol is explained in section 3. Section 4, 5, and 6 describe in detail the brokering components. In section 7 first experimental results that provide insight in the performance of the Migol brokering services are presented. We conclude this paper with an outline of future work.

2 Related Work

Advance reservation strategies and systems are currently under intensive research. Different LRMs, such as PBSPro [8], LSF [14], or a scheduler like Maui [9], have advance reservation capabilities. Unfortunately, these are not sufficient for Grid environments: intransparent workloads and allocation principles prohibit the estimation of a possible start time for an application. Further, PBSPro and LSF require that users possess a certain privilege to place reservations, which is not set by default. In addition, PBSPro grants reservations a higher priority than regular batch jobs. This approach significantly penalizes regular jobs. To ensure fairshare an administrator of a local resource must be able to specify fine granular reservation policies.

Röblitz and Rzadca [11] studied different algorithms for the placement of

reservation requests in job schedules. The described approaches mainly focus on the enabling of LRMs to provide advance reservation capability to Grid brokers.

Different Grid projects are working on the integration of advance reservation features. Viola/Unicore [13] supports the reservation, co-reservation and co-allocation of compute- and network resources. As part of the project the Meta-Scheduling Service (MSS) is currently under development. The MSS provides a WS-Agreement [3] based framework for the negotiation of service level agreements (SLA). In comparison to the Migol Job Broker Service, which implements different allocation procedures, the MSS provides only limited brokering functionality. Further, Globus compute sites can currently not be integrated.

The Globus WS GRAM is currently extended by advance reservations capabilities [2]. In contrast to the Migol Job Broker Service, the WS GRAM will only provide an adapter layer similar to the Migol Advance Reservation Service (ARS) (see section 5).

3 Migol's Brokering Architecture

Migol [10, 7] is a Grid middleware, which addresses the fault-tolerance of long-running and compute-intensive applications. Migol supports e. g. the automatic and transparent recovery respectively the migration of applications. A core feature of Migol is job brokering, i. e. the framework offers services for the discovery, selection, and allocation of resources. Migol is built upon the WSRF implementation of the Globus Toolkit 4 (GT4) [6]. Existing Globus services, e. g. the WS GRAM, are reused whenever possible.

Figure 1 shows an overview of the Migol brokering architecture. The Migol

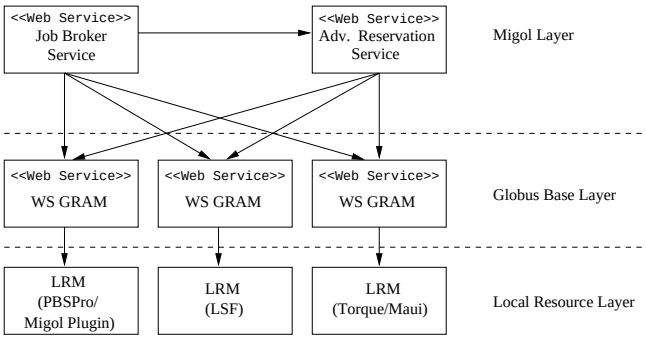

Figure 1: Migol Job Brokering Overview

Job Broker Service (JBS) provides an unified access to the computational resources of the Grid. It finds suitable resources for an application, deploys, and starts it. Before start of the application, the Job Broker stages the necessary libraries and data files automatically.

Different LRMs such as PBSPro, LSF and Maui are capable of creating and managing advance reservations. Unfortunately, each of these services has a different API with a different behavior. The handling of differences in the reservation API should not be part of a meta-scheduling service such as the JBS. Instead an adapter service should be used. The GT4 WS GRAM service [4] offers an adapter API for the creation, monitoring, and destruction of jobs across different LRMs. But, the WS GRAM currently lacks support for advance reservation. Therefore, Migol provides the Advance Reservation Service (ARS) a generic, WSRF-based reservation layer on top of different LRMs.

Further, the reservation negotiation process between Grid broker and local resource management systems is insufficient for Grid usage. Migol therefore extends PBSPro with the capability to accept queries for the earliest possible start time of an application. The following sections will explain in detail the Migol services which support advance reservation.

