Statistical Service Assurances for Applications in Utility Grid Environments

Jerry Rolia[1] Xiaoyun Zhu[1] Martin Arlitt[1] Artur Andrzejak[2]
[1]Internet Systems and Storage Laboratory
Hewlett Packard Laboratories
Palo Alto, CA 94304
{jerry_rolia, xiaoyun_zhu, martin_arlitt}@hp.com
[2]Zuse Institute Berlin (ZIB),
Takustraße 7, 14195 Berlin-Dahlem, Germany
andrzejak@zib.de

Abstract

In this paper we introduce techniques that support advance resource reservation and admission control for applications acquiring information technology (IT) resources from resource utilities. These resource utilities offer programmatic access to resources for complex multi-tier applications. Such utilities may participate in grids. As a workload, we consider business applications which require resources continuously but that have resource demands that change regularly based on calendar patterns such as time of day and day of week. Applications acquire resources as needed to ensure a quality of service to their end users and they release resources when they are not needed to lower their infrastructure costs. We characterize the resource demands of such applications statistically using application demand profiles. The profiles are used to make resource reservations. An admission control system exploits the profiles to enable the overbooking of resources while offering statistical assurances regarding access to resources. Different assurance levels correspond to alternative classes of service. Policing techniques determine whether requests for resources conform to a reservation and therefore whether they must be serviced. We illustrate the feasibility of our approach with a case study that uses resource utilization information from 48 data center servers. Simulation experiments explore the sensitivity of the assurances to correlations between application resource demands, the precision of the demand profiles, and the effectiveness of the policing mechanisms.

Keywords: Resource management, Grid computing, Utility computing, Business applications

1 Introduction

Grid environments offer programmatic access to information technology (IT) resources for applications. These environments have emerged to support the needs of the engineering and scientific communities. For example grid environments [6][14] may harness the unused compute capacity of engineering workstations within an organization or provide access to specialized resources such as supercomputer clusters that are shared by many scientific researchers. These grid environments provide: languages to specify the quantity of resources required by jobs, and, services for brokering, resource discovery and resource management.

To date, grid environments have focused on support for scientific and engineering jobs. There is a substantial literature regarding admission control, resource reservation, and scheduling for the support of these kinds of jobs for grids. Jobs are typically given a start-time/end-time window and a maximum job duration within the window. Peak requirements for resources such as cpus, memory, network bandwidth, and storage capacity are identified and reserved for the duration of the job. This has proven to be effective for the batch job style typically associated with engineering and science workloads. However in the future, grids are likely to support a more diverse set of applications [9].

In this paper, we consider resource management support for business applications. Many of these applications are likely to rely on grids supported by resource utilities. Resource utilities are data centers with tightly coupled resources that offer IT resources on demand. Jointly, we refer to these as *utility grids*. We loosely define *business applications* as those requiring resources on a continuous basis but with resource requirements that vary based on calendar patterns such as time of day and day of week. Examples include enterprise resource management systems, customer relationship management systems, and e-commerce storefronts. For such applications, workload patterns are often repeatable. However, future workload conditions and corresponding resource demands may only be known statistically. We expect such applications to exploit resource utilities via grids. Applications may reserve resources based on their expected demands and then acquire and release resources as needed based on their own current workload conditions.

For business applications, the peak and mean number of resources required can differ by a factor of 1.5 to 20 [2][17]. This, combined with uncertainty regarding precise knowledge of future demands, motivates the development of admission control mechanisms that exploit statistical multiplexing. In other words, there is an opportunity to overbook a utility's resources yet provide high statistical assurance, i.e., high probability, that resources will be made available to applications when they need them.

Our contributions in this paper are as follows. We present an approach for statistically characterizing demand profiles of business applications and a method that uses these profiles to provide statistical assurance regarding the number of resources needed to satisfy aggregate demand. Different assurance levels correspond to alternative classes of service. Policing techniques are described that decide whether an application's request for a resource exceeds its reservation. If so, the request need not be satisfied. We illustrate the feasibility of our approach with a case study that uses resource utilization information from 48 data center servers. Simulation experiments explore the sensitivity of the assurances to correlations between application resource demands, the precision of the demand profiles, and the effectiveness of the policing mechanism.

For the system under study we found that application demands were not highly correlated at the one hour time scale and that there were clear gains possible from statistical multiplexing. Our simulation experiments show that the technique did a good job at estimating the maximum aggregate number of resources needed by applications even with high (simulated) correlations in application demands. The technique was more sensitive to an accurate characterization of demand profiles. We show that the policing technique is effective in limiting bursts in demand that can limit our ability to offer service assurances.

Related work is discussed in Section 2. Section 3 formally defines application demand profiles and presents techniques for estimating the number of resources needed to support the aggregate demand of many applications. Section 4 describes classes of service for different levels of assurance. Policing methods are described in Section 5. Section 6 illustrates the feasibility of the techniques by applying them in a study involving servers from a data center. We explore the robustness of the technique using a sensitivity analysis. Section 7 gives summary and concluding remarks.

2 Related work

The grid community provides infrastructure for the support of scientific batch jobs in utility environments [14]. Jobs are typically described by their peak resource requirements, maximum job duration, start-time, and end-time. A job description is submitted to a resource manager. The resource manager uses resource availability information to decide whether it has sufficient resources to support the job. Current grid technologies rely on resource management systems such as LSF [15][16][24].

Advance reservation is appropriate for jobs that require access to a large number of resources, access to popular resources that are hard to obtain, or when resources are needed with certainty. With advance reservation, time is partitioned into slots. The slots form a calendar. Reservations typically start in the first available slot where all required resources are available.

We note that the above systems do not attempt to exploit the potential resource savings offered by statistical multiplexing while providing statistical assurances regarding resource availability. Statistical multiplexing may be appropriate for business applications because of their continuous operation, potential for large peak-to-mean ratios in resource demands, and a variable number of users. Resource savings lead to lower management costs for infrastructure since less infrastructure is required for the same number of applications. These savings may be passed onto the owners of the business applications. While reducing costs is important for business applications, their owners must have confidence they will have access to resources when needed. The potential advantages of statistical multiplexing for this class of applications has been noted in the literature [20][22].

There are resource management systems that aim to support business applications. We refer to these as *utility computing environments*. Broadly they fall into two categories. The first is a *shared server utility* model where server resources are exploited by multiple customer applications at the same time. The second is a more recent approach we define as a *full server utility* model where applications programmatically acquire and release entire servers as needed.

MUSE [5] is an example of a shared server utility. With MUSE, hosted Web sites are treated as services. All services 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 optimization model shifts load to use as few servers as possible while satisfying application level Service Level Agreements (SLA) for the services. 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 [7][21].

Our approach uses historical and/or anticipated load information to characterize an application's expected (time varying) resource requirements. These requirements are used to make advance resource reservations. This enables support for statistical multiplexing and corresponding statistical assurances regarding resource accessibility [20]. In our view, the utility is responsible for providing resources on demand with a particular level of assurance to its applications. The application must reserve resources with appropriate access assurance to offer its end-users the quality of service they require [8]. We believe this separation of concerns is practical for many kinds of business applications.

A full server utility named the Adaptive Internet Data Center is described in reference [18]. Its infrastructure concepts have been realized as a product [10]. 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 and can also contain systems that internally implement a shared server utility model. A second example of a full server utility approach is Oceano [1], an architecture for an e-business utility.

Ranjan *et al.* [17] consider QoS driven server migration within full server utility environments. 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 full server utility. 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.

