# Minimizing Data Center SLA Violations and Power Consumption via Hybrid Resource Provisioning

Anshul Gandhi
Carnegie Mellon University
Pittsburgh, PA, US
anshulg@cs.cmu.edu

Yuan Chen, Daniel Gmach, Martin Arlitt, Manish Marwah
HP Labs
Palo Alto, CA, US
{firstname.lastname}@hp.com

***Abstract*—This paper presents a novel approach to correctly allocate resources in data centers, such that SLA violations and energy consumption are minimized. Our approach first analyzes historical workload traces to identify long-term patterns that establish a "base" workload. It then employs two techniques to dynamically allocate capacity: predictive provisioning handles the estimated base workload at coarse time scales (e.g., hours or days) and reactive provisioning handles any excess workload at finer time scales (e.g., minutes). The combination of predictive and reactive provisioning achieves a significant improvement in meeting SLAs, conserving energy, and reducing provisioning costs. We implement and evaluate our approach using traces from four production systems. The results show that our approach can provide up to 35% savings in power consumption and reduce SLA violations by as much as 21% compared to existing techniques, while avoiding frequent power cycling of servers.**

***Keywords-data center; power management; performance management; resource allocation***

## I. INTRODUCTION

Data centers are very expensive to operate due to the power and cooling requirements of IT equipment. The EPA predicts that energy consumption in data centers will exceed 100 billion kWh in 2011, at an estimated cost of $7.4 billion [10]. Rising energy costs, regulatory requirements and social concerns over green house gas emissions amplify the importance of energy efficiency. However, energy efficiency is for naught if the data center cannot deliver IT services according to predefined SLA or QoS goals, as SLA violations result in lost business revenue. For example, Amazon found that every additional 100ms of latency costs them a 1% loss in sales, and Google observed that an extra 500ms in search page generation time reduced traffic by 20% [2]. Today, SLA violations are often avoided by over-provisioning IT resources. This results in excessive energy consumption. Thus, an important question in data center resource management is how to *correctly provision* IT equipment, such that SLA requirements are met while minimizing energy consumption.

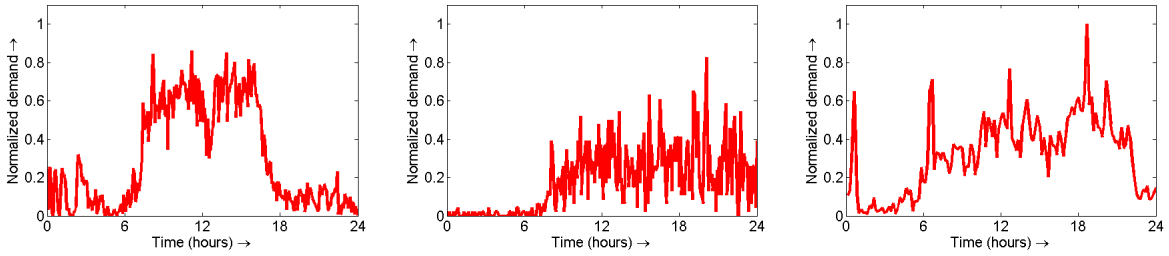
The correct provisioning of resources is a difficult task due to variations in workload demands. Most data center workload demands are very bursty in nature and often vary significantly during the course of a single day. Figure 1 shows a 24-hour workload demand trace for three real-world traces: an SAP enterprise application, a multi-tier business application called VDR, and a web application. The demands have been normalized by the maximum demand in each trace. The key observation is that the workload demand varies a lot. For example, the SAP trace demand varies from a minimum of almost 0 to a maximum of approximately 0.8. Further, for most traces, we find that the maximum demand is typically 4─5 times higher than the corresponding mean value, suggesting huge variations across the trace time-series. Also, for all traces, the 90%ile value is much higher than the mean value, suggesting that demand values are highly variable and well spread out. Thus, *workload demands typically have significant variability*. This makes it difficult to provision resources appropriately. A single size (static provisioning) cannot fit all, and will result in either over-provisioning or under-provisioning.

The solution we propose in this paper is based on three important observations. First, many workloads in data centers (e.g., Web servers) typically exhibit periodic patterns (i.e., daily, weekly and/or seasonal cycles). Figure 2 plots the time-series and the periodogram for five days of demand traces collected from the SAP, VDR and Web applications, respectively. The peak at 24 hours in the periodogram indicates that these traces have a strong daily pattern (period of 24 hours). If we can identify these patterns in the workload, we can then adjust the resource allocation accordingly and hence improve the accuracy of resource provisioning and reduce power consumption. However, demand patterns are statistical in nature and there will be deviations from historical patterns due to unforeseen factors such as flash crowds, service outages, and holidays. Though the volume of such fluctuations is small compared to the total demand, ignoring them completely can result in significant SLA violations. Finally, provisioning is not free; there are various associated costs and risks. Frequent provisioning incurs both performance and energy penalties. For example, turning servers on can take a significant amount of time (up to several minutes) and consume a lot of power (close to peak power consumption) [12]. Frequent power cycling of servers causes "wear and tear", which could result in server failure and service outage(s) [8].

