Server Workload Analysis for Power Minimization using Consolidation

Akshat Verma Gargi Dasgupta Tapan Kumar Nayak Pradipta De Ravi Kothari

IBM India Research Lab

Abstract

Server consolidation has emerged as a promising technique to reduce the energy costs of a data center. In this work, we present the first detailed analysis of an enterprise server workload from the perspective of finding characteristics for consolidation. We observe significant potential for power savings if consolidation is performed using off-peak values for application demand. However, these savings come up with associated risks due to consolidation, particularly when the correlation between applications is not considered. We also investigate the stability in utilization trends for low-risk consolidation. Using the insights from the workload analysis, two new consolidation methods are designed that achieve significant power savings, while containing the performance risk of consolidation. We present an implementation of the methodologies in a consolidation planning tool and provide a comprehensive evaluation study of the proposed methodologies.

1 Introduction

According to an estimate [2] based on trends from American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE)[1], by 2014, Infrastructure and Energy (I&E) costs would contribute about 75% while IT would contribute a significantly smaller 25% towards the overall total cost of operating a data center. While there may be a difference in opinion on the relative proportion of I&E and IT costs, there is little disagreement that I&E costs would comfortably be the largest contributor to the cost of operating a data center. Reducing the I&E costs is, or will soon be, a major initiative of most data centers. One promising approach, prompted by virtualization and hardware-assisted isolation, for reducing the I&E costs is server consolidation.

Server consolidation is based on the observation that many enterprise servers do not maximally utilize the available server resources all of the time. Co-locating applications, perhaps in individual virtual machines, thus allows for a reduction in the total number of physical servers, minimizes *server sprawl* as well as the total data center space requirements.

Consolidation reduces the total power consumed by the applications because existing servers are not energy-proportional, i.e., a significant amount of power is consumed even at low levels of utilization [26]. Though server features like *voltage* and *frequency scaling* modify this curve, there is still substantial power drawn at *idle* or *low utilization*. Consolidation thus provides an opportunity to reduce the overall power consumed by operating the servers in a range with a more attractive performance/Watt. For example, if two identical servers each utilizing, say 40% of the resources and drawing 80% of peak power were consolidated onto a single server, the consolidated server would be able to deliver identical performance at significantly less than the $160\%(80+80)$ of the peak power. However, the key to effective consolidation is to estimate the (time-varying) resource requirements of individual applications (virtual machines) and to utilize these estimates along with the power profile of the physical servers to determine the consolidation strategy that can provide the best space-power benefits.

Server consolidation can be loosely broken into static, semi-static and dynamic consolidation. In static consolidation, applications (or virtual machines) are placed on physical servers for a long time period (e.g. months, years), and not migrated continuously in reaction to load changes. Semi-static refers to the mode of consolidating these applications on a daily or weekly basis. On the other hand, dynamic consolidation spans a couple of hours and requires a runtime placement manager to migrate virtual machines automatically in response to workload variations. Many virtualization vendors provide some tooling support for static consolidation [10, 15] with third party providers providing add-on features [9, 8] for inferring hardware constraints etc.

However, these tools essentially provide a policy-based framework with user defined policies and the placement intelligence is fairly simplistic. While multiple dynamic placement frameworks have been researched, in practise, administrators are often reluctant to migrate virtual machines automatically. Instead they prefer an offline or semi-offline framework, to evaluate the proposed placement and manually approve it. Hence, static and semi-static consolidation, where consolidation is performed daily or weekly is a much more appealing technique for administrators in real data centers. Though consolidation for minimizing server sprawl or power is not new, we are not aware of any prior study that utilizes correlation between workloads in a systematic way for determining the most effective static consolidation configuration.

1.1 Static Consolidation: What is new?

While dynamic workload placement has been a well studied problem, it assumes that there is minimal change in the resource requirement of the application during the (typically short) consolidation interval and hence a single resource size suffices. In the past, it has been assumed that the same assumption holds for static consolidation. However, for longer term consolidation there are significant reasons why this assumption fails. First, over a longer period of time, one is likely to see periods of peak as well as reduced application demand. Should the application size be taken to be the maximum, average or some other statistic? Second, placement decisions made based on historical data may not be accurate due to a systematic drift in the load. Third, there is an opportunity to utilize correlation between resource utilization on different virtual servers to influence the consolidation decision. Finally, long term placement has additional objectives like workload balance on active servers.

In summary, a static consolidation framework needs to deal with stochastic variables instead of fixed variables and the behavior of these variables need to be completely understood. We need to identify the right parameters to size workloads for medium or long intervals and assess their impact. It is also important to understand how correlation between applications can be employed for a more effective consolidation. The stability of various workload parameters need to be studied thoroughly to identify the risks involved in consolidation. Finally, existing placement methodologies need to be seen in light of the results of the workload characterization and should be modified, as needed.

1.2 Contribution

We present in this paper the first systematic server workload characterization of a large data center from the perspective of medium (semi-static) or long term (static) consolidation. We study the distribution of the utilization and occurrence of the peak utilization on servers relative to various percentiles and average metrics. We find that the tail of the distribution does not decay quickly for most servers implying that sizing applications based on average utilization has high degree of risk. We also observe significant correlation between applications hosted on different servers. We make the important observation that certain metrics like the 90-percentile as well as cross correlation between applications are fairly stable over time.

We use the insights obtained from our workload characterization to design two new consolidation methodologies, namely *Correlation Based Placement (CBP)* and *Peak Clustering based Placement (PCP)*. We implement the methodologies in a consolidation planning tool and evaluate the methodologies using traces from a live production data center. Our evaluation clearly establishes the superiority of the proposed algorithms. We also bring out the various scenarios in which each methodology is effective and show how to tune various parameters for different workloads.

The rest of the paper is organized in the following manner. We provide a background of server consolidation and the need for a system-level workload characterization in Sec. 2. A detailed workload characterization of a large data center is presented in Sec. 3. We use the insights from the workload characterization to design new placement methodologies in Sec. 4. We present an implementation and a careful evaluation of the proposed methodologies in Sec. 5. We conclude the paper with a summary of our key findings in Sec. 6.

2 Background

In this section, we first present a generalized formulation for server consolidation. The consolidation exercise can be formally stated as follows. Let there be N applications A_i that we need to place on M physical servers S_j for the period T. For each application A_i, let $C(A_i, t)$ denote the resource required in order in order to meet its SLA at time t. This paper does not deal with the problem of translating an application SLA to a resource value and assumes that $C(A_i, t)$ are available from monitored resource data. Let the capacity of a physical server S_j be denoted by $C(S_j)$ and X denote a specific consolidation configuration to specify the placement of applications on physical servers, i.e., an element of X, say $x_{ij} = 1$ if application A_i is placed on server S_j and 0 otherwise. Consolidation requires finding a configuration that optimizes a given cost function. For example, if the objective of consolidation is to optimize power, then we want to find a configuration X that minimizes $P(X)$, where $P(X)$

is a real valued function that provides the power consumed for a specific placement of applications. Further, the placement should ensure that the resource requirements are all applications are met for the entire duration T, i.e., $\forall t \in T, \sum_{i=1}^{N} x_{ij} C(A_i, t) \leq C(S_j)$. Further, we need to ensure that all applications are placed, i.e., $\sum_{j=1}^{M} x_{ij} = 1$.

Dynamic consolidation assumes that T is very short, leading to a single time-independent capacity demand $C(A_i)$ for each application. Hence, the capacity constraint is no longer stoachastic in nature. In dynamic consolidation, for the estimation of $C(A_i)$ and $C(S_j)$, a popular metric in use is the RPE2 metric from IDEAS and almost all the commonly used servers are bench marked with a fixed RPE2 value [24]. The $RPE2$ value of the server is used for $C(S_j)$ whereas the resource requirements of the application are estimated from the CPU utilization of the server. More specifically, if virtualization is not in use then the RPE2 of the host server multiplied by the maximum CPU utilization of the server in the period is used as an estimate of the resource requirements (size) of the application. If virtualization is in use, then the size is computed based on the entitlement of each virtual server on its host physical server, the CPU utilization of the virtual server, and the RPE2 of the host server.

