Grids for Enterprise Applications

Jerry Rolia, Jim Pruyne, Xiaoyun Zhu, and Martin Arlitt

Hewlett Packard Labs, 1501 Page Mill Rd., Palo Alto, CA, USA, 94304

Abstract

Enterprise applications implement business resource management systems, customer relationship management systems, and general systems for commerce. These applications rely on infrastructure that represents the vast majority of the world's computing resources. Most of this infrastructure is lightly utilized and inflexible. This paper considers the role of Grid technologies in increasing the utilization and agility of enterprise infrastructure. Requirements of enterprise applications upon Grid resource management systems are introduced along with a discussion of how recent advances in resource virtualization make such management possible. We describe a Resource Access Management Framework for enterprise application infrastructure and identify the resulting requirements upon Grid service protocols.

1 Introduction

Grid computing is emerging as a means to increase the effectiveness of information technology. Grid technologies a) enable sharing of critical, scarce or specialized resources and data, b) provide wide area resource allocation and scheduling capabilities, and c) form the basis for dynamic and flexible collaboration. These concepts are enabled through the creation of virtual organizations. This paper is focused on the resource allocation and scheduling aspects of the Grid, particularly as they relate to applications and resource environments found in enterprises.

Until recently, the focus of Grid computing has been on support for scientific or technical job-based parallel applications. In such Grids, a *job's* requirements are specified in a Resource Description Language (RDL). Grid services, such as those offered by Globus [1], match the requirements of jobs with a supply of resources. A resource management system (RMS), such as Condor [2], schedules access to resources by job and automates their deployment and execution. Different kinds of resources can contribute to such Grids. For example, Grids may provide access to: the spare compute cycles of a network of workstations, a supercomputing cluster, a much sought after device, or to shares of a large multi-processor system.

Enterprise applications have requirements on infrastructure that are different and more complex than the typical requirements of parallel jobs. They may have *topology* requirements that include multiple tiers of servers and multiple networks, and may exploit networking appliances such as load balancers and firewalls that govern collaborations and other networking policies. Grids for enterprise applications require resource management systems that deal with such complexity. Resource management systems must not only govern which resources are used but must also automate the joint configuration of these compute, storage, and networking resources. Advances in resource virtualization now enable such cross-domain resource management systems [4]. Grid services may provide an open and secure framework to govern interactions between applications and such resource management systems across intranets and the Internet.

In this paper our contributions are as follows. We enumerate resource management requirements for enterprise applications and how they are addressed by recent advances in resource virtualization technologies. We propose a Resource Access Management (RAM) Framework for enterprise application infrastructure as an example of an RMS and present resulting requirements on Grid service protocols.

Section 2 elaborates on enterprise application requirements for resource management systems. We explain how advances in resource virtualization enable automated infrastructure management. A resource access management framework for enterprise application infrastructure is described in Section 3. Requirements upon Grid service protocols are presented in Section 4. Section 5 describes related work. Concluding remarks are offered in Section 6.

2 Background

In general, enterprise infrastructure is lightly utilized. Computing and storage resources are often cited as being less than 30% utilized. This is a significant contrast to most large scientific computing centers which run at much higher utilization levels. Unfortunately, the complex resource requirements of enterprise applications, and the desire to provision for peak demand are reasons for such low utilization. Configuration and resource requirement complexity can cause support costs for operations staff to exceed the cost of the infrastructure. Furthermore, such infrastructure is inflexible. It can be difficult for an enterprise to allocate resources based on business objectives and to manage collaborations within and across organizational boundaries.

Infrastructure consolidation is the current best practice for increasing asset utilization and decreasing operations costs in enterprise environments. *Storage consolidation* replaces host based disks with virtual disks supported by storage area networks and disk arrays. This greatly increases the quantity of storage that can be managed per operator. *Server consolidation* identifies groups of applications that can execute together on an otherwise smaller set of servers without causing significant performance degradations or functional failures. This can greatly reduce the number, heterogeneity and distribution of servers that must be managed. For both forms of consolidation, the primary goal is typically to decrease operations costs.

Today's consolidation exercises are usually manual processes. Operations staff must decide which applications are associated with which resources. Configuration changes are then made by hand. Ideally, an RMS should make such decisions and automate the configuration changes.

Recent advances in the virtualization of networking, servers, and storage now enable joint, secure, programmatic control over the configuration of computing, networking and storage infrastructure. Examples of virtualization features include: virtual Local Area Networks, Storage Area Networks and disk arrays that support virtual disks, and virtual machines and other partitioning technologies that enable resource sharing within servers. These are common virtualization features within today's enterprise infrastructure.

Programmable Data Centers (PDC) exploit virtualization technologies to offer secure multi-tier enterprise infrastructure on-demand [4]. As an example, Web, Application, and Database servers along with firewalls and load balancers can be programmatically associated with virtual Local Area Networks and virtual disks on-demand. Furthermore, additional servers or appliances can be made to appear on these networks or can be removed from the networks on-demand. Application topology requirements can be described in an RDL, submitted to a PDC and rendered on-demand. This makes the Grid paradigm feasible for enterprise applications [5].