Based on the above observations, we propose a hybrid provisioning approach that dynamically provisions IT resources at multiple time scales. Our main contributions are:

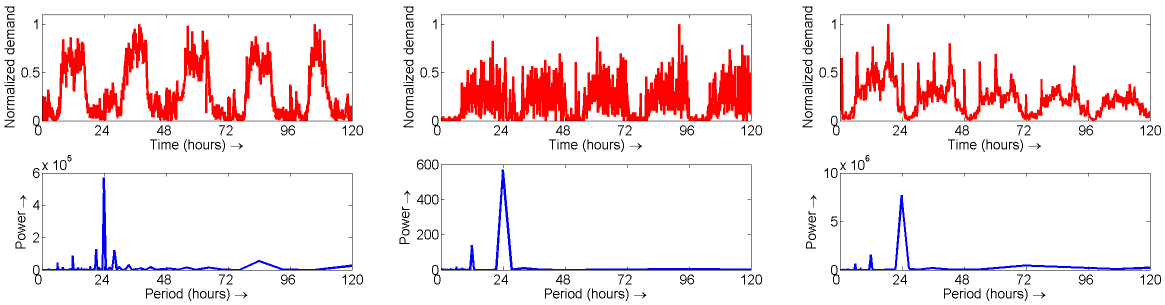
1. A novel approach that combines predictive and reactive control to provision IT resources: predictive control provisions the base workload at coarse time scales (e.g., hours) and reactive control handles any excess demand at finer time scales (e.g., minutes). The coordinated management of these two approaches achieves a

**978-1-4577-1221-0/11/$26.00 ©2011 IEEE**




(a) SAP (b) VDR (c) Web

Figure 1. Workload demand during a single day for (a) the SAP trace; (b) the VDR trace; and (c) the Web trace.




(a) SAP (b) VDR (c) Web

Figure 2. Time-series and periodogram for (a) the SAP trace; (b) the VDR trace; and (c) the Web trace.

significant improvement in meeting SLAs, conserving energy and reducing provisioning costs.

2. A novel analysis technique—"workload discretization", that determines the "base" workload demand for a service. In particular, we propose a dynamic programming algorithm that can accurately capture the demand while minimizing costs and risks from a provisioning perspective.
3. We implement our server provisioning system and validate our solution through both trace driven simulations and real implementation. The experimental results show that our hybrid approach out-performs other provisioning solutions when considering SLA violations, power consumption, and provisioning cost.

The rest of the paper is organized as follows. Section II presents our hybrid provisioning approach and describes our workload discretization technique. Section III discusses our analytical and experimental evaluation. Section IV reviews related work and finally Section V concludes the paper and discusses future directions.

## II. HYBRID PROVISIONING

We now present our *Hybrid Provisioning* solution. As discussed in the introduction, workload demands often have predictable patterns in them. However, there can be significant deviations from these patterns due to the bursty nature of data center workload demands. Excess workload demand or a sudden spike in demand can cause performance problems. Also, there is a cost and risk assoicated with provisioing changes, and thus, we want to avoid frequent provisioning changes. Based on these observations, we propose a *Hybrid Provisioning* solution, which combines predictive provisioning with reactive provisioning. The intuition behind our approach is that we capture periodic and sustained workload patterns from historical data, which we refer to as the *base workload* and then proactively (and predictively) provision for them. At run time, the deviation between the actual and predicted base workloads, which we refer to as the *noise workload*, is handled by reactive provisioning, which is invoked each time the request rate exceeds our prediction. Our solution also takes into account the costs and risks associated with provisioning.

### A. Overview of Our Approach

Figure 3 depicts the conceptual architecture of our solution. We begin with an overview of how it operates.

1. A *base workload forecaster* analyzes historical workload traces (Figure 3(a)) and identifies the patterns that form the base workload. We represent the patterns in a workload by discretizing it into consecutive, disjoint time intervals with a single representative demand value (e.g., mean or 90th percentile demand) in each interval (Figure 3(b)).
2. A *predictive controller* proactively estimates and allocates the proper amount of capacity (e.g., the number of servers) needed to handle the base workload. For example, given the base demand (e.g., time varying request rate in Figure 3(b)), it generates the capacity allocation (e.g., number of servers in Figure 3(c)).
3. A *reactive controller* handles excess demand by adding additional capacity at short time scales (e.g., # of servers in Figure 3(d)), in response to excess workload that is beyond the base workload, i.e., the difference between the actual workload and forecasted base workload.
4. A *coordinator* dispatches workload requests to servers and also communicates with the predictive and reactive controllers to provide information about incoming demand. When the actual workload (e.g., requests in Figure 3(e)) arrives, the dispatcher assigns the portion of the workload not exceeding the base workload (e.g.,

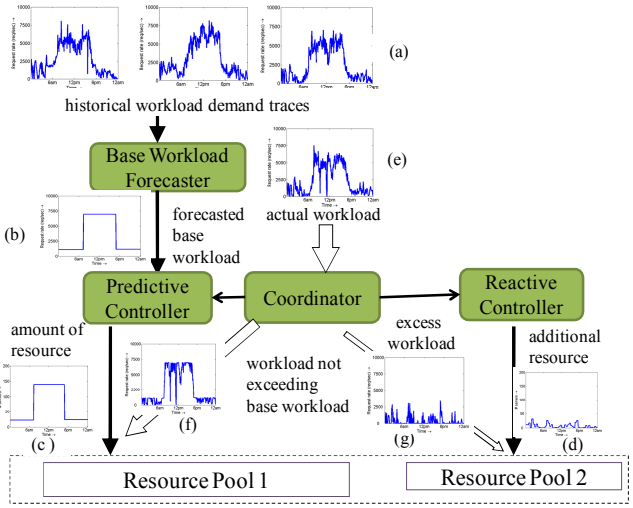



Figure 3. Hybrid provisioning.

requests in Figure 3(f)) to resource pool 1 and any excess workload (e.g., requests in Figure 3(g)) to resource pool.

