Failure Recovery in Distributed Environments with Advance Reservation Management Systems

Lars-Olof Burchard, Barry Linnert
{baron,linnert}@cs.tu-berlin.de

Technische Universitaet Berlin, GERMANY

Abstract. Resource reservations in advance are a mature concept for the allocation of various resources, particularly in grid environments. Common grid toolkits such as Globus support advance reservations and assign jobs to resources at admission time. While the allocation mechanisms for advance reservations are available in current grid management systems, in case of failures the advance reservation perspective demands for strategies that support more than recovery of jobs or applications that are active at the time the resource failure occurs. Instead, also already admitted, but not yet started applications are affected by the failure and hence, need to be dealt with in an appropriate manner. In this paper, we discuss the properties of advance reservations with respect to failure recovery and outline a number of strategies applicable in such cases in order to reduce the impact of resource failures and outages. It can be shown that it pays to remap also affected but not yet started jobs to alternative resources if available. Alike reserving in advance, this can be considered as *remapping in advance*. In particular, a remapping strategy that prefers requests that were allocated a long time ago, provides a high fairness for clients as it implements similar functionality as advance reservations, while achieving the same performance as the other strategies.

1 Introduction

Advance reservations are a way of allocating resources in distributed systems before the resources are actually required, similar to flight or hotel booking. This provides many advantages, such as improved admission probability for sufficiently early reservations and reliable planning for clients and operators of the resources. Grid computing in particular uses advance reservations, which besides reliability of planning simplifies the co-allocation of very different resources and resource types in a coordinated manner. For example, the resource management integrated in the Globus toolkit [6] provides means for advance reservations on top of various local resource management systems. Currently, grid research moves its focus from the basic infrastructure that enables the allocation of resources in a dynamic and distributed environment in a transparent way to more advanced management systems that accept and process jobs consisting of numerous sub-tasks and, e.g., provide guarantees for the completion of such jobs. In

this context, the introduction of service level agreements (SLA) provides flexible negotiation mechanisms for various applications. This demands for control over each job and its required resources at any stage of the job's life-time from the request negotiation to the completion. An example for a resource management framework covering these aspects is the virtual resource manager architecture described in [3].

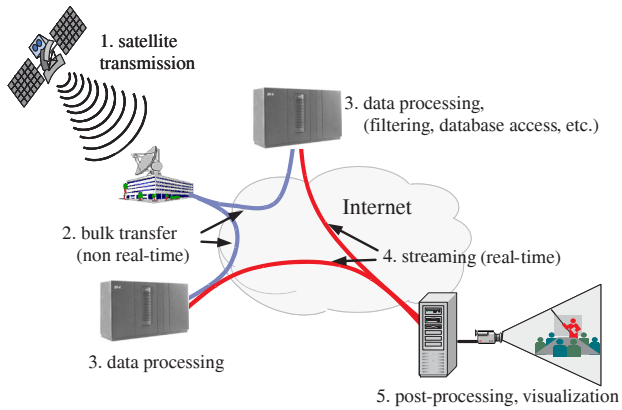

Fig. 1. Example: grid application with time-dependent tasks.

Such an application is depicted in Figure 1. The job processed in the distributed environment consists of a number of sub-tasks which are executed one after another in order to produce the final result, in this case the visualization of the data. This includes network transmissions as well as parallel computations on two cluster computers.

One important aspect in this context is the behavior of the management system in case of failures. While current research mainly focused on recovery mechanisms for those jobs that are already active, in advance reservation environments it is also necessary to examine the impact of failures onto admitted but not yet started jobs or sub-jobs. In contrast to the sophisticated and difficult mechanisms needed to deal with failures for running jobs, e.g., checkpointing and migration mechanisms, jobs not yet started can be dealt with in a transparent manner by remapping those affected jobs to alternative resources.