The automation capabilities of PDCs can be exploited by an RMS in many ways. The first is simply to automate the configuration of enterprise infrastructure with the goal of decreasing operations costs. However, this automation can then be further exploited by the management system to increase the utilization of resources and to better manage the allocation of resources with respect to business objectives. Next, we consider the requirements enterprise applications impose upon such resource management systems.

Enterprise applications differ from parallel compute jobs in several ways. First, as we have already discussed, enterprise applications have different topology requirements. Second, enterprise applications have different workload characteristics. Often, a scientific or technical job requires a constant number of homogeneous resources for some duration. In contrast, enterprise applications require resources continuously over very long, perhaps undetermined time intervals. In addition, enterprise workloads have changes in numbers of users and workload mix which may result in time varying demands for resources, large peak to mean ratios for demand, and future demands that are difficult to predict precisely. Demands need to be expressed for a variety of resources that include: servers, partitions of servers, appliances, storage capacity, and the network bandwidths that connect them.

For an enterprise application, reduced access to resources degrades application Quality of Service (QoS). The result is that the users of the application face greater queuing delays and incur greater response times. The users do not in general repeat their requests in the future. Therefore, unsatisfied application resource demands on enterprise infrastructure are not, in general, carried forward as a future demand.

If an application must satisfy a Service Level Agreement (SLA) with its own users then it must also require an *access assurance* from the enterprise infrastructure RMS that it will get a resource it needs precisely when needed. For example, the assurance could be a probability θ with value 1 for guaranteed access or $\theta = 0.999$ or $\theta = 0.99$ for less assurance. Few applications always need an assurance of $\theta = 1$ for all of their resource requirements. An application may for example require n servers with an assurance of 1 and two additional servers with an assurance $\theta = 0.999$. Service level assurance is a QoS that can be offered by an RMS as a class of service to applications.

Enterprise applications require several kinds of QoS. These include: access assurance, availability, performance isolation, and security isolation. It is the role of a PDC RMS to provide such QoS to applications as they access the enterprise application infrastructure.

In the next section we describe an RMS for enterprise applications. It relies on a PDC to automate the rendering of infrastructure. We use the system to help derive requirements for Grid service protocols.

3 Resource Access Management Framework

This section describes our proposed framework for Resource Access Management (RAM) as an RMS for enterprise infrastructure. The framework includes several components: admission control, allocation, policing, arbitration, and assignment. These address the questions: which applications should be admitted, how many resources must be set aside to meet their needs, which applications are currently entitled to resources on-demand, which requests for resources will be satisfied, and which resources are to be assigned to each application. Answering such questions requires statements of expected application demands and required qualities of service.

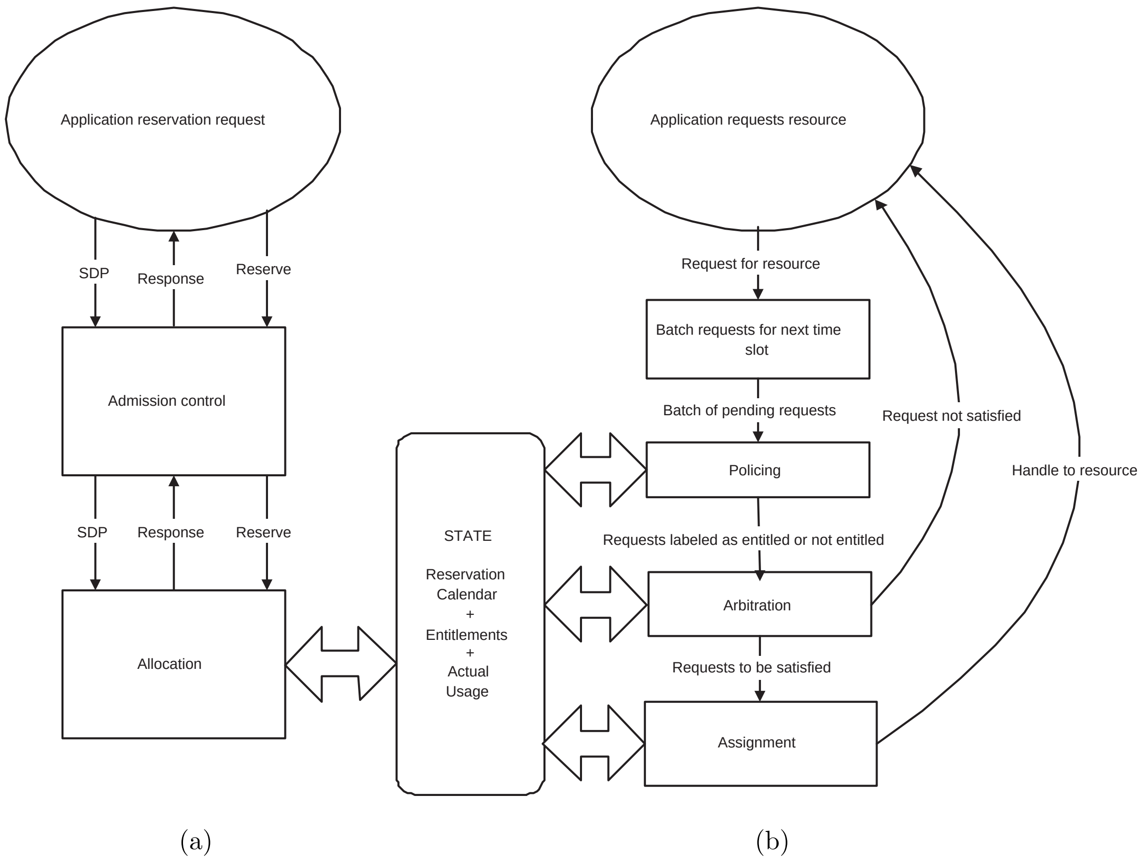
In our framework, we characterize time varying application demands on enterprise infrastructure resources using a *Statistical Demand Profile* (SDP). The profile describes demands for quantities of resources (including bandwidths), and qualities of service. The qualities of service are with respect to the application's requirements upon the enterprise infrastructure resources. An SDP is compact in that it is composed of patterns of demands, such as a typical weekday, a weekend day, or a day with exceptional demands, for example due to a typical end of month surge in sales orders. The profile is used by the resource management system to extrapolate demand over a capacity planning horizon.