Dynamic consolidation, due to its automated nature, is not preferred by data center administrators. Instead, they opt for static or semi-static consolidation strategies, where they can manually verify and approve the new configuration. However, for static or semi-static consolidation, the crux of the problem is to identify a size parameter that is useful for longer periods. Typically, an administrator may migrate virtual machines at the end of the day or on an identified day of the week. For such long durations, it is imperative to use a size that is able to save a lot of power (by consolidating on few power-efficient machines) as well as ensure that no SLA capacity violations would happen during periods of high load. Hence, the two important objectives in static consolidation are (i)*Overall Power Consumption* and *(ii) SLA Violation*, defined as number of time instances, when the capacity of server is less than the demand of all applications placed on it $\sum_{i=1}^{N} x_{ij} C(A_i, t) > C(S_j)$.

2.1 Related Work

Existing research in workload modeling can be classified into (a) aggregate workload characterization and (b) individual server utilization modeling. Aggregate workload characterization of a web server by Iyenger *et al.* [19] and workload models of a large scale server farm by Bent *et al* [3] fall in the first category. Individual server utilization has been studied in [5, 16, 4]. In [5], Bohrer *et al* use peak-trough analysis of commercial web servers to establish that the average CPU utilization for typical web servers is fairly low. Similar observations on the peak-trough nature of enterprise workloads have been made in [16]. In [4], Bobroff *et al* perform trace analysis on commercial web servers and outline a method to identify the servers that are good candidates for dynamic placement. However, none of these studies provide a characterization of the inter-relationship between various workloads, as required for static consolidation.

There is also a large body of work on energy management in web clusters. Most of the cluster energy management literature addresses the problem of distributing requests in a web server cluster in such a way that the performance goals are met and the energy consumption is minimized [6, 21, 25, 17]. There are a number of papers that describe server or cluster level energy management using independent [22, 13] or cooperative DVS techniques [12, 18]. There are other efforts in reducing peak power requirements at server and rack level by doing dynamic budget allocation among sub-systems [14] or blades [23]. The work closest to the semi-static or static consolidation problem addressed in this paper are the dynamic consolidation methods proposed in [7, 26, 27, 4]. However, the relatively long duration for static consolidation introduces a stochastic nature to individual applications that is not captured in any of these frameworks.

3 Server Workload Analysis

We first present the details of the workload analyzed in this paper.

3.1 Trace Workload Details

The workload analyzed in this paper was collected from the production data center of a multi-national Fortune Global 500 company. The data center runs the core business applications of the enterprise as well as a service delivery portal. Each separate application suite was run from its own server cluster with a dedicated application team. Every application component in a suite ran from a dedicated virtual server, with many virtual servers hosted on a (typically) high end physical server. The traces were collected by the MDMS monitoring framework [20] deployed in the data center. The framework used its own sensors to collect CPU utilization for all the virtual servers with one entry every 5 minutes. We use traces collected over a 90 day period in the year 2007 for our analysis. We use the terms server and application interchangeably as each trace data corresponds to exactly one virtual server and application component.

The tracing methodology depends on sensors deployed in actual production servers over a long period. Hence, the data was noisy in parts due to routine system

Suite-Name	# of Servers	# of Days
AppSuite-1	10	19
AppSuite-2	18	13
AppSuite-3	13	25
AppSuite-4	16	37

Table 1: Workload Details for each cluster

maintenance (server reboots, performance troubleshooting that terminated all daemons including the sensors). Thus, the traces had missing or incorrect data for many time intervals during the trace period. We used a simple interval graph technique to identify the longest contiguous interval, where all the servers in one cluster had monitored data available. Hence, for each server cluster we identified a smaller period which had accurate monitored data available and used these smaller periods for our analysis.

The data center had a large number of clusters and we have selected 4 representative clusters (Table. 1) for this analysis. 'AppSuite-1', 'AppSuite-2' and 'AppSuite-4' had a 2 tiered application with application server components and DB server components. 'AppSuite-3' was a 3-tiered application suite with a few web servers, a few application servers, and a few DB servers. In most cases, multiple application servers used a common DB server. However, for 'AppSuite-2', few application servers had a dedicated DB servers assigned to them. There were no restrictions on co-locating two or more components of an application suite on the same physical server. The detailed information about the applications running in the data center and the virtual server to physical server mapping are withheld for privacy and business reasons.

3.2 Macro Workload Analysis

We begin our workload characterization study with server utilization of individual servers. Due to space limitations, we primarily report our observations on only one server cluster 'AppSuite-1', broadening our observations to other clusters only for important findings.

3.2.1 CPU Utilization Distribution

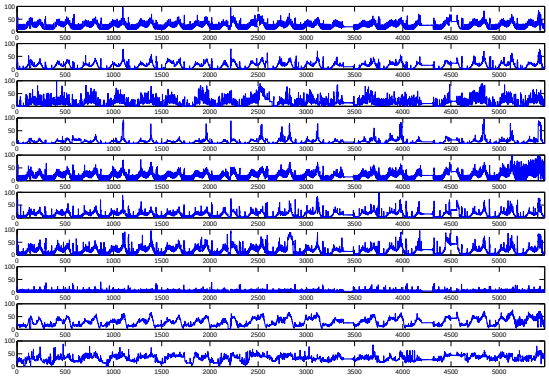


Figure 1: CPU Utilization for AppSuite-1 with Time

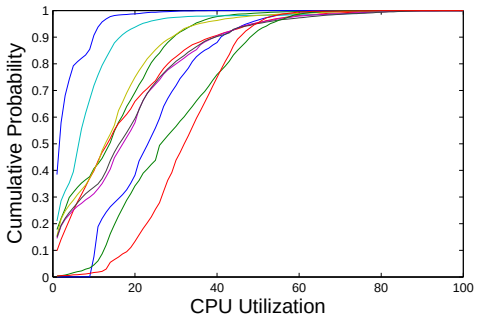
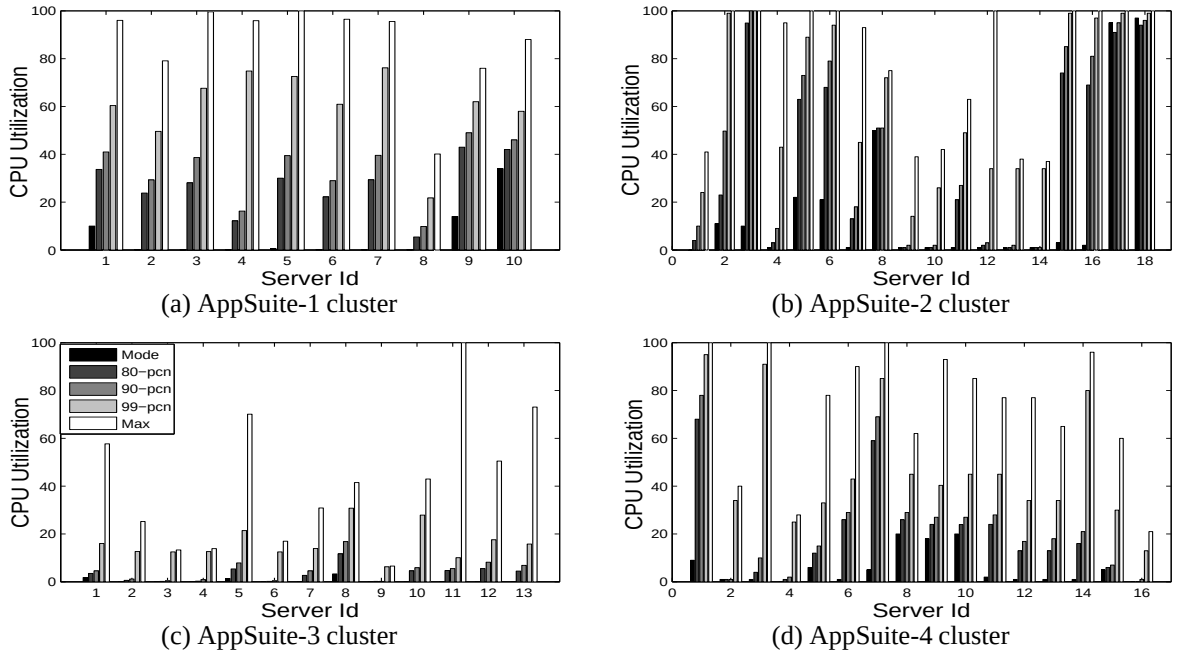


