THE VIRTUAL RESOURCE MANAGER: LOCAL AUTONOMY VERSUS QOS GUARANTEES FOR GRID APPLICATIONS

L.-O. Burchard, H.-U. Heiss, B. Linnert, J. Schneider
{baron,heiss,linnert,komm}@cs.tu-berlin.de
Communications and Operating System Group
Technische Universität Berlin, Germany

F. Heine, M. Hovestadt, O. Kao, A. Keller
{fh,maho,okao,kel}@upb.de
Paderborn Center for Parallel Computing (PC^2)
Universität Paderborn, Germany

Abstract In this paper, we describe the architecture of the virtual resource manager VRM, a management system designed to reside on top of local resource management systems for cluster computers and other kinds of resources. The most important feature of the VRM is its capability to handle quality-of-service (QoS) guarantees and service-level agreements (SLAs). The particular emphasis of the paper is on the various opportunities to deal with local autonomy for resource management systems not supporting SLAs. As local administrators may not want to hand over complete control to the Grid management, it is necessary to define strategies that deal with this issue. Local autonomy should be retained as much as possible while providing reliability and QoS guarantees for Grid applications, e.g., specified as SLAs.

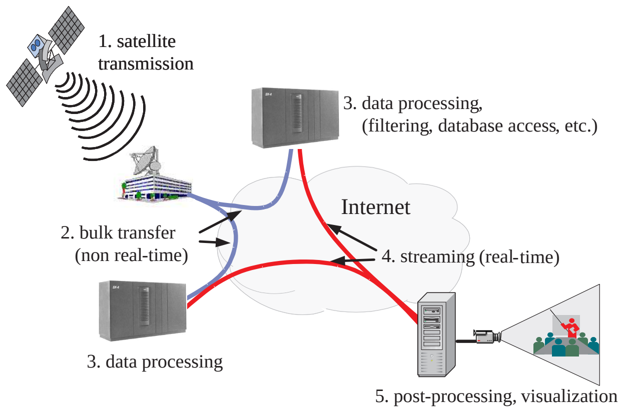
1. satellite transmission
3. data processing, (filtering, database access, etc.)
Internet
2. bulk transfer (non real-time)
4. streaming (real-time)
3. data processing
5. post-processing, visualization

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

1. Introduction

The use of advance reservations in Grid computing environments has many advantages, especially for the co-allocation of different resources. Research on future generation Grids is moving 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 complex jobs and workflows consisting of numerous sub-tasks and even provide guarantees for the completion of such jobs. In this context, the introduction of service level agreements (SLA) enables flexible mechanisms for describing the quality-of-service (QoS) supported by the multiple resources involved, including mechanisms for negotiating SLAs [5]. The introduction of SLAs implies prices for resource usage and also penalty fees that must be paid when the assured QoS is not met. Depending on the scenario, this may be a missed deadline for the completion of a sub-job in a workflow. Consequently, the definition of SLAs 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 (VRM, [4]) architecture discussed in this paper.

Figure 1 depicts a typical example for a complex workflow in a Grid scenario. The workflow processed in the distributed environment consists of five sub-tasks which are executed one after another in order to produce the final result, which is the visualization of the data. This includes network transmissions as well as parallel computations on two cluster computers.

In order to support QoS guarantees for such applications on the Grid level, one important requirement is to gain control over the local resources, at least to a certain extent to be able to guarantee the agreed service level for the Grid application.

In this context, it is necessary to distinguish the policies defined by local administrators and the policies enforced by the Grid resource management. In the VRM architectural model, the resource management resides on top of the local RMS and uses the local facilities in order to submit, process, and monitor jobs. This architectural concept, common in todays' Grid resource management systems, requires the Grid jobs to obey local policies in order to not interfere with local jobs. This assures the autonomy of the local RMS leveraging the acceptance of SLAs in the context of Grid computing.

In this paper, we describe the general problem of trading off local autonomy versus QoS for Grid applications and show that these two aspects are related. In addition, we present the architecture of the VRM and discuss the actual organization of local autonomy within the VRM.

