iOverbook: Managing Cloud-based Soft Real-time Applications in a Resource-Overbooked Data Center

Faruk Caglar and Aniruddha Gokhale
Department of Electrical Engineering and Computer Science
Vanderbilt University, Nashville, TN 37235, USA
Email: {faruk.caglar, a.gokhale}@vanderbilt.edu

Abstract*—Cloud service providers (CSPs) often overbook their resources with user applications despite maintaining service-level agreements with their customers. Overbooking is attractive to CSPs because it helps to reduce power consumption in the data center by packing more user jobs in less number of resources while improving their profits. Overbooking becomes feasible because user applications tend to overestimate their resource requirements, utilizing only a fraction of the allocated resources. Arbitrary resource overbooking ratios, however, may be detrimental to soft real-time applications, such as airline reservations or Netflix video streaming, which are increasingly hosted in the cloud. At the same time, the changing dynamics of the cloud preclude an offline determination of overbooking ratios. To address these concerns, this paper presents iOverbook, which uses a machine learning approach to make systematic and online determination of overbooking ratios such that the quality of service needs of soft real-time systems can be met while still benefiting from overbooking. Specifically, iOverbook utilizes historic data of tasks and host machines in the cloud to extract their resource usage pattern and predict future resource usage along with the expected mean performance of host machines. To evaluate our approach, we have used a large usage trace made available by Google of one of its production data centers. In the context of the traces, our experiments show that iOverbook can help CSPs improve their resource utilization by an average of 33% and save 65% power in the data center.

Keywords*—*resource overbooking, cloud computing, soft real-time performance.

I. INTRODUCTION

Resource overbooking [1], [2], [3], [4] is a common practice adopted by Cloud Service Providers (CSPs) to increase resource utilization in the servers of a data center and reducing the number of physical servers that are powered on. The outcome for the CSPs is a profitable business model and lower energy bills due to lesser number of servers being used. Resource overbooking entails committing more resources, such as CPU and memory, than are actually available on the physical host machines to the applications – in our case more virtual machines (VMs) that host user applications – that are packed onto physical servers than can actually fit. The resource overbooking technique is a feasible option for CSPs to adopt because cloud users often tend to overestimate the resource requirements for their applications; in reality they use just a fraction of the allocated resources.

This claim can be validated by observing the dynamics of a production data center whose usage trace is made available by Google Inc [5]. Figure 1 illustrates a snapshot of a host machine in the data center over a time interval depicting the actual CPU usage, host machine CPU capacity, and the requested CPU capacity (shown as the allocation). As seen from the figure, the actual CPU usage of a task is much lower than the allocated amount of CPU clearly indicating that users overestimate their resource needs. Without overbooking, this situation yields very low resource utilizations in data centers, which is detrimental to the CSP as well as to the environment. It is estimated that in Google's data centers, the resource utilization is maintained between 40-60% whereas this percentage is around 7-25% in other data centers [6].

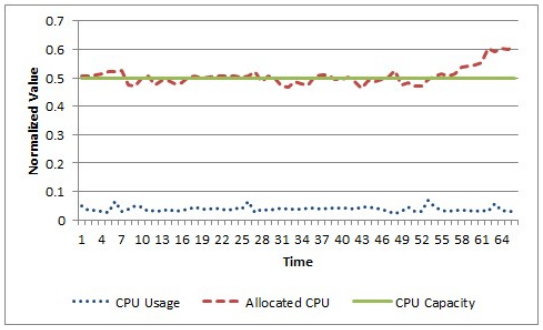


Fig. 1: Allocated resources versus Actual CPU Usage of a Host Machine in the Google Cluster Trace Data

To enable overbooking the servers of data centers with virtual machines, most well-known hypervisors, such as Xen [7], KVM [8], and VMware ESX Server [9] support a configuration option for resource overbooking ratios. Even the cloud infrastructure software that manages the cloud platforms, such as OpenNebula [10], OpenStack [11], and Eucalyptus [12] allow overbooking. For example, OpenStack has a feature for allowing up to 16:1 and 1.5:1 CPU and memory overbooking ratios, respectively. A 16:1 CPU overbooking ratio means that one physical CPU (pCPU) core can be overbooked by up to sixteen virtual CPU (vCPU) cores. Techniques, such as transparent page sharing, memory ballooning, memory compression, and swapping to disk are some of the methods that hypervisors utilize to make memory overbooking possible [13].

The resource overbooking approach adopted by CSPs tends to be suitable for enterprise applications where most jobs are of the batch processing type for whom throughput is more important. However, as more applications with soft real-time requirements, such as airline reservations, Netflix video streaming, real-time stream processing, and massive open online courses, get hosted on the cloud, resource overbooking may cause significant jitter giving rise to unpredictable performance, which is not acceptable for this class of applications. Moreover, in accordance with the Service Level Agreements (SLA) between the CSP and the customer, service providers have to assure certain performance requirements, such as response time and availability, which is hard to assure without a systematic approach to resource overbooking.

To understand the spectrum of overbooking, we observe that at one end of the spectrum exists lower overbooking ratios, which can result in high satisfaction for cloud users, but can be detrimental to CSPs who would not be effectively and economically utilizing their resources. At the other end of the spectrum, higher and arbitrary overbooking ratios might result in CSPs utilizing their resources effectively thereby saving on energy costs and making their services more profitable, but the soft real-time systems hosted in the cloud will suffer from not receiving their desired quality of service (QoS) due to the high resource contention and interference caused by overbooking [14], [15], [16], [17].

The key challenge lies in systematically identifying effective overbooking ratios which will make the right tradeoffs in meeting these conflicting objectives. Secondly, since cloud data centers are made up of heterogeneous machines, a single overbooking ratio may not be effective. Finally, since the cloud environment is highly dynamic, an offline computation of overbooking ratios is not applicable. In the current state of the art, the common overbooking strategy being applied in data centers is to analyze the workloads of the VMs by system administrators through resource monitoring applications or by using optimum overbooking ratios for CPU, memory, and disk established by earlier studies [4]. However, none of these contemporary approaches might be appropriate for all the CSPs because of the workload heterogeneity and the risks of errors due to human involvement. These limitations call for an online and autonomous solution.

To address these limitations, this paper presents *iOverbook*, which provides an autonomous, online and intelligent performance-aware, overbooking strategy for heterogeneous and virtualized environments hosting soft real-time applications. iOverbook autonomously forecasts asymmetric overbooking ratios, *i.e.*, an overbooking ratio per host machine in the data center, by carefully considering the historic resource usage of the applications and not jeopardizing the performance requirements of the soft real-time systems. Specifically, it predicts the mean CPU and memory usage of the physical host machine within the next specified time interval – in our case an hour – by utilizing historic resource usage patterns along with some other features, such as CPU capacity, memory capacity, and requests for CPU and memory, and employing machine learning algorithms. Overbooking ratios for the next hour for CPU and memory are then computed based on a mathematical formula. iOverbook continues to adjust these ratios till they converge to a precise value, which will assure certain QoS levels of the hosted applications. The prediction window then slides to the next hour. Resource overbooking can cause performance interference and affect VM placement, which is the focus of our ongoing investigations [18].