Figure 2: Cumulative Distribution Function of CPU Utilization for AppSuite-1

Fig. 1 shows the CPU utilization of each server in 'AppSuite-1', server1(top) through server10(bottom). The important observation to make here is that all servers barring server8 reach a CPU utilization of 100% at some point in time during the duration of the trace. This has an important implication for consolidation. **Observation 1:** *If consolidation is performed by reserving the maximum utilization for each application, the application may require capacity equal to the size of its current entitlement.* This observation is reinforced by taking a look at the Cumulative Probability Distribution (CDF) of CPU utilization (Fig. 2) for each server of 'AppSuite-1'. An interesting observation in the CDF plot however is the large skew in the distribution. For most of the applications, the CPU utilization at 90-percentile of the distribution is less than half of the peak CPU utilization. Such a skew can be utilized for a tighter consolidation by provisioning less than peak resource consumed by each application.

We drill down further into the skew of the CPU utilization distribution function in Fig. 3(a). We observe that the 99-percentile CPU utilization value is significantly less than the maximum CPU utilization in many cases. This is also in line with observations on other enterprise workloads made in [16]. Interestingly, the 90-percentile CPU utilization is about half or less of the maximum CPU utilization for 9 out of 10 servers. Interestingly, the gap between the 80 and 90-percentile values is less than 10% CPU utilization in all cases and less than 5% in many cases. We also look at the other server clusters in Fig. 3 and find the observations to hold there as well. However, in the 'AppSuite-2' cluster, a few servers have high utilization (Servers 15 to 18) for most of the interval. Hence, in these cases, both the 80 and 90-percentile values are reasonably close to the peak CPU utilization. The above findings lead us to our second important observation. **Observation 2:** *If we could size an application based on 90-percentile CPU utilization instead of maximum CPU utilization, it could lead to significant savings.*

We next observe the variability of the CPU utilization for different servers. To measure the variability, we computed the coefficient of variation (COV) for all the ap-

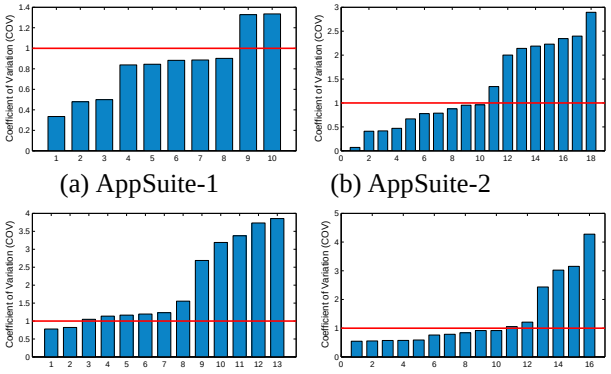


(a) AppSuite-1 cluster (b) AppSuite-2 cluster

(c) AppSuite-3 cluster (d) AppSuite-4 cluster

Figure 3: Comparison of Peak, Mode and Percentile CPU Utilization

plications in a cluster. The coefficient of variation is a normalized measure of dispersion of a probability distribution and is defined as $COV = \sigma/\mu$, where σ is the standard deviation and μ is the mean of the distribution.



(a) AppSuite-1 (b) AppSuite-2

(c) AppSuite-3 (d) AppSuite-4

Figure 4: Coefficient of Variation for all clusters. Servers in each cluster are sorted by COV for easy comparison.

COV is a useful statistic for comparing the degree of variation and equals 1 for exponential distribution. Distributions with $COV>1$ (such as a hyper-exponential distribution) are considered high-variance, while those with $COV<1$ are considered low-variance. The coefficient of variations for all the clusters are shown in Fig. 4. We observe that all clusters have at least a few applications with high-variance distributions and 'AppSuite-3' has the largest number of applications with $COV>1$. There are also applications with low-variance distributions. However, it is well known that combining a heavy tailed distribution ($COV>1$) to another independent (or positively correlated) distribution with an exponentially decaying tail ($COV=1$) leads to an aggregate distribution, which is heavy-tailed. This leads to our third important observation. **Observation 3:** *If a statistical measure that ignores the tail of the distribution is used for sizing an application, the consolidated server may observe a large number of SLA capacity violations.*

3.2.2 Correlation

Our first few observations bring out the potential savings if applications were sized based on percentile values as opposed to peak values. However, sizing based on a non-peak value may lead to significant SLA violations if co-located applications peak together. Hence, we next study the correlation between applications belonging to the same application suite. The correlation between a pair of applications with timeseries $\{x_1, x_2, \ldots, x_N\}$ and $\{y_1, y_2, \ldots, y_N\}$ is represented by the *Pearson correlation coefficient*,

$$r_{xy} = \frac{N\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{N\sum x_i^2 - (\sum x_i)^2}\ \sqrt{N\sum y_i^2 - (\sum y_i)^2}} \quad (1)$$

Fig. 5 shows the pair-wise correlation between the applications of 'App-Suite1'. One may observe that there are many applications that have significant positive correlation. On the other hand, there are also a few applications (e.g., on Server 3,8, and 10) that have minimal correlation with other applications. The observations highlight that (a) there is a risk of SLA violation if consolidation methods are not aware of correlation and (b) there is

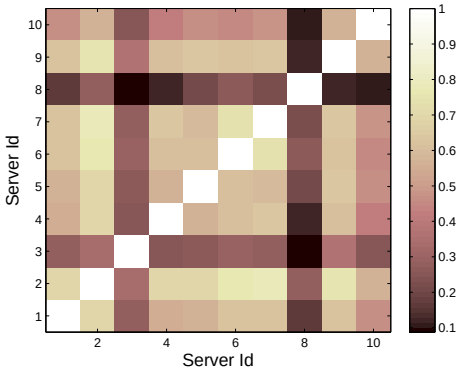


Figure 5: Inter-Server Correlation for AppSuite-1 cluster

potential for placing non-correlated applications to mitigate this risk. The other clusters have less correlation between servers but there are still a significant number of servers (more than 25%) that exhibit correlation with one or more other servers. One may observe that virtual servers that are part of a multi-component application have a high likelihood of being correlated. However, since in most cases, multiple (4 or 8) application servers were sharing a common DB server, the correlation was not strong. 'App-Suite2' however had 4 (application server, db server) pairs that were dedicated. As a result, even though the workload to this suite had low intrinsic correlation, the two-tier nature of the application suite introduced correlation. Hence, multi-tier applications with a one-to-one mapping between servers in different tiers are likely to exhibit correlation even for workloads with no intrinsic correlation. This leads to our next important observation. **Observation 4:** *There are both positively correlated and uncorrelated applications in a typical server cluster. Hence, correlation needs to be considered during placement to avoid SLA capacity violations.*

3.2.3 Temporal Distribution of Peaks

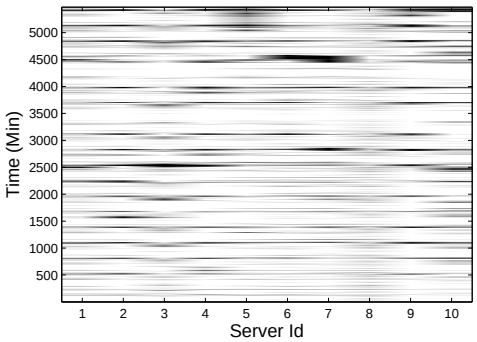


Figure 6: Duration of Peak CPU utilization ($>$ 90-percentile) for AppSuite-1 cluster. Dark lines indicate sustained peaks.

