An Efficient Protocol for Reserving Multiple Grid Resources in Advance

Jörg Schneider[1], Julius Gehr[1], Barry Linnert[1], and Thomas Röblitz[2,3]

[1] Technische Universitaet Berlin, Germany
{komm,jules,linnert}@cs.tu-berlin.de
[2] Zuse Institute Berlin, Germany
roeblitz@zib.de
[3] CoreGRID Institute on Resource Management and Scheduling

Abstract. We propose a mechanism for the co-allocation of multiple resources in Grid environments. By reserving multiple resources in advance, scientific simulations and large-scale data analyses can efficiently be executed with their desired quality-of-service level. Co-allocating multiple Grid resources in advance poses demanding challenges due to the characteristics of Grid environments, which are (1) incomplete status information, (2) dynamic behavior of resources and users, and (3) autonomous resources' management systems. Our co-reservation mechanism addresses these challenges by probing the state of the resources and by enhancing a two-phase commit protocol with timeouts. We performed extensive simulations to evaluate communication overhead of the new protocol and the impact of the timeouts' length on the scheduling of jobs as well as on the utilization of the Grid resources.

1 Introduction

Over the last decade, Grid computing has evolved from providing basic methods to access distributed heterogeneous compute resources to a research discipline that addresses the demanding needs of both scientific and commercial applications. Scientific simulations and large-scale data analyses often involve complex workflows which require the co-allocation of multiple resources. For example, Fig. 1 illustrates a tele-immersion application, which requires different resources such as compute clusters, network links and visualization units. Co-allocating multiple resources can be implemented by different means. First, the resources' management systems coordinate the allocation of resources to a distributed application, i.e., without the need of a higher-level Grid scheduler. Second, a higher-level Grid scheduler coordinates the allocation of the resources. In this model, delivering end-to-end quality-of-service (QoS) guarantees to distributed applications necessitates the use of *advance reservations*, because Grid resources are autonomously managed. Autonomous management means, that a Grid scheduler has only limited control over the start time of a job. While this is sufficient for many single-resource applications, delivery of QoS guarantees to distributed multi-resource applications may be very difficult if not impossible.

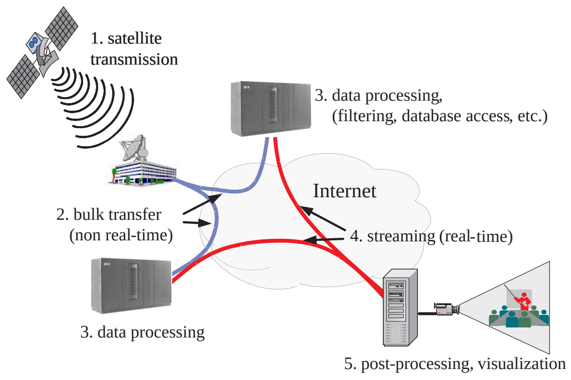

1. satellite transmission
3. data processing, (filtering, database access, etc.)
Internet
2. bulk transfer (non real-time)
4. streaming (real-time)
3. data processing
5. post-processing, visualization

Fig. 1: Example: Grid application with time-dependent tasks.

In this paper, we follow the second approach, that is reserving multiple resources in advance. In particular, we propose a protocol which (1) reduces the number of trials until a co-reservation is admitted, (2) provides means to reduce the impact of the reservation process to concurrent job submissions and (3) implements an *all-or-nothing* semantics for acquiring co-reservations.

We assume that the capacity of *resources* is space-shared among simultaneously active requests. Furthermore, the resources' local management is planning-based, i.e., it accepts requests starting in the future and guarantees that these requests are allocated the needed capacity. *Co-reservation requests* consist of multiple parts each specifying an earliest start time, a duration, a latest end time and a needed capacity. Different parts of a co-reservation request may require resources of different type and different capacities. The relationship between the parts are specified with temporal and spatial constraints.

The first step in reserving is to filter the existing resources with the static requirements of a request, e.g., the operating system of a machine or the available interfaces to a storage server. The resulting set of *resource candidates* is the input to the mechanism we propose here. Our protocol is composed of three main steps:

1. Determine co-reservation candidates by probing the future status of resources.
2. Gather preliminary reservations for each part of a co-reservation candidate.
3. Commit the preliminary reservations.

A *co-reservation candidate* is a set of tuples each describing a time slot at a certain resource. A time slot is defined by a begin time, an end time and a capacity. Also, a tuple associates performance metrics with each time slot such as cost. These performance metrics may be used to select a co-reservation candidate for further processing. While selecting a candidate is an interesting research topic by itself it is beyond the scope of this paper. In the evaluation, we simply used the candidate with the earliest time slots. Note, since the preliminary reservations expire after some timeout, they do not need to be canceled if the whole transaction fails.

