Workload-Driven VM Consolidation in Cloud Data Center

Hao Lin*, Xin Qi†, Shuo Yang‡, Samuel P. Midkiff*

*Purdue University, West Lafayette, IN
Email: {haolin, smidkiff}@purdue.edu
†Amazon.com Email: qixin@amazon.com
‡Huawei R&D Center, Santa Clara, CA
Email: shuo.yang@huawei.com

Abstract—Virtual machines hosted in virtualized data centers are important providers of computational resources in the era of cloud computing. Efficient scheduling of data centers’ virtual machines can reduce the number of physical servers needed to host the virtual machines and, in turn, reduce the energy and other capital costs for maintaining the virtualized data center. In this paper, we propose an innovative approach to achieve efficient pro-active VM scheduling. Our approach uses a multi-capacity bin packing technique that efficiently places VMs onto physical servers. We use time-series analysis techniques to extract not only low frequency information about future VM workloads but also high frequency information for VM workload correlations. We show that the proposed algorithms mathematically guarantee the VM scheduling meets the Service Level Objectives (SLO) and, moreover, guarantee statistically that the desired success probability of the SLO is met.

Evaluation of our technique on production (real) workloads shows that our approach reduces by up to 15% the number of physical machines. We also see improvements of up to 18% for production workloads in machine utilization.

Keywords—Cloud computing; VM consolidation and provisioning; Scheduling

I. INTRODUCTION

Cloud computing has become a major computing infrastructure utility, with public and private clouds gaining in popularity. Public providers such as Amazon, Microsoft, Google, IBM and Oracle are expanding their cloud operations, building new cloud data centers and their own private clouds for resource consolidation and IT automation. Virtualization is the key technology enabling the benefits of cloud computing – scalability, Service Level Objectives (SLO) based service [27], high availability and failure recovery. Data center nodes are virtualized by hypervisor technologies and these same technologies allow nodes to be dynamically placed on physical servers to meet a specific SLO. Multiple virtual machines (VMs) can be placed on a single physical machine to allow flexible partitioning of resources. Efficiently placing VMs onto physical servers translates into a reduction in two major data center costs: the capital expense (CAPEX) of server procurement and the operation cost (OPEX) resulting from providing power to run and cool the physical machines. For example, power consumption is a key OPEX factor in modern data center and enterprise environments and the cost of the electricity consumed for powering the data centers in the U.S. was estimated to exceed $7B per year since 2011 [19]. Being able to consolidate virtual machines onto fewer physical servers enables various saving opportunities such as turning-off servers and other more sophisticated power management techniques [10].

There have been previous efforts to solve this placement problem. PRESS [13] reported that the prediction-based pro-active resource scheduling showed positive results for Google workloads. The work of [5], [11], [8] proposed to estimate historical workloads as a single static value (the static workload) to allow VM placement to be abstracted and solved as a traditional bin packing problem. The physical servers are bins whose sizes are the capacities of a computing resource such as CPU, memory usage, disk I/O or network bandwidth. Summarizing historical workloads with a single value can lose too much information [20]. While Bobroff et al. [6] model VM workloads as a stochastic process they do not model the sum of VM workloads, e.g. the total workload being executed by some physical server, as a stochastic process. Meng et al. [20] proposed a correlation-aware VM selection technique that forms pairs of negatively correlated VMs, and placed each negatively correlated pair on a physical server. An extreme case of this is when one VM is CPU bound and another is I/O bound. While this takes into account the relationship between two VMs, the relationship between the pair of VMs’ and a physical server’s current workload is not taken into account and limits assignments of VMs to servers to groups of two. This is done to minimize the risk of violating an SLO but can lead to underutilization of machines.

Our approach, described in this paper, overcomes these restrictions by abstracting VM placement as a generalized bin packing problem. We represent both the correlation of a VM’s workload with the aggregate workload of one or more VMs on a server and the predictive workloads of individual VMs or groups of VMs on a server as a single number that is used to perform the bin packing. Moreover, while previous work estimates the static workload using only SLO constraints, we try to give equal importance to the SLO success probability and how this probability is guaranteed by estimations of
the static workloads. We explain and exploit the trade-offs between the VM sizes and the correlation relationships among VMs on a server. The core algorithm is described in the context of CPU usage, and further extended to consider additional resource constraints corresponding to the disk I/O rate and so forth.

The technique is used within a framework, shown in Figure 1, with three crucial components: (1) a technique that dynamically predicts future workloads based on current and prior execution of the VM; (2) a technique that accounts for correlation and covariance; and (3) a VM placement algorithm that uses actual or predicted workloads and correlations between workloads. In this paper we focus on new techniques for (2) and (3) and use the pre-existing SARIMA techniques of [9] for (1).