We have used the correlation coefficient as an indicator of the temporal similarity between two applications. However, correlation is a comprehensive metric that captures temporal similarity between two applications at all levels (both peak and off peak). Capacity violations, though, occur when two applications sized by an off-peak value peak together. Hence, we look at the correlation between only the peaks for various applications in Fig. 6. We observe that there are apps with low correlation, but whose peaks may be correlated. Further, there also exists correlated apps whose peaks typically do not occur at the same time (e.g., Server 5 and 7). This leads to our next important observation. **Observation 5:** *Correlated Applications may not always peak together. Similarly, non-correlated applications may also peak together in some cases.*

3.3 Stability Analysis

Static and semi-static placement decisions are made for extended periods of time. Hence, there is a need to analyze the stability of workload statistical properties to ensure the reliability of the placement decisions. In this section, we study the workload periodicity, variation in statistical properties like mean, 90-percentile and correlation co-efficient over the observation period.

3.3.1 Periodicity analysis of utilization data

We first analyze the periodicity of the collected data. It will help to find the repeating patterns, such as the presence of a periodic signal which has been buried under noise. The usual method for deciding if a signal is periodic and then estimating its period is the auto-correlation function. For a discrete timeseries $\{x_1, x_2, \ldots, x_N\}$ with mean μ and variance σ^2, the auto-correlation function for any non-negative integer $k < N$ is given by

$$R(k) = \frac{1}{(N-k)\sigma^2} \sum_{n=1}^{N-k} [x_n - \mu]\,[x_{n+k} - \mu], \quad (2)$$

Essentially, the signal $\{x_n\}$ is being convolved with a time-lagged version of itself and the peaks in the auto-correlation indicate lag times at which the signal is relatively highly correlated with itself; these can be interpreted as periods at which the signal repeats. To enhance the analysis, we also computed the magnitude spectrum of the timeseries, $|X_k|$, where $\{X_k\}$ is the Discrete Fourier Transform (DFT) of $\{x_n\}$ and is defined by

$$X_k = \frac{1}{N} \sum_{n=1}^{N} x_n e^{-\frac{2\pi i}{N}(k-1)(n-1)}, \;\; 1 \le k \le N. \quad (3)$$

We study the auto-correlation function and magnitude spectrum of the utilization data for all the applications and find that some servers exhibit nice periodic behavior, whereas some servers do not follow any particular pattern. Fig. 7 shows a periodic pattern with a time period of one day as the lag between two consecutive peaks in the auto-correlation function is one day and there is a peak in the magnitude spectrum corresponding to it.

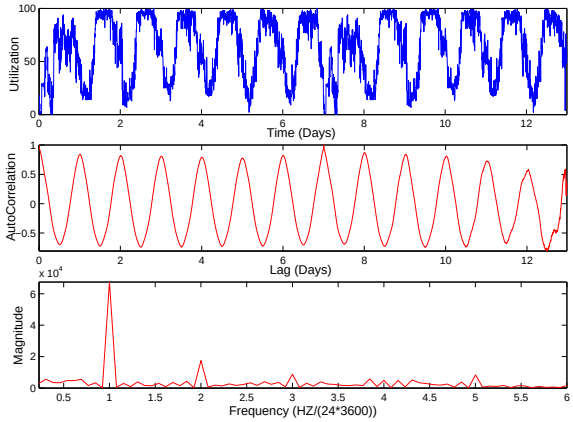


Figure 7: The timeseries, auto-correlation and frequency spectrum of this workload shows a periodicity of 1 day

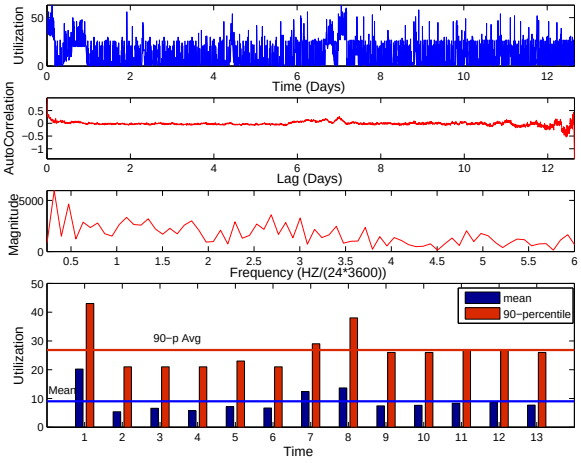


Figure 8: The timeseries, auto-correlation and frequency spectrum plot of the workload do not show any periodicity, but the mean and 90-percentile values show stability.

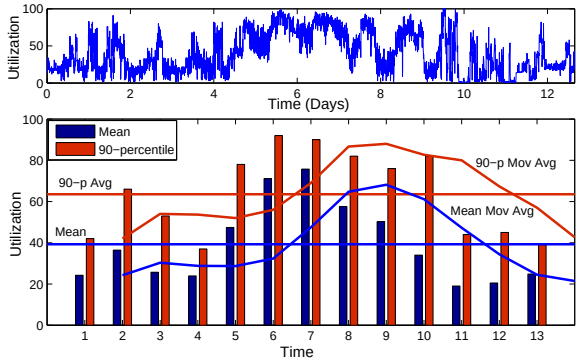


Figure 9: The timeseries has no regular pattern and the mean and 90-percentile statistics also vary significantly over the time period, but the moving averages track the statistic well.

This kind of workloads can be predicted with significant reliability. Many applications do not show any periodic pattern in the observed period, however, the statistical properties remain consistent over a long period. To analyze the consistency, we computed the mean and 90-percentile statistics over several windows of length 1 day. Fig. 8 shows that although the workload has no periodic pattern, the mean and 90-percentile statistics remains stable over most part of the observed period. Hence, for such workloads, the statistics can be estimated reliably. A third category of applications neither show any periodic behavior, nor any statistical consistency over a long period. However, for these applications, the moving averages follows the actual mean and 90-percentiles closely over the observed period (Fig. 9) and can be used for estimation. These observations lead to the following conclusion. **Observation 6**: *Some servers exhibit periodic behavior and the future pattern can be reliably forecasted with a day or a week of data. For many non-periodic servers, the statistical properties are fairly stable over time. For highly variable servers, an adaptive prediction method like MovingAverage should be used to estimate the statistical properties.*

3.3.2 Stability in Correlation

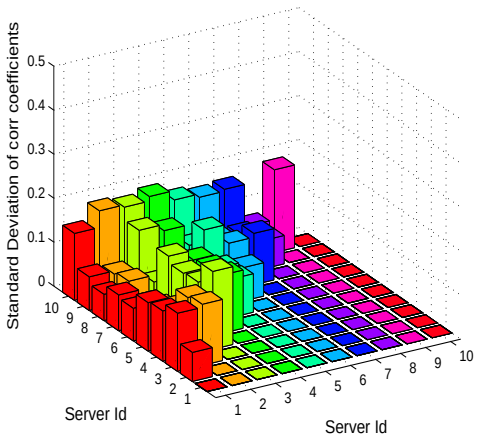


Figure 10: Stability of Correlation for App-Suite1 (Half the values have been deleted for visual clarity)

We have observed that correlation between applications should be used while making placement decisions. We next study the stability in correlation for AppSuite-1, which has the highest correlation amongst all the clusters. For this purpose, we compute the correlation between all pairs of applications for every day separately during the whole duration of the trace and compute the standard deviation across these daily correlation values. We observe in Fig. 10 that the standard deviation is fairly low, indicating the stability of correlation across time. **Observation 7**: *The correlation between the CPU utilization of various servers is fairly stable across time.*

4 Placement Methodologies

We now present various placement methodologies.

