Towards a Performance Interference-aware Virtual Machine Placement Strategy for Supporting Soft Real-time Applications in the Cloud

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

***Abstract*—It is standard practice for cloud service providers (CSPs) to overbook physical system resources to maximize the resource utilization and make their business model more profitable. Resource overbooking can lead to performance interference, however, among the virtual machines (VMs) hosted on the physical resources causing performance unpredictability for soft real-time applications hosted in the VMs, which is unacceptable to these applications. Balancing these conflicting requirements needs a careful design of the placement strategies for hosting soft real-time applications such that the performance interference effects are minimized while still allowing resource overbooking. These placement decisions cannot be made offline because workloads change at run time. Moreover, satisfying the priorities of collocated VMs may require VM migrations, which require an online solution. This paper presents a machine learning-based, online placement solution to this problem where the system is trained using a publicly available trace of a large data center owned by Google. Our approach first classifies the VMs based on their historic mean CPU and memory usage, and performance features. Subsequently, it learns the best patterns of collocating the classified VMs by employing machine learning techniques. These extracted patterns are those that provide the lowest performance interference level on the specified host machines making them amenable to hosting soft real-time applications while still allowing resource overbooking.**

***Keywords*-virtual machine placement, cloud computing, performance interference, resource overbooking, application QoS.**

I. Introduction

Resource overbooking [1], [2] is used as a means to increase resource utilization in servers of a data center and making the cloud-hosted services more profitable. The idea behind resource overbooking in the cloud data centers is to commit more resources, such as CPU and memory, than are actually available on the physical host machines. The intuition behind the overbooking strategy is that users often request more resources than their applications actually need thereby providing an opportunity to the cloud provider to overbook. Considering this trend, contemporary hypervisors, such as Xen [3], KVM [4], and VMware ESX Server [5], provide the necessary support to make overbooking feasible to implement in practice.

At the same time, we are witnessing an increasing trend towards hosting soft real-time applications, such as airline reservation systems, virtual reality applications, Netflix video streaming, and Coursera online digital learning, in the cloud. These applications demand more stringent performance requirements, *i.e.,* these applications are sensitive to fluctuations in latency and response times. However, the resource overbooking used by cloud providers may incur negative impact on their performance because multiple VMs collocated due to resource overbooking can trigger significant performance interference [6], [7], [8], [9] for applications hosted in their respective VMs.

Although there exists prior work on performance isolation [9] among VMs collocated on an overbooked host machine, it is still a challenging task to consider performance interference for VM placement and shield the VMs from its neighbors due to the nature of resource sharing, resource overbooking practices employed, and the fluctuating workload characteristics in the cloud. Therefore, an application running on one VM might still impact the performance of another application running on a separate VM on the same host machine. Specifically, network- and compute-intensive applications might be adversely impacted.

Since performance interference is caused because of how one VM interacts with another VM collocated on a physical host, addressing the performance interference challenges that stem from resource overbooking and satisfying the response time requirements of soft real-time applications will require effective trade-offs involved in the placement of VMs on host machines by carefully considering the actual workload characteristics of the VMs. Due to the changing dynamics of the workloads on the VMs and also because VMs often tend to migrate from one physical machine to another for a variety of reasons, traditional and offline heuristics such as bin packing will not be applicable for interference-aware VM placement in cloud computing. Consequently, we have focused on a VM placement strategy considering not only the performance interference effects but also the workload characteristics of VMs.

Our prior work involving VM-based resource management has considered power-performance trade-offs [10], physical server consolidation using VM overbooking [11], and auto tuning of hypervisor parameter [12] but none of these works perform resource management considering performance interference between collocated VMs.

To assure that latency-sensitive soft real-time applications receive their required Quality of Service (QoS) despite VMs being collocated due to resource overbooking strategies employed by the cloud provider, we present an online VM placement technique based on machine learning and made available in a middleware called *hALT* (The harmonious Art of Living Together). This paper makes two contributions. First, we analyze a trace log of a production data center published by Google [13]. This analysis provides us insights on how collocation of VMs resulting from a migration from one machine to another can cause undue performance interference despite the target machine having more capacity. Second, using these insights we use machine learning to learn about the desired VM placement patterns and use these as a means of making runtime placement decisions.