The remainder of this document is organized as follows: first, related work important for this paper is outlined. After that, the problem is described, including the necessary assumptions and conditions for implementing QoS guarantees based on SLAs. The different levels of QoS that can be implemented depending on the degree of local autonomy are then described and discussed in Sec. 4. Section 5 focuses on the VRM architecture and the mechanisms to support the definition of local autonomy levels is presented. The paper concludes with some final remarks in Sec. 6.

2. Related Work

In [2], a number of qualitative and quantitative characteristics have been determined which can be used to define the QoS of a Grid application execution. It was also shown that the main requirement is to restrict the possible time of execution, e.g., by providing a deadline.

Advance reservations are an important allocation strategy, e.g., used in Grid toolkits such as Globus [8], as they provide simple means for planning of resources and in particular co-allocations of different resources. Besides flexible and easy support for co-allocations, e.g., if complex workflows need to be processed, 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, advance reservations have a particular advantage when time-dependent co-allocation is necessary, as shown in Fig. 1. Support for advance reservations has been integrated into several management systems for distributed and parallel computing [9, 13]. In [4], advance reservations have been identified as essential for several higher level services not only limited to SLA.

One major field of application for advance reservations in general are environments demanding for a large amount of resources. Examples are e-science Grid applications with transfer sizes in the order of several Gigabytes or even Terabytes and complex computations. For this purpose, the common Internet infrastructure is insufficient and therefore, is bypassed using dedicated switched optical circuits in today's research networks. One example for such a Grid environment is the TransLight optical research network [6].

Advance reservations are used by local RMS to achieve the same goals as in Grid environments [10]. Planning based resource managers are available for different kinds of resources, e.g., QoS managed networks [3] and cluster computers [11].

On the other hand, queueing based RMS achieve higher throughput and are therefore widely used, e.g., OpenPBS [12]. In this paper, we also describe how these queueing based, local RMS can be used in a planning based Grid RMS.

3. Problem Definition

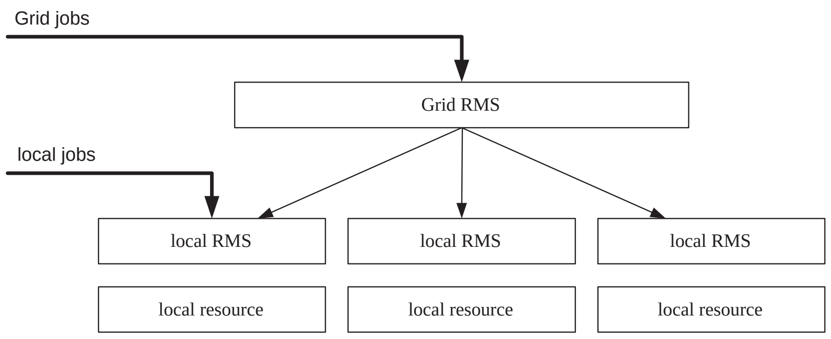

Figure 2. Job submission via Grid RMS and local RMS.

As described before, SLAs provide a powerful mechanism to agree on different aspects of service quality. Although SLAs cover various aspects, which are important for Grid applications, deadline guarantees are most important in this context [2, 4]. Two SLAs for the same aspect may differ in the strength of their defined service level as well as in the agreed prices and fines. The term *technical strength* in this context refers to the extent and quality of the requested resource usage. Informally, strength is defined as follows.

We assume two SLAs are comparable by their technical strength as follows: SLA A is technically stronger than SLA B if the service level required by B is included in the service level of A. For example, the service level "the submitted job gets three hours CPU time exclusively" is included in the service level "the submitted job gets three hours CPU time exclusively within the next two days." In the same way, an order based on the prices and fines can be found. As the combination of these two orders, i.e., technical strength and pricing-based, depends on specific needs of the Grid resource management systems (*Grid RMS*) operator, we assume further that there will be such a combination and refer to it as *strength* of the SLA.