4.1 Workload-unaware Placement

We presented the $pMapper$ power-aware application placement methodology and system in [26] in the context of a runtime placement controller. $pMapper$ minimizes fragmentation using an enhancement of *First Fit Decreasing (FFD)* bin-packing algorithm and uses a novel *Order Preservation* property to select the right server for any application being migrated in order to minimize power. The algorithm optimizes the use of one resource (typically CPU utilization) during packing and treats other resources (e.g., memory, I/O) as constraints. Hence, it always comes up with a feasible packing for all resources but allocates only one resource in an optimized manner. The methodology used by $pMapper$ does not focus on resource sizing of each VM for the next placement interval, which is predicted by a *Performance Manager*. An important thing to note here is that $pMapper$ is designed for short consolidation intervals. There are two important implications of such an approach. Firstly, each application is sized independently and a single number is used to size an application. Secondly, as placement decisions need to be aware of application migration costs, few applications are migrated and the relocation decision takes an existing (old) placement into account. However, such an approach can still be applied for *static* consolidation with much longer consolidation intervals. In such a static placement scenario as the one considered in this paper, the $pMapper$ methodology is naturally adapted by sizing an application based on the peak resource usage of the application in the (longer) placement period. Note further that in the case of SLA governed data centers, one can use less strict sizing functions. For example, if the SLA requires that resource requirements are met for at least 99% of the time, one could use a VM size that ensures a tail violation probability of 1%. Similarly, one may also choose to size all applications based on a metric like mode or median, if short-term violations are acceptable. We term this family of placement methodologies using peak, percentile, mode etc. based sizing as *Existing* placement methodologies.

4.2 Correlation Based Placement

We now present our first methodology that leverages the observations made from our workload analysis to place applications in a more effective manner. This methodology aptly named as the *Correlation Based Placement (CBP)* is based on the following important observations.

- The peak resource usage of an application is significantly higher than the resource usage at most other times (e.g., size at 90% cdf). (Fig. 2, 3)
- If we size applications by an off-peak metric and place correlated applications together, there is a high risk of SLA capacity violation.
- If two uncorrelated applications are placed together and sized individually for a violation probability of $X\%$, the probability that both of them would violate their sizes at the same time is $(X^2)\%$.

To take an example, consider two applications A_1 and A_2. Assume that both A_1 and A_2 have a maximum size of 1000 RPE2 units with a 90 percentile value of 500 RPE2 units. Further, assume that A_1 and A_2 are uncorrelated with each other. It is now easy to see that if we place A_1 and A_2 on a single server and allocate 500 RPE2 units each to both the applications, the probability that both of them would exceed their allocation at the same time is only 1%. Hence, provisioning based on 90 percentile and placing uncorrelated applications together can lead to a potential savings of 50% over the peak-based sizing method. CBP uses exactly these ideas to size individual applications based on a tail bound instead of the maximum size of the application. Further, it adds co-location constraints between positively correlated applications so that two such applications are not placed on the same server. The number of actual constraints added can be controlled using a tunable $Correlation\,Cutoff$. Hence, CBP proceeds in very similar manner to the $pMapper$ algorithm with few key differences: (i) We add co-location constraints between any two positively correlated application pairs (A_i, A_i') that exhibit a correlation coefficient above the $correlationthreshold$ (ii) We size applications based on a tail bound instead of the maximum value and (iii) In the inner loop of $pMapper$ where we find the most power-efficient server S_j that has resources for an application A_i, we also make a check if none of the already placed applications on S_j have a co-location constraint with A_i. If indeed there is such an application, we mark the server ineligible and consider the next server for placing the application.

It is easy to see that CBP incurs an overhead in the computation of correlation for all application pairs.

Theorem 1 *Given N applications and a timeseries with d points, CBP takes $O(N^2 d)$ time to find the new placement.*

We have proposed the CBP methodology that takes an existing dynamic consolidation algorithm and adapts it to work in a static or semi-static consolidation scenario. CBP adds co-location constraints between correlated applications to ensure that an application can be sized based on an off-peak value. However, it adds a hard constraint between correlated applications.

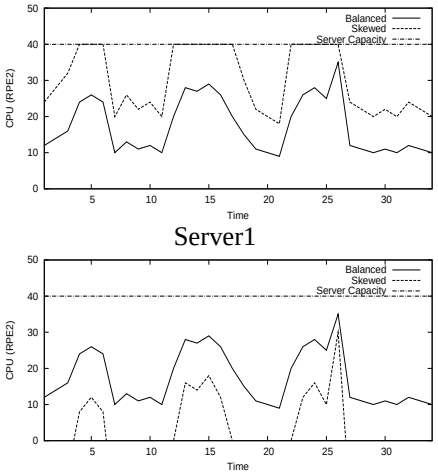


Figure 11: Fractional Optimal Solutions: Balanced and Skewed

We now take a closer look at the problem to understand the nature of the optimal solution. Consider a set of 6 applications that need to be placed on a set of servers, each with a capacity of 40. There are two potentially fractional optimal solutions, as shown in Fig. 11. A balanced solution would pack half of the timeseries in the first server and the other (balanced) half in the other server. A skewed solution would pack the first server to the maximum capacity and pack the remaining applications in the second server. CBP and other dynamic consolidation algorithms aim to approach the skewed optimal solution. However, from an administrative point of view it may be preferred to have balanced workload across all active servers.

A second problem with CBP may arise when there are many correlated applications. In the above example, if there are 3 applications that are positively correlated, we would need a minimum of 3 servers to satisfy the co-location constraints. Finally, computing the correlation between all pairs of applications is expensive (quadratic in nature) and may not be applicable for large number of applications and trace periods. We address these issues in another methodology next.

4.3 Peak Clustering Based Placement

We address the issues with CBP in a new consolidation method called *Peak Clustering based Placement (PCP)*. PCP is based on the following fundamental observations 1) Correlation between peaks of applications is more important than correlation across the complete timeseries of the applications. 2) A set of applications that peak together are distributed evenly across all active servers in the optimal solution. However, two applications with correlated peaks may still be co-located. 3) Co-located applications that do peak together can use a common buffer for their peaks and each have a reservation equal to an off peak value.

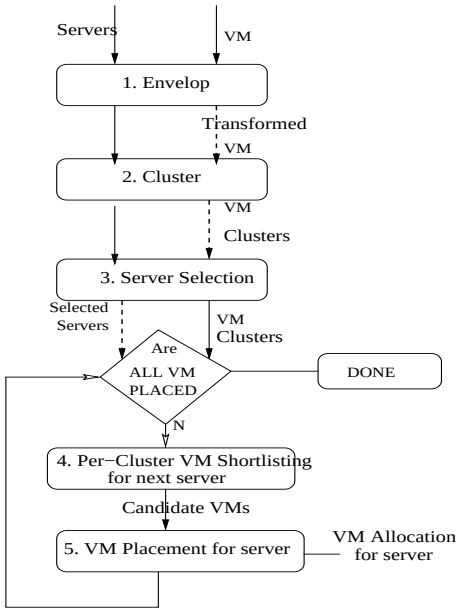
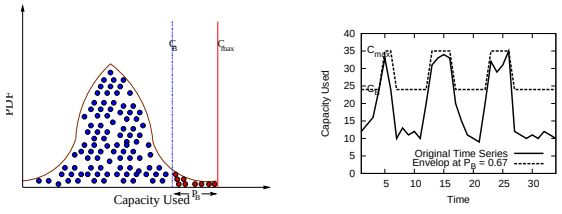


Figure 12: Overall Placement Flow

PCP uses these ideas to first identify clusters of applications with correlated peaks. One may observe that the number of such clusters may become very large if we use the original timeseries with the complete range of values. Hence, PCP uses a novel two-level envelop of the original time-series of each application for clustering. The envelop has a single value to represent all CPU utilization for the body of the distribution and another value for all points in the tail of the distribution. On each active server, it then reserves space for each cluster in proportion to the size (lower level of envelop) of the applications in that cluster and keeps a buffer space equal to the maximum peak across all clusters. Each application cluster shortlists a set of applications for its reservation and PCP does a final selection of applications for the server. The overall flow of PCP is described in Fig. 12.