The remainder of this paper is structured as follows. In Section 2 we discuss related work. The multi-resource reservation protocol we propose is presented in Section 3. The benefits of the probe messages and the timeouts are evaluated through extensive simulation in Section 4. We conclude in Section 5.

2 Related Work

As described, elaborated resource allocation mechanisms are a key requirement to enable complex applications in the Grid. Grid brokers relying on batch management systems at the local resources will only implement co-allocations at a high cost [9]. If the local resource manager already supports advance reservations, the Grid broker has to submit such reservations in a coordinated manner in order to allocate the requested resources. Several protocols have been proposed for this task, all using the basic concept of a two-phase commit. In the first phase, each required resource is preliminarily reserved. If this was successful for each resource these reservations will be committed. In the literature [1, 3, 5–7, 14], this protocol is usually applied on exactly named resources. In our approach, the co-allocation request specifies the type of the resource only. Thus, our Grid broker may automatically try multiple combinations of resources until the requested preliminary reservations are obtained.

Kuo and Mckeown [6] present a two-phase commit protocol and show that in all cases only valid final states will be reached. They discuss possible economic models to deal with preliminary reservations. The protocol also supports timeouts for the preliminary reservations. But the impact of timeouts on the resource utilization and the processing of the workload was only stated and not empirically analyzed.

Czajkowski et al. [3] propose a reservation protocol where the application itself or a co-allocation agent on behalf of the application manages the co-allocation. The authors extended the two-phase commit protocol such that the application can dynamically modify the reservation. For example, the application was enabled to give up a reservation only if it could get another reservation.

Brandic et al. [1] developed a two-phase commit protocol to ensure a requested quality-of-service. The protocol is employed between a Grid user and the Grid broker. The Grid broker will produce a QoS offer consisting of a predicted execution time, an allocation for all needed resources, and a prize based on an economic model. The user confirmation of such an offer equals the commit in the two-phase commit protocol. The offer is valid for a limited time only, but the actual impact of this timeout is not evaluated.

Haji et al. address the issues of the two-phase commit protocol with restricted knowledge [5]. They propose a mechanism to monitor all suitable resources until the preliminary reservation will be carried out. The local resource management systems of the selected resources will inform the Grid broker on each state change. The monitored resources must provide detailed information on their status which contradicts the information hiding aspect of the Grid. Additionally, the protocol operates with immediate reservations only.

The HARC co-scheduler [7] uses Gray and Lamport's Paxos Commit Protocol, a generalization of the two-phase commit protocol. Using this protocol, the commit process may succeed if multiple participants fail or are temporarily unavailable. The HARC co-scheduler requires detailed information about the timetables of the requested resources.

The MetaScheduling Service [14] uses a negotiation protocol for co-allocations. In a first step each resource is asked for the earliest possible start time. This step is repeated using the latest start time as a new lower bound until all resources return the same time. The subsequent steps implement the two-phase commit protocol. The resources are selected by the user before the negotiation begins. Our proposed probe requests works similarly to the first step, but the probe responses contain multiple possible start times in the whole requested time frame.

In the context of transactions the two-phase commit is a fundamental technique for processing distributed transactions. Various approaches extended the basic two-phase commit such that it could also be applied in distributed systems without a central coordinator, i.e., by introducing an additional phase to elect a coordinator [13].

3 The Reservation Protocol

The reservation protocol must cope with the specific characteristics of Grid environments, which are (1) incomplete status information, (2) dynamic behavior of the resources and users, and (3) autonomy of the resources' local management systems. In the theoretic case, that a client would have global information and could control any state change in the resources, it could simply request the reservation of multiple available resources. In a realistic scenario, however, a client neither has global information nor does he or a broker fully control the resources. Hence, a client does not know a-priori, which resources are available. The reservation protocol, we propose, approaches this problem by specifying three methods tailored at the level of information a client possesses and by defining the relationships between these methods. Here a client is any participant who is requesting some service. For example, a user sending a request to a broker, or a broker sending a request to a resource. The participants can compose a multi-level hierarchy, which is common in economic environments. For the sake of simplicity, we will limit the discussion to three levels: clients, broker and resources.

3.1 Building Blocks of the Protocol Phases

The protocol involves the three phases: (1) probing, (2) gathering preliminary reservations and (3) committing. The corresponding methods and the level of information to which they are applicable are summarized in Table 1.

Probing phase. The purpose of the probing phase is to gather detailed information about the state of a resource. If a component receives a probe request it determines reservation candidates. Each reservation candidate is defined by a start time, an end time and a quality-of-service level which satisfy the bounds of the flexible request. Each reservation candidate is associated with a number of evaluation metrics. The used evaluation metrics only depend on what a client is requesting and what a service provides. Typical, evaluation metrics are cost, utilization, makespan, etc. [12].