## B. *Base Workload Forecaster*

The *base workload forecaster* first performs a *periodicity analysis* of the workload demand to reveal the length of a pattern or a sequence of patterns that appear periodically. We use Fast Fourier Transform (FFT) to find the periodogram of the time-series data [5] from historical data. From this, we derive the periods of the most prominent patterns. For example, the periodograms from Figure 2 reveal that our traces have a strong daily pattern (period of 24 hours). The base workload forecaster runs periodically (e.g., once a day). It divides the historical data into daily demands and forecasts the demand for the next day. For workload forecast, we can just take the average of the mean historical daily demand. Advanced forecast techniques can be used, but are out of the scope of this paper [13]. Next, we identify and discretize patterns in the forecasted workload demand. Our goal is to represent the daily pattern in workloads by discretizing their demands into consecutive, disjoint time-intervals with a single representative demand value in each interval. We propose a dynamic programming algorithm to find a small number of time intervals and representative demand for each, such that the deviation from actual demand is minimized. Keeping the number of intervals small is important, as more intervals imply more frequent provisioning changes and thus higher risks and costs. This is one of the key contributions of our work. We now formally define discretization.

*Workload discretization: Given the demand time-series X on the domain [s, t], a time-series Y on the same domain is a workload characterization of X if [s, t] can be partitioned into n successive disjoint time intervals, {[s, $t_1$],[$t_1$, $t_2$],...,[$t_{n-1}$, t]}, such that X(j)= $r_i$, for all j in the $i^{th}$ interval, [$t_{i-1}$, $t_i$].*

Note that, by definition, any time-series, $X$, is a discretization of itself. For our purposes, we set s=0 and $t$=period of workload. In the follow discussion, we assume a period of 24 hours, based on our periodicity analysis.

The idea behind discretization is two-fold. First, we want to accurately capture the demand. To achieve this, the representative values, $r_i$, for each interval, [$t_{i-1}$, $t_i$], should be as close as possible to the actual demand values of the time-series in the interval [$t_{i-1}$, $t_i$]. We determine the demand estimation error incurred by discretization, which we define as the percentage of the area between the original time-series and the discretized time-series with respect to the area under the original time-series. Second, provisioning of IT resources is not free [8]. For this reason, we want to avoid having too many intervals and hence too many changes to the system, as this is not practical and can lead to many problems (e.g., performance loss, wear and tear of servers, system instability, etc.). In summary, we want to minimize the error introduced by discretization and the number of intervals in the discretization. We propose a dynamic programming (DP) solution to discretize the time-series such that we reduce both.

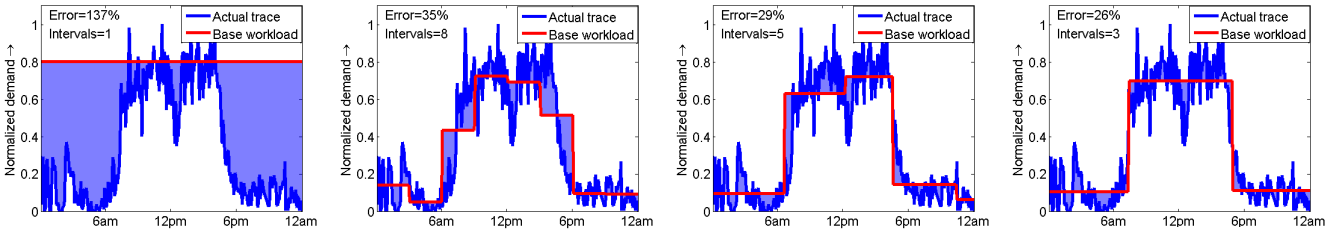
$$\sum_{i=1}^{n}\left[\sum_{t=t_{i-1}}^{t_i}(X(t)-r_i)^2\right]+f(n) \quad (1)$$

Equation (1) is the objective function we want to minimize, where $X$ is the time-series and $f(n)$ is a cost function of the number of changes or intervals, $n$. The goal of Equation (1) is to simultaneously minimize the workload representation error and the number of changes. In some cases, one might prefer to minimize the square of the number of changes (or some other function of the number of changes). For this paper, we set $f(n) = c * n$, where $c$ is a normalization constant which we discuss later. The objective function expresses the goal as minimizing the normalized number of changes and the representation error. The Mean-squared error is used to quantify the workload representation error. Note that in the most general case, both the number of changes and the representational error could be formulated as utility functions. We use DP to minimize the objective function given in Equation (1). Minimizing the Mean-squared error for a given partition results in setting $r_i$ to be the mean of the time-series values on that partition. That is:

$$min.\sum_{t_{i-1}}^{t_i}(X(t)-r_i)^2 \quad \Rightarrow \frac{dy}{dr_i}\sum_{t_{i-1}}^{t_i}(X(t)-r_i)^2 = 0$$

$$\Rightarrow r_i = \frac{\sum_{t_{i-1}}^{t_i} X(t)}{t_i - t_{i-1}+1} \quad (2).$$

Thus, we only need to partition the time-series domain, [$t_0$=0, $t_n$=24hrs], to optimize the objective function. Let us assume that we have the optimal partition of the domain [$t_0$, $t_n$], and it is {[$t_0$, $t_1$], [$t_1$, $t_2$],...,[$t_{n-1}$, $t_n$]}. Now, consider the optimal solution for the domain [$t_0$, $t_{n-1}$]. We claim that this is simply {[$t_0$, $t_1$], [$t_1$, $t_2$],...,[$t_{n-2}$, $t_{n-1}$]}. This is because if the optimal solution for the domain [$t_0$, $t_{n-1}$] was different, then we could simply append the partition [$t_{n-1}$, $t_n$] to it, and that should give us a solution for the domain [$t_0$, $t_n$] with a lower objective function value than {[$t_0$, $t_1$],[$t_1$, $t_2$],...,[$t_{n-1}$, $t_n$]}, which is a contradiction. Thus, the optimal solution for the domain [$t_0$, $t_n$] contains the optimal solution for the domain [$t_0$, $t_{n-1}$]. Therefore, we have the optimal substructure property for this problem, and DP will result in an optimal solution. Note that even in the case of highly irregular traffic where no patterns can be found, the DP solution will still output a workload characterization. In particular, the workload characterization will simply be a single value, e.g., the mean or desired percentile of the observed demand, over




(a) 90%ile (b) Mean/3hrs (c) K-means (d) Our Approach (DP)

Figure 4. Comparision of different discretization techniques to determine the base workload.

the entire domain $[t_0, t_n]$. Therefore, the DP solution is robust to irregularities in the demand pattern.