(a) (b)

Figure 13: (a) Calculation of steps C_B, C_{max} for Envelop and (b) Envelop Creation

In step 1, PCP starts by using an $Envelop$ function that transforms the original time series for each application to a two-level envelop. Given any tail bound P_B (e.g., 10%), the $Envelop$ computes a value C_B such that the probability that application's CPU usage exceeds C_B is bounded by P_B. It also identifies the maximum capac-

ity used by the application as C_{max} (Fig. 13). We then transform the original timeseries by a two-level time series in the following manner. If at a given time, the capacity used by the application is greater than C_B, we replace it with C_{max}. Otherwise, replace it with C_B. Hence, the body of the distribution is replaced by C_B and is referred to as size. The tail is replaced by C_{max}. The timeseries for the transformed VM is stored as a set of ranges during which the size C_B is exceeded. The next step in PCP is to cluster workloads based on the correlation of their peak ranges. The clustering step uses a similarity function to identify workloads with correlated peaks. For each application A_i, the similarity function is used to identify if the envelop of the application is covered by any existing cluster center. If no such cluster center exists, then a new cluster is started with A_i as the cluster center.

Step 3 in the overall flow of PCP is to sort servers based on their power efficiency. We define marginal power efficiency for a server with capacity Cap_j running at capacity ρ_j as the slope of the capacity vs power curve at ρ_j capacity and overall power efficiency of the server as the ratio of the capacity of the server Cap_j to the peak power consumed by the server. The server selection method in our earlier work [26] used marginal power efficiency as the metric to sort servers. Dynamic consolidation requires us to make changes in an incremental, online manner from an existing configuration and hence, marginal power efficiency is a good metric for server selection. On the other hand, static consolidation may involve multiple global changes and hence, we use overall power efficiency to rank servers in PCP.

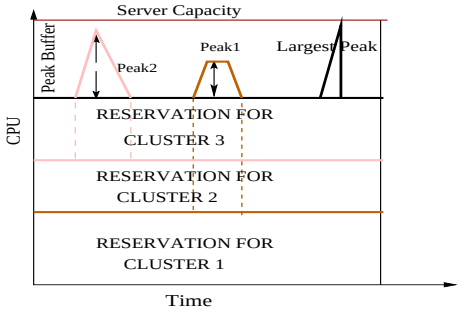


Figure 14: Server Reservation: Each cluster gets a proportional reservation. There is a buffer for the maximum peak across all clusters.

The final steps in PCP pack all the applications on the minimal number of servers, while packing the more power efficient servers first. The method picks the next highest ranked server and selects a set of applications from each cluster in a proportional manner. Given a server with capacity Cap_j to pack and a set of applications yet to be placed, we calculate the sum of the sizes (C_B) of all remaining applications as $TotalSize$. For each cluster M_k, we calculate the sum of the sizes $Size_k$ of its applications and $Peak_k$ as the sum of the peak buffers $(C_{max} - C_B)$ of its applications. We also calculate $MaxBuffer$ as the maximum of the peak buffers across all clusters. Once these metrics are computed, each cluster M_k selects a set of applications such that the overall size $CandSize_k$ and peak buffer $CandBuffer_k$ of the cluster is given by

$$CandSize_k = \frac{Size_k}{TotalSize + MaxBuffer} \quad (4)$$

$$CandBuffer_k \leq MaxBuffer \quad (5)$$

An example server reservation is described in Fig. 14. In this example, there are three clusters and a proportional reservation is made for each cluster that equals $CandSize_k$. Further, capacity is kept spare for peaks and equals the maximum peak across all the three clusters. Since the consolidation can only pick integral solutions, each cluster returns a set of applications whose sizes add up to its proportion or the last added application may exceed its proportion. Hence, as a final selection step, for each cluster that returns k candidates, we automatically select the first $(k-1)$ candidates and add the last candidate to a tentative application pool. We then globally select the candidates from the tentative pool such that the capacity bounds of the server is not violated. In order to reduce fragmentation, at all levels of selection we break ties between applications by preferring the larger applications first. This allows PCP to strike a balance between reducing fragmentation costs and proportionally selecting applications across different clusters.

Analysis Period	120 hrs
Evaluation Period	24 hrs
P_B for PCP	0.9
Correlation Cutoff for CBP	0.5

Table 2: Baseline Parameters

5 Experimental Evaluation

We have performed extensive experiments to evaluate the proposed methodologies. We first detail out our evaluation setup and then present some of our key findings.

5.1 Evaluation Setup

The methodologies have been implemented as part of an Consolidation Planning Tool from IBM called Emerald [11]. Emerald has built-in adapters to input trace data in various formats, a module to collect the current server inventory and a knowledge base that includes a catalog of various server platforms, their RPE2 values and power

models(Power Vs CPU Utilization). The power models are derived from actual measurements on the servers and are used by a *Power Estimation* module to estimate the power drawn by any candidate placement. We feed traces for the 4 application suites described in Sec. 3 for the evaluation study. Each server in this application suite was either a virtual machine or a standalone server in the data center. Hence, a single physical server may host one or more of these virtual servers in the data center. In our baseline setting, the physical servers were kept to be the same as in the data center. Further, Emerald allows an administrator to specify an analysis period for the traces followed by an evaluation period, where the effectiveness of the proposed placement is evaluated by the *Power Estimation* module. PCP uses a tail bound parameter P_B to create the envelop whereas CBP uses a correlation cutoff parameter to identify if two applications are correlated. The baseline settings of all experimental parameters are listed in Table. 2.

We evaluate the performance of the proposed methods in comparison with *Existing* methods based on dynamic consolidation. We run the *Existing* method with two different sizing functions: (i) *Peak_Existing* sizes all applications based on their peak values and (ii) *Mode_Existing* sizes all applications based on the mode of the distribution. There are three different objectives of consolidation: a)minimize power b) minimize risk of consolidation (c) balance workload across active servers. We use the number of capacity violations as the metric for risk. To investigate load imbalance, we estimate the average CPU utilization for all active servers during the evaluation period and identify the server with the maximum average CPU utilization. The difference between the CPU utilization of the highest loaded server and the average utilization across all servers is taken as the metric for load imbalance. We compare all the methodologies along all these objectives. We also measured the running time of various methodologies to assess their scalability.

In order to generalize our results and look at various settings, we also experimented with changes in our baseline setting. Since these methodologies need to deal with fragmentation, we also simulate a placement where the initial number of virtual servers on a physical server are increased or decreased. Towards this purpose, we assume that the virtual servers are placed on a different capacity server of the same platform, i.e., with more or less processors as required for the simulation. Further, seasonal variations were observed in many workloads and hence, we increase and decrease the training period from the baseline setting. Finally, we vary the tuning parameters of CBP and PCP and study their impact.

5.2 Performance Comparison

Power Consumed and SLA Violations: Fig. 15 shows

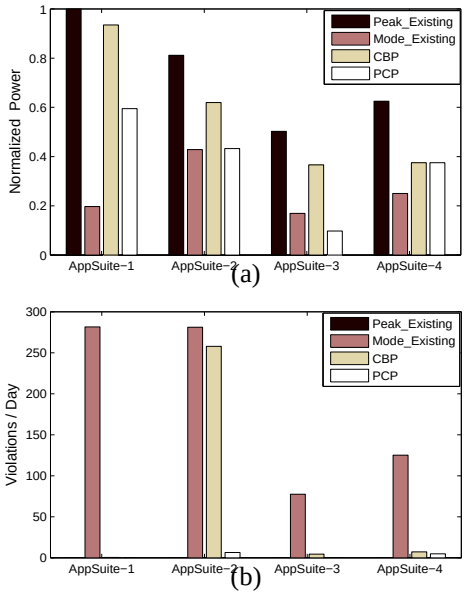


Figure 15: (a) Power Consumed and (b) SLA violations in all Clusters for various placement methodologies. Absence of bars indicate zero violation.

the power consumed in all 4 clusters as a result of placement recommendations made by *Peak_Existing*, *Mode_Existing*, *CBP* and *PCP* and the corresponding SLA violations. A striking observation is that the *Peak_Existing* methodology saves no power in AppSuite-1. Closer look at Fig. 3 reveals that barring one application, all other applications in this cluster have a peak CPU utilization of 80% or more. Hence, a methodology that sizes applications based on peaks is of no use in this cluster. The overall power consumed by *Mode_Existing* is typically the lowest as it sizes aggressively, while the *Peak_Existing* uses the maximum power. On the other hand, *Peak_Existing* has the lowest violations (typically zero) whereas *Mode_Existing* has the highest violations. Both CBP and PCP lie mid-way between these extremes, with PCP attaining about $(20-40)\%$ lower power consumption than CBP and significantly lower violations.