Table 1: Protocol methods operating on different levels of status information.

Method	Description	Level of information (X)
probe	Resources are asked to provide information about their current and future state.	low (0)
reserve	A preliminary reservation is requested. The request can be given with fixed or ranged values.	high for fixed request parameters, medium for flexible request parameters (1)
commit	A previously admitted preliminary reservation is committed.	highest (2)

Gathering preliminary reservations. Preliminary reservations are like normal reservations except that they need to be confirmed until a given time. Before that time, a resource may not allocate the reserved capacity to any other request. Canceling a preliminary reservation (or simply not confirming it) does not impose any cost to the client. Similar to a probe request a reservation request may be given with flexible parameters. Flexibility enables a broker to negotiate the exact parameters. Note, this is not only beneficial for the client side, but also for the resources, which can express their preferences. In a scenario with fixed parameters, a resource can only grant or deny a request.

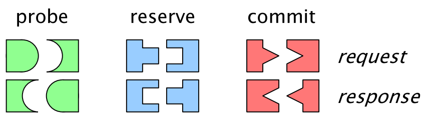
Committing phase. In the committing phase, a client confirms the preliminary reservations it will use. This phase is essential to implement an *efficient* all-or-nothing semantics in a Grid environment. A client will only start committing reservations if it has received preliminary reservations for all request parts. Because a resource must not allocate the capacity of granted preliminary reservations to any other request, it is interested in receiving a confirmation or cancellation as soon as possible.

Figure 2a shows symbols for each phase. While the upper row illustrates the interaction for sending a request, the lower shows symbols for transmitting the response messages. In the following sections, these symbols are used to compose protocol instances.

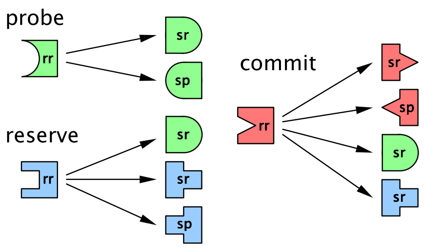
3.2 Rules for Composing Protocol Instances

A client may possess different levels of information. Consequently, a reservation process can be initiated with each of the three methods. The rules for composing protocol instances only limit the actions of a service receiving a request of a client. Figure 2b illustrates these rules using the symbols introduced in Sect. 3.1.

The service receiving a request corresponding to the information level X may only issue requests of the same or a lower level of information $Y \leq X$ (cf. Table 1). For example, if a broker receives a reserve request, it can only issue probe or reserve requests to resources. Only the sender of a request may switch to a method corresponding to a higher level of information.



(a) Symbols representing message exchanges for the three protocol phases: *probing*, *gathering preliminary reservations* and *committing*.



(b) Rules for composing protocol methods to protocol instances. Symbol labels: sr - send request, rr - receive request, sp - send response (cf. Sect. 3.1 for a detailed description of the methods).

Fig. 2: Method symbols and rules for composing protocol instances.

3.3 A Common Protocol Instance for Grid Environments

In many Grid environments, the three main parties involved in job management are the clients, a broker and the resources. Here we present a protocol instance that is adopted to such situations. Figure 3 provides an overview of the interactions of a client with the broker and of the broker with the resources.

The interactions of the components are as follows:

1. *Client Request.* The client sends a reserve request to a broker.
2. *Probe Request.* The Grid broker sends a probe request to all suitable resources specifying the constraints given by the user.
3. *Probe Response.* The requested local resource management systems (RMS) estimate whether the request could be fulfilled. Each RMS constructs a number of reservation candidates matching the requirements, i.e., containing specific values for the number of CPUs, the start time, or the guaranteed bandwidth. Each reservation candidate is further rated with respect to how well it fits into the known workload of the resource.
4. *Reservation.* Using the information from the probe responses, the Grid broker can now calculate a co-reservation candidate. The allocation takes into account the user given requirements and the scheduling strategy of the Grid broker, but also the preferences given by the local resource management systems (the rating associated with each reservation candidate). All selected resources of this co-reservation candidate are now asked for a preliminary reservation.
5. *Reservation Granted* or *Reservation Failed*. The local resource manager tries to incorporate the reservation into their schedule and informs the Grid broker about the result of the operation. Additionally, a timeout period will be negotiated until which the preliminary reservation will be held up.

6a. *Reservation Commit.* After the required resources are preliminarily reserved, the Grid broker commits all reservations. Figure 3 shows an optional step, in which the broker presents the preliminary reservation to the client and lets the client make a final decision.