In this paper, a framework for dealing with those jobs is presented which includes strategies for selecting alternative resources and assigning inactive jobs. Similar to the term *reserving in advance*, we refer to this approach as *remapping in advance*, as those mechanisms perform the remapping ahead of the actual impact of the failure. Besides the description of the failure recovery framework, we show the success of our approach using simulations in a distributed environment.

The failure recovery strategies do not solely apply to actual failures of resources, e.g., hardware failures of processors or network links, but can also be used in a highly dynamic system, where resources are deliberately taken out of the distributed system for maintenance or in order to use the resource for local

requests of high priority. Furthermore, the failure strategies are independent of the underlying resources, i.e., the mechanism is generic in the sense that it is not restricted to a particular resource type such as parallel computers. Instead, it is possible to apply the mechanisms to a wide range of resources as needed, e.g., in grid environments.

The remainder of this document is organized as follows: firstly, related work important for this paper is outlined. After that, the properties of the advance reservation environment are presented and the impact on the failure recovery mechanisms that must be applied. Furthermore, we introduce the notion of expected downtime which describes the estimated time of the failure and outline a number of remapping strategies for affected jobs which can be adopted in a flexible manner depending on the jobs properties. In Sec. 6, the strategies are evaluated using extensive simulations. The paper is concluded with some final remarks.

2 Related Work

Advance reservations are an important allocation strategy, widely used, e.g., in grid toolkits such as Globus [4], as they provide simple means for co-allocations of different resources. Besides flexible and easy support for co-allocations, advance reservations also have other advantages such as an increased admission probability when reserving sufficiently early, and reliable planning for users and operators. In contrast to the synchronous usage of several different resources, where also queueing approaches are conceivable [1], advance reservations have a particular advantage when time-dependent co-allocation is necessary, as shown in Fig. 1. In [3], advance reservations have been identified also as essential for a number of higher level services, such as SLAs.

In the context of grid computing, failure recovery mechanisms are particularly important as the distributed nature of the environment requires more sophisticated mechanisms than needed in a setting with only few resources that can be handled by a central management system. The focus of this paper is on the requirements for dealing with failures and outages of resources that are reserved in advance.

In general, failure detection and recovery mechanisms focus on the requirements to deal with applications that are already active. The Globus heartbeat monitor HBM [6] provides mechanisms to notify applications or users of failures occurring on the used resources. The recovery mechanisms described in this paper can be initiated by the failure detection of the HBM. In [7], a framework for handling failures in grid environments was presented, based on workflow structure. The framework allows users to select different failure recovery mechanisms, such as simply restarting jobs, or - more sophisticated - checkpointing and migration to other resources if supported by the application to be recovered.

In [2], the problem of failure recovery in advance reservation systems was addressed in a similar manner for networks. One important difference is that in contrast to the considerations in [2], jobs cannot be migrated and distributed

during run-time in the same way as network transmissions, where packets can be transmitted on several paths in parallel.

Mechanisms as presented in this paper can be applied in distributed but also in centralized management systems, such as the virtual resource manager (VRM) described in [3]. Residing on top of local resource management systems, the VRM framework supports quality-of-service guarantees and SLA negotiation and with these mechanisms provides a larger variety and improved quality of the services offered for users. In particular, when SLAs were negotiated, e.g., in order to ensure job completion up to a certain deadline, failure recovery mechanisms are essential in order to avoid breaching an SLA.

3 Application Environment

Advance reservations are requests for a certain amount of resources during a specified period of time. In general, a reservation can be made for a fixed period of time in the future, called *book-ahead interval*. The time between issuing a request and the start time of the request is called *reservation time*. In contrast to immediate reservations which are usually made without specifying the duration, advance reservations require to define the stop time for a given request. This is required to reliably perform admission control, i.e., to determine whether or not sufficient resources can be guaranteed for the requested period.