The rest of this paper is organized as follows: Section II describes relevant related work comparing it with our contributions; Section III describes our findings from the analysis of the Google production data; Section IV presents the system architecture for our machine learning-based solution; and Section V presents concluding remarks alluding to future work.

II. Related Work

This section presents related work on VM placement and solutions to address performance interference, and compares it with our solution called hALT.

Q-Clouds [6] is a QoS-aware framework to manage performance interference in the cloud. It works on the principle of provisioning additional resources to alleviate performance interference. It applies an online feedback mechanism to build a model for capturing interference interactions and use it for resource management. Moreover, the system employs a staging server to determine the resource requirements and leaves a head room, *i.e.*, slack resource for performance management. Q-Clouds allows specification of different levels of QoS, known as Q-states, to increase the resource utilization. However, the slack resources still lead to under utilization of the server resources. Frequent resource allocation due to feedback mechanism can also cause performance overhead for the hypervisor.

Zhu et al. [14] proposed an interference model which predicts application QoS. It considers time-variant inter-dependence among the different levels of resource contention. The authors develop a resource usage profile as a vector of matrices for different performance metrics and then apply a consolidation algorithm to accommodate applications to minimize interference and achieve QoS. We believe this work focuses on developing simplistic models for complex resource utilization relationships, whereas we use k-means clustering to group the VMs in different classes to capture the complex relationships and then apply machine learning to determine performance interference.

TRACON [15] is a task and resource allocation framework for data-intensive applications. It develops three interference prediction models: weighted mean method model, linear model and non-linear model using statistical machine learning for reasoning. It then employs an interference-aware scheduler for reducing performance interference. The focus of this technique is network I/O-intensive applications whereas our approach is focused on CPU-based applications. Moreover, the training data used by TRACON is generated with a workload generator comparable to ours which utilizes traces from a production data center.

In [16], Kambadur et al. studied the methodology and several complexities behind measuring performance interference in data centers due to resource contention and proposed a new technique based on finding the performance interference between base application and co-runners on the same machine. They have also used the Google production workloads. In this work, the authors have measured the performance interference in order to identify interference relationships and classes but have not demonstrated its application. We have leveraged some of the insights and parameters from this work.

Moreno et al. [17] proposed a method for interference-aware virtual machine placement by analyzing its impact on energy efficiency in data centers. The combined interference score utilized in this work requires the knowledge of maximum throughput of each workload running on a host machine when mixed with other workload types. This might require employing some applications to reside on VMs to populate this information from the workload which may result in high overhead when a host runs numerous different types of workloads. In contrast, hALT discovers and extracts the best VM patterns by employing machine learning algorithms to predict future performance interference level. hALT also differs from this work based on its VM classification features that uses performance.

Modern day hypervisors like Xen and KVM used for virtualization do not provide an effective solution for performance isolation. Even though resources are sliced and allocated to different VMs, they are still shared. The isolation across VMs provided by hypervisors reduces the visibility of application performance from one VM to other, thus making it difficult to triage the performance issues. The problem is further aggravated as the host machines get overloaded by VMs. This performance interference has been amply demonstrated [18], [8], [6]. The LXC Linux Container [19] is another virtualization technology which promises to provide better performance. It reduces hypervisor overhead by running the guest operating systems within the same host kernel. However, Linux containers suffer from resource contention and security issues. Thus, we need a solution which can minimize the performance interference and provide better results, which is the focus of our work in hALT.

III. Analyzing Performance Impact of Virtual Machine Migration and Collocation

This section presents results from analyzing a trace log of Google's data center [13] revealing how the performance of a VM is affected when it is migrated to a different host machine in the data center. We present this information since it provides key insights into making decisions on what factors to consider in placing a VM such that performance interference can be minimized and soft real-time applications can obtain more predictable performance.