6b. *Reservation Rollback.* If a single reservation fails, all former successful reservations will be canceled and a new co-reservation candidate without the failed resource will be calculated. The processing continues with step 4.

6c. *Timeout.* If the Grid broker does not respond within a given time, the local resource management system cancels the preliminary reservation.

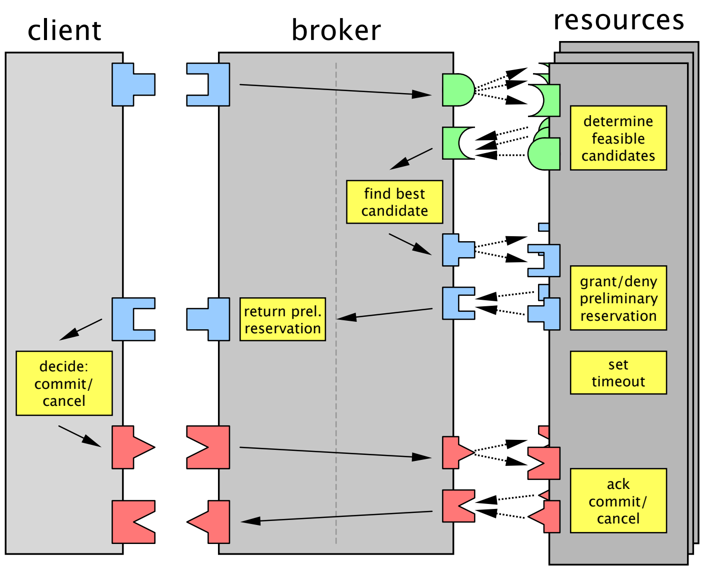


Fig. 3: Example instance of the protocol with three parties – a client, a broker and multiple resources. Solid arrows indicate one control flow. Dotted arrows show the message flow between the broker and the resource.

Using a probe request, the Grid broker is now able to create a set of resources to request, which will – with a high probability – accept the reservation. Note, whether a resource accepts a reservation, depends on the current status of the resource, which may have changed between the probe request was processed and the subsequent reservation request. In general, probing will reduce the number of reservation requests send out until one succeeds. By answering a probe request, a resource provides more detailed information about its current status than what can be derived from standard resource monitoring services [4, 8]. However, each resource can decide on its own how much information it will make available. For example, it could only send back a few reservation candidates despite that more would be free. This can be used for enforcing local scheduling policies and may prevent Grid brokers from reconstructing the precise local utilization.

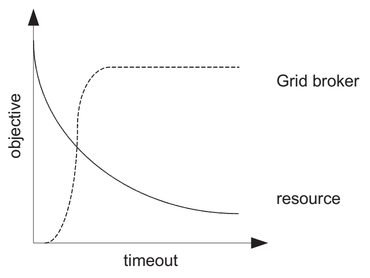
3.4 Trade-offs for Choosing Timeout Values for the Commit Phase

The moment a resource grants a preliminary reservation it also sets off a timeout until which it will keep the reservation and waits for a commit message. In this section, we discuss the trade-offs for choosing timeout values. Analytically, a timeout is composed of two distinct parts: the *technical* part and the *negotiation* part. The technical part is calculated as the round-trip time of the messages between the broker and the resources as well as between the user and the broker. Let N be the number of parts of a co-reservation request and r the message round-trip time. Depending on the communication scheme between the broker and the resources – sequential or parallel – the technical part of the timeout T_t must not be less than $(N+1)r$ or $2r$, respectively. If a resource sets off a timeout smaller than T_t, it can not receive a commit message before the timeout expires. Note, for the sake of simplicity, we ignored message handling times in the above formulas.

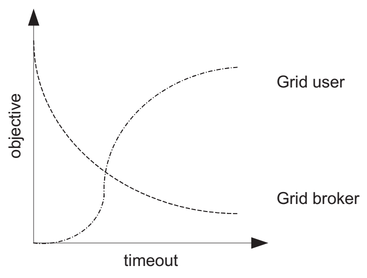
The negotiation part is composed of two parts. The first part is caused by the broker performing re-negotiation and gathering alternative reservations if some of the original candidates were denied. The second part is due to a client, which may want to evaluate the gathered preliminary reservations. For example, a client could query competing Grid brokers, check the available budget or receive a clearance from the accounting. Choosing a value for those parts depends on many parameters, in particular the current workload of the resources, their reservation policies and the co-reservation requests. While the first part can be evaluated through extensive parameter studies, the second part is very difficult to model. Therefore, the experimental evaluation (cf. Sect. 4) focuses on the first part.