We note that each application is responsible for the qualities of service it offers to its own users and that the relationship between application QoS and the expressed demands upon infrastructure is application specific. The specification of application QoS requirements and the corresponding mapping onto QoS requirements upon enterprise infrastructure is a mechanism for expressing the application owner's business objectives with respect to the allocation of infrastructure.

The profile is encoded in an RDL and submitted to the resource management system. Once admitted by the RMS, the application must interact with the system to acquire and release resources as it needs them to meet the QoS needs of its users.

The admission control and resource acquisition processes are illustrated in Figure 1. Figure 1(a) illustrates an application reservation request. The RAM framework checks to ensure that there are sufficient resources available to support the application with its desired QoS. This involves an interaction with an allocation system that determines how many resources must be set aside to support the application. A calendar maintains state information on expected and actual demand for hosted applications over a capacity planning horizon. Figure 1(b) illustrates resource requests from admitted applications. Resource requests are batched by the RAM framework. Remaining steps in the process decide which of the batched requests are honored. These are explained in subsequent sections of this paper. Finally, applications can always release resources, for compactness this process is not illustrated in the figure.

We note that this paper does not consider language support for describing an application's infrastructure topology. Our focus is on a characterization of demand that supports resource access assurance. The following subsections describe this aspect of SDPs in more detail, resource access assurance Class of Service (CoS) as an example of a QoS, and the components of our RAM framework along with the issues



Application reservation request
SDP
Response
Reserve
Admission control
SDP
Response
Reserve
Allocation
STATE
Reservation Calendar + Entitlements + Actual Usage
Application requests resource
Request for resource
Batch requests for next time slot
Batch of pending requests
Policing
Requests labeled as entitled or not entitled
Arbitration
Requests to be satisfied
Assignment
Request not satisfied
Handle to resource
(a)
(b)

Figure 1: Resource Access Management Processes

they address.

3.1 Statistical Demand Profiles and Statistical Assurance

This section describes our approach for characterizing time varying demands upon enterprise infrastructure resources. SDPs [10] represent historical and/or anticipated resource requirements for enterprise applications. Suppose there is a particular type of resource used by an application and that it has a typical pattern of use each weekday. We model the corresponding quantity of resources required as a sequence of random variables, $\{\mathbf{X}_t,\ t = 1, ..., T\}$. Here each t indicates a particular time slot within the day, and T is the total number of slots used in this profile. For example, if each t corresponds to a 60-minute time slot, and $T = 24$, then this profile represents resource requirements by hour of day.