The contributions of this paper are summarized below:

- It analyzes a large Google data center trace and reports on the time-based and machine-based overbooking ratios, which are illustrative of overbooking ratios used in real production servers (Section III).
- It presents an intelligent and autonomous, performance-aware overbooking strategy for each host machine in heterogeneous virtualized environments that satisfies soft real-time application QoS (Section IV).
- Through experimental validations, it analyzes how resource utilization levels can be improved and power consumption reduced in the cloud data centers by utilizing iOverbook (Section V).

The rest of this paper is organized as follows: Section II deals with relevant related work comparing it with our contributions; Section III analyzes an existing usage trace from a Google's production data center and illustrates how Google is using overbooking ratios on different host machines and the resulting application performance and resource utilizations; Section IV presents iOverbook in detail; Section V evaluates the effectiveness of iOverbook; and finally Section VI presents concluding remarks alluding to future work.

II. Related Work

This section compares related work synergistic to our work. Predicting future resource usage of VMs based on historic data is a significant aspect of resource overbooking. Synergistic to our work, machine learning-based approaches are widely used for forecasting the future in different domains. For example, in the energy domain, [19], [20] predict future usage of electrical consumption and hot water production, respectively. In [19], different machine learning approaches are evaluated in the context of energy modeling. In the grid and cloud domain, [21] predicts future workload and [1] predicts resource utilization patterns.

Moreno et al. [1] presented a neural network-based overallocation strategy to increase the energy efficiency in data centers and satisfy performance requirements of real-time applications. The mechanism presented in that paper predicts the customer's resource utilization based on historic data and computes the amount of resources that will be allocated to a VM by employing cost-benefit analysis and an overallocation algorithm. The work in that paper differs from our work in that it does not provide per-host resource overbooking ratios as we do. However, the forecasting of resource consumption has similarity to our work.

Tomas and Tordsson [22] proposed a cloud computing management framework comprising an admission control for horizontal elasticity (*i.e.*, whether to accept more VMs) and scheduling techniques for vertical elasticity (*e.g.*, CPU, memory, and bandwidth). Additionally, they assumed that no SLA violations occur if the used capacity is within the bounds of the physical host machine. This might not be always the case due to the resource contention and interference effects. Our work differs from this work in two ways. First, we provide asymmetric overbooking ratios for a specified timing window (*e.g.*, next one hour). Second, we take many parameters, such as number of VMs on the host machine and mean CPU usage, into account to precisely predict the performance when overbooked. This significantly alleviates the performance interference problem.

In [2], the authors present an approach to determine overbooking, define overload mitigation strategies, and investigate the relationship among overload mitigation techniques and SLAs in the cloud. Birkenheuer et al. [23] address the gain of overbooking in the Grid, Cloud and HPC environments, the restrictions in scheduling algorithms, and how strict SLA is affected by overbooking. They also propose a time-based mathematical model of their overbooking strategy, which is derived from production traces of one year duration consisting 400 processors. Although the insights gained from these approaches are useful for our work, these types of time-based approaches are more appropriate for the Grid environments, which are concerned with executing very long running, high performance computing jobs in controlled environments whereas in our work we primarily target cloud environments – notably public clouds where resources are shared.

Our earlier work [24] developed a model predictive algorithm for workload forecasting based on which an autonomous framework for resource autoscaling for the cloud was developed. This work was also based on insights gained from usage traces of the Soccer World Cup of 1998. Although the goals of our previous and current work are performance assurance, the previous work focused on deciding how many resources are needed for a specific application and how to proactively scale them up or down based on prediction of incoming workload. The end objective was to tradeoff performance with the price the customer pays for using cloud resources. In current work, we take a CSP-centric viewpoint where the objective is to pack as many jobs on the physical resources as possible to maximize resource utilization while being cognizant of application performance.

In the context of supporting real-time applications, Zhang et al. [17] proposed CPI^2 to improve the performance of latency-sensitive jobs when they experience performance interference. CPI^2 detects CPU performance interference incidents by automatically identifying jobs causing the issue, and optionally shielding victim jobs by throttling the triggering task. The authors prove that CPI (cycles-per-instruction) is a good representation of application response time. Using the insights from this work, we have used CPI as the key metric to measure the performance of tasks and develop our algorithms.

The technique we have presented in this paper was made possible after gaining deep insights from a usage trace of a production data center released by Google [5]. Several recent efforts [6], [25], [26] have analyzed this data providing deep insights to us on workload characteristics, task classification, statistical profile and actual resource utilization. These insights have helped us in our research.

III. Insights from Google's Production Data Center Trace

Google Inc. has released a data center cluster trace collected during a period of 29 days in May 2011 and a document called *Google cluster-usage traces: format+schema*, which describes the semantics, format, and schema of the trace in detail [5]. The dataset comprises machine events, machine attributes, jobs, tasks, constraints, and resource usage details. Although sensitive information, such as kernel version, processor speed, actual core count, actual memory size, and external IP address were either obfuscated or normalized, this workload consists of substantial data for more than 12,000 physical host machines. Due to the size of the task- and resource-usage data (about 1.2 billion rows), we utilized only three days worth of data, which we believe is sufficient to gain the overall insights.

The trace describes jobs, where a job is considered to be made up of one or more tasks. The jobs and tasks in the cluster trace have certain event types representing their states during their life cycles. These states are shown in Table I.

TABLE I: Job/Task Event Types in Cluster Trace [5]

Event Name	ID	Description
SUBMIT	0	A task or job was submitted to be scheduled
SCHEDULE	1	A task or job is scheduled to run on a host machine
EVICT	2	A task or job was descheduled due to overcommitting on that host machine or due to another higher priority job or its tasks
FAIL	3	A task or job failed and was descheduled
FINISH	4	A task or job has completed successfully
KILL	5	A task or job was canceled by the user
LOST	6	There is no indication for a task or job after termination
UPDATE_PENDING	7	A task or job info was updated while pending for scheduling
UPDATE_RUNNING	8	A task or job info was updated while running

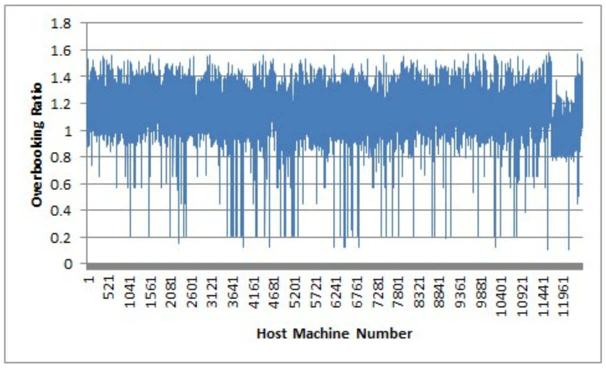
A host machine in the cluster could be in one of the three states shown in Table II.