Enabling the user to manually influence the booking process, also requires a fault-tolerant booking protocol. Users may simply forget to cancel reservations or not able to do so (e.g., due to system crashes, network failures, etc.). In the travel business, advance reservations are often combined with a refund policy such that the client has to pay only a small fee if the reservation is canceled. In economics research, this is known as the "no show"-problem [11]. In online shopping, recent studies [10] show that only a small percentage of customers finally submit an order after filling up the virtual shopping cart. Usually they leave the web site without clearing the shopping cart. Clearly, there is a trade-off between the time of a customer to commit his decision and the costs of a provider whose resources are blocked until the timeout expires.

Figure 4 illustrates the trade-offs for choosing timeout values. At each level in the resource management hierarchy, the service in the role of the provider – the actual resource (solid curve in Fig. 4a) or the Grid broker (dashed curve in Fig. 4b) – favors a short timeout. When the reservation is not committed or canceled before the timeout expires, the system's performance may be unnecessarily degraded. In other words, the longer a resource provider must held up a eventually canceled preliminary reservation, the smaller will be the objective of the provider. Performance degradation can be measured by the number of rejected requests due to blocked resources and by the resource utilization. In contrast, the service in the role of the client – the Grid broker (dashed curve in Fig. 4a) or the Grid user (dash-dotted curve in Fig. 4b) – prefers a long timeout. Note, the different shapes of the clients' curves. A program serving as Grid broker may



(a) Lower level (without human interaction): A Grid broker in the role of a client. A resource in the role of a provider.



(b) Higher level (with human interaction): A user in the role of a client. A Grid broker in the role of a provider.

Fig. 4: The trade-off between the objective and timeout value for negotiation partners at different levels in the Grid resource management hierarchy.

require smaller timeouts to reach a certain objective than a human being. Thus, the whole processing of co-reservation requests should be carried out with as less as possible human interaction.

4 Experimental Evaluation

We evaluated the common protocol instance for Grid environments described in Sect. 3.3 by means of parameter sweep simulations. In particular, we analyzed the impact of the probing phase and the length of the timeouts. In the following, we describe the simulated infrastructure, the workload, the user behavior, the metrics used for evaluation and present the results in detail.

4.1 Simulated Infrastructure and Workload

The simulated hardware infrastructure consists of eight parallel machines with homogeneous processors (512, 256, 256, 128, 128, 96, 32 and 32). For the sake of simplicity, the simulations were made using a simple synthetic job and user interaction model. Each job was assumed to be reserved in advance with the advance booking period being exponentially distributed. Job durations and requested processors were uniformly distributed over the interval $[1250, 3750]$ and $[1, 10]$, respectively. The inter-arrival time was used to model various load situations.

To compare the performance of the proposed protocol, we used two implementations of the Grid broker – one without (version A) and one with probing (version B). In the implementation without probing, the broker tries to reserve without sending probe messages. It just uses the information about those jobs which were submitted through the broker itself, thereby simulating the problem of missing information about the status of resources. That is, the broker has no information about the jobs which were submitted by other brokers. Figure 5 shows the simulation setup. The left path is used for the measurement of different performance metrics. The right path is used to inject the workload which is not under the control of the evaluated broker. The components Administrative Domain Controller (ADC) and Adaptive Interface (AI) are part of the Virtual Resource Manager (VRM) [2] which we used for the simulations. The ADC serves as a Grid broker, while the AI may interface with different local resource management systems (LRMS). For the evaluation, we used a simple planning-based scheduler for compute resources as LRMS.

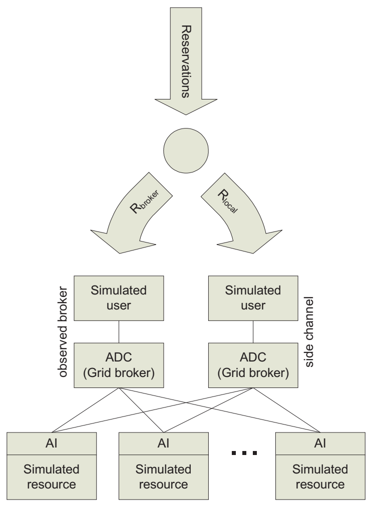


Fig. 5: The simulation setup in the experimental evaluation.

The generated workload R was divided into two sets: the set of jobs submitted to the improved broker R_{broker} and the set of jobs submitted via the side channel broker R_{local}. The jobs where randomly assigned to one of these sets preserving a given ratio $\frac{|R_{broker}|}{|R|}$ and a uniform distribution over the simulation time. Each parameter set – number of jobs, timeouts, acceptance ratios – was tested with several workloads until a sufficiently small confidence interval was reached.