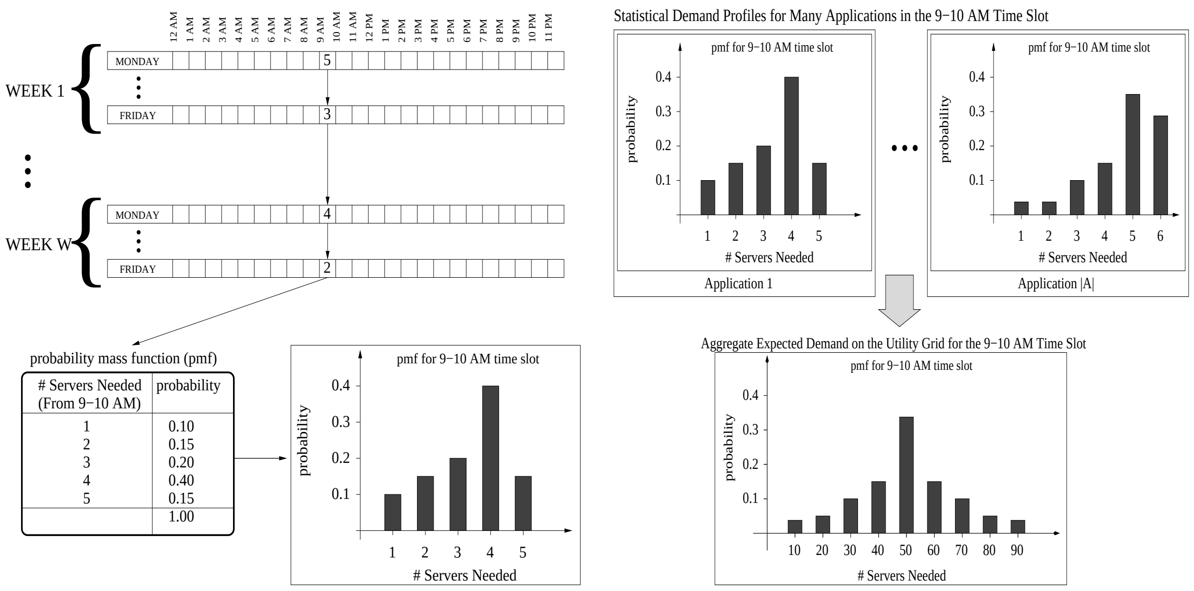
Our assumption here is that, for each fixed t, the behavior of $\mathbf{X}_t$ is predictable statistically given a sufficiently large number of *observations* from historical data. This means we can use statistical inference to predict how frequently a particular quantity of resources may be needed. We use a probability mass function (pmf) to represent this information. Without loss of generality, suppose $\mathbf{X}_t$ can take a value from $\{1, \ldots, m\}$, where m is the observed maximum of the quantity of required resources of a particular type, then the pmf consists of a set of probabilities, $\{p_k,\ k = 1, \ldots, m\}$, where $p_k = Pr[\mathbf{X}_t = k]$. Note that although m and p_k don't have a subscript t for simplicity of the notation, they are defined within each time slot. A demand profile is composed of sequences of pmfs, each characterizing the resource requirement $\mathbf{X}_t$ for a particular resource type in time slot t.

Figure 2(a) shows the construction of a pmf for a given time slot (9-10 am) for an SDP of an application. In the example only weekdays, not weekends, are considered. The application in this example required between 1 and 5 servers over W weeks of observation. Since we are only considering weekdays, there are 5 observations per week, thus there are a total of 5 W observations contributing to each application pmf. Figure 2(b) illustrates how the pmfs of many applications contribute to a pmf for a corresponding resource pool as a whole.

The aggregate demand on a resource pool for a particular resource type is modeled as a sequence of random variables, denoted as $\{\mathbf{Y}_t,\ t = 1, \ldots, T\}$. It is possible to estimate the distribution of the aggregate demand while taking into account correlations in application demands. The characterization of this *SDP of aggregate demand* enables the resource management system to provide statistical assurances to hosted applications so that they have a high probability of receiving resources when needed [10].

So far, we have described an SDP with respect to demand for a specific type of resource. This notion can be generalized to characterize demand for shares of a resource. It can also be augmented to include a characterization of multiple attributes of a resource type. For example, the bandwidth requirements to and from other resource types. In subsequent sections, we assume this more general definition of an SDP.

To summarize, a resource management system can extrapolate the demand from application SDPs onto its own calendar to obtain the SDP of aggregate demand upon its resource pools. If there are sufficient resources available, based on per slot statistical tests as described above, then the application is a candidate for admission.



# Servers Needed (From 9−10 AM)	probability
1	0.10
2	0.15
3	0.20
4	0.40
5	0.15
	1.00

(a) pmf of an application SDP (b) pmf of an SDP of aggregate demand

Figure 2: Statistical demand profiles

3.2 Resource Access Classes of Service

This section describes resource access classes of service for applications as an example of an RMS QoS for enterprise infrastructure. We assume that an application associates multiple classes of service with its profile. For example, the minimum or mean number of resources required may be requested with a very high assurance, for example with $\theta = 1$. Numbers beyond that may be requested with a lower assurance and hence at a lower cost. An application is expected to partition its requests across multiple CoS to achieve the application QoS it needs while minimizing its own costs.

We define the following access assurance CoS for resource requests:

- **Guaranteed:** A request with this CoS receives a 100 percent, i.e. $\theta = 1$, assurance it will receive a resource from the resource pool.
- **Predictable Best Effort with probability θ, PBE(θ)**: A request with this CoS is satisfied with probability θ as defined in Section 3.1.
- **Best Effort:** An application may request resources on-demand for a specific duration with this CoS but will only receive them if the resource management system chooses to make them available. These requests need not be specified within the application's SDP.

The guaranteed and predictable best effort classes of service enable capacity planning for the enterprise infrastructure. The best effort class of service provides a mechanism for applications to acquire resources that support exceptional demands or that may be made available at a lower cost via economic market based mechanisms.

Consider a set of applications that exploit a common infrastructure. Using the techniques of Section 3.1, for the same applications, we will require a larger resource pool for a guaranteed CoS than for a predictable best effort CoS. Similarly, larger values of θ will require larger resource pools. In this way CoS has a direct impact on the cost of resources needed to support the applications. Next, we consider the various components of our proposed framework as illustrated in Figure 1.

3.3 Admission Control

Admission control deals with the question of which applications are to be admitted by a resource management system. Admission control may be governed by issues that include:

- whether there are sufficient resources to satisfy an application's QoS requirements while satisfying the needs of applications that have already been admitted;
- risk and revenue objectives for the enterprise infrastructure; and,
- the goal of increasing asset utilization.

