Removing the Need for State Dissemination in Grid Resource Brokering

Peer Hasselmeyer
NEC Laboratories Europe, IT Research Division
Rathausallee 10
53757 Sankt Augustin, Germany
hasselmeyer@it.neclab.eu

ABSTRACT

Resource brokering in Grids is nowadays handled by resource brokers that require detailed knowledge of the state of the resources that they broker. In business settings, surrendering internal information on resources to an outside party is not an option. The traditional resource brokering method based on intimate resource knowledge is therefore not a viable possibility.

This paper argues that the publication of resource state data is not needed for resource brokering. Instead, we advocate the use of service level agreements (SLAs). The use of SLAs for brokering lets providers keep state information internal and at the same time provides customers with guarantees on the used services.

The proposed model has been implemented in the form of a resource broker that is based on Web Service technology. It shows that the approach of using SLAs for scheduling is a possible solution to keeping resource state private.

Categories and Subject Descriptors

D.4.7 [**Organization and Design**]: Distributed systems; C.2.4 [**Distributed Systems**]: Distributed applications

General Terms

Algorithms, Design, Experimentation

Keywords

Service Level Agreements, Resource brokering, Grid, E-Commerce

1. INTRODUCTION

Scheduling jobs and, in more general terms, scheduling access to any kind of resource is a topic of major importance in Grids. Grid scheduling is defined as "the process of making scheduling decisions involving resources over multiple administrative domains" [12]. While scheduling is a tough problem in itself, the additional challenge in Grids is the integration of multiple administrative domains. Each service provider in a Grid constitutes its own administrative domain and acts autonomously. Each domain has its own policies on resource access and usage. As those policies are usually not made public, Grid resource schedulers can not know them. Making good scheduling decisions in such an environment is problematic at best.

Traditional schedulers (called *resource management systems (RMSs)* in the following) operate on local resources only. They have intimate knowledge of the resources and their state and have full control over the resources. Grid schedulers (called *resource brokers* in the following) are different from local RMSs. They are commonly operated by an organization separate from resource providers. They do therefore not have direct control of resources and lack complete knowledge of the resources' state. To exercise control over resources, resource brokers cooperate with RMSs on local resources as shown in figure 1. They are therefore bound to the information that RMSs provide as well as to the control that RMSs make available to them.

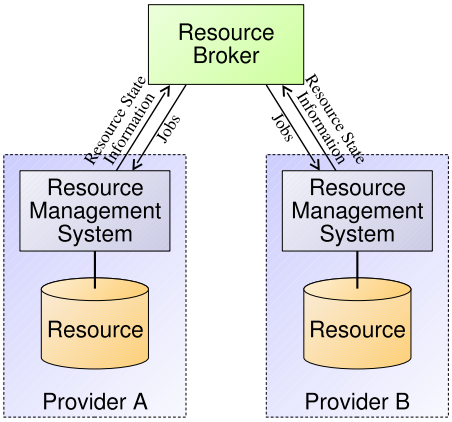

Figure 1: Interaction between Resource broker and RMS

A common solution for increasing the cooperation of resource brokers and local RMSs is the publication of current resource state information, as done for example in the European DataGrid project [7] and Condor [15]. That information is used by resource brokers to perform scheduling decisions. The decision could be based on the current state of the job queues. For example, the broker could submit jobs to the queue with the least waiting jobs. Another strategy is

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
MGC '07, November 26, 2007 Newport Beach, CA, USA
Copyright 2007 ACM 1-XXXXX-XXX-X/07/11 ...$5.00.

to build a history of resource utilization and job throughput and to predict future utilization or job start times based on these [11].

Although such resource brokers work reasonably well, they are mainly restricted to academic environments because of their need of resource status information. In business settings, service providers are reluctant to publish the state of their resources. For compute resources, the state commonly contains data on job queue lengths and the number of processors. Competitors as well as consumers would be interested in learning such data as it gives them insight into resource power, resource utilization, likelihood of successful service provisioning, and even into business processes. Providers therefore want to keep this information private. This is presumably the reason why Amazon's Elastic Compute Cloud [1] web service does not publish real-time utilization data of their resources.