To analyze the performance interference in a production, large-scale data center, we chose the usage trace published by Google [13]. The trace contains a dataset for about 12,000 distinct machines collected over a 29 day period in the month of May, 2011.

Each machine in the cluster was defined by its CPU and memory capacity. There were three distinct CPU capacities and five levels of memory capacities. Having said that, the values provided are relative and we do not know what the actual number for CPU or memory it corresponded to. For our analysis, the relative values are adequate since we utilize normalized data.

According to the Google trace usage document [13], a task or job is migrated into another host machine if either the actual host machine is overloaded, or because of a high priority task or job entering the system, or any other issue related to the physical host machine, such as a failure. Table I shows how these event types are represented in the cluster trace, where:

t_1: time interval between the time when Task 1 is scheduled to run on Host A until it is evicted from Host A and migrated to Host B.

t_2: time interval between the time when Task 1 is scheduled to run on Host B until it is completed on Host B.

t_3: overall time interval for Task 1.

Table I
Some Event Types in Google Cluster Trace

Event Name	ID	Description
SCHEDULE	1	A task or job is scheduled to run on a host machine
EVICT	2	A task or job is descheduled on a host machine
FINISH	4	A task or job has completed its task successfully

All of the resource usage and request measurements (*e.g.*, CPU and memory) are normalized between zero to one by scaling them to the largest capacity of the resource in the cluster [13]. The "Cycles Per Instruction" (CPI) [9] metric is used as a performance metric since it represents the application response time well enough for compute-intensive applications. Based on [9], the lower the CPI value, the better the performance is. Thus, performance values in all the figures in this section are represented by "Instruction Per Cycle (IPC)," which is the multiplicative inverse of CPI. IPC is used because it is more intuitive and effective in interpreting performance data.

How the collocated tasks running on the same physical host machine and heterogeneous host machines affect the performance of the soft real-time applications and how this scenario can be carefully considered by a VM placement algorithm are analyzed for a task in the cluster trace.

To that end, a task in the cluster trace is considered as a VM and the resource overbooking ratio is the value obtained by dividing the total number of resources requested by the capacity of the physical host machine. An overbooking ratio greater than one obviously indicates that actual resource demand exceeds the physical host machine's resource capacity.

The "Total Number of Tasks" column in Table II does not include the tasks which have zero mean CPU and memory utilization values for the tasks shown in Table III. It was assumed that those are just abnormal data (*i.e.*, outliers) or binaries were being copied on host machine [13].

As seen in Table III, Task 1 has highest CPU utilization and quite high memory utilization before migration while the resource requests remain same before and after the migration. The CPU utilization has high fluctuation before migration, but remains low after migration. The reason for over utilization values is because of the usage of exceeding the allocated resource [13].

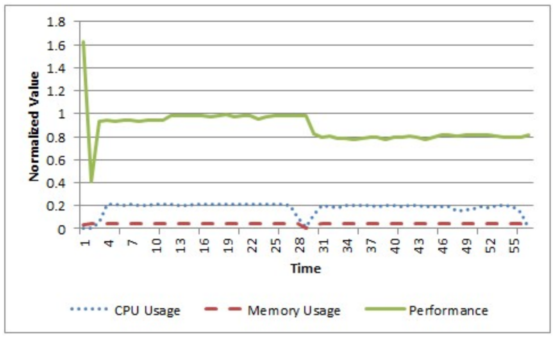

Figure 1. CPU, Memory Usage, and Performance of Task 1 on Host A and Host B

The CPU usage, memory usage, and performance of Task 1 is depicted in Figure 1. Task 1 was initiated to be migrated (i.e., EVICTED event type in cluster trace) from Host A to Host B at the beginning of t_2 as shown by a sudden drop in resource usage values in Figure 1. As seen in Table II, the capacity of the Host A is lower than the capacity of Host B. When the task is migrated to Host B from Host A, the performance of Task 1 dropped about %17. As seen in Table II, the number of tasks on Host B is much more than Host A. Host A and Host B have 9 and 19 tasks, respectively. One of the reasons for the bad performance on Host B is highly likely because of the performance interference caused