Only after an application has been admitted is it able to acquire and release resources.

3.4 Allocation

Allocation decides how many resources must be set aside to satisfy the reservation requirements of applications. The number of resources to be set aside depends on:

- application demands as expressed by SDPs;
- the requested access assurance CoS;
- the granularity of virtualization supported by the enterprise infrastructure; and,
- the topology of the enterprise infrastructure.

Within our framework, calendars keep track of allocations for the enterprise infrastructure. They are used to ensure that allocations do not exceed the expected availability of enterprise infrastructure. The granularity of virtualization supported by the infrastructure, in addition to application security requirements, may determine whether an offered resource is a partition of a physical resource. Allocation must also be sensitive to the topology of the infrastructure. Networking fabrics shared by applications must be able to satisfy joint network bandwidth requirements.

3.5 Policing

We assume that applications describe their expected time varying demands to the resource management system using an SDP. The resource management system uses the information for capacity planning and also to provide statistical assurances. To enable the latter capability, we must ensure that the applications behave according to their SDPs.

The policing component of the framework observes each request for a resource and decides whether the application is entitled to the resource. If it is entitled, then presumably, not satisfying the request will incur some penalty. The choice of which requests will be satisfied is deferred to the arbitration component.

In [11], we described an approach for policing. The approach considers resource access over short time scales, for example over several hours, and long time scales, for example over several weeks or months. Application SDPs are augmented with bounds on resource usage over multiple time scales. These are compared with actual usage to decide upon entitlement. We demonstrate the effectiveness of the proposed approach in a case study. The approach provides an effective way to limit bursts in demand for resources over multiple time scales. This is essential for our RAM framework because it offers statistical guarantees for access to resources.

3.6 Arbitration

Arbitration decides precisely which applications will have their requests satisfied. It is likely to:

- favor entitled requests (i.e. pass the policing test) to avoid penalties;
- satisfy requests that are not entitled if the risk of future penalties are low and if it is possible to charge extra; and,
- withhold a resource to make sure an application does not receive too high a quality of service.

Arbitration aims to best satisfy the needs of the applications and the goals of the enterprise infrastructure.

3.7 Assignment

Assignment decides which of the available resources are to be given to each application. Assignment must take into account the resulting impact of application loads on the performance of any infrastructure that is shared by multiple applications. Specific assignment decisions may also be made to best satisfy the objectives for the enterprise infrastructure. For example, assignment decisions may aim to minimize the use of certain servers so that they can be powered off.

In [9] we describe a mathematical optimization approach for assigning resources to a multi-tier application within a PDC. The method takes into account: the availability of server resources, existing network loads on the PDC's networking fabrics, and the bandwidth requirements between resources in the multi-tier application.

3.8 Summary

We have described our assumptions about the nature of information that must be offered to a resource management system for enterprise infrastructure, the kinds of problems that must be addressed by the system, and components that address them. Together, we refer to these as a Resource Access Management (RAM) framework for enterprise infrastructure.

In the next section, we consider how enterprise applications should interact with such a resource management system. Grid service protocols could be of value here to provide support for admission control and reservation, to acquire and release resources over time, to change reservations, and to deal with issues of multiple resource types.

4 Protocols for Enterprise Application Grids

In the previous sections, we discussed the requirements of applications in an enterprise environment. We also described the framework we have developed for performing resource management for these applications. The final component of a complete system is the external interface that permits customers to interact with the resource management system. Just as the requirements for enterprise applications drives new requirements on the resource management system, so too does it drive changes in how customers interact with resource management systems.

A principal reason for needing a new interaction model is the long running, and time varying demand nature of our application environment. These attributes give rise to a number of observations. First, because of the time varying demand, resource requirements will be complex. Some applications may have mid-day peaks in demand, others may peak in the mornings or near the end of month or business quarter. At times the peaks may overlap.

As discussed in earlier sections, an RMS must take into account such time varying demands when making admission control decisions. Furthermore, we expect that, often times, a resource management system will be unable to satisfy initial requests for resources with, for example, desired resource types or classes of service. This means that we require a negotiation process between the customer and the resource provider to allow the two parties to determine a mutually agreeable allocation pattern.

Second, because applications and their corresponding resource allocations are long-lived, customers of our system will continue to interact with the RMS throughout the application's lifetime. These interactions are needed to submit requests to acquire and release resources as characterized by demand profiles. We therefore require an interaction model that allows customers to request initial allocations of resources, and to alter those allocations, including canceling their use, as their needs change. Similarly, an RMS may fail to satisfy the allocation pattern at some point in time. It must notify the application, possibly in advance. Also, an application may not be returning resources as agreed. A negotiation is needed so that the RMS reclaims the most appropriate resources. In general, these negotiations are needed to guide the RMS so that it makes appropriate decisions.

Finally, the resource management system's planning horizon may actually be shorter than the expected lifetime of the application. That is, the application may be making requests farther into the future than the resource management system is willing to make guarantees at the current time. Therefore, the agreement between the parties may be subject to re-negotiation when the resource management system is prepared to consider new points in time.