We now mention a rule of thumb to pick the normalization constant $c$. The single interval approach with Mean as the representative value gives our objective function a value of:

$$Var = \sum_{t=t_0}^{t_n} [X(t) - Mean(X)]^2 \quad (3),$$

where *Var* is the variance of the time-series under consideration, *X*. By allowing a finite number of changes, say $z$, we want to ensure that we can reduce the representation error in our objective function by at least *Var/2*. Further, this new objective function must be smaller than Var. Thus, we want:

$$\frac{Var}{2} + c * z < Var \quad \Rightarrow \quad c < Var/(2 * z)$$

Thus, for example, setting $z=2$ results in $c<Var/4$. For simplicity, we set $c=Var/4$ in our dynamic programming solutions. In general, a larger value of $z$ may result in a smaller representation error at the cost of an increase in the number of changes, whereas a smaller value of $z$ may result in a larger representation error at the cost of a decrease in the number of changes. The choice of $z$ in a real data center will depend on how often the system administrator is willing to power cycle the servers. We expect a power cycling frequency of once every 4─6 hours to be acceptable. Thus, $z<5$ should be an acceptable value for a 24 hour demand trace.

Figure 4 plots the discretization results of a real SAP application trace using various approaches, including a single representative value of 90th percentile (*90%ile*), multiple intervals with 3 hour partitions and using mean (*Mean/3hrs*), a K-means clustering algorithm (*K-means*), and our approach (*DP*) with $c=Var/4$, which corresponds to $z=2$. The results show that our approach achieves a lower estimation error than *Mean/3hr* and *K-means* with fewer intervals. The single interval with *90%ile* significantly overestimates the demand with an estimation error of 137%.

### C. Predictive Controller

The *predictive controller* is responsible for handling the base workload. It receives the predicted base workload pattern (i.e., the output of discretization) from the *base workload forecaster* and then proactively estimates and allocates the proper amount of capacity required to handle the base workload. In particular, the controller uses a queueing performance model to determine how much capacity will be allocated to ensure that the SLA requirements are met for the forecasted demand without consuming excessive power [5].

We assume that a certain mean response time target (SLA requirement), $t_0$, in seconds, is given. Recall that our goal is to minimize power consumption while ensuring that we meet the mean response time target. We use an M/G/1/PS queueing model to estimate the number of servers needed. Assuming that demand follows a Poisson distribution with a mean request rate λ requests/sec, we have:

$$\frac{1}{\frac{1}{s} - \frac{\lambda}{k}} < t_o \quad (4),$$

where $s$ is the mean job size in seconds, and $k$ is the number of servers. From Equation (4), we derive:

$$k = \left\lceil \frac{\lambda}{\frac{1}{s} - \frac{1}{t_0}} \right\rceil \quad (5).$$

Equation (5) estimates the number of servers needed to ensure that the mean response time target is met. While the assumptions behind Equation (5) may not be the best for real-world data center workloads, they provide a good approximation, as we will see in Section III. Though important, performance and capacity modeling is not the focus of this paper. Note that the capacity allocated by the *predictive controller* is not affected by actual demand and will not change until a new base workload forecast arrives.

### D. Reactive Controller

A reactive controller is invoked each time the actual workload demand is higher than the base workload, to provide additional resources for the excess workload demand. Since the reactive controller is not invoked when the actual demand is lower than the base workload forecast, the impact of over-provisioning is minimized if the predictive provisioning captures the base demand well. The results in Section III show that this hybrid approach works very well in practice.

We use a simple feedback approach with a fixed monitoring interval length to estimate the amount of noise in the workload. For example, if the monitoring interval length is ten minutes, then we estimate the noise (workload demand above the predicted base demand) in the next ten minute interval to be the same as the noise in the current ten minute interval [6]. While more sophisticated approaches, such as ARMA or moving window [9], can be used for noise estimation, we find that a simple feedback based approach is very efficient and works well in practice for noise estimation. Note that errors in noise estimation will only affect the noise provisioning; the base workload demand will not be affected.

### E. Coordinator

We logically divide the server pool into two partitions. One handles the base workload and is managed by the predictive controller. The other handles the noise (excess) workload and is managed by the reactive controller. These two server farms can (and probably will) physically co-exist. The coordinator forwards incoming requests to either the server farm dedicated to the base workload or the server farm dedicated to the noise (excess) workload, based on the predicted base demand and the actual demand. A simple scheme for dispatching requests is to load-balance the incoming requests among all servers irrespective of which server farm they belong to. Under this scheme, all jobs are treated equally. However, one can imagine a more sophisticated dispatching scheme which allows certain important jobs to receive preferential service over other jobs by dispatching the important jobs to the (more robust) base workload server farm. For example, e-commerce sites such as Amazon or eBay may want to prioritize shopping requests over browsing requests. In such cases, shopping requests can be dispatched to the server farm that handles the base workload. This server farm is less likely to incur provisioning changes and can thus provide uninterrupted service. The less important browsing requests can be dispatched to either the base workload server farm, if there is available capacity, or to the noise workload server farm. Compared to load balancing scheduling, our solution isolates the base workload from the noise workload, providing stronger performance guarantees for the base workload.