The NextGRID project [13] is concerned with developing architecture for economically sustainable Grids. Within that project, we developed a model for Grid scheduling that does not rely on the direct surrender of information or control to brokers. Instead, it uses service level agreements (SLAs) to indirectly pass control on to resource brokers. SLAs are used to capture the requirements and constraints of service consumers and providers. Neither party is forced to provide sensitive data to business partners. All private information is kept locally at the owning entity. The only information revealed is the information contained in SLAs.

SLAs describe promises of the resource provider to his clients. In addition, SLAs usually state penalties that are to be paid by the provider to the consumer in case such a promise is broken. Besides the regular payment, the provider thus has an additional incentive to adhere to the promises made in an SLA. A resource broker having an SLA with a provider therefore has some level of control over the resources of the provider via that SLA. Although actual control is exerted by the local resource management system, an SLA "forces" that management system to obey the rules set out in that SLA. As the SLA reflects the needs of the broker and therefore the service consumer, the service consumer indirectly controls the resources via the broker and the local RMS.

The degree of control depends on how much the provider adheres to his promises in his SLAs. As mentioned above, the incentive to follow an SLA is the payment to be received upon successful provisioning of a service and the avoidance of a penalty fee that is due upon SLA violation. Depending on the values of the payment and the penalty, the provider's incentive to fulfil an SLA is larger or smaller.

To investigate the possibilities of SLAs as a method for resource allocation, the author of this paper developed a resource brokering service that bases all scheduling decisions on the outcome of SLA negotiation.

The paper starts in section 2 with a description of the model and the ideas behind our resource bokering system, focussing on SLAs and how they can be used for resource brokering. Section 3 details the architecture of the implementation of our SLA-based resource brokering service. Section 4 shows an example of how resource brokering with SLAs works. Related work is discussed in section 5 and some conclusions are presented in section 6.

2. MODEL

This chapter introduces the model on which our solution is based.

2.1 Grid Scheduling

Scheduling is the process of assigning jobs to resources. The word "job" is used in a very broad sense in this paper. It can be an executable program together with some parameters that are shipped to a generic compute resource. It can also be just some parameters that are sent to a service offering a particular fixed functionality, e.g. a blood flow simulation.

Local resource management systems perform scheduling for their locally available resources. They usually follow some policies that guide the scheduling decisions, e.g. giving higher priority to certain jobs. As already shown in figure 1, Grid scheduling works on a level above the local RMSs. A resource broker therefore assigns jobs to other resource management systems (including, potentially, another resource broker) that are usually owned and operated by different administrative entities.

A resource broker can be a centralized component that is accessed by multiple clients. The clients can be within the same administrative domain as the broker or they can be in different ones. A resource broker can also be a local, private component used by only a single consumer. The system described in this paper does not mandate a particular deployment scenario for the broker. It works equally well with all possible set-ups. The implementation, though, is a stand-alone Web Service.

The notion of scheduling is often associated with the notion of matchmaking, i.e. finding a resource that matches the requirements of a particular job. In this paper, we assume both notions to be different. Matchmaking is more than scheduling and consists of two steps: discovery and scheduling.

Discovery is the process of finding resources that functionally fulfil the requirements of a job. Scheduling is the process of finding the resource that optimizes the non-functional requirements of a job. All resources considered in scheduling are already checked for their functional compatibility.

As an example, imagine an engineer at a car company looking for a crash simulation service. A search for such a service might result in a number of service instances that provide exactly that function (this is the discovery step). To execute a concrete job, the engineer will supply information about that job plus the previously discovered candidate services to the broker which will then select the "best" service out of the supplied candidates (whatever "best" means). A simulation service that can produce the results within three days might not fulfil the non-functional requirement of having the results tomorrow, although the functional interface fits the requirements.