Another observation is that CBP saves significant power in comparison to *Peak_Existing* in all clusters other than AppSuite-1, while it faces very high violations for AppSuite-2. To understand this, we recall from Sec. 3 that the AppSuite-1 cluster has high correlation, AppSuite-2 has medium correlation and the remaining two clusters have a low correlation. As a result, CBP adds many co-location constraints in AppSuite-1 leading to a large number of active servers and not saving any power. In contrast, the medium correlation in AppSuite-2 results in CBP not making recommendations to separate these workloads. However, even though the workloads are not correlated, their peaks show up in the same time interval, leading to high violations by CBP. This peak correlation behavior is exploited only in the PCP

algorithm which decides to judiciously separate offending workloads if their simultaneous peaks can risk overshooting the server capacity, thereby causing violations. Thus PCP sees a nominal number of violations per hour in all clusters.

A surprising result observed is that PCP consumes less power than *Mode_Existing* for AppSuite-3. Recall that the server selection methodology in *Existing* methodology explores locally to reduce migration cost. On the other hand, server selection in PCP can explore new placements that entail a large number of migrations. Since AppSuite-3 was lightly loaded, consolidation would require large number of migrations. Hence, PCP came up with a different server selection than other methodology for this particular cluster, leading to additional power savings.

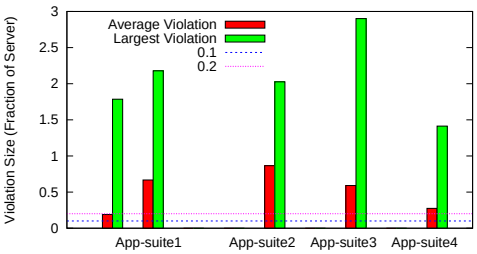


Figure 16: Violations for $Mode\_Existing$ on all active servers

We observed in Fig. 15 that *Mode_Existing* has the potential to save a lot of power but may incur a large number of violations. The correlation in the first two clusters of AppSuite-1 and AppSuite-2 especially impacts *Mode_Existing*, leading to very high violations. Results show that among the 9 active servers in all clusters, 5 of them faced SLA capacity violations. We study the size of the violations in the placement computed by *Mode_Existing* in Fig. 16. In practise, administrators allow a buffer capacity (typically 10% of the server) and hope that any peaks can be served from this buffer capacity. We observe that even a buffer capacity of 20% is not able to handle the burstiness of the workload. Amongst 5 servers, the average violation exceeds 20% of the server capacity in 4 servers and the peak violation exceeds the size of the server itself in all the servers. Hence, a workload-unaware strategy that uses a buffer capacity to handle peaks is not feasible in practise.

Workload Balancing: We next investigate the load balance across the active servers achieved by various methodologies in Fig. 17. We observe that *Mode_Existing* and CBP have servers that are always overloaded (average utilization of highest loaded server is 100%) for AppSuite-1 and AppSuite-2. Such a placement is not suitable in practise. Further, for all methodologies other than PCP, there is a large imbalance between the average utilization of the highest loaded server

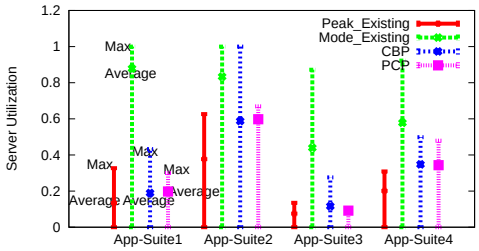


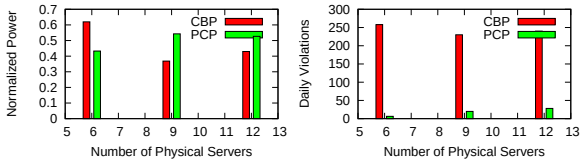
Figure 17: Average and Maximum Utilization of Servers

Cluster (No. of Apps)	Existing (ms)	PCP (ms)	CBP (ms)
AppSuite-1 (10)	10.1	47.7	34
AppSuite-3 (13)	13.5	55.2	55
AppSuite-4 (16)	30.1	39.8	81
AppSuite-2 (18)	21.2	47.7	107

Table 3: Running Time for various methodologies

and the average utilization of the cluster. This is a direct consequence of the design of algorithms, where PCP favors balancing and the other algorithms favor creating a skew.

Running Time: We study the running time of various methods in Table 3. The CBP algorithm very clearly says a super-linear relationship with the number of applications (N) because of the N^2 correlation co-efficient computation between all pairs of applications. The *Existing* method in general scales linearly with the increase in number of applications. PCP has a dependence on (a) the number of applications, (b) the number of peak ranges in a cluster and (c) the number of clusters it creates. Recapitulate that AppSuite-3 has the highest CoV (Fig. 4), which manifests in a large number of peak ranges. As a result, AppSuite-3 has the highest running time, even though the number of applications N is only 13. Overall, PCP has a running time that is double of $Existing$ and fairly stable. The running time of CBP on the other hand increases super-linearly with the number of applications N.



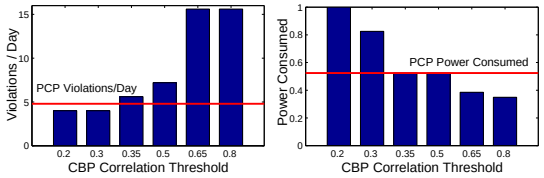
(a) Power (b) Violations

Figure 18: Impact of change in virtual servers per physical server in AppSuite-2

Fragmentation: We next investigate the ability of CBP and PCP to deal with fragmentation (ratio of application capacity $C(A_i)$ to server capacity $C(S_j)$). We change the original servers in the data center and simulate placement on a larger number of lower capacity servers. Fig 18 shows the behavior of CBP and PCP with increase in number of physical servers for AppSuite-2, which has the largest number of virtual

servers. CBP adds a fixed number of co-location constraints and needs at least as many servers as the maximum number of constraints added to any application. Hence, it suffers when few high capacity servers are available. On the other hand, PCP tries to proportionally allocate applications from each cluster (of applications with correlated peaks) and hence should perform better when many applications can be packed on few high capacity servers. Both these intuitions are validated in Fig. 18 as CBP performs better with increase in number of servers and PCP fares worse. However, in both the cases, CBP suffers more violations than PCP. In summary, CBP is more suited when large applications are placed on low or medium capacity servers, whereas PCP is more suitable for consolidating a large number of applications on few high-end servers.

5.3 Tuning CBP



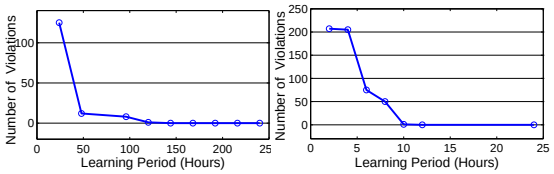
(a) Violations (b) Power

Figure 19: Power drawn and SLA Violations for CBP with changing correlation cutoff in AppSuite-4

The performance of CBP depends on the correlation cutoff parameter that is used to decide if correlation constraints need to be added between a pair of applications. Fig. 19 shows CBP performance with different thresholds, with the corresponding PCP metric shown as a reference line for AppSuite-4. Using a very low correlation threshold (0.2) creates constraints even between weakly correlated workloads thereby reducing the SLA violations (to even below that of PCP). However this comes at a cost of increasing the number of active servers, thereby consuming 50% more power than PCP. On the other hand, using a high correlation threshold (0.8) creates constraints only when workloads exhibit very high degree of correlation. As a result, the power consumed can be lowered below PCP at the cost of higher violations. We recommend an operating range $(0.35 - 5)$ for significant power savings with reasonable number of violations. However one may set the threshold to 0.2 for critical applications and 0.8 for best-effort applications. We do observe that even though CBP can achieve either significant power savings or low violations, it is not able to achieve the trade-off as well as PCP.