Table II
HOST MACHINE INFORMATION FOR ONLY THE TIME INTERVAL WHEN THE TASK BEING ANALYZED RESIDES

Name	ID	Total Number of Tasks	CPU Utilization (Mean)	CPU Utilization (Stdev)	Memory Utilization (Mean)	Memory Utilization (Stdev)	CPU Capacity	Memory Capacity
Host A	294823364	9	%6.02	%13.31	%4.64	%6.14	0.5	0.2493
Host B	4874238388	19	%3.15	%5.70	%3.64	%4.80	1	1

Table III
TASK INFORMATION BEFORE AND AFTER THE MIGRATION FOR THE TIME INTERVAL SPECIFIED

Name	Job ID	Task Index	CPU Utilization (Mean)	CPU Utilization (Stdev)	Memory Utilization (Mean)	Memory Utilization (Stdev)	CPU Request	Memory Request
Task 1 (before)	6276036736	12	%102.38	%34.57	%72.62	%3.39	0.1814	0.06165
Task 1 (after)	6276036736	12	%103.91	%10.94	%74.45	%2.45	0.1814	0.06165

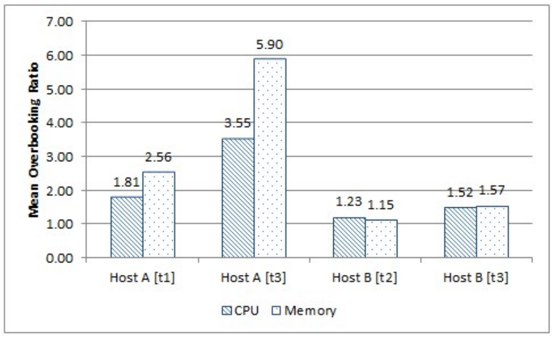

Figure 2. Overall and Time Limited Overbooking Ratios of Host A and Host B

by CPU contention.

As depicted in Figure 2, this could also be seen by overbooking ratios of hosts during t_1 and t_2. Host A has mean overbooking ratios of 1.81 for CPU and 2.56 for memory during t_1 which considerably exceeds the Host A's resource capacity. The mean overbooking ratios for Host A during t_1 is apparently lower than the machine's mean CPU and memory overbooking ratios of 3.55 and 5.90, respectively, during t_3. This could be interpreted as even though (1) the requested resource and mean utilization values remain the same, (2) Host B has higher capacities, (3) more number of tasks on Host B and more resource demand than the actual resource capacity by 1.23 and 1.15, and (4) different resource usage behavior of other tasks running on Host B may have triggered more performance interference and resource contention between tasks.

More importantly, these analyses results indicate that there might be performance differences in soft real-time applications on different host machines even though allocated capacity is identical and resource usage pattern is similar. Therefore, VM placement decisions must be conducted by considering performance interference and resource contention on the host machine. Moreover, latency-sensitive applications must be placed into that host machine where they will receive their desired application performance.

IV. System Architecture for Virtual Machine Placement

This section presents hALT's system architecture that supports soft real-time systems in the cloud to be minimally affected by performance interference.

A. Rationale Behind the Techniques Leveraged

Our solution approach first classifies the Google cluster trace log into meaningful categories using heuristics and then applies machine learning to find best VM collocation patterns. The classification is performed using a k-means clustering algorithm [20]. k-means is an unsupervised learning algorithm that helps to classify the VMs in different classes based on their performance. It provides good results with large datasets such as the one used in our approach. The Silhouette [21] method is used for graphically representing objects within the cluster. It fits well with the k-means clustered data and is employed in our approach to analyze the VM clusters. To capture the non-linear relationships between performance interference amongst the VMs and the large set of input factors for various classes of VMs, we have applied back propagation-based artificial neural network [22]. It is a supervised machine learning technique used to predict the performance interference which is otherwise difficult to estimate in our complex model.