To summarize, this paper makes the following contributions:

1) A general placement strategy that effectively schedules VMs onto physical servers using both workload size and correlation relationships, allowing our approach to be used to facilitate either a pro-active migration or hybrid strategy in a cloud infrastructure instead of being forced to use a reactive migration strategy.

2) A placement algorithm that takes into account the prior SLO violations and the probability of SLO success on future workloads. We give
   a) A statistical analysis and a proof of why our approach meets these conditions.
   b) A placement strategy that uses the residual correlation to reduce the variance in estimating the joint workloads on a physical machine. The strategy is also applied to schedule multiple resource constraints.
   c) Self-adaptive methods to tune the resource limit parameter used by our approach.

3) A quantitative evaluation using both synthetic and production workloads (from a data center), that shows a maximum improvement of up to 20% and an average improvement of 15% in both the number of servers needed and server utilization, respectively.

The paper is organized as follows. Section II provides background information about SLO and SLO success. An overview of our strategy and its workload modeling are provided in Section III. How our approach finds the correlations among servers and VMs is covered in Section IV. Section V describes our placement algorithm in detail. Evaluations and experimental results are shown in Section VI. Section VII discusses the related work. Sections VIII gives our conclusions and describe future work.

II. BACKGROUND OF SLO AND SLO SUCCESS PROBABILITY

In this section, we give a brief background on SLOs [27], SLO success probability [14] and the formulation of our problem based on the SLO and its success probability. Intuitively an SLO is an objective or agreement between the service provider and the consumer, and it can be described as a threshold of the proportion of time that physical resources are over-committed. For example, Figure 2 shows the workload of a server, with its capacity shown by the horizontal dashed line. We see that the server’s workload exceeds its capacity during the periods \([t_1, t_2]\) and \([t_3, t_4]\), thus the overflow time = \((t_2 - t_1) + (t_4 - t_3)\). If the total time is \(L\), the fraction of time the workload exceeds the server capacity (the overflow fraction) is \(\frac{\text{overflow time}}{L}\). The SLO provides an upper bound \(\alpha\) on the overflow fraction.

More formally, let there be \(n\) virtual machines \(VM_1, VM_2, \ldots, VM_n\). The workload of the \(i\)-th VM, \(VM_i\), is represented by a time series \(x_i(t)\) which takes value \(x_i(t)\) at time \(t\). The full set of VMs with their workloads \(W = \{x_1(t), x_2(t), \ldots, x_n(t)\}\) are partitioned into several physical servers \(SV_1, SV_2, \ldots, SV_m\) with resource limits \(c_j \in [0, 1], j = 1, 2, \ldots, m\). Thus, for example, \(c_j = 1\) means 100% of the physical resource can be used.

To measure the overflow time of a server, the following function gives the health status of a server \(SV_j\) at time \(t\), i.e. whether or not its capacity is exceeded:

\[
\chi(t; c) := \begin{cases} 
1 & \text{if } \sum_{i: VM_i \in SV_j} x_i(t) > c_j \\
0 & \text{else }
\end{cases}
\]

(1)

where \(\chi\) is the characteristic function that returns 0 if the total workload on the server \(SV_j\) at time \(t\) is less than or equal to
the capacity of the server, and 1 otherwise. Using \( \chi(t; c) \), the overflow time can be further expressed as:

\[
\text{overflow time} = \int_0^L \chi(t; c) \, dt. \tag{2}
\]

The integration simply sums up all the time points in the interval \([0, L]\) where an overflow occurs.

When a reactive VM placement technique [23], [26] is used, i.e. the VM migration is based on past events, the SLO can be easily checked by summarizing the servers’ workloads’ histories and comparing them with the SLO specified threshold \( \alpha \in [0, 1] \). We, however, use a pro-active VM placement strategy [23], i.e., the VM placement is based on predicted future events. The future workload of a server is modeled as a stochastic process and the ability to fulfill an SLO is modeled as a random variable. The SLO success probability [14], [3], [28] is defined as the probability that the overload fraction does not exceed \( \alpha \) over some long time interval \([0, L]\), i.e.,

\[
P_{\text{suc}} = P \left\{ \frac{\text{overflow time}}{L} \leq \alpha \right\} \tag{3}
\]

Let \( p \) be the lower bound of the SLO success probability that is defined by the SLO. A small value of \( p \) means a higher tolerance of SLO violations and a higher value of \( p \) requires more resources, e.g., more servers, and incurs higher costs. The choice of \( p \) depends on many factors and is beyond the scope of this paper. Given a \( p \), the goal of our algorithm is to place all VMs, each of which may have a different workload, on a finite number of servers while minimizing the number of servers and maintaining the probability of success above \( p \), with the SLO success probability constraint for each server given as

\[
P \left\{ \frac{\text{overflow time}}{L} \leq \alpha \right\} \geq p \tag{4}
\]

For simplicity of discussion, in the remainder of the paper we assume all servers have the same capacity \( c \), the same SLO threshold \( \alpha \) and the same SLO success probability lower bound limit \( p \). This is not, however, a limitation of our method.