A Grid RMS, such as the Virtual Resource Manager described later in this paper, uses the functions provided by the local resource management system (*local RMS*) to submit jobs, start jobs, and control the job execution (see Fig. 2). In the same way as the Grid RMS uses functions of the underlying local RMS, it relies on the SLA given by the local RMS when providing SLAs for Grid jobs. Besides the trivial case, where users demand no SLAs and the Grid RMS does not need to cope with SLA requirements, two cases can be distinguished by the strength of the SLAs provided by the local RMS and needed by the user (see Fig. 3).

In the first case, the local RMS provides the same kind and strength of SLA or a stronger SLA than the user requires. Thus, the Grid RMS can accede the required agreement with the user relying on the given SLA by the local RMS. In both cases mentioned before, the Grid RMS uses only functions provided by the local RMS and does not interfere with the autonomy of the local RMS.

In the second case, the local RMS provides no functionality to give an SLA or it is not able to guarantee the demanded service level. A simple solution is to avoid placing jobs onto the resources without SLA support. Using this solution, the number of resources available would be limited according to the SLA required by the user. Hence the acceptance of jobs i.e., the overall utilization of the Grid RMS domain is reduced. Moreover, the load on the resources may become unbalanced.

Another conceivable solution is to take over the control of the local RMS. This solution reduces the autonomy of the local RMS and thus, interferes with the policies defined by the operator.

In this paper, we describe how to overcome the problem of using resources which are not SLA-aware, but nevertheless guaranteeing an SLA to the user at the same moment. Different trade-offs between the

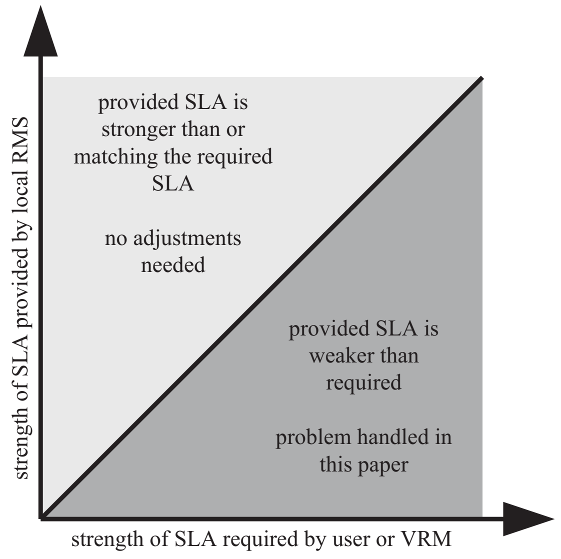
provided SLA is stronger than or matching the required SLA
no adjustments needed
provided SLA is weaker than required
problem handled in this paper
strength of SLA provided by local RMS
strength of SLA required by user or VRM

Figure 3. Different levels of SLA given by the local RMS and required by the Grid RMS generate two cases to handle.

autonomy of the resource operator and the service level to expect are examined.

4. Strategies for Autonomy Provisioning

A trade-off must be made between the degree of the support for SLAs coming from the Grid RMS layer and jobs submitted by local users.

As indicated before, whenever SLAs are not supported by the local RMS, we propose to add an additional software layer on top of the RMS which provides the required functionality and the enhancements necessary to support SLAs. This software layer needs to be configurable by the local administrators in a way that any requested level of autonomy can be implemented. In the following, we describe the conceivable levels of autonomy, ranging from complete control over the resources by the local administrator to full authority fo the Grid RMS, i.e., complete control over the local resource. The former case means in fact the absence of jobs submitted via the Grid, whereas the latter results in preempting local jobs whenever its resources are needed for the execution of a Grid SLA. This shows, the question of autonomy provision is directly related to the question of how well SLAs can be supported by a local RMS.

Between both extreme cases, a number of autonomy levels are conceivable (see Fig. 4). These levels can be implemented solely within the

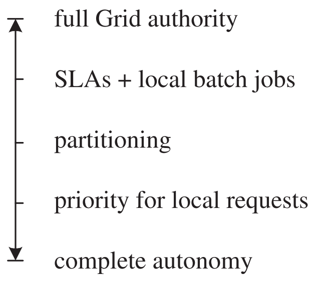

Figure 4. Levels of local autonomy in an RMS connected to an SLA-aware Grid environment.

additional software layer, which is actually a single software component under control of the local administrator (see Sec. 5.3). This interface regulates the way the Grid RMS may use the local resource for SLA provisioning based on policies which are defined by the administrator. Once the policies are defined, no further interaction of the administrator with the Grid RMS is required. The actual choice of the autonomy level may depend on various factors, such as business models, operator's policies, etc.