B. System Architecture

The architectural diagram of our proposed system hALT is depicted in Figure 3. hALT comprises three main components: (1) virtual machine classifier, (2) back propagation neural network, and (3) decision maker for placement. hALT utilizes CPU usage, memory usage, and performance information of the VMs as inputs to the virtual machine classifier component. The virtual machine classifier classifies VMs into specific classes by employing the k-means algorithm and the silhouette method. These classes of VMs are then used by the back propagation neural network to extract best VM patterns, which lead to minimal performance interference on the host machines. After the neural network is trained, if and when a VM migration is requested, then hALT finds the aptly suited host machine which has the minimal performance interference. Details of each hALT component are explained below.

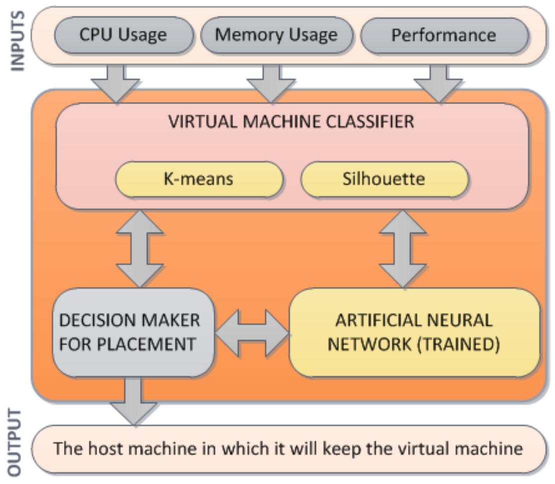
INPUTS
CPU Usage
Memory Usage
Performance
VIRTUAL MACHINE CLASSIFIER
K-means
Silhouette
DECISION MAKER FOR PLACEMENT
ARTIFICIAL NEURAL NETWORK (TRAINED)
OUTPUT
The host machine in which it will keep the virtual machine

Figure 3. hALT architectural diagram

C. Virtual Machine Classifier

The virtual machine classifier component classifies VMs based on their mean CPU, memory usage, and performance metrics. Only the tasks having CPU usage more than 25% are considered for the classifier since the performance metric we use is CPI and it is well correlated with the response times of compute-intensive applications [9].

Total number of distinct tasks we utilize for the classification is 1001 which are the representatives of compute-intensive tasks of cluster trace log. More than 12K samples of resource usage data of these tasks are used. To decide the best number of clusters, the silhouette method is employed for the cluster data we utilize. The higher the silhouette value is, the better the classification is. As seen in Table IV, the best cluster number for the dataset is found as 6 with a maximum mean silhouette value of 0.8051 over other cluster numbers.

Table IV
SILHOUETTE VALUES OF CLUSTERS

Number of Cluster	Mean Silhouette Value
3	0.7129
4	0.7427
5	0.7560
6	**0.8051**
7	0.7372
8	0.6577
9	0.6212
10	0.6170

D. Artificial Neural Network

hALT relies on the historic data to model and capture the relationships between input and output parameters to discover the patterns of VM combinations and the resulting degree of performance interference.

An artificial neural network (ANN) is trained to capture the relationships on how the different types and numbers of VMs impact performance interference. The input parameters for the ANN are as follows:

N_1 = Total number of VMs of *Class 1*
N_2 = Total number of VMs of *Class 2*
N_3 = Total number of VMs of *Class 3*
N_4 = Total number of VMs of *Class 4*
N_5 = Total number of VMs of *Class 5*
N_6 = Total number of VMs of *Class 6*
C = CPU overbooking ratio
M = Memory overbooking ratio

The ANN predicts the performance interference level. The performance interference level is the mean performance difference of a specified VM before and after a VM is migrated on a host machine. The reason to choose number of VMs of each class is to capture the relationships between the different VM combinations along with host machine resource utilization levels and discover the regularities in how these patterns affect the performance interference level on a host machine.