## III. EVALUATION

We obtain workload demand traces from three real applications used in HP data centers and the World Cup 1998 demand trace. We use the four traces to validate our solution via trace driven analysis and implementation on a real test bed. We begin by describing the workload traces and the provisioning policies used for evaluation.

1. SAP is a five-week-long workload demand trace of an SAP enterprise application that was hosted in an HP data center. The trace captures average CPU and memory usage as recorded every 5 minutes.
2. VDR is a ten-day-long trace containing arrival rate and system utilization data recorded every 5 minutes from a high-availability, multi-tier business-critical HP application serving both external customers and HP users on six continents.
3. Web 2.0 is eight days worth of system utilization data from a popular HP Web service application with more than 85 million registered users in 22 countries located in multiple data centers.
4. WorldCup98 is a demand trace obtained from the Internet Traffic Archives [28].

The first three traces exhibit prominent daily patterns but also vary significantly during the course of a single day. The WorldCup98 trace is less predictable, with peak demands influenced by factors such as the outcome of matches, which are difficult to predict.

We compare different provisioning policies that allocate capacity (i.e., number of servers) for a Web server farm during the course of a single day. The *Predictive* policy provisions servers for day 8 based on the 90%ile demand values for the past seven days. We consider four provisioning intervals for the predictive policy: 24 hours, 6 hours, 1 hour and variable length intervals. For example, the *Predictive/1hr* approach considers the corresponding 1 hour demand trace portions from each of the past 7 days and uses the 90%ile demand of these portions as the representative demand value for that hour of day 8. The *Predictive/var* approach uses variable intervals from our workload discretization technique. The *Reactive* approach uses a feedback approach to estimate demand online and allocates capacity accordingly every 10 minutes. We study two hybrid approaches: the *Hybrid/fixed* policy does predictive provisioning every hour and invokes reactive provisioning each time the predictive provisioning underestimates demand; and the *Hybrid/var* approach that combines predictive provisioning with variable intervals and reactive provisioning as described in Section II. We choose the 90%ile demand for all predictive approaches since the 90%ile provides a good tradeoff between resource consumption and quality.

### A. Trace-based Analysis

This section evaluates our hybrid approach for various workload traces via analysis using the following metrics: percentage of SLA violations, total power consumption, and the number of provisioning changes. If the actual demand for a 5-minute interval exceeds the allocated capacity, then that interval is under-provisioned and is counted as an SLA violation. Power consumption of a server is estimated based on its average CPU utilization, and power at idle and fully utilized states as $P_{idle} + (P_{max} - P_{idle})*u$, where $u$ is the CPU utilization, $P_{idle}$ and $P_{max}$ are the observed server power at idle and fully utilized states respectively. Finally, we track the number of provisioning changes, i.e., number of times we change the server farm capacity, throughout the target day.

Figure 5 shows the SLA violations, power consumption and number of provisioning changes for the SAP trace. First, among the four predictive approaches, using variable intervals achieves the best balance of performance and power. The percentage of violations is comparable to the *Predictive/24hrs* approach, but the power consumption is 30% less. The power consumption of *Predictive/var* is similar to that of *Predictive/6hrs* and *Predictive/1hr*, but the performance is much better and the number of provisioning changes is smaller. This demonstrates that our workload discretization technique can effectively improve the accuracy of provisioning.

Second, our *Hybrid/var* approach out-performs all predictive approaches. Compared to the *Predictive-1hr* and *Predictive/6hrs*, our hybrid approach reduces SLA violations from 12% to 5.6% while consuming the same amount of power. Compared to *Predictive/var*, *Hybrid/var* reduces SLA violations from 7.3% to 5.6%. These reductions are a result of the integration of reactive provisioning to handle demand exceeding the predicted base demand.

Third, both hybrid approaches perform significantly better than the Reactive approach in terms of meeting SLA

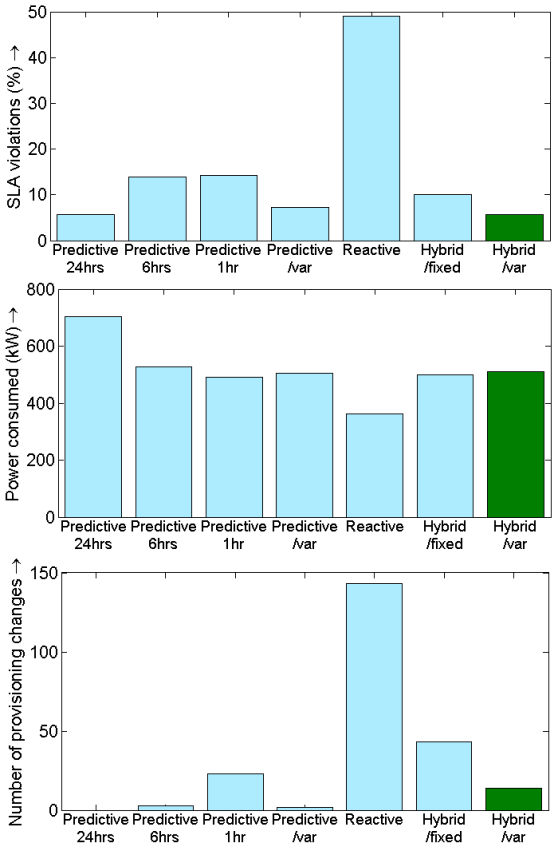



Figure 5. Trace-based analysis results for the SAP trace.

requirements. Also the hybrid approaches incur fewer changes than the reactive approach. The *Reactive* approach has the highest percentage of violations. There are two reasons for this. (1) The workload is highly variable and abrupt changes in short intervals are difficult to track. (2) It takes time to turn on a server, creating a backlog of requests that the reactive controller has to handle when ramping up capacity. Between the two hybrid approaches, our *Hybrid/var* approach outperforms *Hybrid/fixed*. In particular, by using variable intervals for predictive provisioning, *Hybrid/var* reduces SLA violations from 10.1% to 5.6% and the number of changes from 43 to 14, without affecting power consumption. This further shows the value of our DP based workload discretization algorithm.