In the following, Grid jobs and local jobs are distinguished by the interface used for job submission. Local jobs are submitted via the respective interface of the RMS, Grid jobs are submitted using the Grid resource management system. This is depicted in Fig. 2.

4.1 Complete Autonomy

The trivial case where no Grid jobs are accepted by the local RMS requires no further discussion. Even the additional software is not required, i.e., the resource in question is invisible in the Grid. Jobs are submitted only via the interface of the local RMS.

4.2 Priority for Local Requests

In this case, Grid jobs are accepted by the local RMS although these jobs may not be supplied with any QoS guarantee. This means, the local RMS is capable of preempting Grid jobs when needed, e.g., in order to schedule local jobs with higher priority.

In order to implement this strategy, the additional software component on top of the local RMS needs access to the queue of the local RMS. This is necessary to determine whether or not a Grid job can be admitted in the first place, even if it is preempted later.

The highest flexibility is achieved when full access to the scheduler can be provided, i.e., the additional software component is capable of reordering jobs and changing priorities. Thus, Grid jobs can be prioritized lower than local jobs, Grid jobs can be removed from the queue and enqueued again later, and running Grid jobs are preempted or killed once their resources are required to run a local job. Another opportunity is to only use the local management interface to reorganize Grid jobs. In this case, the component only preempts or kills running jobs and removes jobs from the queue. Re-prioritization is not supported.

Using this strategy does not enable the Grid RMS to provide SLA support for a Grid job. It is only possible to run Grid jobs as long as no local load is present i.e., idle cycles are utilized. Whenever checkpointing facilities are provided either by the Grid job itself or the environment, Grid jobs can be preempted and resumed later.

4.3 Partitioning

The partitioning approach provides the highest level of fairness for both local and Grid jobs. Using this strategy, the resources managed by the local RMS are divided into two distinct parts, used more or less exclusively for either job type. The partitioning strategy provides full SLA support for Grid jobs within the respective partition.

Partitioning can be implemented either statically or dynamically, e.g., partitions can adapt to the actual load situation or differ during predefined intervals. For example, Grid jobs may remain on the resource if they have been reserved before all local jobs. As outlined in [7], partitioning is essential whenever jobs with unknown duration are present in a system, e.g., local jobs submitted via a queuing based local RMS. These jobs need to be collected in a separate partition, in order to enable reliable admission control and provide QoS guarantees.

4.4 SLA Guarantee with Local Batch Jobs

Counterpart of the strategy prioritizing local requests is the provision of full SLA guarantees. However, unused space can be filled with local batch jobs. In case local jobs are running and additional Grid jobs are submitted now interfering with local jobs, Grid jobs cannot be placed onto the resource.

Unlike partitioning, this strategy requires to submit the execution time - at least an estimate - together with each job, even local jobs, in order to assure reliable admission control. If the execution time of local jobs is unknown, they must be preempted or even be killed once they interfere with incoming Grid jobs requiring QoS.

4.5 Maximum Authority of the Grid RMS

Job submission is not possible using the local RMS configured with this strategy, i.e., local jobs are killed immediately from the queue. Reliability in this case refers not only to the opportunity to keep SLA guarantees for accepted jobs. In addition, the resource is entirely available to Grid jobs.

It would generally be possible to run local jobs unless they interfere with Grid jobs. However, when striving for maximum reliability for Grid jobs with SLAs, local jobs may compromise the integrity of the resource, e.g., by excessive usage of the network interconnect such that the performance of Grid jobs is affected. In order to avoid such situations as much as possible, one opportunity is to exclude local jobs from the usage of the resource. Local jobs may, however, be submitted using the Grid RMS.