4 Migol PBSPro Plugin

Although many LRMs offer advance reservation, these capabilities are mostly not sufficient for Grid computing. For placement of a reservation at a LRM in general the specification of a start time and duration is necessary. Since a Grid broker is only able to obtain incomplete and fuzzy information from local resources, a prediction of future utilization is not possible. It would be beneficial if LRMs support queries for the earliest possible start time of a job. This functionality could significantly increase the reservation acceptance probability later in the reservation process.

The Migol PBSPro plugin demonstrates how an existing LRM can be extended to support queries for the earliest possible start time of an application. The PBSPro plugin estimates the start time using the application's resource requirements and current PBS statistics. Based on the expected start time a meta-scheduling service like the Migol Job Broker Service can rank reservation responses of multiple resources.

The plugin also enforces fairshare, i. e. it is ensured that reservations and normal batch jobs are equally prioritized. New reservations are only accepted if the start time is greater than the expected end time of all queued jobs.

5 Advance Reservation Service (ARS)

The Advance Reservation Service (ARS) offers APIs on top of different LRMs, such as PBSPro, LSF, and Maui, to create, monitor, and destroy reservations and to bind jobs to existing reservations. Using a plugin architecture and a configuration API the service can be easily extended to support other LRMs. For each user request a shell script, which conducts the respective LRM operations, e. g. calling `pbs_rsub` to place a PBSPro reservation, is generated by the respective plugin. The script is then executed using the WS GRAM (Fork) service of the resource. This approach can be generically applied to any

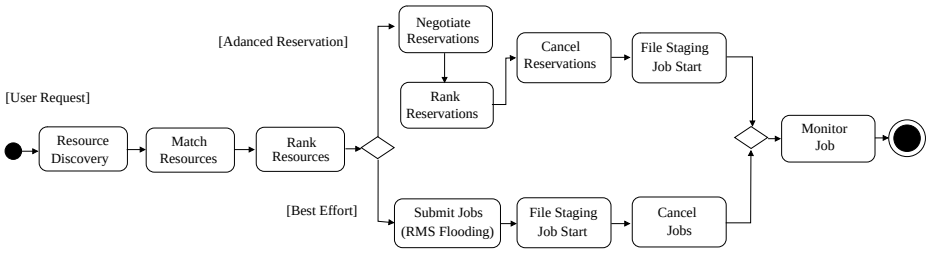

Figure 2: Grid Scheduling Activity Diagram

WS GRAM managed compute resource. The WS GRAM ensures that all LRM requests are securely executed using the privileges of the respective Grid user.

The output of a reservation request is an Endpoint Reference (EPR) representing the reservation resource, which encapsulates the reservation ID of the LRM. The reservation can be managed using this EPR. Since the WS GRAM is not aware of reservations it is not possible to bind GRAM jobs to an ARS reservation. Thus, the ARS must also be able to bind and to manage jobs. For this purpose, the same approach as for reservations is used: the ARS generates a shell script which binds a job to a reservation of the LRM. The same holds for job status queries or cancellations.

6 Job Broker Service (JBS)

The Job Broker Service (JBS) offers a unified access to resources managed by different LRMs. The JBS categorizes jobs into two categories: as default the JBS tries to reserve all required resources for a job. If reservations are not available on all resources, the JBS can only provide best effort services.

Figure 2 shows the submission process for a job. After submission the JBS performs a resource discovery by querying the WS MDS, which returns different static and dynamic information about available Grid resources. The obtained resources are matched against the application requirements. In the next step all resource candidates are ranked.

6.1 Resource Ranking

Many brokers assume that 100 percent of the extremely fine grained information needed to find an optimal resource is available, always correct, and up-to-date. This is not very likely in a dynamic Grid. Since the JBS has no control over local resources, common performance and allocation metrics are likely to fail in a Grid. Therefore, the JBS only uses a simple heuristic to pre-select resources. The broker ranks all resources of the candidate list based on a simple metric, which comprises of different factors:

- **Run time:** The CPU clock rate of a resource is used as speed factor.

- **Waiting time:** The queue length at the LRM and the number of requested nodes per application is used as indicator for the waiting time respectively for the reservation acceptance probability.
- **Transfer time:** The transfer time of application binaries and checkpoint files is estimated based on the available bandwidths.