III. WORKLOAD MODELING

In this section we present our techniques for three key steps shown in Figure 1. First, we describe how we pre-process, model and forecast the VM workloads using time series models. Next we introduce static workload estimation and finish with an explanation of how we model the physical server’s workload as a combination of multiple VMs.

A. Workload Modeling

1) Workload Characteristics and Preprocessing: The workloads in our study are time series whose values are the resource usage at specific times, sampled at a fixed time interval, and typically normalized onto a number from 0 to 1. To use historical time series in prediction, three important characteristics are considered: (1) the length of time represented by each point in the series, i.e. the time scale; (2) the number of points to predict; and (3) the total length of time predicted by the series, i.e. the prediction horizon, which is simply the product of the time scale and the number of points to predict. While a shorter time scale provides finer-grained data, it also requires more points than a longer time scale to produce the same prediction horizon. Unfortunately, predictions further into the future are less accurate than predictions closer to the actual measured data, i.e. the predicted variance increases as the point gets further from the observed data. The best choice for the number of points in the time horizon is a compromise between the data center migration policy and time series prediction accuracy [9].

Usually we need longer scale than raw data that is computationally intractable. We smooth the time series by dividing it into intervals and sampling within each interval. A point in the new time series represents an interval in the original. The time interval of the new time series is the \((1-\alpha)\) quantile such that \((1 - \alpha) \times 100\%\) of the original series’ points in the interval are below the selected value. Each VM is represented by two time series: one from real-time monitoring, denoted \(x_{\text{real}}(t_j), t \in [0, T]\), and one after smoothing with the desired time scale, denoted \(x(t_j), j \in [1, K]\). The smoothed workload is used to predict the future workloads \(x(t_j), j \in [K + 1, K + N]\). We note that \(x(t_j), j \in [1, K]\) is an observation of a random variable (RV), whereas the future workload \(x(t_j), j \in [K + 1, K + N]\) is an RV.

2) Dynamic Workload Modeling: Many time series techniques can be used to forecast future workloads using the smoothed workloads. Typically, a time series model is manually selected for each VM according to its workload pattern [9], [12], which can be challenging with a large number of VMs. New ensemble techniques automatically select a model that performs well [7], [25]. Herbst et al. [15] had a survey of existing prediction methods and proposed an self-adaptive way to choose the prediction methods based on workload category. Our placement strategy neither proposes nor relies on any particular techniques.

Let the time series data be decomposed into a sum of deterministic components according to time series analysis [9], i.e. a trend, cyclic and seasonal, and an irregular error component. We denote the sum of all deterministic components as \(\hat{x}(t_j)\) and the irregular component as the residual \(e(t_j)\). Each time series can be then expressed as \(x(t_j) = \hat{x}(t_j) + e(t_j)\). During the modeling process, we monitor the behavior of each residual \(e(t_j)\), which is also a time series, and predict the mean and variance for future workloads \(x(t_j), j \in [K + 1, K + N]\). For convenience, we will assume each future workload follows a normal distribution, and then based on the predicted information we can construct a confidence interval of the workload at each time \(t\).

B. Static Workload Estimation

In [20] a technique is given to find the static workload, i.e. a single “size” that can be used to bin pack a historical VM
workload as a time series. The static workload is the minimum \( w \) such that the inequality (discretized version):

\[
\frac{1}{T} \sum_{t=1}^{T} \chi_{x \text{real}}(t; w) \leq \alpha
\]

holds, where \( \chi \) and \( \alpha \) are as defined earlier. Essentially we find the smallest server capacity needed such that the VM is properly provisioned all but \((1 - \alpha) \times 100\%\) of the time, as required by the SLO. Therefore, finding the optimal \( w \) based on \( x^{real}(t), t \in [0, T] \) is equivalent to calculating the \( 1 - \alpha \) quantile of the real observations. The time series in Figure 4(a) represents the workloads collected from a server, and its static workload is its \( 1 - \alpha = 0.95 \) quantile (the minimum SLO violation rate \( \alpha = 5\% \)).