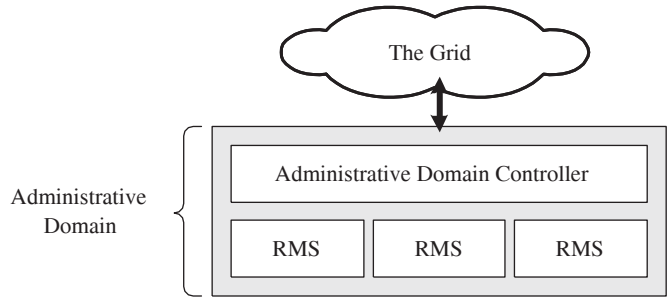

Fig. 2. Outline of the VRM Architecture

The failure recovery mechanisms described here are integrated in the VRM architecture [3] (see Fig. 2). The *administrative domain controller* (ADC) is in charge of the resource management, e.g., resource selection, scheduling, and failure recovery, of a domain consisting of one or more resource management systems. Once a failure is notified to the ADC, the failure recovery searches for alternative resources firstly within its own domain. If this is not successful, other resources available via the grid interface are contacted in order to find a suitable alternative location for a job.

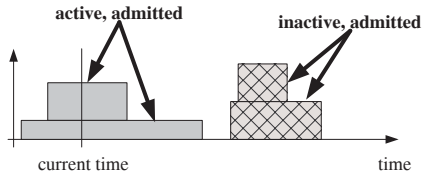

Fig. 3. Active and inactive jobs in the advance reservation environment.

4 Expected Downtime

In advance reservation environments, knowledge is available not only about jobs that are currently active, but also about those that are admitted but not yet started (see Fig. 3). While in other environments, failure recovery strategies need to be implemented only for active jobs, advance reservations require to consider also the inactive ones. For this purpose, we introduce the notion of *expected downtime*. This time represents an estimate of the actual duration of the failure and is the basis for our failure recovery strategies.

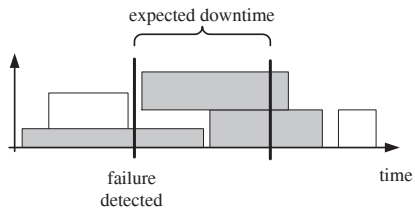

Fig. 4. Jobs within expected downtime (gray) are considered for remapping.

As depicted in Fig. 4, any job that is or becomes active during the expected downtime period is considered for remapping. In contrast to those strategies aiming at only recovering active jobs, e.g., using checkpointing and migration, remapping of inactive jobs has the advantage of requiring much less efforts, since this is done entirely within the management system. The emphasis in this paper is on remapping inactive jobs.

5 Remapping Strategies

In the context of this study, jobs running at the moment the failure occurs are considered to be not remappable. The reason is the difficulty to implement suitable recovery mechanisms, such as checkpointing and migration facilities. For many resource types, such as cluster systems or parallel computers, such functionality lacks completely or has to be implemented explicitly by the application. However, our assumption is not crucial for the evaluation of our approach or the success of the remapping strategies themselves.

In case, the failure of a specific resource system, e.g., a cluster, is notified, the management system has to face different tasks to minimize the impact of

Fig. 5. Timeline of the failure recovery process

the failure, which means, as many affected jobs as possible have to be remapped to alternative resources. The amount of affected jobs to be remapped is defined by the time the failure occurred and the expected downtime. Therefore, at first it is necessary to investigate the actual allocation of the local resource system affected by the failure, which means all jobs that have a reservation for the time span between failure and end of the expected downtime have to be processed. Other jobs are not required to be taken into account. The temporal sequence of events during the recovery process is depicted in Fig. 5.

For all jobs that must be remapped, alternative resources have to be found. Hence, the resource management system must have information about all available resources which are able to run the affected jobs. Because grid environments can provide different and heterogeneous resources, the management system has to make sure that only computing systems feasible to deal with the jobs to be remapped are considered during the recovery process.