TABLE II: Host Machine Event Types in Cluster Trace [5]

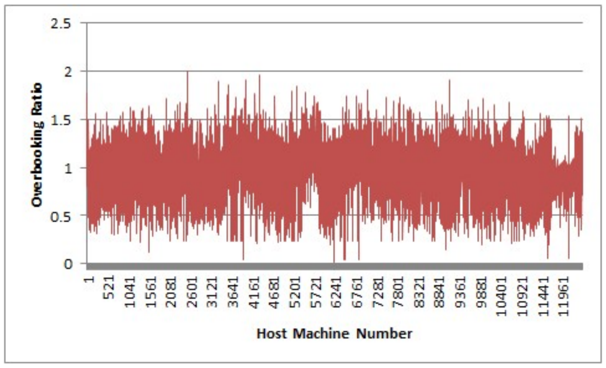
Event Name	ID	Description
ADD	0	A machine is added to host tasks
REMOVE	1	A machine was removed from the cluster
UPDATE	2	A machine's available resources were updated

A. Time-based Overbooking Ratio Analysis

In this analysis our aim is to show what the overbooking ratios for each host machine in the cluster were at time t=0, which is the beginning of the trace. Note that the data center was already operational when the trace was collected. Thus, t=0 does not mean the time when the data center was powered up. The motivation behind selecting a specific time is to investigate the overbooking ratios of all the host machines in the cluster from a high-level perspective. The CPU and memory overbooking ratios for each host machine in the cluster trace at t=0 are depicted in Figure 2. The formula in Equation (2) shows how the overbooking ratios in the figures are computed. Based on the criteria mentioned in [5], Equation (1) considers task event types in Table I and machine event types in Table II.



(a) CPU



(b) Memory

Fig. 2: Initial Overbooking Ratios in the Cluster Trace (at time 0)

The following steps are followed to compute the overbooking ratios for each host machine in the cluster:

- **Step 1 –** When a task is scheduled (*SCHEDULE* event type in the trace), the resource request is accumulated to the total value in Equation (1).
- **Step 2 –** When a task is evicted, failed, finished, killed, or lost (*FAIL*, *FINISH*, *KILL*, *LOST* event types in the trace), the resource amount of these tasks are subtracted from the total value in Equation (1). If a task's resource requirements or a machine's resource capacity is updated, then it is also taken into consideration.
- **Step 3 –** Based on the total resource requested and actual available physical host machine resource capacity, the overbooking ratio is computed in Equation (2).

$$TotalResourceAllocated = \sum_{i=0}^{n} ResourceAllocated_i \qquad (1)$$

$$OverbookingRatio = \frac{TotalResourceAllocated}{HostCapacity} \qquad (2)$$

where

$TotalResourceAllocated$: Total amount of resources allocated to all the tasks on host machine

n : is the total number of the tasks

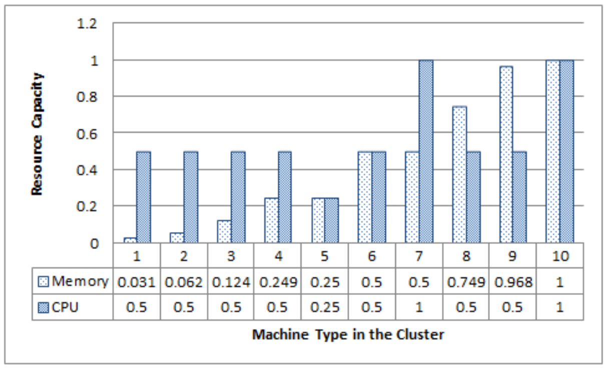
$ResourceAllocated$: size of allocated CPU or memory

$HostCapacity$: Resource capacity of host machine

As shown in Figure 2, it can be seen that host machines in the cluster are overbooked to some degree at the beginning of the trace for better utilization.

B. Machine-based Overbooking Ratio Analysis

In this analysis our aim is to show the overbooking ratios of certain host machines throughout their lifetime in the cluster. Since the cluster is heterogeneous, we first show the types of host machines and their hardware attributes in Figure 3. Due to confidentiality reasons, certain values in the cluster trace are released as encrypted or normalized. For example, machine attributes are transformed to hashed strings and machine capacities and resource requests are rescaled to the values between [0,1] by the normalization process. Therefore, the values seen in Figure 3 are the normalized capacity values in the cluster trace. As shown in Figure 3, there are ten different types of machines having different CPU and memory capacities in the cluster trace.



	1	2	3	4	5	6	7	8	9	10
▨ Memory	0.031	0.062	0.124	0.249	0.25	0.5	0.5	0.749	0.968	1
▨ CPU	0.5	0.5	0.5	0.5	0.25	0.5	1	0.5	0.5	1

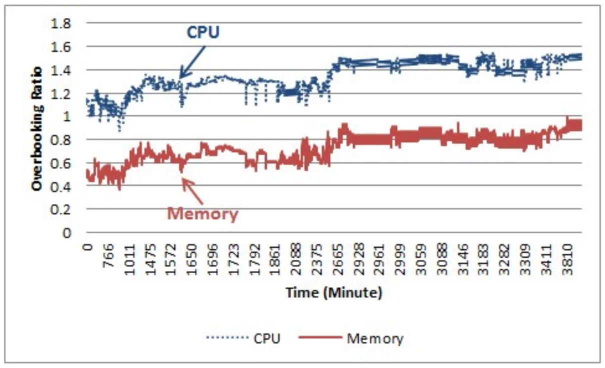
Fig. 3: Machine Types in the Cluster Trace

Among these machines, we have picked machines of types #3, #4, #6, #8, and #10 of the cluster from Figure 3 and depicted them in Table III because these machine types are the ones that appear the most prominently by numbers in the entire cluster.