There is a rich literature on statistical multiplexing with regard to Network Quality of Service [4][13][23]. Statistical multiplexing provides the potential for supporting more work with the same number of resources. The techniques provide statistical assurance, i.e., a probability, that routers will be able to support a large number of independent network traffic flows subject to delay and/or loss constraints. In general the techniques rely on either the central limit theorem or large deviation theory for estimating the number of flows that can be supported by a router with a given number resources and scheduling policy.

Our use of the central limit theorem is similar to the above. However our application resource demands are not expected to be independent. We explore the impact of correlations among application demands on the accuracy of our statistical assurance. Note that our approach for admission control with advance resource reservation applies to both styles of utility computing. For shared server models we can treat fractions of resources in the way we treat servers in server pools.

3 Application Demand Profiles and Assurances

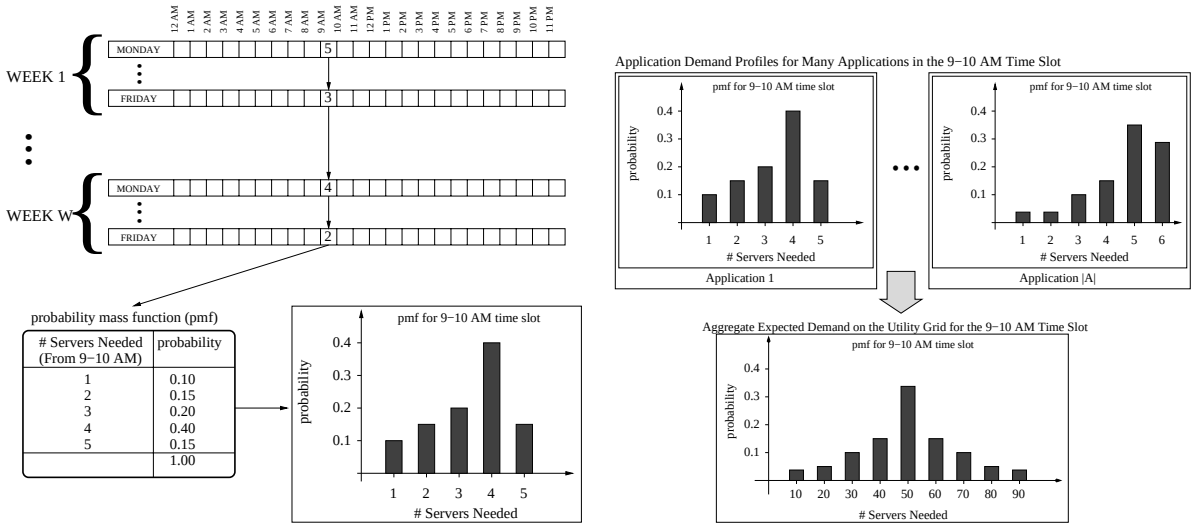
This section introduces Application Demand Profiles (ADP)s along with the statistical framework for estimating the number of resources needed to satisfy joint application resource demands.

3.1 Application Demand Profiles

ADPs represent historical and/or anticipated resource requirements for applications. Suppose there is a particular type of resource used by an application. We model its corresponding required number of resources as a sequence of random variables, $\{\mathbf{X}_t,\ t = 1, ..., T\}$. Here each t indicates a particular time slot, 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 number of resources may be needed. We use a probability mass function (pmf) to represent this information. Suppose $\mathbf{X}_t$ can take a value from $\{1, \ldots, m\}$, where m is the observed maximum of the required number of 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 time slot t and resource type.

Figure 1(a) shows the construction of a pmf for the weekday 9-10 am time slot for an ADP of an application. For this example, only weekdays are considered. The application required between 1 and 5 servers over W weeks of observation. Since there are 5 observations per week there are a total of $5\ W$ observations contributing to each application pmf. Figure 1(b) illustrates how the pmfs of many applications contribute to a pmf for the utility as a whole. As with applications, the ADP for the utility has one sequence of pmfs per resource type.



(a) A pmf of an application's ADP (b) A pmf of a utility ADP

Figure 1. Pmfs of application and utility ADPs

The advantages of the pmf approach are mainly two fold. First, it does not rely on a priori knowledge of the underlying distribution for $\mathbf{X}_t$. It can be estimated directly using a sample of independent observations, which can be obtained from historical measurement of the application demand. Second, compared to a mean-variance characterization alone, it offers more flexibility in terms of how it can be used. If the mean and variance of the demand are needed, they can be easily computed from the pmf. Furthermore the observed minimum and peak demands are also provided. Moreover, we will see later in the paper how the pmfs of individual applications can be used to bound the potential correlations between applications.

The pmfs in the ADP are estimated in the following way. Suppose $\{p_k,\ k = 1, \ldots, m\}$ is the "true" pmf for $\mathbf{X}_t$, which is unknown to us. What we have is a sample of N observations for $\mathbf{X}_t$. Let $\mathbf{Z}_k$ denote the number of observations that have a value k, and let $\hat{p}_k = \frac{\mathbf{Z}_k}{N}$. Then $\hat{p}_k$ gives us an estimate of the true probability p_k.

Inherent uncertainty exists in these estimated probabilities, especially when the sample size N is relatively small. This uncertainty can be bounded by computing confidence intervals for each estimated probability $\hat{p}_k$.

3.2 Statistical Assurance

The previous subsection introduced application demand profiles for applications. This subsection considers the number of resources needed by a utility to support many applications with a specific statistical assurance.

The utility has shared pools of resources that support its applications. The required size of a shared pool for a particular resource type is modeled as a sequence of random variables, denoted as $\{\mathbf{Y}_t,\ t = 1, \ldots, T\}$. As with applications, the random variables are used to define a *utility ADP* as sequences of pmfs, with one sequence per resource type, that describe the statistical behavior of its aggregate resource requirements for each time slot t. The construction of a pmf for a utility ADP is illustrated in Figure 1(b).

Let $A = \{1, 2, \ldots, |A|\}$ denote the set of indices for applications that require resources of the same type from the utility, where $|A|$ is the cardinality of the set A. Then at any particular time, the number of resources needed by the utility is the total number of resources needed by all the applications in A. Let $\mathbf{X}_t^a$ be the random variable for the number of resources needed by application a in time slot t. Then $\mathbf{Y}_t = \sum_{a=1}^{|A|} \mathbf{X}_t^a,\ t = 1, \ldots, T$. Based on this relationship, there are two ways to compute the utility ADP using the demand profiles for individual applications. The first is to compute the joint pmf of

application demands directly, which assumes mutual independence between application demands. The second is based on the Central Limit Theorem (CLT). It lets us consider correlations between application demands and is described here.

The CLT states that the sum of many independent random variables with finite variances tends to have a Normal distribution. Therefore, when the number of applications is large and their individual demands are independent, we can characterize the aggregate demand $\mathbf{Y}_t$ for each time slot t by its mean and variance, μ_t and σ_t^2. They are estimated in the following way.