For the smoothed data \( x(t_j), j \in [1, K], \) we can derive a static workload by taking its maximum \( w = \max_{j=1}^{K} \{x(t_j)\} \).

We prove in the Appendix that \( w \) satisfies inequality 5 for the real historical data \( x^{real}(t), t \in [0, T] \). Figure 4(b) shows the smoothed time series from 4(a) and its workload is the maximum value of all smoothed points.

For our technique we must find a VM’s static workload for a future time period instead of history, which is more complicated. The static workload for the time series \( x(t_j), j \in [K+1, K+N] \) is the minimum \( w \) such that

\[
P \left( \frac{1}{N} \sum_{j=K+1}^{K+N} \chi_{x(t_j; w)} \leq \alpha \right) \geq p
\]

holds and \( \chi(t; w) \) is as defined in Equation 1 and \( p \) and \( \alpha \) are as before, specified by the SLO. Equation 6 holds when \( w \) is large enough that a VM provisioned with \( w \) resources and a workload of \( x(t_j), j \in [K+1, K+N] \) is guaranteed to fulfill the SLO with a probability greater than \( p \).

Equation 6 often does not have an explicit form and will need to be treated with an (approximate) surrogate model for the stochastic process (time series) [14]. Suppose \( \hat{x}(t_j), j \in [K+1, K+N] \) is a deterministic process (a series of fixed numbers that bounds \( x(t_j), j \in [K+1, K+N] \)) from above with an overall probability \( p \), i.e., \( \hat{x}(t_j) \) defines the upper side of a confidence band of \( x(t_j) \). If we take \( w \) as the static workload for the deterministic workload process \( \hat{x}(t_j), j \in [K+1, K+N], \) that is \( w = \max_{j=K+1}^{K+N} \{\hat{x}(t_j)\} \). Then we claim \( w \) is a solution of the inequality in Equation 6 but may not be the optimal one. A simple deductive proof is given in the Appendix.

To summarize, Figure 3 shows the work-flow to estimate static and dynamic workloads for a VM on a future interval. The last part in Figure 4(c) represents the predicted workloads, and its static workload is the maximum of the bound that controls it with a \( p = 0.95 \) confidence.

**C. Server Workload Estimation**

In VM placement we need to check if the static workload of a server, after placing some VM onto it, will exceed the server capacity \( c \) and unacceptably increase the SLO failure probability. One way to estimate the server workload is to directly aggregate server and VM workloads and obtain a new time series. However, the computational cost of repeatedly modeling the aggregated time series whenever a new VM is placed on a server is very high. Instead, individual VMs and their residuals’ correlations are used to update the server workload predicted interval of \( [t_{K+1}, t_{K+N}] \), where \( N \) is the number of points in the predicted interval. Suppose at time \( t \) in \( [t_{K+1}, t_{K+N}] \), VM\(_i\)’s and VM\(_j\)’s workloads, \( x_i(t) \) and \( x_j(t) \), follow two normal distributions with parameters \((\mu_m(t), \sigma_m(t))\) and \((\mu_n(t), \sigma_n(t))\), respectively. The correlation \( \rho_{m,n} \) between the two monitored residuals \( e_k(t) \) and \( e_j(t) \) can be used to estimate the correlation between \( x_k(t) \) and \( x_j(t) \) and their summation \( s_{new}(t) \) follows a normal distribution with parameter pairs

\[
(\mu_m(t) + \mu_n(t), \sigma_m^2(t) + \sigma_n^2(t) + 2\rho_{m,n}\sigma_m(t)\sigma_n(t)).
\]

A \( p \)-upper bound \( \hat{s}_{new}(t) \) for \( t \in [t_{K+1}, ..., t_{K+N}] \) can then be built. All that remains is to check if the inequality \( \max_{j=1}^{N} \{\hat{s}_{new}(t_{K+j})\} \leq c \), where \( \hat{s}_{new}(t_{K+j}) \) is the \( p \)-quantile of the updated server workload at \( t_{K+j} \).

**IV. FINDING CORRELATIONS**

Both the VM static workload and correlations can play important roles in resource utilization. When a server’s workload is not stable, i.e., it has a large variance, and the candidate VMs to be placed have similar sizes, it is better to place a VM on the server whose workload is negatively correlated with the VM’s workload to help “stabilize” the server’s workload. Conversely, when the “peaks” of these VMs overlap there is a higher chance of violating the SLO and degrading the application’s performance. There are many kinds of correlation metrics. For example, “peak ratio” [5] measures the net increase of the peak workload to a server after adding a new VM and it works well in our algorithm if the static workload is substituted for the peak. Another correlation metric, which is also used in our experiments, is the cross correlation at lag 0 Pearson correlation [9], which is adopted by our strategy. Instead of
the smoothed data, the original collected data should be used to compute correlations since cross correlation is a discrete estimation of an integration and the accuracy is always better with more data points. Forecast data may not contain enough information for correlation because of the small number of data points and larger variance in those points.