Because of the mentioned Grid properties, this metric is only a rough approximation. The JBS will not be able to obtain all global information necessary to make a precise forecast. For example, the waiting time of a job is, due to insufficient information and in-transparent allocation principles of the LRM, hardly predictable. Further, it is impossible to precisely forecast the run time of an application solely based on the CPU clock rate.

6.2 Resource Selection

Depending on the availability of advance reservation on all required resources, the JBS will either try to reserve resources using the ARS or it will use a selective flooding approach for job submission.

6.2.1 Reservation-based Brokering: Shortest Expected Delay

If all resources support advance reservation, the candidate list obtained by the ranking metric is now used to send reservation requests to the top three sites using the ARS. In the case of PBSPro the Migol plugin is used to estimate the earliest possible placement for the application. Using the returned start time all accepted requests can be compared based on the shortest expected delay [12]. For the top resource the broker commits the reservation. All other reservations are cancelled. The JBS then plans and executes the transfer of all necessary files. The job is then submitted using the assigned reservation ID (respectively the EPR) through the ARS.

6.2.2 Selective Flooding

In the case of best effort jobs Migol uses a very simple but practical approach to minimize the waiting time. The job is dispatched to the top three sites from the candidate list – we call this approach Resource Manager Selective Flooding (RMSF). When one of these jobs gets active, all other jobs are cancelled. This approach avoids orphan jobs and ensures that the overhead caused by an allocation of more than necessary resources is minimized.

There are a couple of disadvantages: For example, since the LRM decides when a job is started, no commitments on a start time can be given. Thus, it is not possible to plan e. g. the transfer of large files in case of a migration. Therefore, Migol relies on advance reservation whenever possible.

6.3 Job Management

After start of the job the JBS will return an EPR referencing the JBS resource. Using the JBS resource properties the local job endpoint of the WS

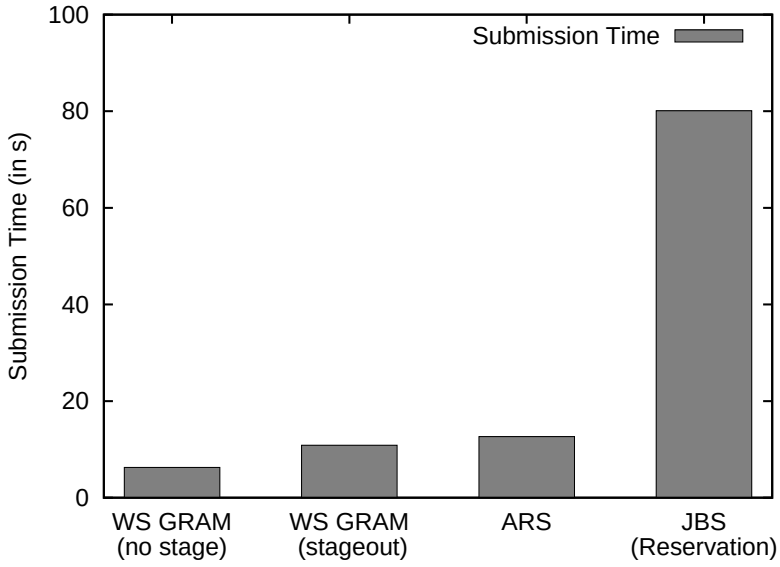

Figure 3: WS GRAM, ARS, and JBS Performance

GRAM respectively ARS job can be obtained. The job state can then be monitored using the WS GRAM respectively the ARS.

7 Experimental Results

We conducted a first performance analysis of the Job Broker and the Advance Reservation Service. For our experiments we used a prototype of the JBS 2.0, which lacks some features of the final version, e. g. WS MDS resource discovery. The measurements therefore represent a baseline.

The test bed comprises of two clusters. Each cluster consists of eight nodes and is managed by a WS GRAM and PBSPro on the front node. The front nodes are equipped with a Intel Xeon 2.66 GHz processor and 2 GB of RAM. The Migol services were deployed on a dedicated machine with AMD XP 2000 processor and 1 GB of RAM. The Globus Toolkit 4.0.4 (GT4) was installed in a Tomcat 5.5.23 container running on Linux. The Java Virtual Machine (JRE 5.0 SP 8) was configured with a heap space of 512 MB. As test application a simple 9-point stencil simulation, the cellular automaton [1], was used. We studied the mean response time of different Globus and Migol services using 20 iterations for each experiment.