Suppose the demand of application a for time slot t, $\mathbf{X}_t^a$, has a mean μ_t^a and a variance $(\sigma_t^a)^2$, which can be derived from its pmf. Then the mean and the variance of the aggregate demand for time slot t, $\mathbf{Y}_t$, can be computed as:

$$\mu_t = \sum_a \mu_t^a, \qquad \sigma_t^2 = \sum_a (\sigma_t^a)^2. \tag{1}$$

Hence, the distribution of $\mathbf{Y}_t$ is approximated by the continuous distribution $Pr[\mathbf{Y}_t \leq k] = \int_{-\infty}^{k} p_t(x)$, where

$$p_t(x) = \frac{1}{\sqrt{2\pi\sigma_t^2}} e^{-\frac{(x-\mu_t)^2}{2\sigma_t^2}}. \tag{2}$$

When the demands of individual applications are correlated we can still approximate the aggregate demand by a Normal distribution, but the quality of the approximation may be poorer due to the deviation of the real distribution from the Normal distribution. In addition, the variance of $\mathbf{Y}_t$ needs to be revised as follows:

$$\begin{aligned} \sigma_t^2 &= \textstyle\sum_a (\sigma_t^a)^2 + 2\sum_{a<b} Cov(\mathbf{X}_t^a, \mathbf{X}_t^b) \\ &= \textstyle\sum_a (\sigma_t^a)^2 + 2\sum_{a<b} \rho_t^{ab} \sigma_t^a \sigma_t^b, \end{aligned} \tag{3}$$

where $Cov(\mathbf{X}_t^a, \mathbf{X}_t^b)$ is the covariance between the demand of application a, $\mathbf{X}_t^a$, and the demand of application b, $\mathbf{X}_t^b$, in time slot t, and ρ_t^{ab} is the corresponding correlation coefficient. For any given pair of applications, ρ_t^{ab} cannot be computed solely from the pmfs in the ADPs. Instead, its evaluation requires access to the raw observations for $\mathbf{X}_t^a$ and $\mathbf{X}_t^b$, which typically will not be available to a resource utility resource management system when an admission control decision must be made. We propose the following two schemes for bounding and estimating correlations between applications.

Worst-case Bound:

Based on the observation from Equation (3) that the variance of the aggregate demand $\mathbf{Y}_t$ increases when the correlation between each pair of applications increases, a technique called *correlation bounding* is designed to find the highest possible positive correlation between two discrete random variables with known marginal distributions. The idea is to solve a linear programming problem on the joint pmf using the marginal pmfs as the constraints and the maximization of the covariance as the objective. Using this technique, we can compute the exact upper bound on ρ_t^{ab}, $\hat{\rho}_t^{ab}$, for any pair of $\mathbf{X}_t^a$ and $\mathbf{X}_t^b$ in each time slot t. Then an upper bound on σ_t, $\hat{\sigma_t}$, can be computed as:

$$\hat{\sigma_t}^2 = \sum_a (\sigma_t^a)^2 + 2\sum_{a<b} \hat{\rho}_t^{ab} \sigma_t^a \sigma_t^b. \tag{4}$$

This calculation provides an estimate of the variability of the aggregate demand in the worst case.

Measurement Approach:

In practice we expect the probability that every pair of applications are correlated to the maximum extent possible to be low, so decisions based on the above estimate tend to be pessimistic. An alternative in an online environment is to use an approach which monitors the pairwise application correlations, and computes a single parameter, ρ_t, that captures the impact of correlations on σ_t. ρ_t can be computed using raw observations of application demands as follows:

$$\rho_t = \frac{\sum_{a<b} Cov(\mathbf{X}_t^a, \mathbf{X}_t^b)}{\sum_{a<b} \sigma_t^a \sigma_t^b}. \tag{5}$$

Note that ρ_t expresses the magnitude of the aggregate correlation as it impacts the aggregate demand. To reduce the number of parameters that characterize application correlation, by substitution, we can rewrite Equation (3) with all pairs of applications having correlation coefficient ρ_t:

$$\sigma_t^2 = \sum_a (\sigma_t^a)^2 + 2\rho_t \sum_{a<b} \sigma_t^a \sigma_t^b. \tag{6}$$

Let ρ be the maximum of ρ_t over all time slots t. Then the parameter ρ can be used as a calibration factor for the impact of correlations on the aggregate demand.

For admission control tests for resource utilities in operation we expect to use a combination of the above two approaches. When an application is first submitted to the utility, correlations between the new application and applications already within the utility can be estimated using the pessimistic correlation bounding approach. As the new application executes, its correlation with other applications can be observed and gradually used to replace the pessimistic bound. We expect to explore this issue and whether ρ can be considered as a resource utility invariant for specific kinds of workloads in our future research.

The characterization of the utility ADP enables the utility to provide statistical assurances to hosted applications so that they have a high probability of receiving a resource when it is needed. In any time slot t, suppose the number of resources in a particular resource pool is Γ_t. We define the *statistical assurance* θ to be the expected probability that a single resource is available to an application that needs it as:

$$\theta(\Gamma_t) = E[min(\frac{\Gamma_t}{\hat{\mathbf{Y}}_t}, 1)], \tag{7}$$

where $\hat{\mathbf{Y}}_t = min(\mathbf{Y}_t, M_t)$, and M_t is the peak demand of $\mathbf{Y}_t$ computed as the sum of peak demands of individual applications. Our motivation is as follows. Recall that $\mathbf{Y}_t$ represents the aggregate demand on a resource by all the applications. If $\mathbf{Y}_t > \Gamma_t$, among the $\mathbf{Y}_t$ resources that are needed, only Γ_t of them can be served by the utility. So the satisfaction rate is $\frac{\Gamma_t}{\mathbf{Y}_t}$. If, on the other hand, $\mathbf{Y}_t \leq \Gamma_t$, then the satisfaction rate is 100%. Note that $\mathbf{Y}_t$ is cut off at M_t because the aggregate demand should not exceed M_t based on the pmfs in the ADPs.

Using the CLT approach, for any given value of Γ_t, θ can be computed as

$$\theta(\Gamma_t) = \int_{-\infty}^{\Gamma_t} p_t(x)dx + \int_{\Gamma_t}^{M_t} \frac{\Gamma_t}{x} p_t(x)dx + \frac{\Gamma_t}{M_t} \int_{M_t}^{\infty} p_t(x)dx, \tag{8}$$

where $p_t(x)$ is defined in Equation (2).

Conversely, given any desired value for θ, we can determine the corresponding required number of resources Γ_t for each time slot t. Let $\Gamma = \max_t \Gamma_t$. Then Γ is the required size of the resource pool in the utility so that the targeted assurance level θ can be achieved at all times.

4 Resource Access Classes of Service

This section describes resource access classes of service as an example of a QoS for a resource pool. We assume that an application associates multiple classes of service with its profile. For example, the time varying minimum or mean resource capacity that is required may be requested with a very high assurance, for example with probability $\theta = 1$. Capacity 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 classes of service for requests for a resource from a resource pool:

- **Guaranteed:** A request with this CoS receives a 100 percent, i.e., $\theta = 1$, assurance it will receive the requested resource.
- **Predictable Best Effort** with probability θ, or **PBE(θ):** A request with this CoS is satisfied with a certain assurance θ where $0 < \theta < 1$.
- **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 – there is no assurance, $\theta = 0$. These requests need not be specified within the application's ADP at the time of admission control.

The guaranteed and predictable best effort classes of service enable capacity planning for the system's resources. The best effort class of service provides a mechanism for applications to acquire resources that support exceptional demands. They may be made available via economic market based mechanisms [5].

5 Policing and Entitlements

ADPs provide a mechanism for characterizing expected resource requirements for each time slot t. They are appropriate for sizing the resource pools needed by the utility. However to provide the expected levels of statistical assurance, applications must behave according to their ADPs. Though a pmf of an ADP limits the maximum number of resources an application is entitled to for its corresponding time slot, we still require a method to ensure that each application adheres to its pmfs over time. The purpose of our *policing* mechanism is to specify an application's entitlement to resources over both short and longer time scales and to recognize when an application exceeds these entitlements. Requests that are not entitled to resources can then be dealt with in a systematic manner.