2.2 Service Level Agreements

Service level agreements are used for codifying service properties. Properties are expressed as terms in an SLA. Terms can be separated into high-level, often legal terms and low-level, technical terms. While the high-level terms are to be processed and understood by humans, technical terms can be processed automatically when put into an appropriate format. They typically describe the provided service and the quality of service (QoS) levels that have been agreed between a provider and a consumer. Technical terms

are usually described in the form of name/value-range pairs. The name refers to a concept that is known to all the parties involved in the SLA (usually the provider and the consumer). The associated value range expresses a set of values that determines the valid range of the values for the named variable.

SLAs are contracts (or parts of contracts) that describe QoS properties of the service to be provided. A service can be offered with different properties that are independent of its function. For example, a job submission service could be offered with different numbers of processors. A service can therefore be offered with multiple different SLAs.

SLAs help both service provider and service consumer to run their businesses more efficiently. Service consumers often need certain minimum QoS levels on service access maintained to run their business satisfactorily. They determine a set of non-functional quality of service properties that are important to them and put them into SLAs established with their service providers. Service providers use SLAs to plan service provisioning and to optimize service utilization. SLAs express the intent of future service access and can therefore be used to predict future resource utilization. Service providers can use this data to plan ahead and dedicate the right amount of resources to the services used and calculate prices for the services offered.

2.3 Negotiation

Establishing an SLA involves a negotiation phase in which provider and consumer try to find SLA terms that both can agree on. In our model, negotiation starts with the consumer getting *SLA templates* from the provider. An SLA template is a non-binding agreement proposal. Based on an SLA template, consumer and provider repeatedly exchange *SLA proposals* containing altered terms until both sides are willing to accede to that proposal or they cancel negotiations. Multiple negotiations can happen in parallel, both on the provider and on the consumer side.

2.4 SLAs as Advance Resource Reservations

Historically, SLAs have a lifetime of months to years. Lately it became clear that such a long lifetime does not fit all customer requirements and shorter validity periods are desired [8]. Shorter periods can easily be accommodated by SLAs. With shortening the validity period, terms in SLAs are likely to change. The terms used in short-term SLAs tend to be more technical as with the shrinking time span, requirements on services become clearer and more specific.

When the lifetime of an SLA becomes reasonably short, e.g. a few hours, it can usually be seen as an advance resource reservation. SLAs contain information on the service to be provided, the time frame within which the service is provided and the circumstances under which it will be provided. The first two items represent exactly the information that is commonly used in advance resource reservations. As SLAs contain additional terms, they are actually a superset of such reservations and can substitute them. In an SLA-based resource reservation system start and end times of service provisioning become just ordinary QoS parameters that can be negotiated. A separation of reservation period and service properties that is common in traditional advance resource reservation systems is not needed.

Short-term, resource-reservation-like SLAs are expected to contain machine-processable terms only. As such SLAs have a short lifetime only, negotiation should happen fast and consume as little resources as possible. It is therefore expected that such SLAs are negotiated by computers without human intervention. As legal terms usually do not make sense to computers, they are left out of short-term SLAs. The SLAs therefore only contain technical terms. The resource broker described in this paper only works on those technical terms.

Legal terms are nevertheless important in resource brokering systems. Short-term SLAs are therefore expected to exist within the context of longer term SLAs that contain all the legalese. Such framework SLAs need to be established (manually) before short-term SLAs can be negotiated for individual service access. Many short-term SLAs are created within the context of one long-term SLA. Short-term SLAs only reference the containing long-term SLA. Particular SLAs are thus much easier to set up and maintain. Short term SLAs therefore resemble resource reservations while framework contracts resemble collaboration agreements such as those commonly used for establishing virtual organizations.