Advance reservation is becoming increasingly common in traditional parallel processing domains [13, 20, 14], and we look to extend these current methods to suit the needs of enterprise applications as much as possible. In particular, SNAP [13], provides a SLA basis for resource management. This appears to be suitable for our environment, although additional issues and requirements are raised in the enterprise environment. We outline extensions to what has thus far been proposed, particularly in the area of Reservation Service Level Agreements (RSLA).

4.1 Properties of a RSLA

We use the RSLA to represent the agreement between the customer and the resource management system. We require that specification of this agreement be flexible enough to represent a wide variety of attributes that can be specified by either the customer or the resource management system. We use the following tuple to represent the RSLA:

$$< I, c, t_{dead}, < \{(r, SDP)^{Q}\}_{\tau} >_{R}> .$$

Much of this notation is based on SNAP. The first three fields are exactly as defined by SNAP: I is the identifier of the RSLA, c is the customer of the SLA, and t_{dead} represents the end time (deadline) for the SLA. The fourth field is the description of the requested resources in a particular RDL, R, used by the system. We allow for a set of resource types, r, each described by attributes such as physical characteristics like a CPU architecture. Each resource type has demand characterized by its SDP which includes policing information. The SDP provides information about each of a resource's demand attributes such as quantities of servers or bandwidth to and from other resource types. Each resource type and SDP pair also has QoS requirements, Q as described in Section 2. Topology requirements across all resource types are contained in τ.

We note that by simply setting $\theta = 1$, i.e. the guaranteed CoS within Q, for all of our demands, we have an agreement that requires guaranteed access to resources, so including the assurance in the RSLA specification need not weaken the original SNAP approach.

Enterprise applications often have complex dependency requirements among the resources they use. For example, multi-tier application architectures, such as those employed by many application servers are common. The wide use of virtualization technologies make it possible to render the required topology in a PDC environment. The topology description τ defines how the resources will be inter-connected. So, in the multi-tier example, we could describe a single logical Internet address provided by a load balancer for servers in the application tier connected by a private sub-net, to a back-end database layer consisting of machines of another logical type. Each type of resource, load balancers, application tier servers, and database servers are of different logical types.

SNAP captures the notion of multiple identical resources through an *Array* construct. However, the use of the SDP is central to our specification as described in Section 3.1. So, we make the SDP an explicit component of our RDL. Within the SNAP context, it is straightforward to make the SDP part of the Array in the form $SDP \times r \times C$, where r is a resource type in τ and C is the number of CoS used by the SDP, instead of the constant $n \times r$ notation in SNAP, where n is a specific number of resources.

A successful negotiation leads to an agreed allocation pattern. The attributes of the RSLA are used by the application to identify context when requesting and releasing resources at runtime and when making best effort requests for resources, and when a negotiation is needed so that the RMS can pre-emptively take back the most appropriate resources.

One of the desirable properties of the SLA abstraction for reservation is that it can be re-negotiated over time as needs or available resources change. However, in some cases, change in need may be known *a priori*, such as when a trend toward increased or decreased demand is known. The SDP can be

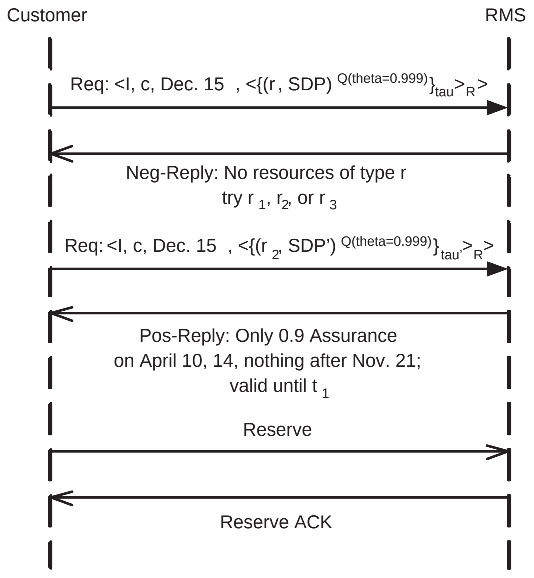

Customer
RMS
Req: <I, c, Dec. 15 , <{(r, SDP) Q(theta=0.999)}tau>R>
Neg-Reply: No resources of type r try r 1, r2, or r 3
Req: <I, c, Dec. 15 , <{(r 2, SDP') Q(theta=0.999)}tau'>R>
Pos-Reply: Only 0.9 Assurance on April 10, 14, nothing after Nov. 21; valid until t 1
Reserve
Reserve ACK

Figure 3: Example of negotiation process

augmented with trending information and used when extrapolating the profile onto the RMS's capacity planning horizon. As a simple example, one could state that the demand will increase 10% per month over the life of the SLA. Finally, we note that the SLA termination time, may be extremely large in an enterprise context. It may be determined by the resource management system due to its capacity planning horizon rather than by the customer as an expected job length as is commonly seen in today's reservation systems.