Finding the alternative resources for a set jobs is a classical bin packing problem [5]. In order to determine feasible resources, strategies such as gangmatching have been developed [10]. Once the set of feasible resources has been determined, the remapping mechanism determines the amount of unused capacity, e.g., compute nodes, on all alternative compute systems, e.g., cluster computers. Then, the task is to maximize the success of the remapping according to some optimization criterion, e.g., the amount of successfully remapped jobs. Other optimization criteria are conceivable as well although not targeted in this paper, e.g., minimizing the penalty to be paid for terminated jobs.

The bin packing problem discussed here deals with different bins of different, but fixed size, to be filled with objects of fixed size. In our case these objects are rectangles, symbolizing the reservations, fixed in height and width, and the bins are defined by the expected downtime (width) and the amount of unused resources on the potentially available alternative resource locations (height). This means, we have to deal with a special case of the multidimensional bin packing problem - a rectangle packing problem, which is NP-complete [8]. Hence, in this paper heuristics are used in order to determine how jobs are remapped onto alternative resources.

Because the reservations are fixed in time it is not possible to shift the jobs to the future on the local system or alternative resources. This differs from scheduling bin packing approaches using time as variable dimension. Thus, it is essential to find free resources during the specific downtime interval, for example, using the available resources within the grid (see Sec. 3). On the other hand, free resources on any of the alternative systems may not be available for any request. Therefore, it the necessary to decide in which order jobs are being remapped to unused resources.

Some assumptions can be made to motivate the decision for suitable remapping heuristics, as outlined in the following.

First Come First Served (FCFS) In order to maximize the acceptance of grid environments and advance reservation systems, a predictable behavior of the system has to be assured – even in cases of failures. One opportunity is to prefer reservations allocated a long time ago. This implements a similar mechanism as advance reservations themselves, i.e., early reservations assure preferred access to the requested resources. Hence, this remapping strategy, called *first come first served*, matches best the users' expectation of the behavior of the failure recovery mechanisms. For this purpose, the reservation time, i.e., the time interval between allocation and resource usage (see Sec. 3), is stored with each request.

Earliest First (EF) Since the problem of remapping all jobs afflicted by the expected downtime is NP-complete, the search for free resources can last a significant amount of time by itself. Furthermore, in distributed management systems it is necessary to accommodate for the communication costs for status checks and remapping requests (see Fig. 2). Therefore, the termination of jobs due to the long lasting recovery process must be reduced. This is achieved by the *earliest first* strategy, which orders job according to their start time.

Smallest Job First (SJF) The *smallest job first* strategy aims at reducing the total number of terminated jobs resulting from insufficient amount of free resources. In contrast to FCFS, this strategy may be preferred by operators more than by users. This strategy orders jobs according to their total resource consumption, i.e., the product $resource\ usage \times time$, e.g., CPU hours.

Largest Job First (LJF) The *largest job first* strategy deals with the effect of fragmentation of free resources on the grid environment. Using this strategy it is likely to optimize the utilization of the whole environment. Many small requests will not congest alternative resources.

Longest Remaining First (LRF) This strategy prefers jobs with long remaining run-time. Thus, jobs which utilize resources for a long period of time will get higher remapping probability.

Shortest Remaining First (SRF) The counterpart of LRF is *shortest remaining first*, which gives priority to jobs with low remaining run-time. Thus, more jobs are likely to be remapped successfully which may be the goal of operators.

In Fig. 6, an example of jobs to be remapped during the expected downtime is shown. Using FCFS, the jobs are prioritized according to the time intervals r_1, r_2, r_3, i.e., the remapping order is J_2, J_3, J_1, whereas when using EF, only the start time of the resource is of interest, i.e., the resulting order is J_1, J_2, J_3.

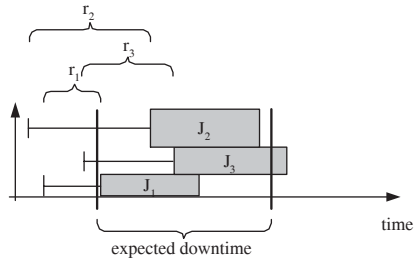

Fig. 6. Example for job ordering and remapping.