3. RESOURCE BROKERING SERVICE

The resource brokering service introduced in this paper is based on service level agreements and builds on the ideas presented in the previous chapter. In particular, it performs SLA negotiation to acquire advance resource reservations. As SLA negotiation is a complex task it is a suitable candidate for outsourcing [9]. We therefore implemented the resource brokering service as a Web Service that can be operated by the service consumer's organization or some external third party. This approach is also consistent with other Grid resource brokering systems, e.g. the European DataGrid workload management system [7].

During the brokering process, the broker negotiates access with candidate resource providers. It gets offers from a number of possible providers and then selects the one with the best offer. The metric for "best" depends on the client of the broker and could be price, time, etc. In the following, interactions between the broker and just one provider are described. In reality, though, interaction with multiple providers is occuring. As the terms within SLA proposals are the deciding factor for doing business with a particular provider, the broker's scheduling decisions are based on those SLAs.

3.1 Architecture

The architecture of the resource brokering system is shown in figure 2. As is common in service-oriented architectures, resources are abstracted by services that grant access to the resources via well-defined interfaces. Furthermore, the architecture follows the WS-Agreement [14] conceptual layered service model in which the agreement layer and the service layer are separated. While regular service access happens at the service layer, SLA negotiation occurs at the agreement layer. This clean separation of functionalities makes it possible for the resource broker to work on the agreement layer only. It is not concerned with subsequent service access. Service consumers have to first get an SLA (possibly via the resource broker) and then directly access the needed service together with a reference to the SLA.

At the agreement layer, the resource brokering service and the resource provider's agreement service manage the cre-

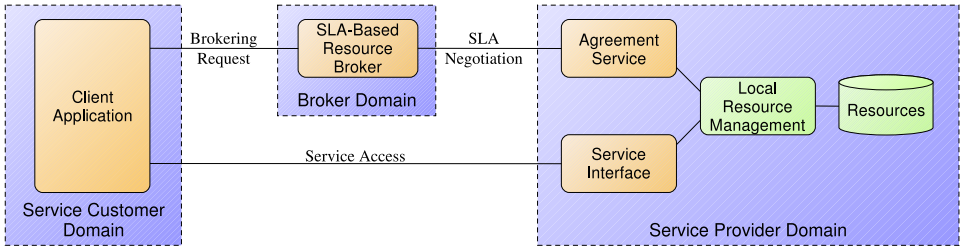

Figure 2: SLA-Based Resource Brokering Architecture

ation and representation of SLAs through Web Service interfaces. The service provider hosts an agreement service which exposes the local resource management system via an SLA negotiation interface. The most important function of the agreement service is to perform SLA negotiation based on the state of local resources. In addition, the agreement service is also responsible for managing SLA templates, checking the realisability of agreement offers, and managing established SLAs.

The provider's agreement service is resource-specific and tightly coupled to the resource management system it negotiates SLAs for. It needs to know the kind of resources it is establishing SLAs for as well as the current state of the resources. It performs SLA negotiation based on the resource's state, but it does not directly reveal that state to its clients.

In addition, negotiation is influenced by provider-specific policies. The agreement service needs to enforce such policies. The agreement service is therefore not only resource-specific – it is also provider-specific. The policies are usually proprietary information and are hidden inside the agreement service. The architecture can therefore be generic with the agreement service being a defined point for introducing proprietary behaviour of individual service providers.

3.2 SLA Representation

SLAs can be represented in different formats and a number of standards exist. As Web Services are based on XML, it is an obvious and adequate choice to use XML for representing SLAs in those environments. The NextGRID project developed an XML schema for describing SLAs. The schema provides an extensible structure to represent SLAs containing arbitrary terms. It does not specify any SLA terms itself or mandate the use of particular terms. It only provides a well-known place for storing those terms and attaching them to contextual data, e.g. contact information of stakeholders involved in a service exchange. The actual terms used depend on the application domain. Both service consumer and provider need to use the same structure (e.g. the NextGRID schema) as well as the same term language for SLAs in order to be able to cooperate.