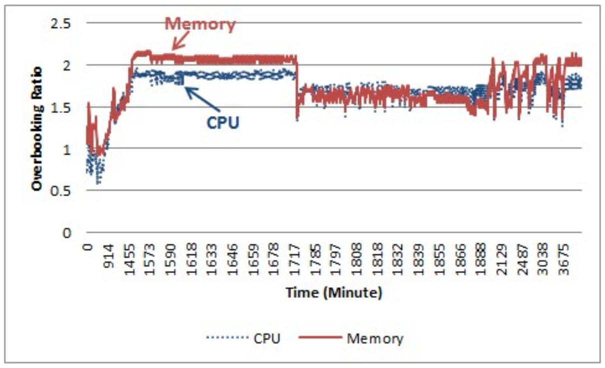
TABLE III: Host Machine Information (Normalized)

Host Name	ID in Cluster Trace	Machine Type	CPU Capacity	Memory Capacity
Host A	4837752655	#10	1	1
Host B	257337162	#8	0.5	0.749
Host C	381113	#6	0.5	0.4995
Host D	1094687	#4	0.5	0.2493
Host E	1094687	#3	0.5	0.1241

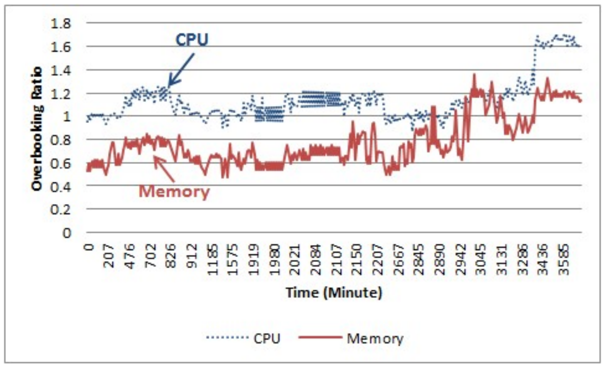
In Figure 4, overbooking ratios of Host A, Host B, Host C, and Host D in Table III are depicted. All four types of machines are overbooked in excess of the available physical resources at some point except Host A in Figure 4a. Host B in Figure 4b is overbooked up to 1.96 and 2.17 times more than its actual CPU and memory capacity, respectively. Compared to Host A, Host C and Host D in Figure 4, Host B has the



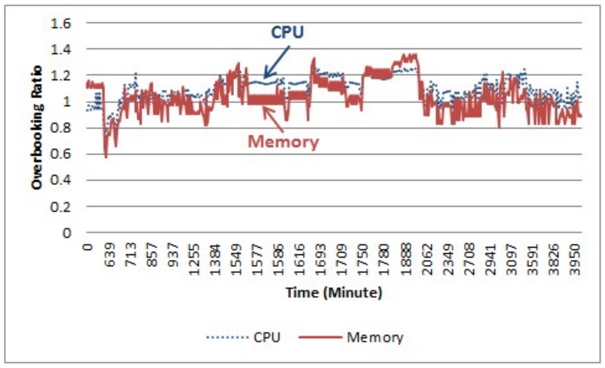
(a) Host A - Machine ID:4837752655 CPU:1 Memory:1



(b) Host B - Machine ID:257337162 CPU:0.5 Memory:0.749



(c) Host C - Machine ID:381113 CPU:0.5 Memory:0.4995



(d) Host D - Machine ID:1094687 CPU:0.5 Memory:0.2493

Fig. 4: CPU and Memory Overbooking Ratios for Different Types of Machines in the Cluster

highest overbooking ratios. It is noticeable in Figure 4a that the highest capacity machine type (*e.g.*, CPU: 1 and Memory: 1) does not overbook memory despite its noticeable CPU overbooking ratios greater than 1. We surmise that these type of machines in the cluster are likely reserved for latency sensitive, compute-intensive jobs. In summary, it can be inferred from both Figure 2 and Figure 4 that host machines in the cluster are overbooked to make the services more profitable and increase utilization level. This provides us an opportunity to investigate a systematic approach to overbooking in a way that soft real-time systems can be hosted in the cloud.

IV. iOverbook System Architecture and Design

Figure 5 depicts the architecture of iOverbook, which is our intelligent, machine learning-based approach to online determination of effective overbooking ratios for the machines of a data center. The goal of iOverbook is to online compute the CPU and memory overbooking ratios for each individual host machine within the next specified time interval. Since our aim is to compute the effective overbooking ratios online that will continue to assure the performance of soft real-time applications, we require an understanding of how the resources are currently utilized and the properties of existing applications so that we can predict the resource usage for a future specified time interval. Once we know this information, it should be feasible to determine how much overbooking is feasible and if it is acceptable for soft real-time applications.

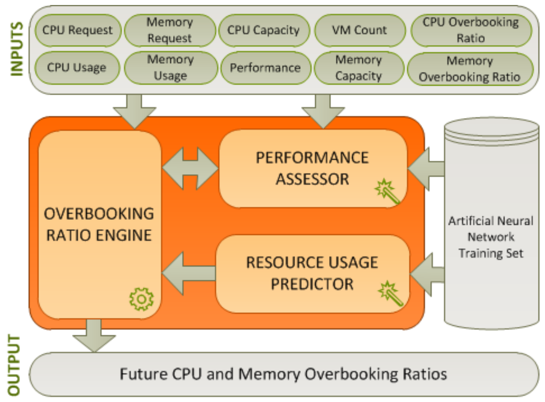


Fig. 5: iOverbook System Architecture

These three responsibilities motivated a three stage design for iOverbook, which comprises: (1) a resource usage predictor, (2) an overbooking ratio prediction engine, and (3) a performance assessor. The resource usage predictor and performance assessor components retrieve historic data from a training set repository to train their internal neural networks. iOverbook utilizes mean CPU and memory request, mean CPU and memory usage, mean performance, mean VM count, mean CPU and memory capacity, and CPU and memory overbooking ratios as input parameters. For this paper, we have showcased how iOverbook predicts the overbooking ratios for a time window of one hour, however, this property is tunable. The rest of this section explains the three components of iOverbook.

A. Resource Usage Predictor

The purpose of the resource usage predictor is to predict the mean CPU and memory usage of the host machine within the next hour (or the specified time interval). A two layer, feed forward artificial neural network (ANN) is employed for prediction. ANNs have a powerful ability to model and generalize both linear and non-linear relationships between input and output, and only a hidden layer is sufficient to make any prediction [27]. The sliding window mean CPU and memory resource usage data, and mean CPU and memory requests along with the host machine's resource capacity are the extracted features that are provided to the resource usage predictor. The structure of the ANN is depicted in Figure 6. The Levenberg-Marquardt back-propagation algorithm is employed for training the ANN.