4.6 Improvements

The granularity of the autonomy levels can be increased by defining less stringent rules in the additional software component. As an example, the allowance for a job may be exceeded for a certain period of time, e.g., as possible using dynamically changing partition sizes (see Sec. 4.3). Furthermore, combinations of different SLA support levels are conceivable, e.g., the actual level depends on the actual time. Thus, during night time SLA guarantees for Grid jobs may be supported whereas only local jobs may be allowed during day time.

5. Application Environment

In order to discuss the trade-off between the demands for autonomy and for supporting QoS guarantees in an actual management architecture, in this section the *Virtual Resource Manager* (VRM) is described. In particular, the components of the VRM are examined which deal with the QoS and SLA requirements.

5.1 VRM Overview

The task of the VRM is to provide high level services, e.g., co-allocation and combination of various local system resources. Using these services, the VRM is able to provide support for SLAs which itself can be seen as a higher level service. For this purpose, the VRM has to deal with different and distributed resources. Additionally, the VRM conceals the underlying structures and components of the local systems such that it is possible to provide *virtual resources*. This is done by establishing

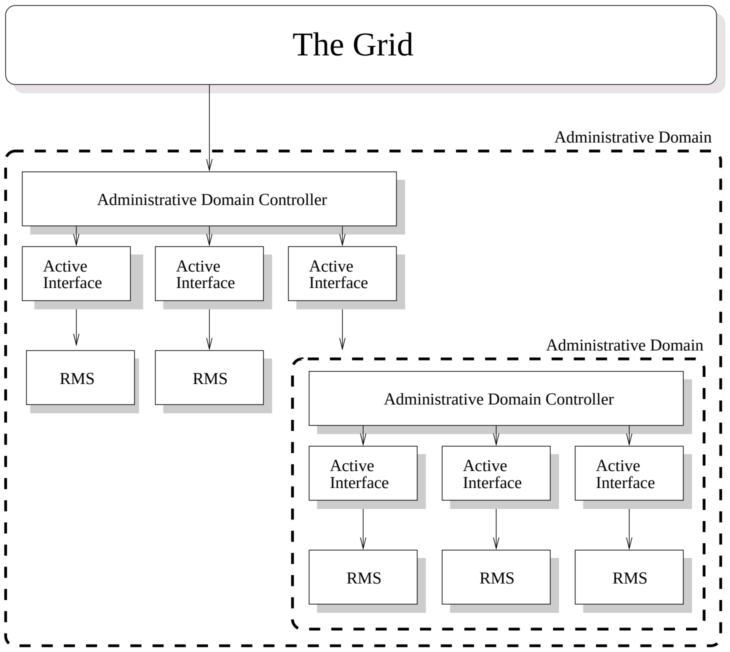

Figure 5. Hierarchical Administrative Domain Structure

so called *Administrative Domains* (AD). ADs may be seen as a pool of resources with the same set of global policies. Such global policies describe the visibility and accessibility of resources and how this access can be provided. Usually, these domains accord to companies in a business environment. In order to meet the demands for flexibility and adaptivity in research or business organizations a hierarchical subdivision for these policies is provided. While global policies, such as accounting or billing are handled uniformly for a whole AD, visibility and monitoring of local resources can be managed by the local administrator, too. This leads to a finer granularity of the set of policies as supporting only global approaches would render possible. As the VRM supports information hiding and provides more autonomy of the local administrative structures, a higher acceptance in integrating local resources into Grid infrastructures can be achieved.

Additionally, ADs may be nested to build up hierarchical structures of the ADs itself, so that resources within an AD may be combined to form virtual resources (see Fig. 5).

The VRM architecture comprises three layers (all described in the following sections) as depicted in Fig. 6. The different parts of the VRM

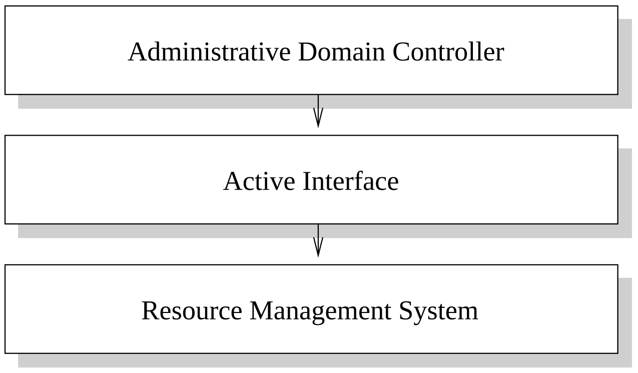

Figure 6. Layers of the VRM Architecture

that constitute the QoS management are present at any of the VRM layers.