SNAP purposely avoids addressing issues relating to the auditing of an SLA, but does discuss dimensions such as pricing that may be subject to auditing. Our framework suggests two varieties of audits. The first is policing as discussed in Section 3.5. In this case, we know how to audit to determine that an application is staying within the agreed upon resource levels. The other dimension is assurance. The specified statistical assurance can be tested against actual performance. Such measurements are possible, but they make take a very long time when assurance levels are high, but not 1.

4.2 Negotiating a RSLA

A RSLA will in most cases be the result of a negotiation process. The customer and the resource management system can be expected to go through multiple iterations to reach a mutually agreeable set of parameters for the SLA. Ultimately, the negotiation is completed in a two-phase protocol where both sides commit to the results of the negotiation.

One possible negotiation scenario is shown in Figure 3. In this example, the customer contacts the RMS, and proposes an initial RSLA comprising the termination date, resource description, SDP, QoS including the desired level of service assurance, 0.999, and topology. The RMS cannot provide the type of resources requested, so proposes alternative resources of types r_1, r_2, r_3 in the form of a *Negative Reply.*

The negative reply indicates that no RSLA has been formed because the requirements could not be met. The consumer then alters its requirements to use resources of type r_2, updating the SDP and topology to match the capability of the selected resource type. This results in an RSLA which the RMS can render, but with a lesser QoS than was requested. In this case, the service assurance level will only be 0.9 on two specific dates, and no guarantee can be made at all after November 21 even though the request was for a lifetime through December 15. Although the conditions are different, the new RSLA is valid, so the RMS provides a *Positive Reply* with the limit that it must be accepted before time t_1. The RMS puts this time limit on the agreement so that it can negotiate with other customers without having to hold this reservation indefinitely. Here, we assume that the customer commits to the new RSLA within the specified time by sending a reserve message, and the RMS acknowledges that reservation.

Even when the SLA is agreed upon, we expect that either party may unilaterally need to change the parameters of the SLA. The conditions upon which this can occur must be agreed to within the original RSLA including possible penalties. This can be easily handled as a re-initiation of the negotiation protocol. This does put a requirement on the customer that it be available to re-negotiate at any time which is often not the case in current systems. The common model today is "submit and forget" where once a reservation has been made, the customer assumes it will be fulfilled. To enable re-negotiation, customers will require a persistent presence on the network with which negotiation can be initiated. The simplest approach is to provide an e-mail address that the RMS can send a message to requesting a new negotiation to occur. A variety of events may trigger this change from the resource management system's side, including failures of resources, arrival of a higher priority customer, or other arbitrary changes in policy that invalidate a current RSLA.

5 Related Work

There are many examples of application control systems that are emerging in the literature. In general, these are control systems that are coupled with applications that recognize when additional resources should be acquired or released and may determine how much an application is willing to pay for additional resources. In our context, a PDC is the resource provider for applications. This section describes examples of application control systems, related work on offering statistical assurance, and examples of PDCs.

First, we consider several examples of application control systems. With MUSE [6], Web sites are treated as services that run concurrently on all servers in a cluster. A pricing/optimization model is used to determine the fraction of cpu resources allocated to each service on each server. The over-subscription of resources is dealt with via the pricing/optimization model. When resources are scarce costs increase thereby limiting demand. Commercial implementations of such goal driven technology are emerging [15][16]. Levy *et al.* consider a similar problem but use RMS utility functions to decide how to deal with assignment when resources are scarce [17].

Ranjan *et al.* [8] consider QoS driven server migration for PDCs. They provide application level workload characterizations and explore the effectiveness of an online algorithm, Quality of Infrastructure on Demand (QuID), that decides when Web based applications should acquire and release resources from a PDC. It is an example of an approach that could be used by an application to navigate within its demand profile, i.e., to decide how many resources it needs in its next time slot.

Next, we consider statistical assurances for access to resources. Rolia *et. al.* consider statistical assurances for applications requesting resources from a RMS. They consider the notion of time varying demands and exploit techniques similar to those from the Network QoS literature [18][19] to compute assurance levels. Urgaonkar *et. al.* also recognize that significant resource savings can be made by offering less than a guaranteed level of service [12] in computing environments. However, they do not consider time varying demands or classes of service. We proposed the RAM framework, classes of service, and policing mechanisms in [11] and offered a study to evaluate the effectiveness of the policing methods.

Finally, we consider examples of PDCs. A PDC named the Adaptive Internet Data Center is described in [3]. Its infrastructure concepts have been realized as a product [4]. It exploits the use of virtual LANs and SANs for partitioning resources into secure domains called virtual application environments. These environments support multi-tier as well as single-tier applications. A second example is Oceano [7], an architecture for an e-business utility.

Work is being performed within the Global Grid Forum to create standards for Grid-based advance reservation systems in the Grid Resource Allocation Agreement Protocol (GRAAP) [20] working group. The group's goal is to make advance reservation systems interoperable through a standard set of protocols and interfaces. The group's approach is also influenced significantly by SNAP, and so is consistent with our approach and goals. We are working with this group to help insure the standards are applicable to enterprise grids as well as scientific grids.