In summary, compared to purely predictive and reactive approaches, our *Hybrid/var* approach achieves better performance and conserves power. The *Hybrid/var* approach consumes 27% less power than the *Predictive/24hour* approach, which has similar performance. Compared to the *Reactive* approach, the *Hybrid/var* reduces SLA violations from 50% to 5.6%. *Hybrid/var* outperforms a hybrid approach with fixed provisioning intervals in terms of SLA violations and power consumption. Finally, *Hybrid/var* incurs 14 changes, less than one every hour, which is considered acceptable in practice.

To further understand how these different approaches allocate resources in response to workload demand changes, we show time series of the demand, the number of SLA

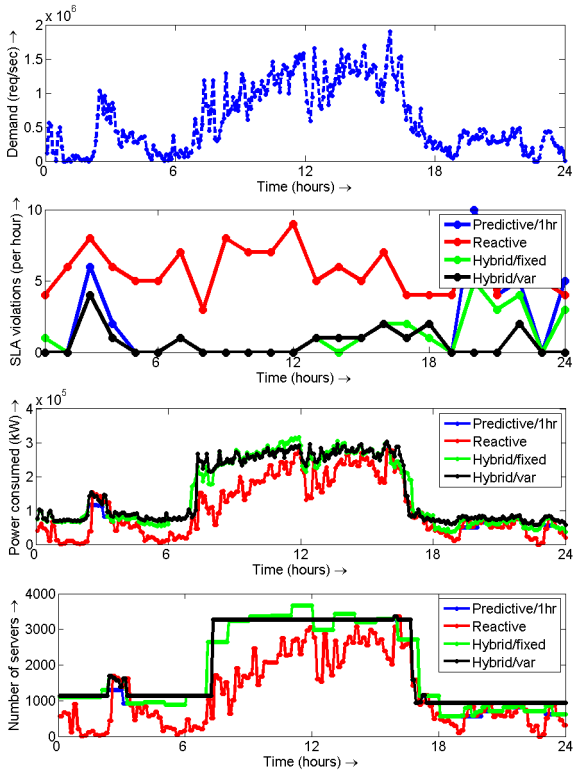



Figure 6. Time-series for the demand, SLA violations, power consumption and the number of servers for the SAP trace.

violations per hour, the power consumption and the number of active servers in Figure 6. We see that the *Reactive* policy makes many provisioning changes and incurs many violations, since a change in provisioning is triggered by either an under-provisioning or an over-provisioning in the previous interval. The figure shows that our *Hybrid/var* approach captures most of the incoming workload demand. Thus, only a few changes are triggered by the reactive part of our approach. However, there are instances when the reactive controller is invoked, for example, around 2am–3am. This again justifies that a hybrid approach is required to handle real-world workload traces.

We conduct a similar analysis study for the Web, VDR and WordCup98 traces and achieve similar results. The *Predictive/var* approach that uses our workload discretization is still among the best of all four predictive approaches. Importantly, our *Hybrid/var* approach provides the best tradeoff between SLA violations, power consumption and the number of provisioning changes among all policies.

### B. *Experimental Results on a Real Testbed*

To further verify that our solution is superior to existing approaches and ensure that we have a practical and robust solution, we implement a prototype system and experimentally evaluate our policy against other policies using the real workload traces described above.

We implement a single-tier server farm application, which dynamically provisions servers and processes user requests. Our test bed includes 10 blade servers. Each server has two Intel Xeon E5535 quad-core processors running at 2.66 GHz, and 16GB of memory. One server was used as a

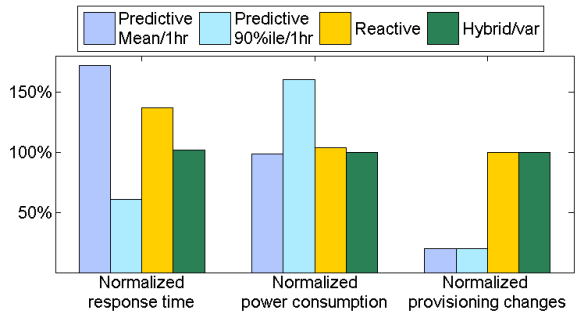



Figure 7. Experimental results for the Web trace.

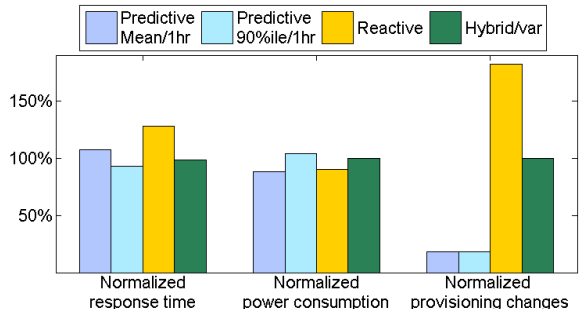



Figure 8. Experimental results for the VDR trace.

front-end load generator running httperf [18], which replays the workload demand traces in the form of cgi requests. Another server was employed as a proxy/load-balancer, which distributes jobs and requests among the remaining eight Apache Web servers. We scale the workload demand from real traces suitably to match our server farm capacity, since most of the traces were collected from production systems running thousands of servers. Business logic processing in enterprise and Web applications is often the bottleneck and the workloads are often processor intensive. Thus, we use LINPACK [17], a multi-threaded CPU bound program, to simulate the request or job processing in real applications. By adjusting the LINPACK job sizes, we model the variability of service times in real applications [11]. We use the blade servers' built-in management processor to turn servers on and off remotely and programmatically, and to obtain power consumption readings. We collect response times from httperf logs.