5.4 Tuning PCP

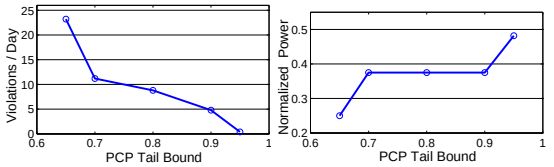
The important configuration parameters for PCP are the length of the training period and the tail bound P_B used for creating envelopes. We first show the impact of



(a) (b)

Figure 20: Performance of PCP with change in training period: (a)AppSuite-2 with weekly periodicity (b)AppSuite-4 with daily periodicity

available history (length of training period) on the performance of PCP. Fig. 20 shows SLA violations of AppSuite-2 and AppSuite-4 of PCP with change in the analysis period. We observe that without adequate history of workload repeatability, PCP can have very high number of violations. However, once adequate history is available, the number of violations fall to 0 for both the clusters. We have included AppSuite-2 and AppSuite-4 because the first cluster has a daily pattern whereas the second cluster has a weekly pattern. We observe that PCP is able to infer the characteristics of the workload in about half of the length of the period. Hence, for AppSuite-2, it takes about half a day of training data to reduce violations, whereas for AppSuite-4, we require about 4 days of training data. We also observe that the impact of historical trends happens in a discrete manner. Hence, for AppSuite-4, the availability of 2 full days of data leads to a significant decrease in violations as diurnal trends are available. However, any further data is not useful until we reach 5 days, when weekly trends become available. A key strength of PCP is that a user can easily determine the length of training period for a server cluster using a few sample runs and then store only the relevant history for future analysis. We next investigate



(a) (b)

Figure 21: Performance of PCP with change in tail bound: (a) Hourly Violations (b) Power Consumed

the impact of the tail bound on the performance of PCP in Fig. 21. A high tail bound in PCP leads to a conservative size and smaller durations of peak activity. Hence, a high tail bound may lead to lesser violations but may lead to a higher power consumption. We observe this intuition holds in the experiments, as the violations fall to 0 for a tail bound of 0.95 but at the cost of higher power. Hence, the administrator can choose a bound based on the criticality of the applications in the cluster.

6 Conclusion

In this work, we have presented the server workload analysis of a large data center. We have investigated a large number of characteristics relevant for medium (semi-static) to long term (static) consolidation in order to save power. The workload study shows that there is a large potential for power savings by using off-peak metrics for sizing applications. However, correlation between applications can lead to significant capacity violations if consolidation methodologies do not take them into account. We design two new consolidation methodologies CBP and PCP that use an off-peak metric for sizing and another metric to ensure that peaks do not lead to violations. Our experimental evaluation shows that PCP achieves superior power savings, low violations and good load balance across active servers. Our work opens up further research in re-design of placement methods in light of the workload characteristics observed in our work.

7 Acknowledgements

We would like to thank our shepherd Mahadev Satyanarayanan and anonymous reviewers for insighted comments that have helped improve the paper.

References

[1] ASHRAE Technical Committee 9.9. Datacom equipment power trends and cooling applications, 2005.

[2] C. Belady. In the data center, power and cooling costs more than the it equipment it supports. http://www.electronics-cooling.com/articles/2007/feb/a3/, 2007.

[3] L. Bent, M. Rabinovich, G. M. Voelker, and Z. Xiao. Characterization of a large web site population with implications for content delivery. In *WWW*, 2004.

[4] N. Bobroff, A. Kochut, and K. Beaty. Dynamic placement of virtual machines for managing sla violations. In *IM*, 2007.

[5] Pat Bohrer, Elmootazbellah N. Elnozahy, Tom Keller, Michael Kistler, Charles Lefurgy, Chandler McDowell, and Ram Rajamony. The case for power management in web servers. In *Proc. Power aware computing*, 2002.

[6] J. Chase and R. Doyle. Balance of Power: Energy Management for Server Clusters. In *Proc. HotOS*, 2001.

[7] J. S. Chase, D. C. Anderson, P. N. Thakar, A. Vahdat, and R. Doyle. Managing energy and server resources in hosting centers. In *Proc. ACM SOSP*, 2001.

[8] Cloud computing software for data centers from Cassatt. http://www.cassatt.com/products.htm.

[9] Server Consolidation and Virtualization Analysis by CiRBA. http://www.cirba.com/.

[10] VMWare Guided Consolidation. http://www.vmware.com/products/vi/vc/features.html.

[11] G. Dasgupta, A. Sharma, A. Verma, A. Neogi, and R. Kothari. Emerald: A tool to help data centers go green. In *Under Review*, 2008.

[12] E. Elnozahy, M. Kistler, and R. Rajamony. Energy- efficient server clusters. In *Proceedings of the Workshop on Power-Aware Computing Systems.*, 2002.

[13] M. Elnozahy, M. Kistler, and R. Rajamony. Energy conservation policies for web servers. In *Proc. of USENIX Symposium on Internet Technologies and Systems*, 2003.

[14] W. Felter, K. Rajamani, T. Keller, and C. Rusu. A performance-conserving approach for reducing peak power consumption in server systems. In *Proc. of International Conference on Supercomputing*, 2005.

[15] Virtual Iron: True Server Virtualization for Everyone. http://www.virtualiron.com/.

[16] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper. Workload analysis and demand prediction of enterprise data center applications. In *IISWC*, 2007.

[17] T. Heath, B. Diniz, E. V. Carrera, W. Meira Jr., and R. Bianchini. Energy conservation in heterogeneous server clusters. In *PPoPP*, 2005.

[18] Tibor Horvath. Dynamic voltage scaling in multitier web servers with end-to-end delay control. *IEEE Trans. Comput.*, 56(4), 2007.

[19] A. K. Iyengar, M. S. Squillante, and L. Zhang. Analysis and characterization of large-scale web server access patterns and performance. In *Int'l World Wide Web Conference*, 1999.

[20] Bharat Krishnamurthy, Anindya Neogi, Bikram Sengupta, and Raghavendra Singh. Data tagging architecture for system monitoring in dynamic environments. In *Proc. NOMS*, 2008.

[21] K. Rajamani and C. Lefurgy. On evaluating request-distribution schemes for saving energy in server clusters. In *Proc. ISPASS*, 2003.

[22] Karthick Rajamani, Heather Hanson, Juan Rubio, Soraya Ghiasi, and Freeman L. Rawson III. Application-aware power management. In *IISWC*, pages 39–48, 2006.

[23] Parthasarathy Ranganathan, Phil Leech, David Irwin, and Jeffrey Chase. Ensemble-level power management for dense blade servers. In *Proc. of International Symposium on Computer Architecture*, 2006.

[24] About IDEAS Relative Performance Estimate 2 (RPE2). http://www.ideasinternational.com/performance/.

[25] Cosmin Rusu, Alexandre Ferreira, Claudio Scordino, and Aaron Watson. Energy-efficient real-time heterogeneous server clusters. In *Proceedings of RTAS*, 2006.

[26] A. Verma, P. Ahuja, and A. Neogi. pmapper: Power and migration cost aware application placement in virtualized servers. In *Proc. of ACM/IFIP/Usenix Middleware*, 2008.

[27] A. Verma, P. Ahuja, and A. Neogi. Power-aware dynamic placement of hpc applications. In *Proc. of ACM ICS*, 2008.