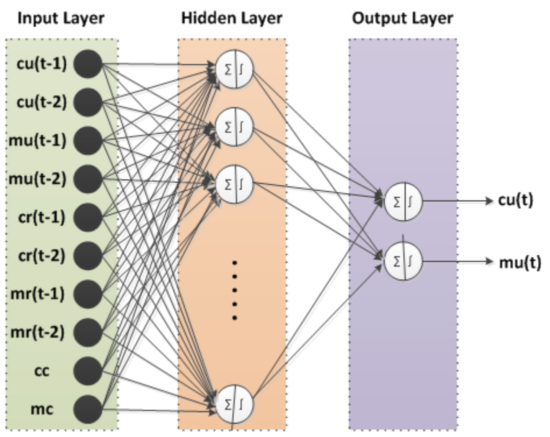


Fig. 6: Structure of the Resource Usage Prediction Artificial Neural Network

The topology of the ANN for predicting mean CPU and memory usage within the next specified time interval – in our case one hour – is shown in the mathematical formulation of the ANN below.

$$\begin{aligned}
\text{Input Layer} : \ & cu(t-1), cu(t-2), mu(t-1), mu(t-2), \\
& cr(t-1), cr(t-2), mr(t-1), mr(t-2), \\
& cc, mc \\
\text{Hidden Layer} : \ & 23 \text{ neurons} \\
\text{Activation Function} \ & \text{(in hidden layer)} \\
: \ & \text{Tangent Sigmoid} \\
\text{Output Layer} : \ & cu(t), mu(t) \\
\text{Transfer Function} \ & \text{(in output layer)} \\
: \ & \text{Pure Linear}
\end{aligned}$$

where

$$\begin{aligned}
t &= \text{the predicted hour} \\
cu(t-1) &= \text{mean CPU usage at hour } t-1 \\
cu(t-2) &= \text{mean CPU usage at hour } t-2 \\
mu(t-1) &= \text{mean memory usage at hour } t-1 \\
mu(t-2) &= \text{mean memory usage at hour } t-2 \\
cr(t-1) &= \text{mean CPU request at hour } t-1 \\
cr(t-2) &= \text{mean CPU request at hour } t-2 \\
mr(t-1) &= \text{mean memory request at hour } t-1 \\
mr(t-2) &= \text{mean memory request at hour } t-2 \\
cc &= \text{CPU capacity of the host machine} \\
mc &= \text{Memory capacity of the host machine} \\
cu(t) &= \text{mean CPU usage at hour } t \\
mu(t) &= \text{mean memory usage at hour } t
\end{aligned}$$

For testing and experimentation, 67 of the host machines which have the mean CPU usage greater than 10% (*i.e.* max percentage in the three days usage of the entire cluster trace to pick sufficient number of host machines for experimental study) are utilized. The idea behind this filtering is to study only those host machines having more compute-intensive tasks.

The reason behind utilizing these input parameters for resource usage prediction is that they are the common factors affecting the CPU and memory usage of a host machine. CPU and memory capacity are also provided to ANN due to the heterogeneity of data center machines, and help convey better correlation between input and output.

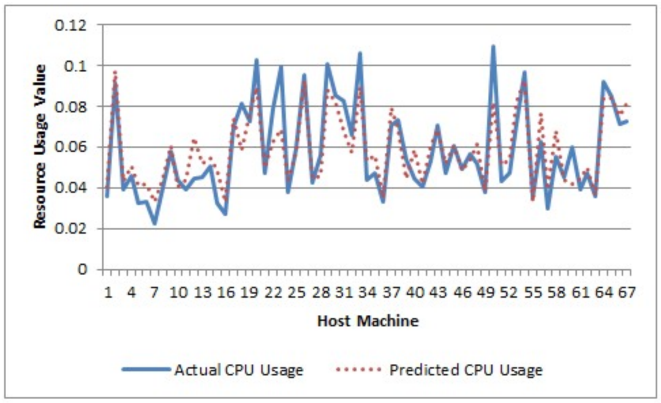
The best performance of the ANN was produced with 23 neurons in the hidden layer with the mean squared error value (MSE), which is the averaged squared difference between inputs and outputs, of 0.00009. The regression (R) value, which is the correlation between inputs and outputs, is 0.9. The generated MSE and R values indicate that the resource usage predictor predicts outputs with a negligible error value, and that the outputs of the ANN are very well correlated with its inputs.

The selection of activation function made in hidden layer and output layer are based upon the ANN type (*e.g.* back propagation dictates an activation function in hidden layer providing derivative), desired output value constraints, and based on trial-and-error performance results of ANN.

The predicted CPU and memory usage values along with the actual usage values for each host machine is illustrated in Figure 7. The training ANN involved using 72 hours of the cluster trace except the 49th hour. The prediction was made for the 49th hour. As seen in Figure 7, the predicted resource usage value follows the actual usage values well enough because of the decent MSE and R values.

B. Overbooking Ratio Prediction Engine

After the resource usage predictor predicts the CPU and memory usage for the next one hour time window, the overbooking ratio prediction engine computes the CPU and memory overbooking ratios per machine, and hands it to the performance assessor. The performance assessor component predicts the performance by using these new overbooking ratios and hands it back to the overbooking ratio prediction engine. This two way communication between overbooking ratio prediction engine and performance assessor iterates until the predetermined convergence values (calculated manually from the trace) in Table IV is satisfied.



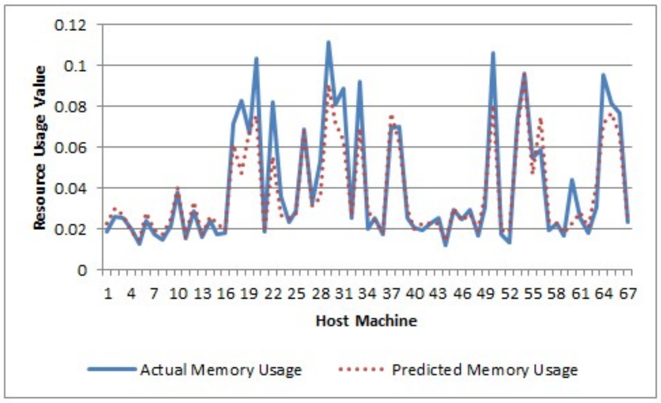


Fig. 7: Actual and Predicted Hourly Mean CPU and Memory Value Comparison

The details of the computation are shown in Equation (3).

$$OverbookingRatio(t) = \frac{HostCapacity(t) - SecuritySlack(t)}{PredictedUsage(t)} \tag{3}$$

$$SecuritySlack(t) = HostCapacity(t) \times SecurityPercentage \tag{4}$$

$$ResourceRequest(t) = OverbookingRatio(t) \times HostCapacity$$

where

$OverbookingRatio(t)$: CPU and memory overbooking ratios at hour t for a host machine

$HostCapacity(t)$: resource capacity (e.g. CPU and memory) of a host machine at hour t

$SecurityPercentage(t)$: elastic capacity on a host machine to converge the best ratio at hour t

C. Performance Assessor

The performance assessor component is responsible for predicting the performance thereby providing an assurance that the new overbooking ratios computed by the overbooking ratio engine do not violate the SLAs. The Performance assessor uses the 1/CPI as the performance metric which means that the higher the value, the better the performance. The SLA violation is checked based on the mean performance values of the same type of host machine in Table IV. If the machine under consideration's mean performance is greater than the overall mean performance of the same type of host machine in the cluster, iOverbook assumes that SLA is not going to be violated thereby providing performance assurances to soft real-time applications. In Section V, this SLA violation logic is elaborated upon by taking standard deviation of same type of host machines into account for better performance results.