We evaluate four different polices: *Predictive Mean/1hr*, *Predictive 90%ile/1hr*, *Reactive* and our hybrid approach, *Hybrid/var*. The predictive provisioning policies use the corresponding 1 hour demand trace portions from the past days (Mean and 90%ile respectively) to predict the demand for the hour of the target day. *Reactive* invokes the reactive controller every 10 minutes and employs a feedback model that provisions servers for the next 10 minutes. For all policies and a given response time target, the number of active back-end servers is determined by the performance model described in Section II.C.

Figure 7 shows our experimental results of performance, power consumption and number of provision changes for the Web trace. The performance is the average response time normalized by the target response time, and the power consumption is normalized by the power consumption of our hybrid approach. We see that the *Predictive Mean/1hr* does not provision sufficient capacity for the workload demand while the *Predictive 90%ile/1hr* tends to over-provision capacity. The *Reactive* policy misses its response time target due to the lag in booting additional servers. By contrast, our *Hybrid/var* policy meets its response time target, while keeping power consumption low. In particular, our *Hybrid/var* policy provides a 35% savings in power consumption when compared to the *Predictive 90%ile/1hr* and lowers response time by as much as 41% when compared to the *Predictive Mean1/hr*.

Figure 8 shows our experimental results for the VDR trace. Again, we see that the *Predictive Mean1/hr* and *Reactive* policies fail to meet the target mean response time. Further, the *Reactive* policy incurs a lot of provisioning changes. Both *Hybrid/var* and *Predictive 90%ile/1hr* succeed in meeting the response time target but our approach uses slightly lower power. Note that *Predictive 90%ile/1hr* significantly over-provisions resource for the Web trace (Figure 7), which indicates that its performance really depends on the characteristics of the workload. By contrast, our approach works well for both traces.

In summary, we conclude that our Hybrid approach outperforms other provisioning approaches such as the predictive and reactive policies, and combinations of both with fixed intervals, when considering performance, power consumption, and number of provisioning changes.

## IV. Related Work

Existing server provisioning solutions can be broadly classified into predictive and reactive solutions. *Predictive provisioning* assumes there is a predictable and stable pattern in demand and allocates capacity typically at the time-scale of hours or days based on the pattern. Castellanos *et al.* exploit the predictability in business applications' demand to improve the effectiveness of resource management [6]. With the adoption of virtualization technology, server consolidation has emerged as a promising technique to improve resource utilization and reduce power consumption [21],[25]. Though these approaches can be effective to a certain extent, choosing the proper provisioning size is still a very difficult task. Our workload discretization technique can help with this. However, large, unpredicted demand surges could cause severe SLA violations.

*Reactive provisioning*, on the other hand, allocates resources in short intervals (e.g., every few minutes) in response to workload changes. One approach is to use reactive control loops that are based on feedback control [1],[20],[24]. Others use reactive provisioning strategies for resource allocation and power management in virtualized environments. Typically, these dynamically allocate CPU shares to virtual machines and/or migrate VMs across physical servers at runtime [19],[27]. Purely reactive policies can potentially react quickly to changes in workload demand, but issues such as unpredictability, instability and high provisioning costs limit their use in practice.

There are several approaches that *combine predictive and reactive control* [4],[14],[23]. While these approaches share some features with our hybrid approach, they differ in several aspects. First, our approach aims to optimize the

performance, power consumption and provisioning cost simultaneously. The provisioning cost is particularly important to consider as it can significantly impact the performance and power consumption. Second, we propose and apply a novel workload analysis technique, *workload discretization*, to determine provisioning intervals with variable lengths whereas predictive provisioning in other approaches uses simple fixed intervals. Third, our results show that our approach outperforms these other approaches.

Numerous studies have examined workload demand traces. Characterization studies of interactive workloads such as web or media servers indicate that demands are highly variable, although daily and weekly patterns are common [3],[7]. Rolia et al. find similar patterns in business application workloads [22]. A lot of research has been conducted to predict future workload demands [26] [16] [13] [15]. What distinguishes our work from others is the idea of workload discretization and the incorporation of the provisioning cost in our workload analysis, which is important from a capacity planning and resource provisioning perspective.

## V. Conclusion and Future Work

It is a challenging task to correctly provision IT resources in data centers to meet SLA requirements while minimizing power consumption. In this paper, we propose a hybrid approach that proactively allocates resources for the predictable demand pattern and leverages a reactive controller to deal with excess demand. In particular, we develop a novel workload discretization technique to accurately determine workload demand patterns. We demonstrate via experimentation that by using the proposed workload discretization technique and combining predictive and reactive control, our approach successfully meets SLA requirements, is more power efficient than existing approaches, and avoids unnecessary provisioning changes.

In the future, we plan to incorporate dynamic CPU frequency scaling and power states into our approach. Further, we plan to validate our hybrid provisioning approach on larger test beds and with multi-tier applications. We are also interested in applying our workload discretization technique to other capacity planning and resource provisioning problems, including workload consolidation in virtualized environments.

## References


[1] K. Appleby, S. Fakhouri, L. Fong, G. Goldszmidt, M. Kalantar S. Krishnakumar, D. P. Pazel, J. Pershing, B. Rochwerger, "Océano-SLA Based Management of a Computing Utility", in Proc. of IEEE/IFIP Int. Symposium on Integrated Network Management, Seattle, WA, USA, pg. 855─868, 2001.