4.2 Modeling the User Behavior for the Committing Phase

Modeling the behavior of actual users would require to describe how many reservations are canceled, committed or just forgotten and how the response times (cancel or commit) are distributed. In addition, users may adapt their behavior to specific settings of timeouts and may behave differently depending on the properties of the requested reservation. Because, only few information is available for defining a model addressing all these issues, we used a very simple model of the user behavior.

A user can either be an *accepter* or a *rejecter*. An accepter would send a commit within the timeout. For a local resource management system, however, it does not matter when this message is received. Hence, we assume that it is send immediately after the reservation was preliminarily booked. A rejecter might send a cancel within the timeout or will not send any message at all. The later is the worst case from the point of view of the local resource management system. Hence, we assume that a preliminary reservation of a rejecter is automatically canceled when its timeout expires.

Thus the only variables in our user model are the number of accepters and the number of rejecters. In the evaluation, we used a set of accepters $R_{accepter}$ with a fixed number of $10,000$ co-reservations and a fixed inter-arrival time distribution. For each evaluated accepter-rejecter-ratio $ratio_i$, a corresponding set $R_{rejecter,ratio_i}$ containing $10,000/ratio_i$ co-reservations requests was generated. These requests were automatically canceled when the timeout expired.

4.3 Metrics

We used different metrics for evaluating the impact of the probing and the committing phases.

Metrics for the probing phase. The purpose of the probe messages is to minimize the communication needed to negotiate the terms of a reservation. Hence, we measured for each request the total number of messages n_{probe} exchanged between the broker and the resources. Because, the negotiation for a complex workflow application with a large number of sub-jobs will involve more resources than a simple job with only a few sub-jobs, the communication cost for probing C_{probe} is defined as

$$C_{probe} := \frac{n_{probe}}{n_{jobs}} \tag{1}$$

where n_{jobs} is the numbers of sub-jobs. We also measured the impact of the probing on the number of actual reservation attempts r. The more the probe messages improve the knowledge of the status of the resources, the less reservation attempts should be necessary.

Metrics for the committing phase. We evaluated the timeout settings by measuring the impact of the workloads on the acceptance rate α defined as

$$\alpha := \frac{|R_{accepted}|}{|R_{accepter}|} \tag{2}$$

with $R_{accepted}$ being the set of reservations that are accepted and committed and $R_{accepter}$ the set of reservation requests for which a user will accept the obtained preliminary reservations.

4.4 Results for the Probing Phase

Figure 6 shows the number of reservation attempts (vertical axis) over different numbers of $\frac{|R_{broker}|}{|R|}$-ratios (horizontal axis). The higher the ratio, the more reservations are submitted at the evaluated broker or in other words the less

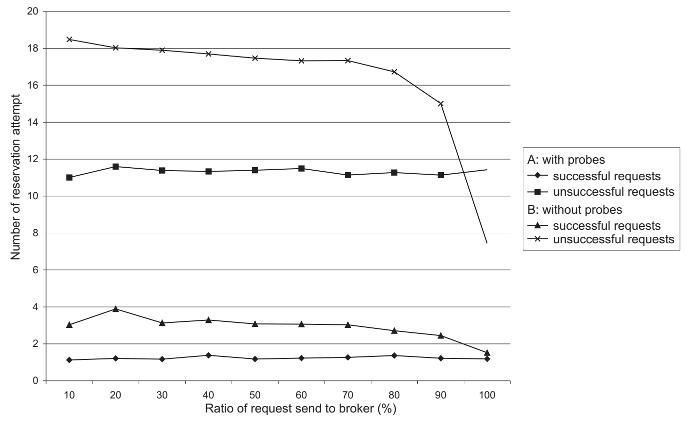


Fig. 6: Reservation attempts for different $\frac{|R_{broker}|}{|R|}$ ratios.

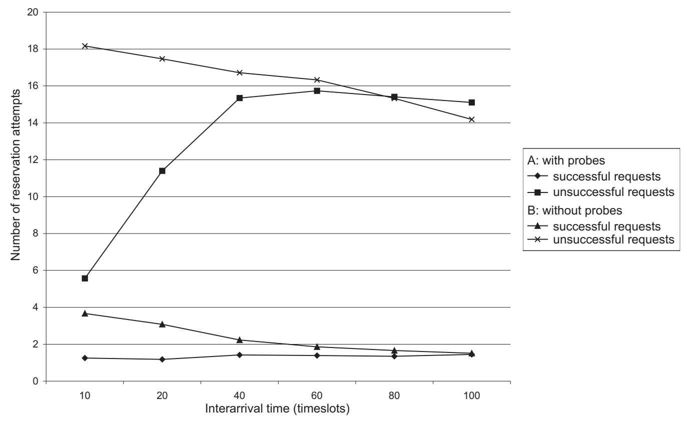


Fig. 7: Reservation attempts for different load situations.

status information was missing. All experiments were performed with an inter-arrival time of 20 time slots. The worst case in terms of reservation attempts is if all co-reservation candidates fail to be reserved. Therefore, the diagram shows individual graphs for successful and unsuccessful co-reservations. The graphs demonstrate that the probing phase significantly reduces the number reservation attempts for both successful and unsuccessful requests. Only in the case of $\frac{|R_{broker}|}{|R|} = 100\%$, i.e., all requests are submitted via the evaluated broker, the broker (version B) using the knowledge on previous reservations needs less co-reservation candidates than the probing broker (version A).