E. Decision Maker

When a VM placement request is made or if a VM must be migrated, the decision maker component is responsible to iterate over all the host machines in the cluster, run the trained ANN, and return the host machine info which will provide the lowest performance interference level. The VM can then be placed in the machine despite the cloud provider utilizing overbooking strategies.

V. CONCLUSION

This paper presented our preliminary work on a performance interference-aware virtual machine placement algorithm named hALT that is used as an online algorithm for VM placement to support the QoS requirements of soft real-time, cloud-hosted applications. The approach comprises two steps. First, a large, trace log of a production data center from Google is analyzed to glean away key insights into performance interference caused due to VM collocation. These insights are used in finding an aptly suited host machine for VMs to minimize the performance interference effects and reduce the performance degradation in soft real-time applications. To achieve this goal, a classification-based VM placement algorithm was designed by utilizing feed forward, back propagation neural network.

In this work, we considered only the compute-intensive applications because of the performance metric available in the cluster trace log. A more generic performance metric such as response time and throughput which might be representative of a wide range of applications in the cloud is planned as future work. Additionally, the middleware that provides a pluggable framework to utilize this algorithm is still in preliminary stages of development. To that end, we will conduct the experimental study at our in-house private cloud to precisely analyze only the performance interference effects of VMs to each other and convert it to a pluggable component. Additionally, analyzing hALT's energy efficiency effects in data center is left as future work.

Acknowledgments

This work was supported in part by the National Science Foundation CAREER CNS 0845789 and AFOSR DDDAS FA9550-13-1-0227. 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 and AFOSR.

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

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] F. Caglar, S. Shekhar, and A. Gokhale, "iPlace: An Intelligent and Tunable Power- and Performance-Aware Virtual Machine Placement Technique for Cloud-based Real-time Applications," in *17th IEEE Computer Society Symposium on Object/component/service-oriented real-time distributed Computing Technology (ISORC '14)*. Reno, NV, USA: IEEE, Jun. 2014.

[11] F. Caglar and A. Gokhale, "iOverbook: Managing Cloud-based Soft Real-time Applications in a Resource-Overbooked Data Center," in *The 7th IEEE International Conference on Cloud Computing (CLOUD' 14)*. Anchorage, AL, USA: IEEE, Jun. 2014, p. 10.

[12] F. Caglar, S. Shekhar, and A. Gokhale, "iTune: Engineering the Performance of Xen Hypervisor via Autonomous and Dynamic Scheduler Reconfiguration," in *Submitted to the 6th ACM/SPEC International Conference on Performance Engineering*. Austin, TX, USA: ACM/SPEC, 2015, p. 10.

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

[14] Q. Zhu and T. Tung, "A performance interference model for managing consolidated workloads in qos-aware clouds," in *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*. IEEE, 2012, pp. 170–179.

[15] R. C. Chiang and H. H. Huang, "Tracon: Interference-aware scheduling for data-intensive applications in virtualized environments," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2011, p. 47.

[16] M. Kambadur, T. Moseley, R. Hank, and M. A. Kim, "Measuring interference between live datacenter applications," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE Computer Society Press, 2012, p. 51.

[17] I. S. Moreno, R. Yang, J. Xu, and T. Wo, "Improved energy-efficiency in cloud datacenters with interference-aware virtual machine placement," in *Autonomous Decentralized Systems (ISADS), 2013 IEEE Eleventh International Symposium on*. IEEE, 2013, pp. 1–8.

[18] Y. Koh, R. Knauerhase, P. Brett, M. Bowman, Z. Wen, and C. Pu, "An analysis of performance interference effects in virtual environments," in *Performance Analysis of Systems & Software, 2007. ISPASS 2007. IEEE International Symposium on*. IEEE, 2007, pp. 200–209.

[19] (2013, Sep.) lxc linux containers. [Online]. Available: http://lxc.sourceforge.net/

[20] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[21] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[22] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *Neural Networks, 1989. IJCNN., International Joint Conference on*. IEEE, 1989, pp. 593–605.