In the first experiment we analyzed the submission times of the WS GRAM, ARS, and JBS. Figure 3 presents the results of this experiment. Since our implementation relies on Java we focused our investigation on the Java WS GRAM client. The performance of the WS GRAM has significantly improved

compared to the GT3 GRAM (see [10]). Without file staging the performance of the WS GRAM is comparable to the Pre-WS GRAM (see Feller et. al. [5] for details). In our setup we measured an average completion time of 6.3 s for a basic job without file staging.

Especially relevant for the ARS is the execution of a small job script including the stage out of the job result. The stage out even of a small output file (∼10 K) increases the job completion time by 4.6 s to 10.9 s in average. Staging seems to be due to the overhead of the Reliable File Transfer (RFT) service and the credential delegation process in particular for small files a very expensive operation. The execution of a reservation request using the ARS takes approximately 12.7 s and thus adds 1.8 s overhead to the WS GRAM. This is the normal overhead required for credential delegation and the invocation of multiple secure Web service operations.

The JBS heavily depends on the ARS performance: For each scheduling operation the JBS must conduct the following operations:

- Try to reserve resources at three sites using the ARS (3 ARS requests).
- Start job at the resource with shortest expected delay (1 ARS request).
- Cancel other reservations (2 ARS request).

That means in total at least 6 requests to the ARS are required. For each request a WS GRAM job must be executed. This explains the overhead of 74 s for a JBS submission, which is also much higher than the overhead we observed for a Resource Manager Selective Flooding (RMSF) submission in earlier experiments [10]. These experiments were conducted on a similar hardware. The JBS showed with RMSF response times of about 24 s. The RMSF procedure does not require as much WS GRAM interactions as the reservation-based approach. Further, the monitoring of the job state can be done directly using the WS GRAM – no execution of a job script is required.

Since the JBS is heavily depended on the base services ARS and WS GRAM, we also conducted a stress test using up to 16 concurrent users (Figure 4). In our environment the WS GRAM and ARS scale equally well with up to four concurrent clients. With more than four clients the response times increase almost linearly. A reason for this behavior is the saturated CPU on the front node running the WS GRAM, which showed CPU loads higher than 80 %. This high CPU load shows the high resource requirements that are associated with deployment of WSRF services such as the Globus WS GRAM and the Migol services.

8 Conclusions and Future Work

Common performance and allocation metrics, which are based solely on precise and complete information are likely to fail in a Grid. Since a Grid broker has no local control these metrics can only be seen as an indicator. With advance reservation a LRM can commit resources to a certain time and for a certain duration to a Grid broker.

To provide a reservation-based broker, support for advance reservation on

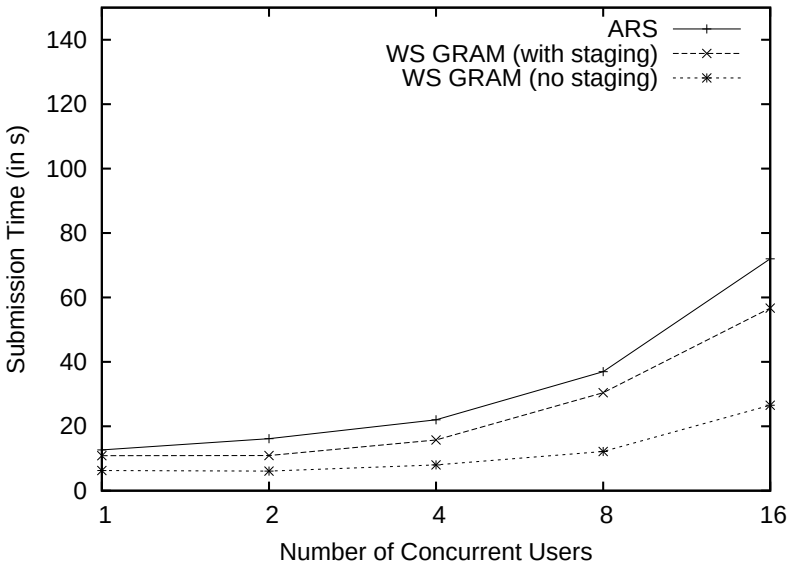

Figure 4: WS GRAM and ARS Performance under Load

Grid-level and LRM-level is required. With the Advance Reservation Service Migol provides an unified service API to advance reservation capabilities of different local LRMs.