A. Updating Correlations Among Servers and VMs

If a server $SV$ hosts only one VM, $VM_j$, the server’s workload $SV_{old}(t)$, is the same as $VM_j$’s workload, i.e., $x_j(t)$ and the correlations between the server and other VM workloads are the same as the correlations between $x_j(t)$ and other VM workloads. Once another VM, $VM_k$, is placed onto the server the workload dynamics of the server change from $SV_{old}(t) = x_j(t)$ to $SV_{new}(t) = x_j(t) + x_k(t)$. The correlations among the server and all non-scheduled VMs also change. It is straightforward to update the correlations by recalculating the cross correlations. But when the number of VMs is large, the computation can be very time consuming. Instead, since all the correlation and covariance coefficients are stored in the calculated matrices, we can apply the following linear properties:

\[
cov(x_j + x_k, x_l) = \cov(x_j, x_l) + \cov(x_k, x_l),
\]

\[
\var(x_j + x_k) = \var(x_j) + \var(x_k) + 2\cov(x_j, x_k)
\]

where $x_l$ is the dynamic workload of another VM $VM_l$, $l \neq j$ and $l \neq k$. Let the calculated correlations among all VMs be stored in a matrix $C$ where element $C[j, k]$ holds the correlation between $VM_j$ and $VM_k$. Then the updated correlation between server and $VM_l$ can be calculated as:

\[
cor(SV_{new}, x_l) = \frac{C[l, j] + C[k, l]}{\sqrt{(C[j, j] + C[k, k] + 2C[j, k])C[l, l]}}
\]

V. VM PLACEMENT

A VM workload is represented as a time series and the problem of VM placement becomes a stochastic bin packing problem where the object size follows a distribution forecast by the time series. Since the problem is NP-hard, two widely used heuristic approaches are first-fit decreasing (FFD) [29] and least-loaded (LL) [4]. Most previous VM placement algorithms use these two approaches based on a static workload, however they may map highly correlated VMs to the same server.

Our placement algorithm is summarized in Figure 5. The algorithm walks along the VMs in the non-increasing order of their priorities, and picks the first several VMs as candidates. We propose a bin packing heuristic by defining a simple pri-
ority model that balances the two factors: (1) maximizing the static workload and (2) maximizing the “stability” workload. One way to measure “stability” is the future variance of the server workload, which can be estimated by historical metrics. This method requires more computation on summarizing data points along time (in complexity of $O(n^2)$ in [16]). We instead use correlations when we consider the priorities of which VM to consolidate. The priority for some VM $i$ with $M$ resource constraint is written as

$$
priority_{SV,i} = \sum_{m=1}^{M} (1 - \rho_{SV,i}^m) \times \left(\frac{X_i^m}{cap_{SV}^m}\right)$$

where $SV$ represents the current destination server and $X_i^m$ is its static workload for resource $m$, as computed in Section III. $cap_{SV}^m$ is the physical capacity for resource $m$. By dividing $X_i$ by this capacity, we normalize the workload size to prevent biased values. When correlation $\rho_{SV,i}$ is, or is close to, 0, the priority depends almost entirely on workload size, which means the algorithm behaves similarly to FFD. When correlation $\rho_{SV,i}$ is close to either 1 or -1, scheduling is based almost entirely on correlations. A positive correlation reduces the priority of the VM, while negative correlation increases the priority. When there is little difference among the correlations of candidate VMs and the server, however, it makes little sense to violate the FFD order. It is worth mentioning that when a server is empty, its correlations with other VMs are 0, and it starts with the largest available VM.

As listed in Figure 5, all remaining VMs are listed in descending order of their $priority_{SV}$ and the scheduler will follow this list to find the first several VMs that the destination server can hold as candidates. We finally place the VM from the candidates list that leads to the maximum predicted server workload size by using Equation 7. In case of multiple resources, the server workload size is simply defined as the product of size in all resource dimensions. After the scheduler places the selected VM onto the server, the server workload, each VM’s correlation with the server and the score is updated, and then the list of VMs to be scheduled is re-sorted by their priorities. The process will repeat until no VM on the list will fit on the server, at which time a new server is needed. VM placement ends when all VMs have been placed.