3.3 Negotiation

The broker does currently not implement a real negotiation protocol. It follows a simple accept/reject protocol. Consumers have to present a completely filled-out SLA proposal and the provider can then decide whether to accept or reject it.

In an environment without negotiation, SLA templates must already contain all the information that is necessary for a consumer to determine whether an SLA based on a template would fulfil its requirements. It also means that the SLA templates are non-negotiable and the price tag must be part of the template. As the price is fixed, the valid value ranges for SLA terms are rather small (otherwise the provider could not operate with a profit). To compensate for that, a number of templates need to be published, with each one applying to just a small subset of term values. This approach is reflected in the current practice of offering bronze, silver and gold service levels. That method works well with a small number of terms, resulting in a small number of templates. But with an increasing number of variable terms the number of templates also increases.

4. EXAMPLE

The implementation of the resource brokering model has been used with the example introduced in this section. The scenario deals with a digital media production company that is creating rendered video animations. The company produces the models for their films by themselves, but has them rendered by external rendering service providers. Thos service providers supply both hardware and software to perform the particular function of video rendering. For outsourcing the rendering the company has framework agreements (negotiated face-to-face by humans) with a number of rendering service providers. Under these framework agreements, individual rendering jobs can be submitted. For every job submission, an SLA must be negotiated with the respective service provider. The resource broker takes care of selecting the most appropriate service for a given job and negotiates an SLA with that provider. The selected provider can accept or reject the broker's SLA proposal. In case of rejection, the broker tries to get an SLA with the next best provider. If none of the providers is willing to accede to the SLA proposal, the job is rejected.

The SLA terms used in the example are:

- **start time**: this is an obligation on the service consumer and it describes the time range in which uploading the rendering source files has to happen in order not to breach this SLA. If the files are uploaded outside this range, the service provider does not guarantee the timely delivery of the results.
- **end time**: the time range within which the results of the rendering process are to be made available. If the input files are supplied on time (i.e. within the start time time range), the service provider guarantees to produce the result at some point within this time range.
- **resolution**: the resolution of the produced video images. This information is needed by the provider to

```
<Term>
 <Name>http://nextgrid.org/endTime</Name>
 <Value>
  <xs:simpleType>
   <xs:restriction base="xs:dateTime">
    <xs:maxInclusive value="2007-08-24T00:00:00Z"/>
   </xs:restriction>
  </xs:simpleType>
 </Value>
</Term>
<Term>
 <Name>http://nextgrid.org/frameCount</Name>
 <Value>
  <xs:simpleType>
   <xs:restriction base="xs:positiveInteger">
    <xs:maxInclusive value="45000"/>
   </xs:restriction>
  </xs:simpleType>
 </Value>
</Term>
```

Figure 3: SLA Template Excerpt

```
<Term>
 <Name>http://nextgrid.org/endTime</Name>
 <Value>
  <xs:simpleType>
   <xs:restriction base="xs:dateTime">
    <xs:maxInclusive value="2007-08-24T08:00:00Z"/>
   </xs:restriction>
  </xs:simpleType>
 </Value>
</Term>
<Term>
 <Name>http://nextgrid.org/frameCount</Name>
 <Value>
  <xs:simpleType>
   <xs:restriction base="xs:positiveInteger">
    <xs:enumeration value="60000"/>
   </xs:restriction>
  </xs:simpleType>
 </Value>
</Term>
```

Figure 4: SLA Requirements Excerpt

calculate the amount of resources needed for the job.

- **frames**: the number of frames of the produced video file. This information is needed by the provider to calculate the amount of resources needed for the job.

- **objects**: the average number of objects in the files to be rendered. This information is needed by the provider to calculate the amount of resources needed for the job.