[2] G. Linden, "Make data useful," http://glinden.blogspot.com, 2006.

[3] M. Arlitt, C. Williamson,"Web Server Workload Characterization: The Search for Invariants", in Proc. of ACM SIGMETRICS, Philadelphia, PA, USA, pg. 126─137, 1996.

[4] M. Bennani, D. Menasce, "Resource Allocation for Autonomic Data Centers using Analytic Performance Models", in Proc. Int. Conf. on Automatic Computing. IEEE Computer Society, Washington, DC, USA, pg. 229─240, 2005.

[5] D. Brillinger, "Time series: data analysis and theory", Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

[6] M. Castellanos, F. Casati, M. Shan, U. Dayal, "iBOM: A Platform for Intelligent Business Operation Management", in Proc. of Int. Conf. on Data Engineering (ICDE), Tokyo, Japan, pg. 1084─1095, 2005.

[7] L. Cherkasova, M. Gupta, "Characterizing Locality, Evolution, and Life Span of Accesses in Enterprise Media Server Workloads", in Proc. Workshop on Network and Operating Systems Support for Digital Audio and Video, New York, NY, USA, pg. 33─42, 2002.

[8] A. Coskun, R. Strong, D. Tullsen, S. Rosing, "Evaluating the impact of job scheduling and power management on processor lifetime for chip multiprocessors", in Proc. of ACM SIGMETRICS, Seattle, WA, USA, pg. 169─180, 2009.

[9] P. Dinda, D. O' Hallaron, "Host Load Prediction Using Linear Models", in Cluster Computing, Vol. 3, No. 4, pg. 265─280, 2000.

[10] "Report to congress on server and data center energy efficiency, public law," U.S. Environmental Protection Agency, ENERGY STAR Program, 2007.

[11] A. Gandhi, M. Harchol-Balter, R. Das, C. Lefurgy, "Optimal Power Allocation in Server Farms", in Proc. of ACM SIGMETRICS 2009, Seattle, USA, 2009.

[12] A. Gandhi, M. Harchol-Balter, I. Adan, "Server farms with setup costs", Performance Evaluation, Vol. 67, pg. 1123─1138, 2010.

[13] D. Gmach, J. Rolia, L. Cherkasova, A. Kemper, "Capacity Management and Demand Prediction for Next Generation Data Centers", in Proc. of IEEE Int. Conf. on Web Services, Salt Lake City, Utah, USA, pg. 43─50, 2007.

[14] D. Gmach, S. Krompass, A. Scholz, M. Wimmer, A. Kemper, "Adaptive Quality of Service Management for Enterprise Services", in ACM Transactions on the Web, Vol. 2, No. 1, 2008.

[15] J. Hellerstein, F. Zhang, P. Shahabuddin, "A Statistical Approach to Predictive Detection", in Computer Networks, Vol. 35, No. 1, 2001.

[16] P. Hoogenboom, J. Lepreau, "Computer System Performance Problem Detection using Time Series Models", in Proc. of Summer USENIX Conference, pg. 15─32, 1993.

[17] Intel, Intel Math Kernel Library 10.0, LINPACK, http://software.intel.com/en-us/intel-mkl/.

[18] D. Mosberger, T. Jin, "httperf-A Tool for Measuring Web Server Performance", in Proc. of ACM Sigmetrics Performance Evaluation Review, Vol. 26, pg. 31─37, 1998.

[19] P. Padala, K. Hou, X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, K. Shin, "Automated control of multiple virtualized resources", in Proc. of EuroSys, Nuremberg, Germany, 2009.

[20] S. Ranjan, J. Rolia, H. Fu, E. Knightly, "QoS-Driven Server Migration for Internet Data Centers", in Proc. of IEEE Int. Workshop on Quality of Service. Miami Beach, FL, USA, pg. 3─12, 2002.

[21] J. Rolia, A. Andrzejak, M. Arlitt, "Automating Enterprise Application Placement in Resource Utilities", in Proc. of IFIP/IEEE Int. Workshop on Distributed Systems: Operations and Management. Heidelberg, Germany, pg. 118─129, 2003.

[22] J. Rolia, X. Zhu, M. Arlitt, A. Andrzejak, "Statistical Service Assurances for Applications in Utility Grid Environments", in Performance Evaluation, Vol. 58, No. 2+3, pg., 319─339, 2004.

[23] B. Urgaonkar, P. Shenoy, A. Chandra, P. Goyal, "Dynamic Provisioning of Multi-tier Internet Applications", in Proc. of IEEE Int. Conf. on Autnomic Computing, June 2005.

[24] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, A. Tantawi, "An Analytical Model for Multi-tier Internet Services and its Applications", in ACM Transactions on the Web, Vol. 1, No. 1, 2007.

[25] A. Verma, G. Dasgupta, T. Nayak, P. De, R. Kothari, "Server Workload Analysis for Power Minimization using Consolidation", in Proc. of USENIX Annual Technical Conference, 2009.

[26] R. Vilalta, C. V. Apte, J. L. Hellerstein, S. Ma, S. M. Weiss, "Predictive Algorithms in the Management of Computer Systems", in IBM Systems Journal, Vol. 41, No. 3, pg. 461─474, 2002.

[27] J. Xu, M. Zhao, J. Fortes, R. Carpenter, M. Yousif, "Autonomic resource management in virtualized data centers using fuzzy logic-based approaches", in Cluster Computing Journal, Vol. 11, No. 3, pg. 213─227, 2008.

[28] The internet traffic archives: WorldCup98. Available at http://ita.ee.lbl.gov/html/contrib/WorldCup.html.