For short time scales, for example over several hours, a *horizontal profile* characterizes the percentage of resource usage H_t^w that may occur over a sliding window of w slots starting at time slot t, for one or more values of w, for all time slots. For example, H_t^w for time slot t = 7am to 8am and $w = 3$, describes the interval between 7am and 10am on the same day. If the maximum possible demand for the interval, based on the pmfs of application's demand profile, is Δ units, then the horizontal entitlement H_t^w is a percentage of Δ, for example 80%.

For longer time scales, for example over weeks and months, a *vertical profile* characterizes the percentage of resource usage V_t^w that may occur over a sliding window of w instances of the same time slot t, for one or more values of w, for all slots. For example, an application may be entitled to a total of Δ resource units over O instances of a 9am to 10am time slot. The vertical profile of V_t^w may specify that no more than 50% of the Δ units may be consumed in $w = O/3$ successive instances of a slot and no more than 100% of Δ units may be consumed in O successive instances of the slot. In this way the vertical component captures medium to long-term entitlements.

We define the H_t^w and V_t^w for an ADP's time slots for multiple values of w as *entitlements* for the ADP. We use the entitlements to implement our *policing mechanism* that governs per-interval requests for resources. If an application requests more resources than it is entitled to over any of the values for w then the application's demand goes beyond the capacity set aside by the reservation. These requests must be dealt with according to some policy. For example, we may reject the surplus requests, without contributing to the utility's service level violations, or treat them as best effort.

The entitlements should be based on automated observation of the application's demand behavior. They should be derived while constructing other aspects of the ADPs.

Lastly, it could be the case that an application acquires resources but holds them longer than expected thereby exceeding its next time slot's H_t^w or V_t^w for some w. In this case the application may be required to release some resources. We refer to this as *clawback*.

6 Case Study

This section presents a case study involving cpu utilization from 48 servers in a data center. The purpose of the study is to evaluate the effectiveness of our techniques.

6.1 Measurement Data

For the purpose of our study we were able to obtain cpu utilization information for a collection of 48 servers. The servers have between 2 and 8 cpus each, with the majority having either 4 or 6 cpus. The data was collected between September 2, 2001 and October 20, 2001. For each server, the average cpu utilization across all processors in the server was reported for each five minute measurement interval. This information was collected using MeasureWare (Openview Performance) Agent [11].

We *interpret* the load of each server as an application for a full server resource utility environment. Whereas the groups' servers have multiple cpus in our study we assume the utility has many servers with one cpu each. If a server only required one cpu in an interval then its corresponding application requires one server in the utility. If the actual server required four cpus in an interval then its corresponding application requires four servers in the utility. We exploit the fact that changes in server utilization reflect real changes in required activity for its applications. Our applications have the same changing behavior. However since our purpose is simply to validate our techniques we are not concerned with whether it is feasible for the loads on the multi-cpu servers to be hosted on many single cpu servers. Similarly we do not scale the loads with respect to processor speed and/or memory capacity.

We define the number of cpus required by a server based on a per-cpu target utilization $\hat{u}$. We consider a target $\hat{u} = 0.5$ as it is likely to be appropriate for interactive work. A target of $\hat{u} = 0.8$ may be acceptable for latency insensitive loads.

The required number of cpus k of a server for a measurement interval is computed as follows:

$$k = max(\lceil \frac{U\ \hat{k}}{\hat{u}} \rceil, 1),$$

where U is the average utilization of the server's $\hat{k}$ cpus. We assume that at least one cpu must be available at all times. For the corresponding application, k is therefore the number of single cpu servers required for the interval.

6.2 Application Demand Profiles

For this case study, we chose to characterize the application ADPs by weekday and by hour of day. We consider all weekdays to be equivalent, and we omit data from weekends. As a result, our profile consists of 24 one hour time slots. Since we have utilization data for 35 days, there are 35 data points that contribute to each pmf in each application's ADP.

The original measurement data was gathered at 5 minute intervals. When constructing pmfs for time slots with duration greater than 5 minutes we require one extra processing step. Consider an interval of duration b with 5 minute sub-intervals $a_1 \cdots a_j$, where j is an integer such that $j = \frac{b}{a}$. The number of servers required for b is chosen as the maximum of the number of servers required over sub-intervals $a_1 \cdots a_j$. The number of servers required for b is used to construct the pmf for its corresponding time slot. For the purpose of our study we consider time slots that are one hour in duration. They are constructed from measurement intervals of duration $b = 60$ minutes.

Table 1 lists three consecutive pmfs for a randomly selected application. In time slot t, the probability of requiring one server is 0.14; in time slot t + 1 the probability of requiring one server is 0.11. For each time slot, the sum of probabilities adds to one.

Time Slot	1 server $\hat{p}_1$	2 servers $\hat{p}_2$	3 servers $\hat{p}_3$	4 servers $\hat{p}_4$
t	0.14	0.00	0.66	0.20
t + 1	0.11	0.03	0.75	0.11
t + 2	0.14	0.00	0.83	0.03

Table 1. Subsequence of pmfs for a randomly selected application, $\hat{u} = 0.5$, with 60 minute time slots

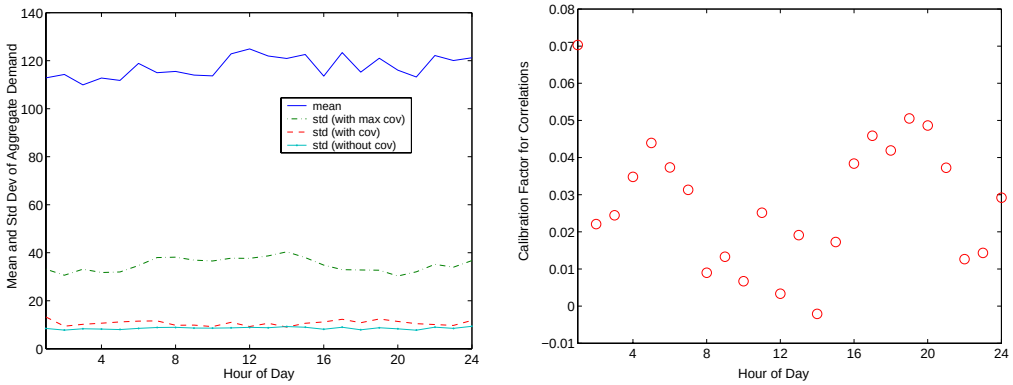
6.3 Utility ADP

In this section, we verify that the aggregate number of resources required per time slot t appears to be Normally distributed. Once verified, we are able to take the ADPs of applications and use the CLT approach to compute the number of servers needed (Γ) to provide resources at specific levels of assurance (θ).

First, we consider whether the aggregate load for each interval t appears to be Normally distributed. The Bera-Jarque test [3][12] evaluates the null hypothesis that a given series of data, in our case $\mathbf{Y}_t$, has a Normal distribution with unspecified mean and variance against the alternative that $\mathbf{Y}_t$ does not have a Normal distribution. We found that the aggregate number of servers required per interval does indeed appear to behave in a similar manner as randomly chosen data from the Normal distribution. These results are not shown in this paper.

Next, we estimate the parameters of the Normal distribution (μ_t and σ_t) for each time slot t. Figure 2(a) uses raw data from the system under study to illustrate the mean and standard deviation for the utility ADP. The top curve in Figure 2(a) represents the mean (μ_t) of the total demand over time, and the bottom two curves are the standard deviation (σ_t) with and without the covariance term in the computation. The middle curve represents $\hat{\sigma}_t$, the maximum value for the standard deviation computed using the correlation bounding method. Figure 2(b) shows the value for ρ_t obtained directly from the raw data. In this case ρ, the maximum of ρ_t over all time slots, is 0.07, which happens at $t = 1$. This indicates a relatively low level of correlation between applications. Accordingly, the increase in σ_t after adding the covariance is not large relative to the value of the mean. Figure 2(a) also shows that assuming the worst case for correlation (std with max cov) can be very pessimistic.