Figure 3 shows an excerpt from an example SLA template. For brevity, it only shows the end time and the number of frames. Figure 4 shows a similar excerpt from the requirements document of the consumer. It can be seen directly that the value ranges specified for the number of frames have no overlap and the given SLA template can therefore not fulfil the requirements of the consumer. The broker will therefore not try to negotiate an SLA with the provider of that template. An actual SLA is not shown here as it would look the same as the template.

From the example it can be seen that the provider does not need to hand out any internal resource state information to the consumer. All terms in the SLA template are on an abstract, application-related level. Not needing to give out details about its resources' state gives the provider a great amount of freedom in organizing its resources, including using them for other, unrelated work or for overbooking them.

The specification of an end time range gives the provider the freedom to schedule the job on its internal resources as he sees fit. For example, it allows him to use parallelized rendering software if he wants to speed up processing. The actual number of processors used for rendering is completely up to the provider. The consumer cannot influence this choice nor can he get to know it.

The benefit to the consumer is the guaranteed delivery time. By the end of the end time time range, the consumer can get his results. In case the provider is late delivering results, the consumer is eligible for a penalty fee which should be set at a value that covers all the costs incurred by a late delivery.

5. RELATED WORK

Traditional Grid resource brokers work on a best-effort basis. To decrease time to completion, they try to estimate the start or end times of jobs at different providers and select the one with the (presumably) closest start/end time. The workload management system of the European DataGrid is an example of such a broker [7]. It can base scheduling decisions on different parameters such as queue length, number of processors, processing power, etc. GRUBER [3] works in a similar way. Both brokers need large amounts of current resource state information to perform their function. All that information must be supplied by the service providers. With the system described in this paper, such information surrender is not needed.

Although there are many dedicated protocols for advance resource reservation, e.g. [10, 6], SLAs are equally well suited for that purpose. SLA negotiation offers the same features as the dedicated protocols. In addition, details on stakeholders, pricing, penalties, payment methods, etc. are an integral part of SLAs and are not regarded as mere add-ons. The examples given in the WS-Agreement specification [14] already hint at the use of SLAs as a means of advance reservation. Czajkowski et al. also mention that possibility in [2]. Elmroth and Tordsson use WS-Agreement as resource reservation protocol in [4]. WS-Agreement replaces a proprietary reservation protocol that was used in earlier versions of their broker.

That broker is one of the few examples that use resource reservation [5]. Although, on an abstract level, the functionality of their broker is the same as the broker described in this paper, there are a number of significant differences: reservations only cover job submission, information on resource state is required, and prediction of execution time is needed. One reason for the differences is that their broker is aimed at academic users. Neither objectives of service providers nor costs to consumers are taken into account. An example is the main aim of their broker, which is to minimize time to delivery, i.e. the time taken by a job including data transfer and waiting time. Monetary costs – which might serve as an alternative optimization goal – are not taken into account by their approach.

6. CONCLUSION

This paper argues for the replacement of the dissemination of detailed resource state information by the negotiation of bi-lateral SLAs in Grid scheduling. The benefits of that approach are:

- guaranteed delivery time: as SLAs are advance resource reservations, jobs will be executed by a certain negotiated deadline. Traditional resource brokers do only work on a best-effort basis.
- secrecy of state information: resource providers can keep the state information of their resources hidden from external parties. They do therefore not run the risk of revealing internal resource usage practices.
- reduced network load: resource state data, which can be rather large, does not need to be transfered. On the other hand, negotiations might involve many possible service providers, reducing this benefit.
- up-to-date state information: as SLA acceptance and rejection decisions are made by the local RMSs, up-to-date state information is used for these decisions. State information transferred to a broker is always old.

Our model results in increased autonomy for service providers and looser coupling between consumers, providers, and the broker while promoting the use of quality of service guarantees for consumers.

The implementation of our resource brokering model showed that SLAs can be used for advance resource reservation and scheduling decisions. It also proved that SLA-based resource brokering can eliminate the requirement for providing sensitive resource information to external parties. However, the success of the model greatly depends on the service provider's ability to dynamically calculate competitive, yet lucrative prices. This topic needs to be actively researched.