Figure 7, shows the number of reservation attempts (vertical axis) over the inter-arrival time of jobs (horizontal axis). Smaller inter-arrival times correspond to higher load. For these experiments, the workload was evenly split, such that 50% of the requests were send to the evaluated broker and 50% were sent to the side channel. The graphs show that, the higher is the load (small inter-arrival

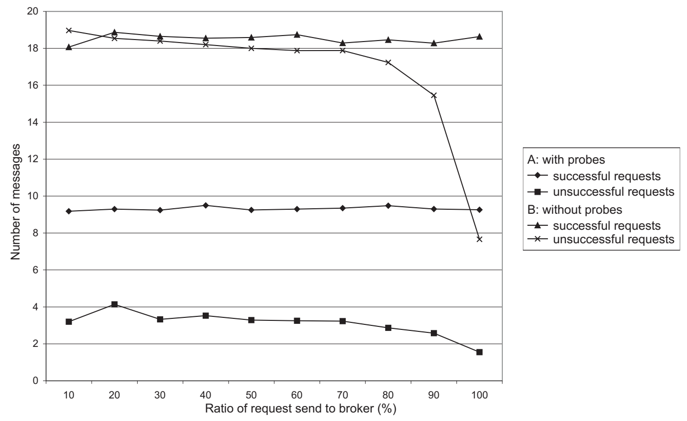


Fig. 8: The total number of messages.

times), the more does probing reduce the number of reservation attempts. For larger inter-arrival times, probing does not yield an improvement.

Figure 8 shows the total normalized number of exchanged messages exchanged messages between a broker and the resources C_{probe} (vertical axis) over different numbers of $\frac{|R_{broker}|}{|R|}$-ratios (horizontal axis). The graphs show that, for most ratios, the total normalized number of messages is higher for the broker employing the proposed protocol both for successful and unsuccessful requests. This is due to the additional probe messages. In our evaluation, each resource receives a probe message. Thus, eight probe messages were sent out for each sub-job of a co-reservation request (our simulation environment consists of eight resources, cf. Sect. 4.1). In contrast, the broker not using the probing mechanism (version B), only sends the reservation request. However, the results for the probing protocol may be improved by sending probing messages in parallel, resulting in a smaller overhead.

4.5 Results for the Committing Phase

The simulated hardware infrastructure as well as the synthetic job and user interaction model was same as in the experiments evaluating the probing mechanism (cf. Sect. 4.1 and Sect. 4.2). In contrast to the evaluation of the probing mechanism, we did not distinguish between local and Grid jobs as both suffer from unnecessary blocked resources in the same way. Hence, the number of jobs submitted via the side channel (cf. Fig. 5) were set to zero.

Figure 9 shows the acceptance rate (vertical axis) over the timeout (horizontal axis). Different curves represent rejecter-accepter-ratios (depicted by the number of rejecters per 1,000 accepters). The timeout is measured in number of time slots. The graphs show that, the smaller is the number of rejecters, the smaller is the impact of the timeout on the acceptance rate. This graph can be used to calculate timeouts dynamically based on the estimated rejecter-accepter-ratio and the target acceptance rate. For example, if we expect one rejecter per

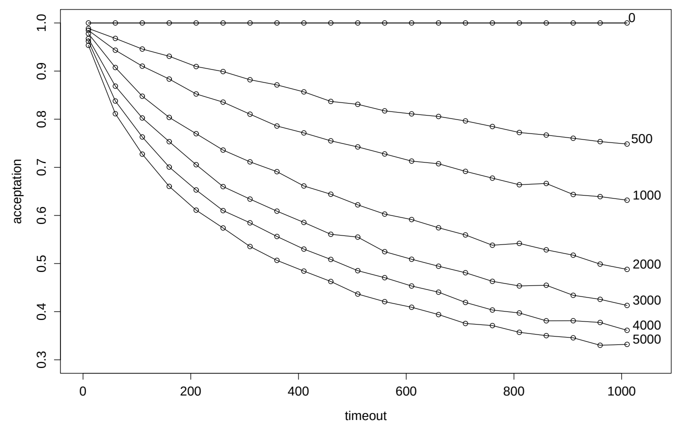


Fig. 9: The average acceptance rate α for all jobs. Each curve corresponds to a specific ratio of rejecters to accepters.

accepter and the target acceptance rate is approx. 0.8 the timeout needs to be set at about 300 time slots (cf. to curve 1,000).