6 Summary and Conclusions

Grid computing is emerging as a means to increase the effectiveness of information technology. Open standards for interactions between applications and resource management systems are desirable. Ideally such interaction methods can be unified for parallel and scientific computing and enterprise applications. Although the infrastructure requirements in the enterprise are often more complex, the advent of virtualization technologies and PDCs enable dynamic configuration of resources to meet these demands. The challenge becomes developing a resource management framework that meets the demands of enterprise applications. We have observed that these applications have long running times, but varying demand. This leads to under utilized infrastructures due to over provisioning based on peak demands.

Our solution is a resource allocation framework that permits customers to describe, probabilistically, their applications' demands for resources over time. We capture this behavior in a Statistical Demand Profile, and allocate resources based on a service assurance probability. We also augment the SDP with information needed to police applications which go outside their profile. We have shown that this approach leads to higher resource utilization for enterprise infrastructure [11].

Users interact with our system using an approach based on the Grid, and Reservation Service Level Agreements. The RSLA describes the agreement between the customer and the resource management system for resource utilization over time. Our formulation builds on existing work in this area, and extends it to support the topology, SDP, and service assurance used in our framework. We also base the RSLA formulation process on a two-phase negotiation protocol.

We are striving for a converged environment that supports equally scientific and enterprise applications. We believe that our approach is consistent with, and extends on-going work in the parallel scheduling community, so we believe this vision can be a reality.

References

[1] Czajkowski K., Foster I., Karonis N., Kesselman C., Martin S., Smith W., and Tuecke S.: A Resource Management Architecture for Metacomputing Systems. JSSPP, 1988, 62-82.

[2] Litzkow M., Livny M. and Mutka M.: Condor - A Hunter of Idle Workstations. Proceedings of the 8th International Conference on Distributed Computing Systems, June, 1998, 104-111.

[3] Rolia J., Singhal S. and Friedrich R.: Adaptive Internet Data Centers. Proceedings of the European Computer and eBusiness Conference (SSGRR), L'Aquila, Italy, July 2000, Italy, `http://www.ssgrr.it/en/ssgrr2000/papers/053.pdf`.

[4] HP Utility Data Center Architecture, `http://www.hp.com/solutions1/infrastructure/solutions/utilitydata/architecture/index.html`.

[5] Graupner S., Pruyne J., and Singhal S.: Making the Utility Data Center a Power Station for the Enterprise Grid, HP Laboratories Technical Report, HPL-2003-53, 2003.

[6] Chase J., Anderson D., Thakar P., Vahdat A., and Doyle R.: Managing Energy and Server Resources in Hosting Centers, Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles (SOSP), Oct. 2001.

[7] Appleby K., Fakhouri S., Fong L., Goldszmidt G. and Kalantar M.: Oceano – SLA Based Management of a Computing Utility. Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management, May 2001.

[8] Ranjan S., Rolia J., Zu H., and Knightly E.: QoS-Driven Server Migration for Internet Data Centers. Proceedings of IWQoS 2002, May 2002, pp. 3-12, Miami, Florida, USA.

[9] Zhu X. and Singhal S.: Optimal Resource Assignment in Internet Data Centers, Proceedings of the Ninth International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, pp. 61-69, Cincinnati, Ohio, August 2001.

[10] Rolia J., Zhu X., Arlitt M., and Andrzejak A.: Statistical Service Assurances for Applications in Utility Grid Environments. Proceedings of the Tenth IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems 12-16 October 2002, pp. 247-256, Forth Worth, Texas, USA.

[11] Rolia J., Zhu X., and Arlitt M.: Resource Access Management for a Resource Utility for Enterprise Applications, The proceedings of the International Symposium on Integrated Management (IM 2003), March, 2003, pp. 549-562, Colorado Springs, Colorado, USA.

[12] Urgaonkar B., Shenoy P., and Roscoe T.: Resource Overbooking and Application Profiling in Shared Hosting Platforms, Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI), Boston, MA, December 2002.

[13] Czajkowski K., Foster I., Kesselman C., Sander V.: Tuecke S.:, SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems, http://citeseer.nj.nec.com/538954.html.

[14] Jackson D., Snell Q., Clement M.: Core Algorithms of the Maui Scheduler. 7th Workshop on Job Scheduling Strategies for Parallel Processing, Cambridge, MA, June 2001.

[15] Sychron Enterprise Manager, http://www.sychron.com, 2001.

[16] Utility computing white paper, http://www.ejasent.com, 2001.

[17] Levy R., Nagarajarao J., Pacifici G., Spreitzer M, Tantawi A., and Youssef A.: Performance Management for Cluster Based Web Services, The proceedings of the International Symposium on Integrated Management (IM 2003), March, 2003, pp. 247-261, Colorado Springs, Colorado, USA.

[18] Zhang Z., Towsley D., and Kurose J.: Statistical Analysis of Generalized Processor Sharing Scheduling Discipline, IEEE Journal on Selected Areas in Communications, vol. 13, Number 6, pp 1071-1080, 1995.

[19] Knightly E. and Shroff N.: Admission Control for Statistical QoS: Theory and Practice, IEEE Network, Vol. 13, Number 2, pp. 20-29, 1999.

[20] Global Grid Forum GRAAP Working Group Web Page, http://www.fz-juelich.de/zam/RD/coop/ggf/graap/graap-wg.html.