The implemented brokering system does not make use of a negotiation protocol. All SLA templates contain fixed, pre-defined value ranges which are non-negotiable. As service offerings are becoming increasingly dynamic, flexible, and personalized, we expect a migration from this approach to one that is based on "real" negotiation. Further research needs to be done that evaluates the possibility of dynamically adjusting term values to requirements of consumers.

Acknowledgements

This work has been supported by the NextGRID project and has been partly funded by the European Commission's IST activity of the 6th Framework Programme under contract number 511563. This paper expresses the opinions of the author and not necessarily those of the European Commission. The European Commission is not liable for any use that may be made of the information contained in this paper.

7. REFERENCES

[1] Amazon.com. Amazon Elastic Compute Cloud (Amazon EC2), 2007. http://aws.amazon.com/ec2.

[2] Karl Czajkowski, Ian Foster, Carl Kesselman, Volker Sander, and Steven Tuecke. SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems. In: Job Scheduling Strategies for Parallel Processing. Springer Verlag, 2002.

[3] Catalin L. Dumitrescu and Ian Foster. GRUBER: A Grid Resource Usage SLA Broker. In: Euro-Par 2005 Parallel Processing. Springer Verlag, 2005.

[4] Erik Elmroth and Johan Tordsson. An Interoperable, Standards-based Grid Resource Broker and Job Submission Service. In: Proceedings of the First International Conference on e-Science and Grid Computing (e-Science '05), Melbourne, Australia, December 2005.

[5] Erik Elmroth and Johan Tordsson. A Grid Resource Broker Supporting Advance Reservations and Benchmark-based Resource Selection. In: Applied Parallel Computing – State of the Art in Scientific Computing. Springer Verlag, 2006.

[6] Ian Foster, Carl Kesselman, Craig Lee, Bob Lindell, Klara Nahrstedt, and Alain Roy. A Distributed Resource Management Architecture that Supports Advance Reservations and Co-Allocation. In: Proceedings of the 7th International Workshop on Quality of Service (IWQoS '99), London, England, June 1999.

[7] G. Avellino et al. The EU DataGrid Workload Management System: towards the second major release. In: Proceedings of the International Conference on Computing in High Energy and Nuclear Physics (CHEP 2003), San Diego, CA, March 2003.

[8] Kamel Haddadou, Samir Ghamri-Doudane, Yacine Ghamri-Doudane, and Nazim Agoulmine. Designing Scalable On-Demand Policy-Based Resource Allocation in IP Networks. IEEE Communications Magazine, 44(3):142–149, March 2006.

[9] Peer Hasselmeyer, Changtao Qu, Lutz Schubert, Bastian Koller, and Philipp Wieder. Towards Autonomous Brokered SLA Negotiation. In: Proceedings of the eChallenges e-2006 Conference, Barcelona, Spain, October 2006.

[10] Dean Kuo and Mark Mckeown. Advance Reservation and Co-Allocation Protocol for Grid Computing. In: Proceedings of the First International Conference on e-Science and Grid Computing (e-Science '05), Melbourne, Australia, December 2005.

[11] Hui Li, David Groep, Jeff Templon, and Lex Wolters. Predicting Job Start Times on Clusters. In: Proceedings of the IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2004), Chicago, IL, April 2004.

[12] Jarek Nabrzyski, Jennifer M. Schopf, and Jan Weglarz. Grid Resource Management: State of the Art and Future Trends. Kluwer Academic Publishers, 2004.

[13] NextGRID Consortium. NextGRID Homepage. http://www.nextgrid.org/.

[14] Open Grid Forum. Web Services Agreement Specification (WS-Agreement), March 2007. http://www.ogf.org/documents/GFD.107.pdf.

[15] Todd Tannenbaum, Derek Wright, Karen Miller, and Miron Livny. Condor: a distributed job scheduler. MIT Press, 2001.