6 Evaluation

All of the strategies previously described have their advantages and may be chosen depending on the focus of operator or user perspectives. Simulations were conducted in order to show how the different strategies perform in actual grid environments.

6.1 Simulation Environment

The simulations were made assuming an infrastructure of several cluster and parallel computers with homogeneous node setting, i.e., each job is capable of running on any of the machines involved. The reason is, that although grid computing in general implies a heterogeneous infrastructure, an alternative resource used for remapping a job needs to be equipped such that the respective job is runnable. Hence, it is sensible to simplify the infrastructure.

The simulations only serve the purpose of showing the general impact of failures and since according to [9] the actual distribution of job sizes, job durations etc. do not impact the general quality of the results generated even when using simple models, the simulations were made using a simple synthetic job and failure model. Each job was assumed to be reserved in advance with the reservation time being exponentially distributed with a mean of 100 slots. Job durations were uniformly distributed in the interval $[250, 750]$ and each job demanded for a number of nodes being a power of 2, i.e., $2, 4, 8, \ldots, 256$ nodes with uniform distribution. Each time a failure occurred, a resource was chosen randomly with uniform distribution. The time between failures followed an exponential distribution with a mean of 250 slots. The hardware infrastructure consisted of different parallel computers with varying number of compute nodes, in total there were eight machines with different amount of nodes, i.e., 1024, 512, 256, 128, 96, and 16. Obviously, some jobs cannot be executed on any machine.

Each simulation run had a duration of 10,000 slots and the results presented in the following sections each represent the average of 10,000 simulation runs.

In order to assess the performance of the different strategies, two metrics were chosen that reflect both the amount of jobs that were affected but could not be successfully remapped onto alternative resources and the reduction of the

utilization that resulted from terminated jobs. The first metric is the *termination ratio*, which is defined as follows:

$$\text{termination ratio} := \frac{|\bar{A}|}{|A|},$$

with A being the set of affected jobs and $\bar{A} \subset A$ being the set of terminated jobs. The second metric is called *utilization loss rate*, defined as

$$\text{utilization loss ratio} := \frac{\sum_{j \in \bar{A}} t(j)c(j)}{\sum_{j \in A} t(j)c(j)},$$

with $t(j)$ denoting the duration of job $j \in A$, and $c(j)$ denoting the extend of the resource usage of j. For example, when the resource in question is a cluster computer, the amount of CPU hours lost due to a failure is captured by the utilization loss ratio.

For the sake of simplicity, it was assumed that jobs can only be finished completely or not at all. In certain cases, users may also be satisfied with a reduced quality-of-service in the sense that even partial results or a reduced number of nodes can be tolerated. However, as the emphasis in this paper is on the general behavior of a management system using our failure recovery strategies, this was not taken into account.

6.2 Performance of the Remapping Strategies

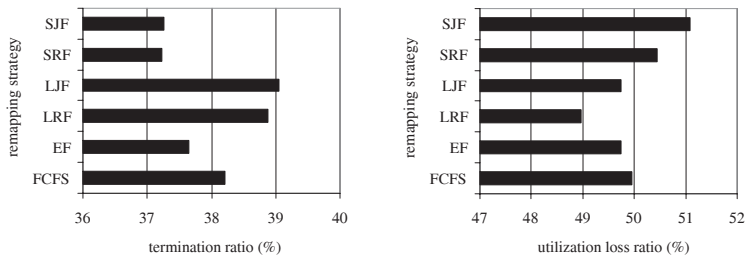

Fig. 7. Performance of the remapping strategies

In Fig. 7, the performance of the different strategies is depicted with respect to termination ratio and utilization loss ratio. The general result is that, the differences between the individual strategies is rather low. This means, it may be possible to select a strategy that matches the expectations of operators or users best.