5.2 Administrative Domain Controller

The task of the *Administrative Domain Controller* (ADC) is to build up the administrative domain and to promote it to the Grid. The ADC is responsible for the SLA negotiation, the co-allocation of the required resources. It also has the responsibility for jobs during their run-time as requested by an SLA.

Thus, the ADC serves as a gateway between the internal resources and the outside world. This requires the ADC to implement the global policies dealing with global system access, responsibility at runtime, accounting and billing. Additionally, the ADC is able to merge all of the local policies building a global policy on aspects defined by local resource management environment in cases this global view is needed.

In order to serve requests demanding a higher amount of resources than can be provided by a single local RMS, the ADC is able to join the physical resources to so-called *virtual resources*. This *virtualization* is a mechanism which allows to re-combine the physical resources to new resource classes. For example, we assume a site operating two clusters, one with 100 Opteron nodes and one with 50 IA32 nodes. The ADC is now able to establish a virtual resource comprising 150 IA32 nodes. This new resource class can be made available to internal and/or external users by means of the policies described above. Furthermore, in case of resource failures such virtual resources are useful to find alternative resources where a job affected by a failure may be resumed in order to keep a given SLA.

Beside virtualization it is also necessary to *combine* resource types in order to support SLAs. For example, the combination of the compute power of different nodes and the performance of the node interconnect is necessary to accept SLAs for parallel programs with high communication complexity. For large jobs with a huge amount of data it is also reasonable to combine the performance of the compute nodes at a specified time and the bandwidth of the external network to ensure that all of the data for the job is at place in time. Furthermore, it may be desirable to consider disk storage required by an application. An example is a teleimmersion application where visualization information, rendered by a cluster system, must be transferred in real-time over an external network to the visualization device, e.g. a cave. In this case, the availability of the compute resources, the network connection, and the visualization device is guaranteed by an SLA.

Before a new job is accepted, the contents of the SLA are negotiated with the ADC. In our implementation of the ADC the negotiation process is performed using the Web-services agreement specification defined by the Global Grid Forum [1].

In detail, the negotiation process works at follows: The ADC first checks which resources are able to comply with the SLA requirements according to their static properties, e.g., processor type. The ADC then starts a negotiation within the VRM framework (see Fig. 5). The internal negotiation may be less complex then the external, e.g., limited to the selection from a list of matching time intervals. The results of the internal negotiations are combined and returned to the external requester. The requester, e.g., a Grid based application, has then the opportunity to change its SLA request. The negotiation process iterates until the requester accepts or declines, or the ADC is unable to offer any other matching resource combination.

If a job is accepted, the ADC is in charge of complying with the terms of the corresponding SLA. Thus, the ADC maintains responsibility during run-time in the case of exceptions like resource failures. If a problem with any local resource arises that prevents the compliance with an accepted SLA, the local RMS first tries to solve the problem on its own. If this is not possible, a monitoring facility provides feedback to the related ADC. The ADC then tries to find spare resources in other internal RMS. If resources with the desired properties are not available, the ADC can either return the request to a higher layer in the hierarchy (see Fig. 5) or try to retrieve required resources within the Grid. The latter applies when the ADC resides at the top of the management hierarchy and is connected to other Grid RMS, e.g., Unicore.

The prototype of the ADC, as part of our implementation of the VRM framework, today just integrates parts of this functionalities. Actually, we focus on resource virtualization and job control. Our prototype deals with the challenge to integrate different information and access policies of the local RMS. This is achieved by building subsets of all policies defined by the local resource management systems in case an integration of all of these resources is needed. In other cases the ADC is able to provide different policies representing the local policies defined by the local RMS.