The structure of the performance predictor ANN is similar to the resource usage predictor ANN in Figure 6 with different input and output. The topology of this ANN for predicting performance is provided in the mathematical formulation below. It is considered that allocated amount of resources and mean VM count on a host machine are changed once the overbooking ratio engine computes new ratios. Based upon newly computed ratios in Equation (3), the performance assessor is employed to check whether these new overbooking ratios may trigger any SLA violations. As long as the predicted performance is less than the mean performance value (*i.e.*, a SLA violation will occur), the performance assessor increases the security slack value in Equation (4) by 0.5% and requests new ratios from overbooking ratio engine till the ratios converge to the values not violating SLA.

TABLE IV: Mean Performance Values of Each Host Machine Type in the Cluster

CPU	Memory	Mean (μ) Performance	Stdev (σ) Performance
1	1	0.1384	0.0896
0.5	0.749	0.1416	0.1142
0.5	0.4995	0.1474	0.1157
0.5	0.2493	0.1474	0.1227
0.5	0.1241	0.1647	0.1485

Input Layer : $cr(t), mr(t), cor(t), mor(t), vm(t), cc, mc$

Hidden Layer : 22 neurons

Activation Function (in hidden layer) : Tangent Sigmoid

Output Layer : $P(t)$

Transfer Function (in output layer) : Pure Linear

where

t = the predicted hour

$cr(t)$ = mean CPU request at hour t

$mr(t)$ = mean memory request at hour t

$cor(t)$ = CPU overbooking ratio at hour t

$mor(t)$ = memory overbooking ratio at hour t

$vm(t)$ = mean VM count at hour t

cc = CPU capacity of the host machine

mc = Memory capacity of the host machine

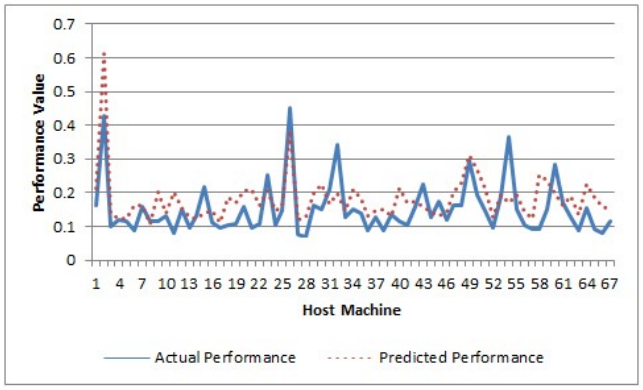


Fig. 8: Actual and Predicted Performance Results Comparison

The best performance of the ANN was produced with 22 neurons in the hidden layer with the MSE of 0.0053 and the R of 0.67. The generated MSE and R values indicate that the performance assessor component predicts performance output value with a somewhat negligible error compared to our resource usage predictor ANN. The outputs of this ANN are well correlated with its inputs.

The predicted performance value along with the actual performance values for each host machine is illustrated in Figure 8. The predicted hour is the same as the overbooking ratio prediction engine, which is chosen as the 49th hour in the timeline. As can be seen in Figure 8, the predicted performance value follows the actual usage values well because of the lower MSE and good R values.

V. Validating the iOverbook Approach

This section validates the iOverbook approach using the Google cluster usage trace.

Validation Approach: Since it is not possible to recreate the Google's data center trace, we have used an alternate approach to validating iOverbook. We use part of the usage trace to train iOverbook. Subsequently, we use iOverbook to predict overbooking for a time interval that was not used in the training phase. The results of this prediction are then compared to the actual numbers appearing in the usage trace.

To that end we have used 72 hours of usage trace data to train iOverbook's ANNs except the 49th hour in that interval, and instead used iOverbook to predict the overbooking rates for the 49th hour. The predicted overbooking rates (both CPU and memory) and performance are compared to the actual overbooking and performance seen from the usage trace. In our experiments, two different overbooking ratios are computed under two different conditions: (1) if the predicted performance value ($P(t)$) is greater than or equal to the mean performance value of the same type of host machines in the cluster (*i.e.*, $P(t) >= \mu$), and (2) if the predicted performance value is greater than or equal to the sum of the mean performance value of the same type of host machines and two times the standard deviation of this value (*i.e.*, $P(t) >= \mu + 2\sigma$). The motivation behind computing overbooking ratios under two different conditions is to provide results under tighter constraints. The mean and standard deviation values are shown earlier in Table IV.

We then analyze how these predicted overbooking ratios for each host machine help to improve the resource utilization and reduce power consumption in the data center. To determine the power consumption, we have utilized SPECpower_ssj2008 [28], an industrial benchmark measuring power and performance values of different computer architectures, to compute power consumption of host machines.

Comparing Actual versus Predicted Overbooking and Performance: Figure 9 compares Google host machines' actual and iOverbook's overbooking ratios computed by overbooking ratio prediction engine at t=49.

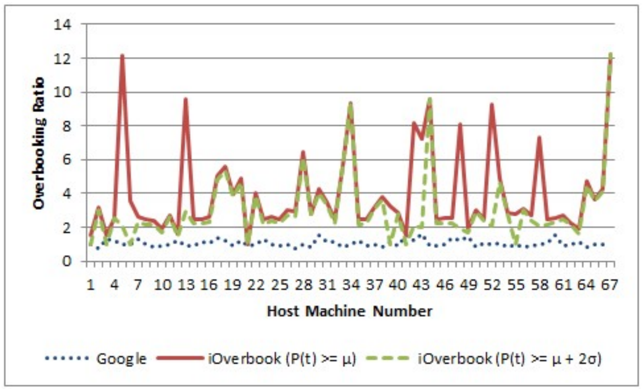


Fig. 9: Comparison of Google's Host Machines' Actual and iOverbook's CPU Overbooking Ratios under Different Performance Considerations

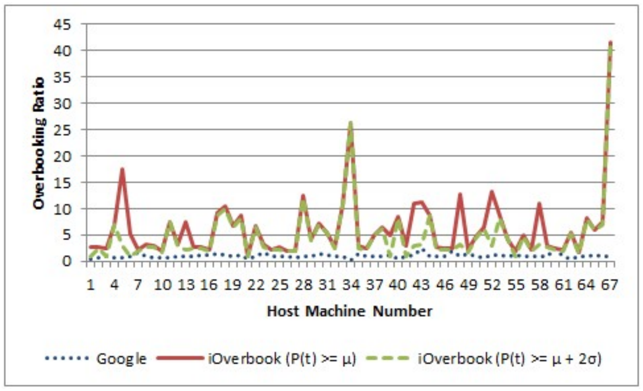


Fig. 10: Comparison of Google's Host Machines' Actual and iOverbook's Memory Overbooking Ratios under Different Performance Considerations

In the context of the Google trace, the following inferences can be made from Figures 9 and 10. Overall, iOverbook was able to predict higher overbooking ratios for host machines compared to Google's overbooking without SLA violations.