The above parameters are used with Equation (8) to compute Γ for several values of θ. The results are summarized in Table 2. The first column lists the different values for θ, and the rest of the columns show the corresponding values for



(a) μ_t and σ_t (b) ρ_t

Figure 2. Parameters for the utility ADP

Γ, the number of servers required by the utility to provide a service assurance of level θ. The second column uses the standard deviation σ_t without considering correlations, while the third column is similar but σ_t incorporates the covariance term computed using raw data. The results of these two columns are similar, showing that the amount of correlation between applications and its impact on the aggregate demand for the utility is small for the system under study. We do note that correlation has a greater impact on the higher levels of assurance θ.

To better assess the advantages of statistical multiplexing for the system under study, we choose to add a migration overhead to the demand profiles of applications and evaluate its impact on the utility ADP. Given that applications acquire and release server resources, the original traces of server requirements from the raw data are augmented in the following way. If an additional server is required in a time slot t, then the request is made for the server in time slot $t-1$. This provides time to migrate the server into the application. Similarly if a server is no longer needed in time slot t, it does not become available to other applications until time slot $t+1$. This provides time to migrate the server from the application. We note that with 60 minute time slots this is very pessimistic. However this is still optimal from the application's point of view, in that a server is always ready when it is needed, and as such represents a best case scenario for statistical multiplexing. The augmented traces contribute to the pmfs in the ADPs, which in turn contribute to the utility ADP that is used to compute Γ.

The numbers in the last column of Table 2 take into account both application correlations and the migration overhead. The comparison between the last two columns show that the extra number of servers required due to overhead is roughly 15-20%. For a more fair comparison with the static allocation of resources, the application demand profiles used in the next subsection include the migration overhead unless stated otherwise.

θ	Γ no correlation	Γ with correlation	Γ with correlation and overhead
0.80	101	101	117
0.90	113	113	133
0.99	131	134	158
0.999	141	147	172

Table 2. Comparison of assurance (θ) vs. number of servers required (Γ)

6.4 Sensitivity Analysis

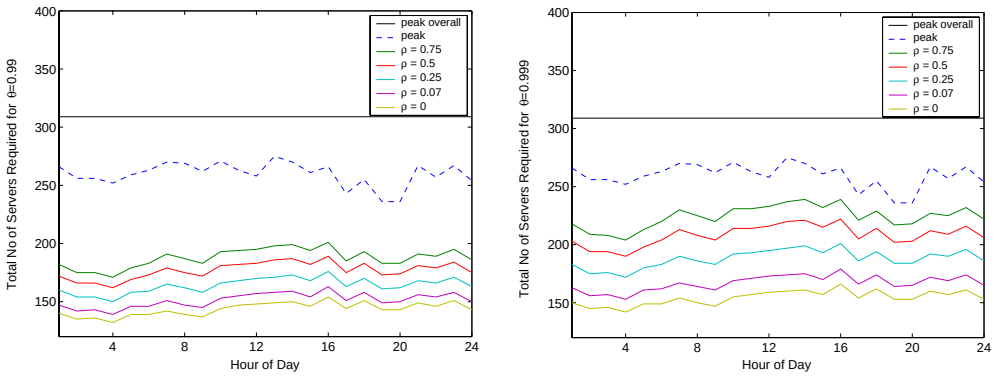
This section considers further the sensitivity of our estimate for the number of servers required to correlations in application demands. It then uses a simulation environment to validate the analytic approach. We then explore the accuracy of the statistical assurance with respect to correlations in application demands and the precision of the pmfs in application ADPs.

6.4.1 Sensitivity of Γ to correlations between application demands

In this subsection we use our analytic technique to investigate the relationship between statistical assurance θ, correlations between application demands ρ, and the required number of resources per time slot Γ_t.

Figure 3 shows the expected value for Γ_t for $\theta = 0.99$ and $\theta = 0.999$. The top curve (solid horizontal line) corresponds to the scenario where each application is always allocated its maximum required number of servers. This is the only case where the migration overhead is not included because it represents the static allocation of resources. In this case, the number of servers needed by the utility is 309 servers. The second curve (dashed line) illustrates the scenario where the peak number of resources m_t^a are allocated to each application a for each time slot t. For this scenario the utility as a whole needs 275 servers to satisfy its aggregate demand for all the time slots. This demonstrates that for this system under study the potential savings due to time sharing but without statistical multiplexing is about 10% compared to a static resource allocation.

The remaining curves correspond to the scenario where each application acquires only its *required* number of servers. For these cases statistical multiplexing comes into play. Its benefit varies with the degree of correlation between applications demands. Using the correlation coefficient ρ defined in Section 3, we computed the values for Γ_t as the level of correlation decreases from 0.75 down to 0. For this data, $\rho = 0.75$ corresponds to the case where the correlation between each pair of applications is at its highest level for our system under study. In this case, the number of servers required by the utility is 201 and 239 for assurance levels of $\theta = 0.99$ and $\theta = 0.999$, respectively. We note that even in this worst-case scenario for correlations, statistical multiplexing offers advantages over peak allocation (27% and 13%, respectively) and static allocation schemes (35% and 23%, respectively) for the system under study. As the degree of correlation goes down, the aggregate resource requirement for the utility decreases accordingly. The bottom curve represents the case when demands of all applications are mutually independent, where the benefit of statistical multiplexing is the greatest. From Figure 2(b), we see that $\rho = 0.07$. For this value the advantages of statistical multiplexing are clear for the system under study.



(a) $\theta = 0.99$ (b) $\theta = 0.999$

Figure 3. Aggregate server requirements for several correlation levels

6.4.2 Validation by simulation

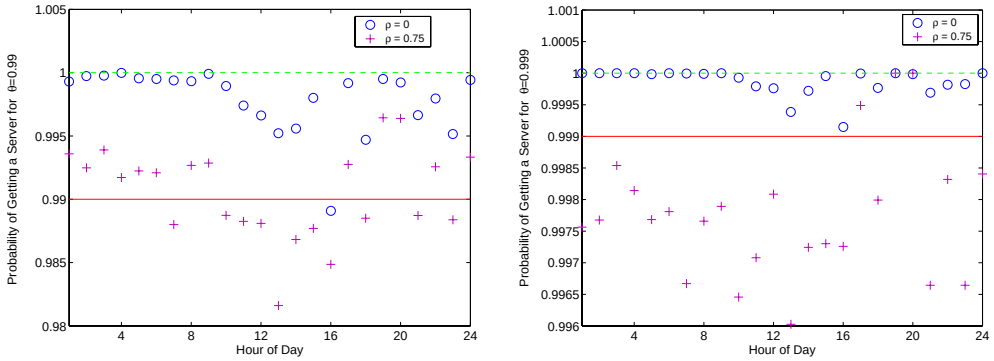
We use a simulation environment to help validate our analytic results. For each application, the simulator generates traces of requests for the required number of servers in successive time slots. For each time slot the simulator draws a uniform pseudo-random number to index into the application's corresponding pmf. Correlation among the load of applications is introduced by correlating the uniform random number streams that index the application pmfs. Once the traces are prepared they are traversed on a slot by slot basis.