The ADC plays only a passive role concerning the specification of local autonomy as it only places jobs on RMS which support the desired level of QoS (see Sec. 4). The actual degree of QoS support for Grid jobs can be queried from the underlying software components, i.e., active interfaces as described in the following section.

5.3 Active Interfaces

Active Interfaces (AI) act as adapters between the demands of the ADC and the capabilities of the specific local RMS. The AI represents the software component configured by the local administrator as defined in Sec. 4. Hence, the actual level of local autonomy is defined within an AI.

As the SLA negotiation between the AI and the ADC is always the same, independent of the underlying RMS, the AIs contain not only the SLA negotiation functionality. Additionally, the AIs provide also the capability to either monitor the underlying components (in case of resource failures) or to accept feedback from an underlying RMS, if the RMS supports this. Such a feedback is provided to the ADC which then may react in order to avoid breaching an SLA.

In order to implement local policies, the administrator of the local RMS has to install and configure a specific instance of an AI. Thus, the local administrator keeps control of his own system at any time. The local policies describe to what extent the local resources are available for Grid jobs (see Sec. 4).

In addition, access policies may be tailored to specific user groups. For security reasons and customization, the ADC allows the administrator of the specific AI to define the granularity of the information that is published about this specific resource. If privacy is not a key issue, it is possible to publish all available information about the internal infrastructure. However, this is not reasonable in most cases. Based on the published information the external requester can decide, as part of the negotiation process, if the ADC is potentially able to fulfill the requirements of a job.

The VRM is designed to span over heterogeneous environments. This means the VRM framework has to be able to integrate different local RMS via the AI layer. Since the AIs ensure that all of the information and functionality necessary for implementing higher QoS can be provided to the ADC, the AIs do not simply wrap the underlying RMS but provide additional functionalities. For example, AIs add advance reservation capabilities if not supported by the local RMS.

With respect to the RMS functionalities, we have to distinguish between planning based and queueing based RMS.

Planning Based RMS. Since advance reservations are essential in order to implement most of the QoS mechanisms provided by the VRM, planning based RMS provide the foundation for such mechanisms. Examples for planning based RMS are the Computing Center Software CCS [11] or the network management system described in [3]. Using advance reservations, the VRM is capable to implement co-allocation of different resources and is able to provide guarantees like specific start times and deadlines for the completion of a job.

Queueing Based RMS. Any RMS that does not support planning of resource reservations, such as OpenPBS [12], must be enhanced by a specific AI in order to offer this functionality.

Similar to the considerations in [8], in such a case the AI keeps track of the allocations for the RMS. In detail, this works as follows: the AI keeps an internal data structure where all advance reservation requests are stored. These requests can be accepted up to the limit given by the capabilities of the actual resources. In order to implement advance reservations on top of such an RMS, only a single queue must be used and no preemption of running jobs is allowed. The AI then starts allocated jobs when they are due, according to the advance reservation schedule stored in the internal data structure of the AI.

The procedure previously described is analogue to the implementation of the *LRAM* described in [8]. However, the AI provides an extended functionality. It provides monitoring functionality, based on the monitoring tool provided by the RMS or by the AI itself. This is required in order to provide the ADC with status information about exceptions and failures of the underlying RMS.

The different QoS types rely on properties and capabilities of the underlying RMS or have to be provided by the related AI. Thus, when only a subset of these properties is implemented by the RMS or the AI, fewer

QoS Type	AI/RMS Functionality
Advance Reservation	Planning (Advance Reservations)
Malleable Reservation	Planning
Deadline Guarantee	Planning
First Interval Search	Priority Queue or Planning
Control of Long Running Jobs	Planning
Failure Recovery	Resource Monitoring, Planning[1]

[1] Remapping inactive jobs requires planning

Table 1. QoS types and required RMS functionalities.

QoS guarantees can be realized by the respective VRM infrastructure. The QoS types and the required capabilities are depicted in Table 1. It can be observed that advance reservations are essential for nearly any of the QoS types (see Sec. 3). Using the AI, it is possible to mix queueing and planning based RMS in our architecture on any of the QoS levels discussed.