5 Conclusion

In this paper, we presented a novel protocol for reserving multiple Grid resources in advance. Most existing reservation protocols implement a two-phase commit. Applied to scenarios including complex workflows or multi-site applications these protocols may lead to a high number of requests by a Grid broker trying to allocate resources. The lack of knowledge about the availability of the resources causes this high number of requests. In contrast to these approaches, our reservation protocol implements a probe phase to gather the availability information and try only co-reservation candidates with a high success probability.

We also investigated the impact of timeouts implemented in our reservation protocol as an efficient mechanism to reduce resource wastage due to canceled or not committed reservations. The results of our experiments show a significant impact of the choice of the timeout length on the overall performance of the Grid resource management system. Based on our research the Grid providers and manager is now able to adjust the timeout length to their objectives. By observing the acceptor-rejecter-ratio the timeout may be adapted dynamically.

Acknowledgments

The work of the forth author was partially funded by the German Ministry for Education and Research under grant 01AK804C and by the Institute on Resource Management and Scheduling of the EU Network of Excellence CoreGRID (Contract IST-2002-004265).

References

1. I. Brandic, S. Benkner, G. Engelbrecht, and R. Schmidt. QoS Support for Time-Critical Grid Workflow Applications. *Proceedings of the 1st International Conference on e-Science and Grid Computing (e-Science 2005)*, pages 108–115, 2005.
2. L.-O. Burchard, M. Hovestadt, O. Kao, A. Keller, and B. Linnert. The Virtual Resource Manager: An Architecture for SLA-aware Resource Management. In *4th Intl. IEEE/ACM Intl. Symposium on Cluster Computing and the Grid (CCGrid), Chicago, USA*, pages 126–133, 2004.
3. K. Czajkowski, I. Foster, and C. Kesselman. Resource Co-Allocation in Computational Grids. *Proceedings of the Eighth IEEE International Symposium on High Performance Distributed Computing (HPDC-8)*, pages 219–228, 1999.
4. K. Czajkowski, C. Kesselman, S. Fitzgerald, and I. Foster. Grid Information Services for Distributed Resource Sharing. *Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10)*, pages 181–194, 2001.
5. M. H. Haji, P. M. Dew, K.Djemame, and I. Gourlay. A SNAP-based community resource broker using a three-phase commit protocol. In *18th International Parallel and Distributed Processing Symposium*, pages 56–65, 2004.
6. D. Kuo and M. Mckeown. Advance Reservation and Co-Allocation Protocol for Grid Computing. *Proceedings of the 1st International Conference on e-Science and Grid Computing (e-Science 2005)*, pages 164–171, 2005.
7. J. MacLaren, B. Rouge, and M. Mc Keown. HARC: A Highly-Available Robust Co-scheduler. Technical report, Center for Computation and Technology, Louisiana State University, 2006.
8. M. L. Massie, B. N. Chun, and D. E. Culler. The Ganglia Distributed Monitoring System: Design, Implementation, and Experience. *Parallel Computing*, 30(7):817–840, July 2004.
9. H. H. Mohamed and D. H. J. Epema. Experiences with the KOALA co-allocating scheduler in multiclusters. *IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2005)*, 2005.
10. H. K. Omwando, J. Favier, I. Cremers, and T. van Tongeren. The Best Of Europe's Online Retail. Technical report, Forrester Research, October 2003.
11. S. Ringbom and O. Shy. Reservations, refunds, and price competition. Technical Report 5/2003, Svenska handelshgskolan, Swedish School of Economics and Business Administration, November 2003.
12. T. Röblitz, F. Schintke, and A. Reinefeld. Resource Reservations with Fuzzy Requests. *Concurrency and Computation: Practice and Experience*, 18(13):1681–1703, November 2006.
13. D. Skeen. Nonblocking commit protocols. *Proceedings of the 1981 ACM SIGMOD international conference on Management of data*, pages 133–142, 1981.
14. O. Wäldrich, P. Wieder, and W. Ziegler. A Meta-scheduling Service for Co-allocating Arbitrary Types of Resources. In *Proc. of the Second Grid Resource Management Workshop (GRMWS05) in conjunction with the Sixth International Conference on Parallel Processing and Applied Mathematics (PPAM 2005)*, volume 3911 of *LNCS*, pages 782–791. Springer, 2005.