The chosen values for θ and ρ, as illustrated in Figure 3, are used with Equation (8) to give values for Γ for the simulations. When processing the traces, for each slot, we measure the fraction of total server requests that are satisfied. This gives the achieved value for θ for each experiment. For example, if applications require a total of 100 servers in an instance of a time slot but only 99 are available then 0.99 contributes to the value achieved for θ as averaged over all simulated instances of that time slot.

We consider two scenarios for validation. In the first case application demands are independent, i.e., $\rho = 0$. The uniform random numbers that index different application pmfs are mutually independent. In the second case they are maximally

correlated, which corresponds to $\rho = 0.75$. This is accomplished by using the same sequence of uniform random numbers to index all application pmfs. For each scenario we consider θ as 0.99 or 0.999. Based on our analysis, for $\theta = 0.99$, we compute the number of servers required, Γ, as 154 and 201 for $\rho = 0$ and $\rho = 0.75$, respectively; for $\theta = 0.999$, we compute $\Gamma = 166$ and 239 for $\rho = 0$ and $\rho = 0.75$, respectively. These numbers were used for the simulation.

For the simulation, we generated 1,000 weekdays worth of observations. The results are shown in Figure 4. When application demands are independent, the estimated value for Γ is nearly always sufficient to provide the assurance level θ, with the exception of the violation for $\theta = 0.99$ at $t = 16$. However, when application demands are very highly correlated the results become poorer; this is the case despite that we take into account the correlations when estimating Γ by augmenting the variance of the aggregate demand. This is mainly due to the deviation of the aggregate distribution from the Normal distribution in the high correlation case. Nevertheless, when we specify an assurance level of $\theta = 0.99$ we actually provide an assurance level of 0.98, when we specify an assurance level of $\theta = 0.999$ we provide an assurance level of 0.996. This is encouraging as it is the worst case for correlation for the system under study.



(a) $\theta = 0.99$ (b) $\theta = 0.999$

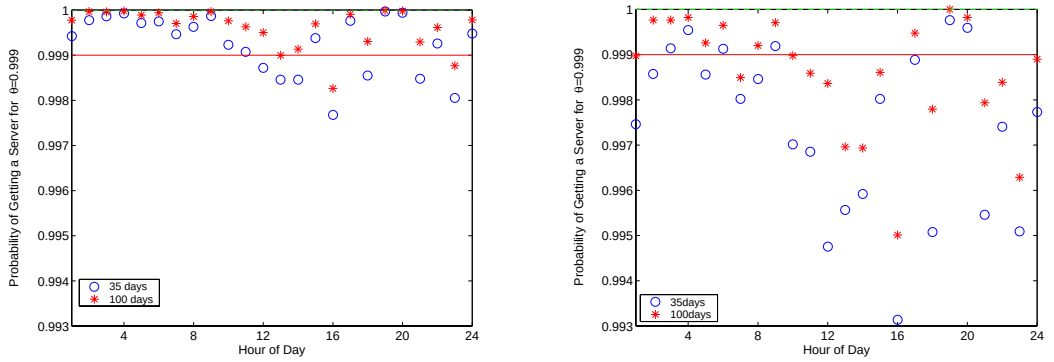
Figure 4. Simulated level of assurance

6.4.3 Sensitivity to precision of ADPs (by simulation)

Next, we consider the sensitivity of our approach to the precision of the pmfs in the ADPs. To illustrate we pick $\Gamma = 166$ which corresponds to $\rho = 0$ and $\theta = 0.999$. We can approximate each probability in a pmf using a binomial distribution with parameters N and $\hat{p}_k$, where $\hat{p}_k$ is the estimated probability and N is the number of observations used in the estimation. The value of N *determines* the confidence level of the probabilities; the larger the value of N the tighter the confidence intervals. To simulate the variability of the "true" probabilities, we generated the probabilities for each pmf randomly using the binomial distribution subject to the constraint that all the probabilities in each pmf sum to one. Each *perturbed pmf* was then used to draw 100 samples for the number of required servers. This represents 100 days worth of data. This was repeated one hundred times, each offering a test case with different perturbed pmfs so that a total of 10,000 days were simulated. The achieved value of θ was computed in the same way as in our validation study.

The mean and 5-percentile values for θ achieved for the 100 test cases are displayed by circles in Figure 5(a) and 5(b), respectively. The achieved values for θ are lower than the desired level due to the inherent uncertainty in the estimated pmfs. The mean of the achieved values for θ are always greater than 0.9975. Ninety-five percent of the achieved values for θ were greater than 0.993. However this is nearly an order of magnitude worse than the desired value. The lowest value achieved was 0.989. We note that the perturbed pmfs are based on the original data where $N = 35$.

To emulate an increase in confidence levels for the probabilities of the pmfs we repeat the experiments with $N = 100$, i.e., we assume nearly three times as much data was used to create the original pmfs. This provides for perturbed pmfs that are closer to the original pmfs than when $N = 35$. For comparison, the results are plotted in the same figures using stars. Ninety-five percent of the achieved values for θ were greater than 0.995. Increasing the number of observations that contribute to pmfs increases our confidence in the ADPs. In our simulation environment, this leads to an increase in the statistical assurance level that is achieved. Experiments with other parameter values of θ and ρ lead to similar findings.



(a) $\theta = 0.999$, $\rho = 0$, mean of θ from 100 runs (b) $\theta = 0.999$, $\rho = 0$, 5-percentile of θ from 100 runs

Figure 5. Simulated level of assurance with perturbed pmf probabilities

slot t	$I\%$	$w_{I=25}$	$I\%$	$w_{I=50}$	$I\%$	$w_{I=75}$	$I\%$	$w_{I=100}$
1	25	5	50	13	75	23	100	35
2	25	6	50	15	75	25	100	35

(a) Vertical Entitlements

slot t	w	$J_{w=2}$	w	$J_{w=4}$	w	$J_{w=6}$	w	$J_{w=8}$
1	2	0.77	4	0.73	6	0.79	8	0.81
2	2	1	4	0.80	6	0.73	8	0.70

(b) Horizontal Entitlements

Table 3. Examples of entitlements for an application

6.5 Policing

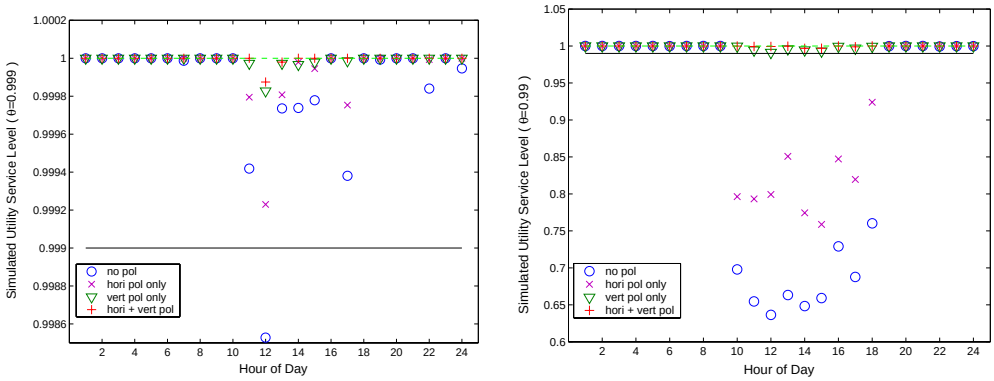
For our experiments, we examine how effectively our policing mechanisms curtail bad behavior. *Bad behavior* is defined as the case where certain applications demand more servers than they have reserved.