While the strategies that prefer small or short jobs (SJF, SRF) achieve a low termination ratio, the strategies which give high priority to long or large jobs (LJF, LRF) achieve superior utilization loss ratio. The strategies related to the time, i.e., EF and FCFS, range between the worst and the best, with EF being near the best for both metrics.

6.3 Impact of the Downtime Estimation

The computation of the expected downtime is a crucial task in the whole failure recovery process. This estimation can, e.g., be based on knowledge about the type of the actual failure or statistics about previous failures. For example, replacing a failed hardware part such as a processor or interconnect can strongly depend on the time required for shipping the failed part which usually is known in advance. However, as it can not be assured that the estimation is accurate, it is important to study the impact of inaccurate downtime estimations on the termination ratio and utilization loss ratio.

Two cases must be examined: an overestimation means that the actual failure lasted shorter than expected, an underestimation means the actual failure lasted longer than originally assumed.

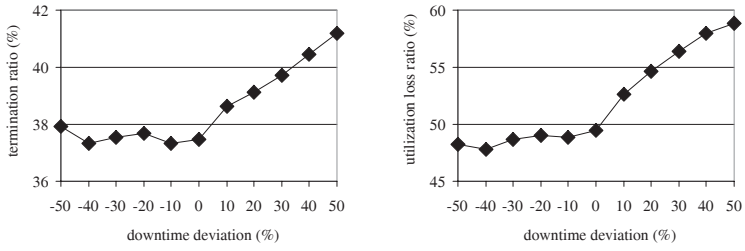

Fig. 8. Impact of inaccurate downtime estimation on the termination ratio and utilization loss ratio

In Fig. 8, the influence of over- and underestimations is depicted for the FCFS strategy as an example. It can be clearly observed, that with a positive downtime deviation, i.e., the actual failure lasted longer than expected, both the termination ratio and utilization loss ratio increase significantly. In contrast, overestimations of the actual downtime do not show significant effects with respect to both metrics.

The reason for this behavior is that with overestimations, the amount of jobs that must be terminated does not differ from the case of an exact estimation. Once the failure is removed, e.g., by replacing a failed hardware item, the management system simply changes the status to *running*. No further actions is required. In case of an underestimation, this is different. Once the end of the estimated failure period is reached and the system is still not operable, the management needs to extend the estimated downtime period and then remap the jobs within the extended downtime. Since at this time additional jobs may have arrived and assigned to the set of alternative resources, it is more likely that remapping is not successful.

While an overestimation of the actual downtime has no negative impact on the job termination ratio, this is slightly different when investigating the amount of jobs that can be accommodated by the distributed system and the achievable

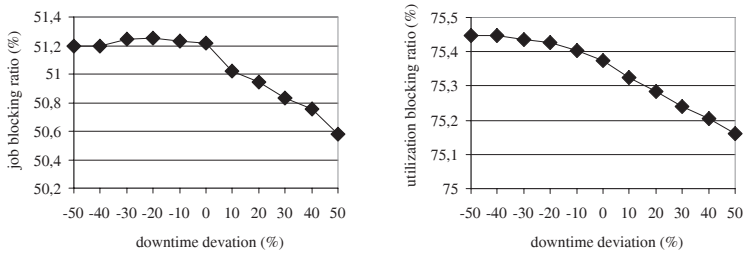

Fig. 9. Impact of inaccurate downtime estimation on the job blocking ratio and utilization blocking ratio

utilization. This is depicted in Fig. 9, showing the job blocking ratio and utilization blocking ratio which capture the percentage of rejected jobs in total and the utilization these jobs would have been generated. It can be seen that both metrics decrease with increasing overestimation resulting from the assumption that the downtime lasts longer and since jobs are not admitted to a system which is failed, fewer jobs are admitted to the system. However, in this case underestimations admit more jobs at the expense that fewer jobs actually survive failures. Furthermore, the impact on the overall utilization depends on the amount of failures and their duration. As failure situations can be considered as exceptions, the actual impact of inaccurate downtime estimations remains low.