Compared to traditional stochastic bin packing algorithms with high computational complexity, our method runs efficiently. We use only around 10 minutes, on average, to schedule 500 VMs in a two-day period. Both time series prediction and the correlation computation can be done in parallel. We have deployed our scheduling algorithm on the RABID [18] distributed parallel R framework on an 8 node cluster and reduced the scheduling time to 120 seconds, on average. Should the number of VMs grow so large that scheduling becomes a bottleneck we can do two things. First, we can adjust the time scale, allowing the migration cost to be amortized over a longer execution time before rescheduling. Second, we can partition the VMs in the data center and effectively have a smaller scheduling problems.

A. Automatic Resource Limit Selection

The consolidation algorithm of Figure 5 needs a resource upper bound limit to pack the VMs so that the resulting physical server will not violate the SLO. This parameter is very important and has a great impact on the consolidation result. Usually, it is impossible to know the value of these parameters in advance. We have observed the relationship between the physical server’s SLO violation rate and its resource limits for consolidation, as shown in Figure 6(a), where the SLO violation rate varies over CPU resource limit. We consider a violation rate of less than $\alpha = 0.1$ to be an acceptable target [21] (the horizontal line). Nguyen et al. [21] proposed an online resource pressure model that adaptively adjusts the resource allocation to meet an application’s SLO.

We are solving a different problem: SLO is based on each physical server and each server may have a resource model for each resource constraint. The model can vary along the time according to the VM workload on it. We model the pairs of resource limit and SLO violation rate in a simple Logistic Regression model and train it by using Gradient Descent [24]. Unlike the method in [21], which collects a set of pairs of profiling data and fits them against a set of polynomials with different orders, we do not need to store a sequence of historical data since Gradient Descent approach adjusts the model incrementally by each data sample. The red curve in Figure 6(b) is an illustration of a Logistic Regression model fitting the profiling data. In the figure, profiling data is aggregated to calculate the probability of SLO violation rate greater than the target of 0.1. Therefore we can derive a resource limit threshold for future consolidation by setting the target probability to 0.5.

VI. EXPERIMENTAL EVALUATION

Our experiments evaluate the effectiveness of our framework and techniques. We use the results of the Multiplexing approach proposed by Meng et al. in [20] and the commonly used FFD packing algorithm (used in OpenStack [1] open source cloud framework) as baseline numbers to evaluate the above metrics across different scenarios and setups. The FFD packing algorithm sorts the VMs in the decreasing order of their static sizes, and iteratively picks largest VM to place on
the first physical servers on which it will fit. If there is no such server, the VM is placed in a new empty server. No correlation information is involved in such case.

A. Experimental Setup

To evaluate our approach both production and synthetic workloads are collected and used. The production workload data were collected from ITaP [2], Purdue’s computing services organization. This well structured data consists of 225 VMs, with a data point every 30 minutes and spanning approximately three months. The data includes many different resource constraints, and we focus on the “CPU usage” and disk I/O rate information in our experiments. The synthetic dataset contains data for 500 VMs spanning 3 months, with a data point every 10 minutes. This series dataset was generated using the SARIMA model [9] with $p = P = q = Q = 1$ and $d = D = 0$, which are commonly used.

As mentioned in Section III-A, we first smoothed the time series by using the $(1 - \alpha)$ quantile value (See Section III-A1) to represent the interval given by the new time scale. In our experiments, the new time scale is one hour. Therefore we pick two adjacent points as an interval for the production data and six points as an interval for the synthetic data. The new time series with fewer data points is smoother. We assume the seasonal period is one day (24 hours with one data point per hour, i.e. 24 data points) and use 2 weeks of data, i.e. $2 \times 7 \times 24 = 336$ data points to predict the next 2 days (48 points). This allows the cloud operator to migrate VMs every two days. The real workloads for the predicted 2 days are used as ground truth. The process of time series analysis is illustrated in Figure 7.

We use three metrics to evaluate the quality of our placement algorithm: SLO violations, the number of servers and the utilization of the servers. SLO violations are the key metric for customer satisfaction and the number and utilization of servers are key metrics for the cost of hosting the VMs. The utilization of servers is measured by the median utilization, or 50% quantile of utilization of all physical servers. The median utilization should be as high as possible, but typically should not exceed the SLO threshold $\alpha$.

The scheduling procedure is repeatedly performed at two-day intervals. In the experiments, we repeated the scheduling for 50 times on 116 days worth of data, with 100 days being predicted by the experiments. The averaged result of the three metrics over 50 runs of the experiment is presented. In our evaluation, we used 500 VMs for the synthetic workloads and 225 VMs for the real workloads.

Our experimental data does not have short running jobs. Short running jobs or jobs that need to be scheduled between scheduling periods are placed based on the historical training data for the job. If they live to the next scheduling period they will be scheduled like other jobs. The migration time is not relevant to our offline scheduling algorithm and is not considered in the evaluation.