For our experimental design, we have three groups of applications. Each corresponds to the set of 48 applications as described earlier. The groups make reservations according to the Guaranteeed, PBE(0.999), and PBE(0.99) CoS, respectively. Each PBE class is given access to unused surplus servers of the other PBE class. We refer to this as pool sharing. The number of *surplus servers* for a CoS for a time slot t is defined as the difference between the total size of the server pool for that CoS (over all slots) and the number of servers needed to offer the assurance level θ for slot t. PBE(0.999) applications are given higher priority of access than PBE(0.99). For bad behavior, we consider the cases where all applications of the PBE(0.99) CoS run at the peak of their pmfs between time slots 10 and 18 (9am to 6pm). These are the slots with the greatest aggregate demand. Applications start their bad behavior at random within the first 35 simulated days. Though this is not the worst possible behavior, we believe it represents significant hostile behavior with respect to applications in a resource utility.

For policing, we chose entitlements as described in Section 5. Table 3 illustrates examples of entitlements for several values of w and t for one arbitrarily chosen application.

Table 3(a) illustrates vertical entitlements. They show bursts in demand for time slots $t = 1$ and $t = 2$. Let Δ_t be the total resource capacity reserved by an application for a slot t as measured over O instances of the slot. For our example, $O = 35$. Let I be a percentage of Δ_t. The table shows that $I = 25\%$ of the application's reservation capacity for slot $t = 1$ is used in a sliding window of only $w_{I=25\%} = 5$ slots. That is, one quarter of the demand can occur in approximately one seventh of the slots. Hence, the entitlement $V_{t=1}^{w=5}$ is one quarter of $\Delta_{t=1}$. Similarly, one half of $\Delta_{t=1}$, as measured over $O = 35$ slots, may materialize in a sliding window of approximately thirteen slots, which is approximately one third of the O slots.

Table 3(b) illustrates horizontal entitlements. Consider the maximum potential demand for a horizontal sliding window of w slots starting at a slot t. Define this as Δ_t^w. This demand is bounded by the sum of the peaks of the pmf for the adjacent slots. Let J be the maximum observed percentage of Δ_t^w with regard to the data used to compute an application's demand profile. Next, consider a horizontal sliding window of size $w = 4$ that starts at slot $t = 1$. The table shows that for the raw data, over all observations of the sliding window, at most $J_{w=4} = 0.73$ of $\Delta_{t=1}^w$ was used. Hence, $H_{t=1}^{w=4}$ is seventy



(a) $\theta = 0.999$, PBE(0.99) Bad behavior (b) $\theta = 0.99$, PBE(0.99) Bad behavior

Figure 6. Impact of bad behavior and policing on utility service levels

three percent of $\Delta_{t=1}^{w}$. Such vertical and horizontal entitlements can augment an application's demand profile and hence its reservation. They can then be used for policing exercises.

For the results we present in this case study, we use one value of w for both vertical and horizontal profiles. For the vertical profile we use $w = 35$. For the horizontal profile we use $w = 4$. When servers are clawed back an application must return them immediately.

We illustrate and evaluate the effectiveness of our techniques based on the the following metrics: utility service level violations for the predictable best effort classes (θ achieved for the utility), and, application service levels for the predictable best effort classes (θ perceived by each application). These metrics verify that the utility provides the levels of service that are planned, that applications receive qualities of service distinguished based on CoS, and that policing has the desired impact. We also show that that there remain significant opportunities for exploiting servers via a best effort CoS.

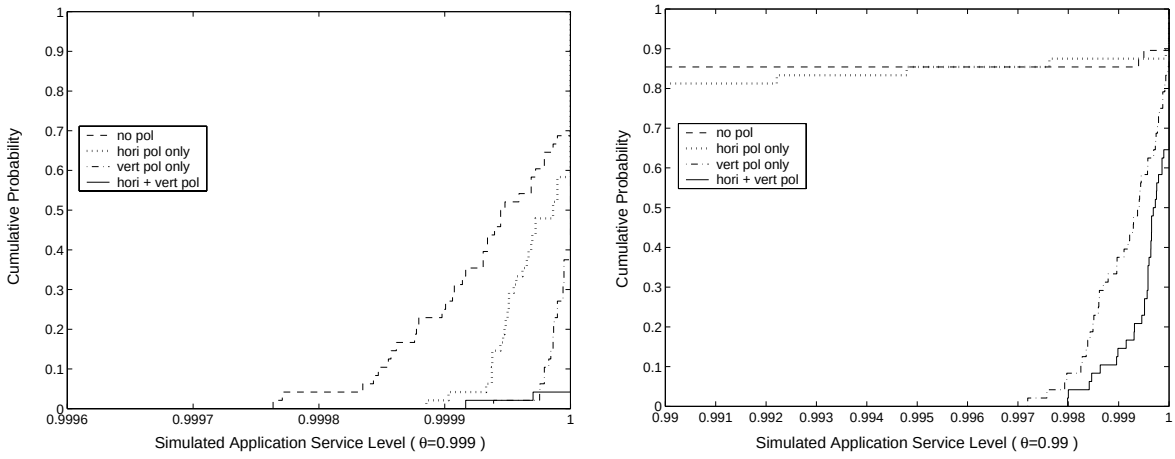
Figures 6(a) and (b) illustrate service levels achieved by the utility over the 1,000 day time scale with applications of the PBE(0.99) CoS exhibiting bad behavior between the 10th and 18th time slots. The impact of horizontal, vertical, and a combination of horizontal and vertical policing are shown. The horizontal line at θ represents the boundary between acceptable and unacceptable performance. Without policing, there are a large number of service level violations for the PBE(0.99) class. Vertical policing is effective as a mechanism to limit application demands. The combination of horizontal and vertical policing appears to further reduce service level violations. The PBE(0.999) is spared from the bad behavior because its applications have higher priority access to resources.

Figure 7 provides Cumulative Distribution Functions (CDF) to illustrate service levels obtained by the individual applications. For the figure, x-axis labels have been chosen on a case by case basis to best illustrate density between θ and 1. The figures correspond to the scenarios of Figure 6. As in Figure 6, Figure 7(b) shows that vertical policing is an effective mechanism for ensuring application service levels. The combination of horizontal and vertical policing further reduces service level violations. Policy mechanisms for the utility may be used to ensure that an application does not receive a better quality of service than it is entitled to even if resources are available.

Figure 8 provides CDFs for the number of servers available for the best effort CoS. These represent the unused servers from the Guaranteed and PBE classes of service. Figure 8(a) shows resource availability with pool sharing for the scenarios without bad behavior. We note that over all time slots, there are between 150 and 170 servers that are unused – which is approximately 30% of the total resource pool. These can be used to support best effort workloads. Figure 8(b) shows results for the case with *both* PBE classes of service exhibiting bad behavior between the 10th and 18th time slots. It shows that policing limits demand thereby increasing the number of unused servers. From the figure, we note that the no policing and horizontal policing only scenarios overlap while the combination of vertical and horizontal policing is the most effective.