The results presented in this section show clearly, that the introduction of the expected downtime, i.e., performing remapping in advance, is an effective mean to reduce the amount of actually terminated jobs. Otherwise, the effect is similar to an underestimation, i.e., termination ratio and utilization loss ratio increase significantly. Although it is unrealistic that the actual downtime can always be accurately predicted, it is useful to have at least any rough estimate in order to increase the amount of successfully remapped jobs. Overestimations, although reducing the amount of jobs that can be accommodated, do not harm the systems performance with respect to the amount of terminated jobs. As indicated by the performance results, the estimation of the downtime is more important than the choice of the actual remapping strategy. In particular, an underestimation of the downtime by only 10 percent leads to a worse performance than selecting a different remapping algorithm.

7 Conclusion

In this paper, failure recovery strategies for advance reservation systems, e.g., several distributed parallel computers or grid environments, were presented. It could be shown, that particularly remapping in advance, i.e., remapping inactive but admitted jobs, is important to reduce the impact of failures. Furthermore, remapping of inactive jobs does not interfere with running applications but can instead be performed completely within the management system. The strategies presented in this paper are generic, i.e., they can easily be applied to almost any

resource type and any resource management system, either centralized or distributed. This is particularly important for next generation grid systems, which essentially need to support higher level quality-of-service guarantees, e.g., specified by SLAs.

The results of the simulations showed, that the impact of a wrong downtime estimation is much higher than the differences between the remapping strategies. This means, the choice of the remapping strategy can be selected according to the needs of the actual environment. Concluding, the remapping of jobs in advance proved to be a useful approach for dealing with failures in advance reservation systems.

References

1. Azzedin, F., M. Maheswaran, and N. Arnason. A Synchronous Co-Allocation Mechanism for Grid Computing Systems. *Journal on Cluster Computing*, 7(1):39–49, January 2004.
2. Burchard, L.-O., and M. Droste-Franke. Fault Tolerance in Networks with an Advance Reservation Service. In *11th International Workshop on Quality of Service (IWQoS), Monterey, USA*, volume 2707 of *Lecture Notes in Computer Science (LNCS)*, pages 215–228. Springer, 2003.
3. Burchard, L.-O., 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*, 2004.
4. Foster, I., C. Kesselman, C. Lee, R. Lindell, K. Nahrstedt, and A. Roy. A Distributed Resource Management Architecture that Supports Advance Reservations and Co-Allocation. In *7th International Workshop on Quality of Service (IWQoS), London, UK*, pages 27–36, 1999.
5. Garey, M. and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.
6. The Globus Project. http://www.globus.org/.
7. Hwang, S. and C. Kesselman. Grid Workflow: A Flexible Failure Handling Framework for the Grid. In *12th Intl. Symposium on High Performance Distributed computing (HPDC), Seattle, USA*, pages 126–138. IEEE, 2003.
8. Karp, R., M. Luby, and A. Marchetti-Spaccamela. A Probabilistic Analysis of Multidimensional Bin Packing Problems. In *16th annual ACM Symposium on Theory of Computing (STOC)*, pages 289–298. ACM Press, 1984.
9. Lo, V., J. Mache, and K. Windisch. A Comparative Study of Real Workload Traces and Synthetic Workload Models for Parallel Job Scheduling. In *4th Workshop on Job Scheduling Strategies for Parallel Processing, Orlando, USA*, volume 1459 of *Lecture Notes in Computer Science (LNCS)*, pages 25–46. Springer, 1998.
10. Raman, R., M. Livny, and M. Solomon. Policy Driven Heterogeneous Resource Co-Allocation with Gangmatching. In *12th Intl. Symposium on High Performance Distributed Computing (HPDC), Seattle, USA*, pages 80–90. IEEE, 2003.