B. Evaluation Results

**Synthetic workload results.** Figure 8 shows the physical server counts and the utilization for the two approaches with their averaged SLO violation rate, for different numbers of VMs (50 to 500) on our synthetic data. Also, to make a fair comparison of utilization and the number of servers needed by the two approaches, we control their SLO violation rate to be as close to each other as possible. We present the server count and utilization result in Figure 9. Our approach uses a smaller number of servers and utilizes them more efficiently. This is because our approach, unlike the FFD approach, takes into account the correlations among VMs and the server workloads. As more VM workloads that are negatively correlated with a server workload are picked for placement on the server the resulting workload tends to be smoother (less variability) and capable of hosting more VMs. With the FFD approach it is more likely that positively correlated workloads are placed together, leading to more peaks that exceed the server capacity. Thus the FFD approach is more likely to underestimate the workloads at peaks and overestimate the overall workload and so, in practice, have more SLO violations and achieves less utilization.

![Image](image-url)
(a) The number of physical servers needed to host a VM using the different methods.
(b) The number of SLO violations.
(c) The median utilization of servers for different number of VMs, where the number of servers is given in (a).

Fig. 8. Placement results for synthetic workloads.

Adaptive resource limit for VM consolidation. Figure 10 shows how VM consolidation can benefit from self-adaptive resource limit selection introduced in V-A. We compare our packing algorithm with self-adaptive resource limit selection (with initial limits of 75% and 90%), and without it, i.e., using fixed resource limits of 75% and 90% respectively. The results show that the resource limit model leads to better utilization while giving an acceptable SLO violation rate (< 0.1). Although the utilization using fixed limit of 0.9 is greater than the one using the adaptive approach, the former gives much higher SLO violation rate that is not acceptable.

Production workload result. The effectiveness of our placement algorithm used with the production workload data is shown in Figure 11. As before, we keep the number of SLO violations similar. Overall, our approach deploys fewer physical servers and has better utilization. Although the benefit of our approach is less than with the synthetic workloads, it is still significant and much better than what the FFD and multiplexing placements give.

A key reason our approach is less effective with the production than the synthetic VM workload is that they are less negatively correlated than the synthetic workloads, as seen in Figure 12. This reduces the effectiveness of our placement algorithm since more positively correlated workloads must be placed together (there is no negatively correlated workload to use) which leads to more variable joint workloads with higher peaks. Another reason might be that the production workloads are, overall, not as good a fit for the SARIMA(1,0,1,1,0,1) model so the accuracy of prediction is worse for them.

We next explore the effect of prediction on our technique. Figure 13 shows that if we have perfect knowledge of future VM demands our approach uses fewer physical machines and achieves better utilization compared with VM placement using the imperfectly predicted results of Figure 11. From these results we can conclude that our technique will benefit
A. Time Series Model Selection

The ARIMA model [9] can be applied to most time series, but its predictive ability depends on the characteristics of the time series and much effort is needed to find a suitable model for observed data. In a data center, however, the number of VMs that need to be analyzed can be extremely large, and it is expensive and impractical to independently forecast all VM workloads. Bobroff et al. [6] introduce a method to schedule VMs dynamically under the constraints imposed by SLOs. This work adopts an adaptive time series model when estimating the workload of servers and VMs. Stokely, et al. [25] introduces a platform and forecasting methodology that implements time series forecasting robustly with high accuracy. Herbst [15] summarized a survey of a broad range of prediction methods proposed an approach to self-adaptively select workload forecasting methods. Hguyen et al. [21] used a Wavelets to forecast medium-term workloads. Our strategy builds on seasonal ARIMA models used to predict the future time series and can take advantage of ongoing efforts to more accurately predict future VM workloads.

B. Resource Provisioning

Hguyen et al. [21] introduced a resource pressure model to control the SLO violation rate. Schwarzkopf et al. [22] proposed a new resource offering framework to increase the efficiency of parallel job scheduling in a large scale cluster. In contrast, our methodology considers the correlations among all the VMs and servers, and places them onto servers based on both their correlations and sizes. We also learn the SLO violation rate in a Logistic Regression model to predict a better resource limit.

C. Dynamic Placement of VMs

Dynamic VM Placement has been studied ever since dynamic VM migration became available to data centers. The bin packing problem, or stochastic bin packing problem, is often used to abstract dynamic VM consolidation. The Multi-Capacity bin packing algorithm is applied to scheduling jobs in [17]. Hwang et al. [16] used a hierarchical strategy to increase the efficiency of the placement algorithm. Meng et al. [20] proposed an approach for resource provisioning and VM consolidation by forming two negatively correlated VMs into a joint VMs. This helps achieve a more compact consolidation, reduces the number of servers used and improves resource utilization. Our work models server workloads that are based on the dynamic workloads of multiple VMs that have been placed on them, and their residual covariance. We make fully use of dynamic workloads of all VMs rather than only their static estimations.