(1) 66/67 host machines could have been overbooked without SLA violation under $P(t) >= \mu$ condition (recall that performance is measured as 1/CPI so any value less than the mean is a SLA violation).

(2) 20/67 and 36/67 host machines could have had CPU and memory overbooking ratios greater than 4, respectively, without SLA violation under $P(t) >= \mu$ condition.

(3) 60/67 host machines could have been overbooked without SLA violation under $P(t) >= \mu + 2\sigma$ condition. These results are somewhat inferior compared to #1 due to a tighter performance constraint.

(4) 12/67 and 26/67 host machines could have had CPU and memory overbooking ratios greater than 4, respectively, without SLA violation under $P(t) >= \mu + 2\sigma$ condition. These results are somewhat inferior compared to #2 due to a tighter performance constraint.

In Figure 11, Google's host machines' actual (*i.e*, at t=49) and iOverbook's predicted performance values associated with the overbooking ratios in Figure 9 are depicted. A zero value in the figure means that machine is not overbooked. As seen from the Figure 11, there is one host machine under $P(t) >= \mu$ condition and seven host machines under $P(t) >= \mu + 2\sigma$ condition that iOverbook predicted a SLA violation and did not allow those host machines to be overbooked.

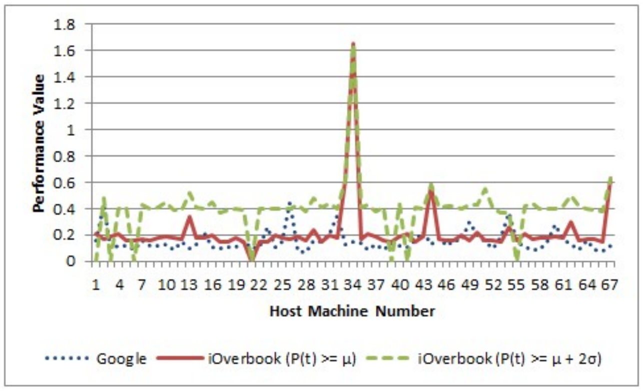


Fig. 11: Google's Host Machines' Actual and iOverbook's Predicted Performance Value under Different Performance Considerations

Improved Utilizations and Power Savings due to iOverbook: Recall from Section III that the cluster trace provides obfuscated configurations of machine attributes, which does not allow us to identify and compute the exact power consumption of host machines. Therefore, we surmise the potential configuration of the machine such that it is in tune with the normalized resource capacities of physical host machines shown in Table III. The estimated configurations are shown in Table V.

Table VI compares the actual (from the trace) and iOverbook's resource utilization effects. The "Total CPU" request value in the table is based upon the overbooking ratios computed by iOverbook. The ratio of total CPU usage over total CPU requested is used to calculate the utilization value when the host machines are overbooked with iOverbook. As seen in Table VI, the actual utilization for 67 host machines in the trace was 11%. In contrast, the utilization level using iOverbook could have been 44% under $P(t) >= \mu$ and 34% under $P(t) >= \mu + 2\sigma$ conditions, which shows an improvement of 33% and 23%, respectively.

TABLE V: Estimated Host Machine Configurations [28]

Host Name	Processor Name	CPU (cores)	Memory (GB)	Mean Watts @100%	Mean Watts @Idle
Host A	Intel Xeon E5-4607	24	64	368	109
Host B	Intel Xeon X5675	12	48	293	118
Host C	Intel Xeon E5-2640	12	32	231	57
Host D	Intel Xeon E5645	12	16	200	63.1
Host E	AMD Opteron 2419 EE	12	8	178	74.6

TABLE VI: Resource Utilization Results in Test Set

	Actual Value	iOverbook ($P(t) >= \mu$)	iOverbook ($P(t) >= \mu + 2\sigma$)
Total CPU Request	36.08261	138.72779	106.16678
Total CPU Capacity	34	34	34
Total CPU Usage / Total CPU Requested	0.10821	0.10821	0.10821
Total CPU Usage	3.90452	15.01184	11.48839
Mean CPU Utilization	11%	44%	34%

The comparison of power consumption results for different consolidation cases are shown in Table VII. Consolidation refers to packing as many tasks on as less number of machines as possible and leave the rest of the machines at either powered off or idle mode. *Actual Case* is the current status of the host machines. *Case-1* and *Case-2* represent the status of the host machines under $P(t) >= \mu$ condition representing the expected power consumption if the host machines with no tasks on it after the consolidation were powered off or remain powered on but in idle mode conditions, respectively. *Case-3* and *Case-4* are the same cases as *Case-1* and *Case-2* except they show the results under the $P(t) >= \mu + 2\sigma$ condition.

TABLE VII: Power Consumption Results in Test Set

	Total Watt	Number of Servers @On	Number of Servers @Off	Number of Servers @Idle	Savings
Actual	6597.85	67	0	0	0%
Case-1	2337.92	11	56	0	65%
Case-2	6661.72	11	0	56	-1%
Case-3	4873.33	31	36	0	26%
Case-4	7204.53	31	0	36	-9%

As seen from Table VII, iOverbook helped save roughly 65% and 26% of energy for *Case-1* and *Case-3*, respectively. However, if the host machines with no tasks on it are left in the idle state (*Case-2* and *Case-4*), then it has a negative impact on energy consumption, which we surmise can be attributed to the power consumption of host machines at idle mode.

Lessons from the Validation Experiments: These validation results demonstrate that adding higher standard deviations gives us less beneficial results but probably tighter and a preferred result to assure SLAs. CSPs can utilize our technique by first training our ANNs with their own historic data and then integrating iOverbook with their actual job and VM scheduler in the data center. Therefore, the scheduler could overbook each host machine by considering the overbooking ratios provided by iOverbook. Since our approach allows runtime updates to overbooking ratios, it can adapt autonomously to changing workloads.