In order to provide the full functionality of the VRM, it is essential that local resource allocations must also pass the AI which needs to be informed about the resource status at any point in time. If this cannot be guaranteed, local administrators may assign a certain partition of the resource to be controlled exclusively by the AI (see Sec. 4). Obtaining local administration responsibility was an important design decision of the VRM [4].

5.4 Local Resource Management Systems

The local RMS administered within an AD provide access to various kinds of physical resources. In the current setting, the ADC allows to access RMS for clusters [12] and *network management systems* [3]. Instead of such RMS, the management system accessed via the AI layer may also be another VRM. Thus, it is possible to build hierarchical VRM structures.

6. Conclusion

In future generation Grid environments, the provision of QoS guarantees for Grid applications will be an important factor.

We showed that there is a close relation between the degree of autonomy for local administrators and provisioning QoS guarantees for Grid applications. Especially, if local resource management systems do not provide SLA support, this is a useful and necessary feature in order to improve the utilization and thus, the profit of the resource. This can be achieved with only minor changes to the management infrastructure, i.e., only an additional active interface must be set up and configured. The configuration is subject to the level of autonomy desired by the local administrator. The different levels of local autonomy and their consequences for the achievable Grid application QoS were discussed.

In addition, the architecture of the virtual resource manager (VRM) was presented. The VRM provides SLA-aware management of Grid applications based on the autonomy schemes described in the paper. The behavior of the active interface, i.e., the extent to which local autonomy is retained, can be adjusted by local administrators and defines the extend of QoS guarantees given by the VRM.

References

[1] Global Grid Forum. http://www.ggf.org/, visited 05.01.2005.

[2] R. Al-Ali, K. Amin, G. von Laszewki, O. F. Rana, D. W. Walker, M. Hategan, and N. Zaluzec. Analysis and Provision of QoS for Distributed Grid Applications. *Journal of Grid Computing*, 2004.

[3] L.-O. Burchard. Networks with Advance Reservations: Applications, Architecture, and Performance. *Journal of Network and Systems Management, Kluwer Academic Publishers*, 2005 (to appear).

[4] 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.

[5] K. Czajkowski, I. Foster, C. Kesselman, V. Sander, and S. Tuecke. SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems. In *8th Intl. Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP), Edinburgh, Scotland, UK*, volume 2537 of *Lecture Notes in Computer Science (LNCS)*, pages 153–183. Springer, January 2002.

[6] T. DeFanti, C. de Laat, J. Mambretti, K. Neggers, and B. S. Arnaud. TransLight: A Global-Scale LambdaGrid for E-Science. *Communications of the ACM*, 46(11):34–41, November 2003.

[7] D. Ferrari, A. Gupta, and G. Ventre. Distributed Advance Reservation of Real-Time Connections. In *5th Intl. Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV), Durham, USA*, volume 1018 of *Lecture Notes in Computer Science (LNCS)*, pages 16–27. Springer, 1995.

[8] I. Foster, 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.

[9] The Globus Project. http://www.globus.org/, visited 05.01.2005.

[10] M. Hovestadt, O. Kao, A. Keller, and A. Streit. Scheduling in HPC Resource Management Systems: Queuing vs. Planning. In *Job Scheduling Strategies*

for Parallel Processing: 9th International Workshop, JSSPP 2003 Seattle, WA, USA, June 24, 2003 Revised Papers, 2003.

[11] A. Keller and A. Reinefeld. Anatomy of a Resource Management System for HPC Clusters. In *Annual Review of Scalable Computing, vol. 3, Singapore University Press*, pages 1–31, 2001.

[12] OpenPBS. http://www.openpbs.org/, visited 05.01.2005.

[13] Q. Snell, M. Clement, D. Jackson, and C. Gregory. The Performance Impact of Advance Reservation Meta-scheduling. In *6th Workshop on Job Scheduling Strategies for Parallel Processing, Cancun, Mexiko*, volume 1911 of *Lecture Notes in Computer Science (LNCS)*, pages 137–153. Springer, 2000.