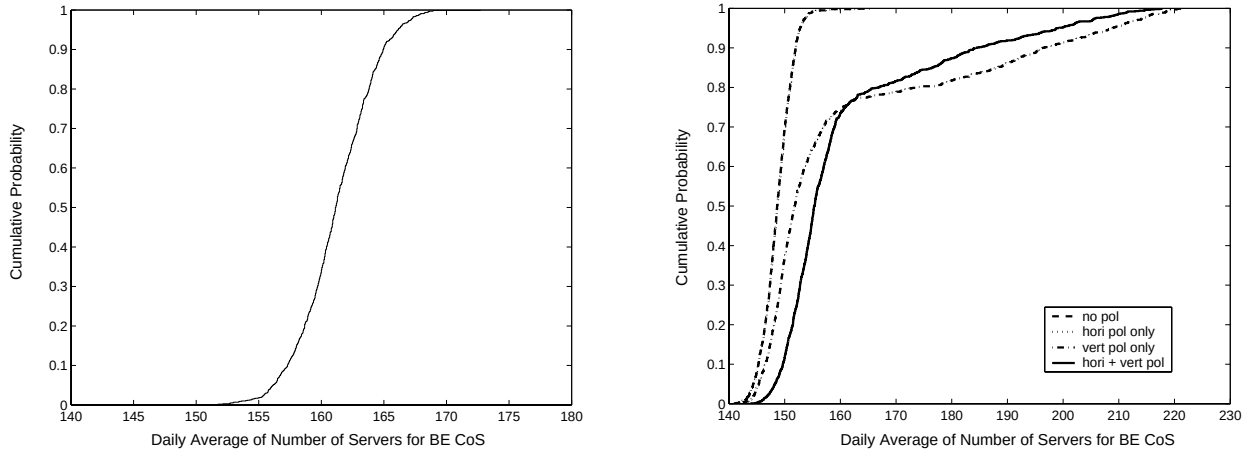
7 Summary and Conclusions

In this paper, we focus on techniques to support admission control and advance resource reservation for utility grid environments that support business applications. We present an approach for statistically characterizing the demand profiles of business applications. Reservations are based on these application demand profiles. The profiles provide the resource



(a) $\theta = 0.999$, PBE(0.99) Bad behavior (b) $\theta = 0.99$, PBE(0.99) Bad behavior

Figure 7. Impact of bad behavior and policing on application service levels



(a) No bad behavior (b) Bad behavior in both PBE classes of service

Figure 8. Best effort server availability

management system of a resource utility with time varying expectations and bounds on application resource requirements. We exploit the central limit theorem to estimate the number of resources needed to provide statistical assurance for access to resources while exploiting the advantages of statistical multiplexing. Statistical multiplexing permits a more efficient use of utility grid resources.

Application demand profiles may also be appropriate for some scientific and engineering loads that are effectively described using task graphs or whose resource requirements are best characterized statistically. The profiles are appropriate when an application/job has significant variation in its resource requirements over its lifetime.

We illustrated the feasibility of our approach with a case study that uses resource utilization information from 48 data center servers. Simulation experiments explore the sensitivity of the assurances to correlations between application resource demands, the precision of the demand profiles, and the effectiveness of our policing mechanism.

Based on our sensitivity analysis, we find that for the system under study there are clear gains to be made via statistical multiplexing with respect to simple time sharing based on peak demands. The technique we present appears to be robust with respect to correlations between application demands. However high values of correlation can distort the distribution of aggregate demand so that it is no longer Normal; this can lead to an optimistic estimate for the assurance that application requests for resources will be satisfied. The technique is less robust with respect to inaccuracies in the demand profiles. However, increasing the accuracy of the profiles had a clear and positive impact on the accuracy of the statistical assurance. The policing mechanism appears to be an effective approach for limiting bursts in application demands that may also impact offered assurance levels.

Future work includes applying the techniques to additional utility grid scenarios, positioning the methods within a Quality of Service management framework [19], exploring the impact of correlation and bursts in demand on sharing, and providing integrated support for the management of multiple resource types.

Acknowledgements

Many thanks to Marsha Duro and Sharad Singhal of HP Laboratories for helpful discussions and comments regarding this work.

References

[1] K. Appleby, S. Fakhouri, L. Fong, G. Goldszmidt, and M. Kalantar. Oceano – SLA based management of a computing utility. In *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management*, May 2001.

[2] M. Arlitt and T. Jin. Workload characterization of the 1998 world cup web site. *IEEE Network*, 14(3):30–37, May/June 2000.

[3] A. Bera and C. Jarque. Model specification tests: a simultaneous approach. *Journal of Econometrics*, 20:59–82, 1982.

[4] R. Boorstyn, A. Burchard, J. Liebeherr, and C. Oottamakorn. Statistical service assurances for traffic scheduling algorithms. *IEEE Journal on Selected Areas in Communications. Special Issue on Internet QoS.*, 18(12):2651–2664, December 2000.

[5] J. Chase, D. Anderson, P. Thakar, A. Vahdat, and R. Doyle. Managing energy and server resources in hosting centers. In *Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles (SOSP)*, Oct. 2001.

[6] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke. A resource management architecture for metacomputing systems. In *JSSPP*, pages 62–82, 1998.

[7] Ejasent. Utility computing white paper, Nov. 2001. `http://www.ejasent.com`.

[8] I. Foster, C. Kesselman, C. Lee, B. Lindell, K. Nahrstedt, and R. A. A distributed resource management architecture that supports advance reservation and co-allocation. In *IEEE International Workshop on Quality of Service (IWQoS)*, pages 27–36, June 1999.

[9] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration, www.globus.org, January 2002.

[10] Hewlett-Packard. HP utility data center architecture. `http://www.hp.com/solutions1/infrastructure/solutions/utilitydata/architecture/index.html`.

[11] Hewlett-Packard. Measureware (openview performance) agent. `http://www.openview.hp.com/products/performance/`.

[12] T. M. Inc. Matlab statistics toolbox, 2001. `http://www.mathworks.com/access/helpdesk/help/toolbox/stats/jbtest.shtml`.

[13] E. Knightly and N. Shroff. Admission control for statistical qos: Theory and practice. *IEEE Network*, 13(2):20–29, 1999.

[14] K. Krauter, R. Buyya, and M. Maheswaran. A taxonomy and survey of grid resource management systems for distributed computing. *Software-Practice and Experience*, 32(2):135–164, 2002.

[15] M. Litzkow, M. Livny, and M. Mutka. Condor - a hunter of idle workstations. In *Proceedings of the 8th International Conference on Distributed Computing Systems*, pages 104–111, June 1988.

[16] A. Natrajan, M. Humphrey, and A. Grimshaw. Grids: Harnessing geographically-separated resources in a multi-organisational context. In *Proceedings of High Performance Computing Systems*, June 2001.

[17] S. Ranjan, J. Rolia, H. Zu, and E. Knightly. Qos-driven server migration for internet data centers. In *Proceedings of IWQoS 2002*, pages 3–12, Miami, USA, May 2002.

[18] J. Rolia, S. Singhal, and R. Friedrich. Adaptive Internet Data Centers. In *Proceedings of SSGRR'00*, L'Aquila, Italy, `http://www.ssgrr.it/en/ssgrr2000/papers/053.pdf`, July 2000.

[19] J. Rolia, X. Zhu, and M. Arlitt. Resource access management for a resource utility for enterprise applications. In *The proceedings of the International Symposium on Integrated Management (IM 2003)*, pages 549–562, March 2003.

[20] J. Rolia, X. Zhu, M. Arlitt, and A. Andrzejak. Statistical service assurances for applications in utility grid environments. In *Proceedings of the Tenth IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 247–256, October 2002.

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

[22] B. Urgaonkar, P. Shenoy, and T. Roscoe. Resource overbooking and application profiling in shared hosting platforms. In *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI)*, December 2002.

[23] Z. Zhang, D. Towsley, and J. Kurose. Statistical analysis of generalized processor sharing scheduling discipline. *IEEE Journal on Selected Areas in Communications*, 13(6):1071–1080, 1995.

[24] S. Zhou. Lsf: Load sharing in large-scale heterogeneous distributed systems. In *Workshop on Cluster Computing*, 1992.