VI. Conclusion

This paper presented insights from an analysis of Google's production data center usage trace. Using these insights we presented iOverbook, which is an intelligent and online resource overbooking strategy for supporting cloud-based soft real-time applications and effective server utilization. iOverbook employs two artificial neural networks for predicting a host machine's future resource usage and performance. It requires historical usage data that cloud providers can provide for use in their data centers. The forecasted values are used in computing significantly better CPU and memory overbooking ratios than those used by Google in their production data center without triggering SLA violations.

In this work, we considered an hour interval for our prediction mechanism, however, this value is tunable. Based on the needs and requirements of CSPs, this interval could easily be changed to another interval by modifying the training set of neural networks. Another approach could be to train different neural networks for different time intervals required and employ the one for which time interval is desired for prediction.

In the current work we did not consider the potential for noise in the available traces. Our future work will investigate effective filtering of noise and using confidence intervals.

Acknowledgments

This work was supported in part by the National Science Foundation CAREER CNS 0845789. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of NSF.

References

[1] I. S. Moreno and J. Xu, "Neural network-based overallocation for improved energy-efficiency in real-time cloud environments," in *Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC), 2012 IEEE 15th International Symposium on*. IEEE, 2012, pp. 119–126.

[2] S. A. Baset, L. Wang, and C. Tang, "Towards an understanding of oversubscription in cloud," in *Proceedings of the 2nd USENIX conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services*. USENIX Association, 2012, pp. 7–7.

[3] I. S. Moreno and J. Xu, "Customer-aware resource overallocation to improve energy efficiency in realtime cloud computing data centers," in *Service-Oriented Computing and Applications (SOCA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1–8.

[4] (2013, Sep.) Best practices for oversubscription of cpu, memory and storage in vsphere virtual environments. [Online]. Available: https://software.dell.com

[5] C. Reiss, J. Wilkes, and J. Hellerstein, "Google cluster-usage traces: format+ schema," *Google Inc., White Paper*, 2011.

[6] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Towards understanding heterogeneous clouds at scale: Google trace analysis," *Intel Science and Technology Center for Cloud Computing, Tech. Rep*, 2012.

[7] T. Abels, P. Dhawan, and B. Chandrasekaran, "An overview of xen virtualization," 2005.

[8] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, "kvm: the Linux virtual machine monitor," in *Proceedings of the Linux Symposium*, vol. 1, 2007, pp. 225–230.

[9] A. Muller and S. Wilson, "Virtualization with vmware esx server," 2005.

[10] M. Lee, A. Krishnakumar, P. Krishnan, N. Singh, and S. Yajnik, "Supporting soft real-time tasks in the xen hypervisor," in *ACM SIGPLAN Notices*, vol. 45, no. 7. ACM, 2010, pp. 97–108.

[11] (2013, Sep.) openstack.org. [Online]. Available: http://www.openstack.org

[12] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The Eucalyptus Open-source Cloud-computing System," in *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*. IEEE Computer Society, 2009, pp. 124–131.

[13] C. A. Waldspurger, "Memory resource management in vmware esx server," *ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 181–194, 2002.

[14] R. Nathuji, A. Kansal, and A. Ghaffarkhah, "Q-clouds: managing performance interference effects for qos-aware clouds," in *Proceedings of the 5th European conference on Computer systems*. ACM, 2010, pp. 237–250.

[15] O. Tickoo, R. Iyer, R. Illikkal, and D. Newell, "Modeling virtual machine performance: challenges and approaches," *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 3, pp. 55–60, 2010.

[16] X. Pu, L. Liu, Y. Mei, S. Sivathanu, Y. Koh, and C. Pu, "Understanding performance interference of i/o workload in virtualized cloud environments," in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*. IEEE, 2010, pp. 51–58.

[17] X. Zhang, E. Tune, R. Hagmann, R. Jnagal, V. Gokhale, and J. Wilkes, "Cpi2: Cpu performance isolation for shared compute clusters," in *Proceedings of the 8th ACM European Conference on Computer Systems*, ser. EuroSys '13. New York, NY, USA: ACM, 2013, pp. 379–391.

[18] F. Caglar, S. Shekhar, and A. Gokhale, "A Performance Interference-aware Virtual Machine Placement Strategy for Supporting Soft Real-time Applications in the Cloud," in *Submitted to the 2nd International Workshop on Real-time and Distributed Computing in Emerging Applications (REACTION) of the 34th IEEE Real-Time Systems Symposium (RTSS '13)*. Vancouver, BC, Canada: IEEE, Dec. 2013.

[19] R. E. Edwards, J. New, and L. E. Parker, "Predicting future hourly residential electrical consumption: A machine learning case study," *Energy and Buildings*, vol. 49, pp. 591–603, 2012.

[20] T. Olsson, "Evaluating machine learning for predicting next-day hot water production of a heat pump," in *4th International Conference on Power Engineering, Energy and Electrical Drives - IEEE POWERENG 2013*, May 2013. [Online]. Available: http://www.mrtc.mdh.se/index.php?choice=publications&id=3379

[21] M. Imam, S. Miskhat, R. Rahman, and M. A. Amin, "Neural network and regression based processor load prediction for efficient scaling of grid and cloud resources," in *Computer and Information Technology (ICCIT), 2011 14th International Conference on*. IEEE, 2011, pp. 333–338.

[22] L. Tomás and J. Tordsson, "Improving cloud infrastructure utilization through overbooking," in *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference*, ser. CAC '13. New York, NY, USA: ACM, 2013, pp. 5:1–5:10. [Online]. Available: http://doi.acm.org/10.1145/2494621.2494627

[23] G. Birkenheuer, A. Brinkmann, and H. Karl, "The gain of overbooking," in *Job Scheduling Strategies for Parallel Processing*. Springer, 2009, pp. 80–100.

[24] N. Roy, A. Dubey, and A. Gokhale, "Efficient Autoscaling in the Cloud using Predictive Models for Workload Forecasting," in *4th IEEE International Conference on Cloud Computing (Cloud 2011)*. Washington, DC: IEEE, Jul. 2011, pp. 500–507.

[25] Z. Liu and S. Cho, "Characterizing machines and workloads on a google cluster," in *Parallel Processing Workshops (ICPPW), 2012 41st International Conference on*. IEEE, 2012, pp. 397–403.

[26] A. K. Mishra, J. L. Hellerstein, W. Cirne, and C. R. Das, "Towards characterizing cloud backend workloads: insights from google compute clusters," *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 4, pp. 34–41, 2010.

[27] B. Kröse, B. Krose, P. van der Smagt, and P. Smagt, "An introduction to neural networks," 1993.

[28] (2013, Oct.) Standard performance evaluation corporation. [Online]. Available: http://www.spec.org/power_ssj2008/