The Migol Job Broker Service relies on advance reservation to find the shortest expected delay for an application. If advance reservation is not supported by all LRMs, a selective flooding mechanism is used.

The Migol PBSPro plugin enables the querying of the earliest possible start time for an application and therefore makes a meta-scheduling based on the shortest expected delay metric possible. Further, it ensures the fair allocation of resources to all jobs.

While our experiments show that the Advance Reservation Service (ARS) and the Job Broker Service (JBS) have substantial overhead, we think that this is acceptable compared to the higher level of assurance reservations offer. The JBS and ARS will be further optimized in the future e. g. by minimizing network calls with local service invocations and parallelizing ARS requests using multiple threads. Further, most operations can also be done asynchronously without blocking the user's machine.

Acknowledgements

We thank Altair for providing us access to the internals of PBSPro. Without their support the creation of the PBSPro plugin would not have been possible.

References

1. The Cellular Automaton. http://www.cs.uni-potsdam.de/bs/cellular/, 2006.
2. Globus Alliance. Globus Toolkit Advance Reservation Architecture. http://bugzilla.globus.org/globus/attachment.cgi?id=1108, 2006.
3. Alain Andrieux, Karl Czajkowski, Asit Dan, Kate Keahey, Heiko Ludwig, Toshiyuki Nakata, Jim Pruyne, John Rofrano, Steve Tuecke, and Ming Xu. Web Services Agreement Specification (WS-Agreement). https://forge.gridforum.org/sf/go/projects.graap-wg/, 2006.
4. Karl Czajkowski, Ian T. Foster, Nicholas T. Karonis, Carl Kesselman, Stuart Martin, Warren Smith, and Steven Tuecke. A Resource Management Architecture for Metacomputing Systems. In *IPPS/SPDP '98: Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing*, pages 62–82, London, UK, 1998. Springer-Verlag.
5. M. Feller, I. Foster, and S. Martin. GT4 GRAM: A Functionality and Performance Study. In *Proceedings of the Teragrid 2007 Conference*, Madison, WI, USA, 2007.
6. Ian Foster. Globus Toolkit Version 4: Software for Service-Oriented Systems. In *Proceedings of IFIP International Conference on Network and Parallel Computing*, pages 2–13. Springer-Verlag LNCS 3779, 2006.
7. Nicole Hallama, André Luckow, and Bettina Schnor. Grid Security for Fault Tolerant Grid Applications. In *ISCA 19th International Conference on Parallel and Distributed Computing Systems*, pages 76–83, San Francisco, USA, 2006.
8. Robert Henderson and Dave Tweten. Portable Batch System: External Reference Specification. Technical report, NASA Ames Research Center, 1996.
9. David Jackson, Quinn Snell, and Mark Clement. Core Algorithms of the Maui Scheduler. *Lecture Notes in Computer Science*, 2221:87, 2001.
10. André Luckow and Bettina Schnor. Migol: A Fault Tolerant Service Framework for MPI Applications in the Grid. *Future Generation Computer Systems – The International Journal of Grid Computing: Theory, Methods and Application*, to appear 2007.
11. Thomas Röblitz and Krzysztof Rzadca. On the Placement of Reservations into Job Schedules. In *Euro-Par 2006 Proceedings*, volume 4128, pages 198–210. Springer, 2006.
12. Bettina Schnor. *Scheduling and Migration Strategies for Parallel Applications on Distributed Systems*. Shaker Verlag, Aachen, 1999.
13. Achim Streit, Oliver Wäldrich, Philipp Wieder, and Wolfgang Ziegler. On Scheduling in UNICORE - Extending the Web Services Agreement based Resource Management Framework. In Gerhard R. Joubert, Wolfgang E. Nagel, Frans J. Peters, Oscar G. Plata, P. Tirado, and Emilio L. Zapata, editors, *PARCO*, volume 33 of *John von Neumann Institute for Computing Series*, pages 57–64. Central Institute for Applied Mathematics, Jülich, Germany, 2005.
14. S. Zhou. LSF: Load Sharing in Large-scale Heterogenous Distributed Systems. Workshop on Cluster Computing, 1992.