VIII. Conclusions and Future Work

Our technique shows that a flexible use of correlation data along with predicted VM workload data provides uniformly better performance on both synthetic and production data relative to FFD and Multiplexing scheduling approaches. It does this while almost always having fewer SLO violations.

---

**Figure 13.** Placement results on production workloads with oracle knowledge of prediction.

**Figure 14.** VM consolidation by using two resource constraints (CPU usage and disk I/O rate), with different resource (disk rate) limits.

---

<table>
<thead>
<tr>
<th>Approach</th>
<th>Number of servers used (normalized)</th>
<th>Median of utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours</td>
<td><img src="image1.png" alt="Diagram" /></td>
<td><img src="image2.png" alt="Diagram" /></td>
</tr>
<tr>
<td>CPU only</td>
<td><img src="image3.png" alt="Diagram" /></td>
<td><img src="image4.png" alt="Diagram" /></td>
</tr>
<tr>
<td>FFD</td>
<td><img src="image5.png" alt="Diagram" /></td>
<td><img src="image6.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

(a) The number of physical servers used.
(b) Median Utilization

<table>
<thead>
<tr>
<th>Approach</th>
<th>Number of servers used (normalized)</th>
<th>Median of utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours</td>
<td><img src="image7.png" alt="Diagram" /></td>
<td><img src="image8.png" alt="Diagram" /></td>
</tr>
<tr>
<td>CPU only</td>
<td><img src="image9.png" alt="Diagram" /></td>
<td><img src="image10.png" alt="Diagram" /></td>
</tr>
<tr>
<td>Disk rate only</td>
<td><img src="image11.png" alt="Diagram" /></td>
<td><img src="image12.png" alt="Diagram" /></td>
</tr>
<tr>
<td>Multiplexing</td>
<td><img src="image13.png" alt="Diagram" /></td>
<td><img src="image14.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

(a) The number of physical servers used (disk rate limit 500MB/s).
(b) SLO violation rates and probability of SLO vio rate > 0.1 with two setups.

---

**VII. Related Work**

We now discuss related work not discussed earlier.
than the other approaches. Our approach is relatively agnostic to prediction strategies and with better predictions performs better in both absolute terms and relative to the other approaches we compare against.

Our future work will proceed in two directions. Our current work clearly benefits from using both correlation and workload data of multiple resources to place VMs. We plan to investigate how workloads clearly benefits from using both correlation and workload data of multiple resources to place VMs. We plan to investigate the scalability of the algorithm when the number of resources increases.

APPENDIX
Proof of Static Estimation of Historical Workloads Satisfying SLO

Let all $N$ points be divided into $K$ intervals. Let $Q_k$ be the $\alpha$-quantile on the $k$th interval, for $k \in [1, K]$. Suppose $Q = \max_{k=1}^K Q_k$. We now prove that $Q$ is larger than the $\alpha$-quantile of all the $N$ points. Here is the proof:

On the $k$–th interval, there are exactly $N_k = \alpha \times N$ * points smaller than $Q_k$. Since $Q \geq Q_k$, there are more than $N_k = \alpha \times N$ * points smaller than $M$. Summarizing all these points on all intervals, there are more than $K \times \sum_{j=1}^{N_j} = \alpha = N \times \alpha$ points smaller than $Q$. Thus $Q$ is larger than the $\alpha$-quantile of all these $N$ points.

Proof of Static Estimation of Predicted Workloads Satisfying SLO Success Probability

We denote the event $B$ as the event that $x_i(t_j) \leq \hat{x}_i(t_j)$ for all $j \in [1, K]$, then $B$ happens with probability $p$, and then we have the SLO success probability:

$$P\left(\frac{1}{N} \sum_{j=1}^{N} x_i(t_j) \leq \alpha \right) = P\left(\frac{1}{N} \sum_{j=1}^{N} x_i(t_j) \leq \alpha | B \right) P(B) + P\left(\frac{1}{N} \sum_{j=1}^{N} x_i(t_j) \leq \alpha | B^c \right) P(B^c) \geq P\left(\frac{1}{N} \sum_{j=1}^{N} x_i(t_j) \leq \alpha | B \right) P(B) = \frac{k}{p} = p$$

Therefore, the SLO success probability is above the lower bound $p$.

REFERENCES