Modeling VM Performance Interference with Fuzzy MIMO Model

Lixi Wang, Jing Xu, Ming Zhao
School of Computing and Information Sciences
Florida International University, Miami, FL, USA
{lwang007, jxu, mzhao}@fiu.edu

ABSTRACT

Virtual machines (VM) can be a powerful platform for multiplexing resources for applications workloads on demand in datacenters and cloud systems. However, it remains challenging for the resource management in such a virtualized system to deliver performance guarantees because of the contention on non-partitionable sources such as last-level CPU cache, memory bandwidth, and on-disk buffer, which introduces performance interference between co-hosted VMs and cannot be directly controlled. To address this challenge, this paper proposes to use fuzzy modeling to establish a multi-input-multi-output performance model for co-hosted VMs in order to capture their coupling. Based on the model, the level of contention on the competing resources is quantified by the model parameters, and can be used to guide VM placement and resource allocation decisions. Experimental evaluations on application benchmarks TPC-H and RUBiS demonstrate that this fuzzy modeling approach can detect and quantify the interference from VMs competing for both CPU and I/O; it can better capture the variation of such contention compared to a linear model.

1. INTRODUCTION

Emerging virtualized systems such as utility datacenters and clouds promise to be important new computing platforms where resources could be utilized efficiently. However, virtualization-based *high consolidation* presents a unique challenge in that virtualized applications interfere with each other's performance in dynamic and complex ways. In addition to the resources that can be controlled, consolidated virtual machines (VM) also compete for resources that cannot be well partitioned such as shared last level cache (LLC), memory bandwidth, and on-disk buffer. For example, as our previous work [16] and the related work [4] show, a VM's performance can vary substantially when the level of cache contention changes from the co-hosted VMs. Such contention makes it difficult to deliver application-desired performance guarantees in virtualized systems because one VM's performance is coupled with others on the same host and it can vary substantially even if the VM's allocation of partitionable resources is fixed.

Therefore, it is important to *identify* and *quantify* the contention on non-partitionable resources between VMs in order to optimize the resource management of a virtualized system. First, although there is no mechanism to directly control the contention generated from non-partitionable resources, the impact to application performance can be indirectly controlled through the allocations of the partitionable resources. For example, when CPU is highly utilized, there can be more contention on shared last-level CPU cache; when disk I/O is intensive, there can be more contention on the shared disk buffer. Second, a good understanding of non-partitionable resource contention can help optimize the VM placement decision in a way that the contention among co-located VMs is minimized and the performance of the entire virtualized system is maximized.

This paper proposes a new multi-input-multi-output (MIMO) fuzzy modeling approach to accurately *identify* and *quantify* the contention on non-partitionable resources between VMs. The proposed approach learns the relationship between multiple co-hosted VMs' resource allocations and the multiple applications' performance by establishing an MIMO model using a set of fuzzy rules. In such a fuzzy model, each rule characterizes a certain aspect of the system using a simple sub linear model, whereas with all the rules aggregated together it can efficiently capture complex VM behaviors and system dependencies without requiring any a priori knowledge.

This proposed approach is evaluated on Xen-based VM environments using typical online transaction (RUBiS [7]) and database benchmarks (TPC-H [17]). The result demonstrates that the fuzzy MIMO model is able to capture the interference between VMs which share the non-partitionable resources in terms of both CPU and I/O and further quantify the level of contention on such shared resources. Compared to a typical linear modeling approach, the fuzzy modeling approach can obtain up to 22% better accuracy in capturing performance interference and successfully reflect the variation in contention.

The rest of this paper is organized as follows. Section 2 describes the background and motivation. Section 3 discusses the proposed approach in details and Section 4 presents an experimental evaluation. Section 5 examines the related work and Section 6 concludes this paper.

2. BACKGROUND AND MOTIVATION

2.1 Resource Management in Virtualized System

One of the key tasks in a resource management solution for a virtualized system is to correctly understand the relationship between the resource allocations to VMs and the performance of hosted applications. With the understanding of such system behaviors, it can accurately allocate resource demand based on the application's Quality of Service (QoS) target and further effectively optimize the resource allocation across VMs based on resource-provider objectives. However, the major difficulty lies in the following aspects.

First, the challenge remains in allocating partitionable resource on an application's demand. The resources that can be well partitioned include CPU cycles, memory capacity, and I/O bandwidth. The dynamics in an application's workload can lead to complex behaviors in multi-types of resource consumptions by its hosting. Both the intensity and composition of the application workload may change over time. For instance, adjusting CPU allocations to a web workload's request rate depending on the time of day and the occurrence of events [15]; varying both CPU and I/O allocations to a database VM when the query workload

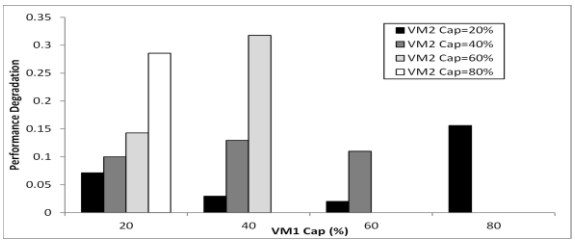

Figure 1. The performance interference of VM1 with CPU contention

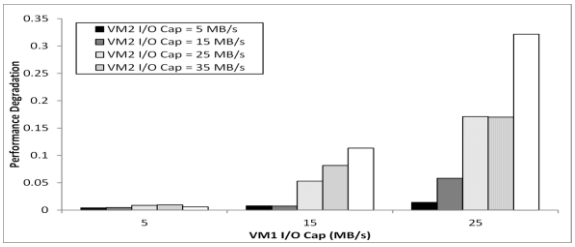

Figure 2. The performance interference of VM1 with I/O contention

changes in its composition of a wide variety of queries with different levels of CPU and I/O demands[11].

Second, the contention on non-partitionable resources, such as shared CPU cache, memory bandwidth, and disk buffer, causes interference among co-hosted VMs. It can also lead to complex and dynamic resource usage behaviors as they compete for various types of resources that cannot be strictly partitioned. For example, when co-hosted VMs compete for the shared LLC and memory bandwidth, the relationship between each VM's resource allocation and its application's performance is known to be nonlinear [4][5][14]. Such contention can also vary dynamically as the VM workload change over time. As a result, a VM's performance cannot be effectively predicted and managed, even if the partitionable resources can be well-controlled.

Our prior work has studied fuzzy-logic-based VM modeling approach and shown that it can accurately and efficiently capture the nonlinear behaviors from the aforementioned first aspect [10][11][18]. Fuzzy models that capture VMs' demands of partitionable resources for meeting their QoS targets are employed to guide resource allocations in a virtualized system. Due to the speed and adaptability of this approach, the fuzzy models can be used and updated at the same time based on the data observed online. To further extend such a fuzzy-modeling-based resource management approach, this paper addresses the aforementioned second aspect and proposes to use MIMO fuzzy modeling to capture the coupling behaviors of multiple co-hosted VMs caused by non-partitionable resource contention.

2.2 Motivating Example

In this section, we use concrete examples to provide evidence of how the interference from non-partitionable resources affects the performance of applications that compete for them.

In the first example, two CPU-intensive workloads derived from RUBiS benchmarks are running concurrently on two separate VMs that are competing for the same CPU core. Figure 1 shows the performance degradation of one of the applications by fixing the CPU allocation to its own VM but varying the CPU cap to another (w.r.t. the case that single VM runs alone without CPU cap). Note that the total amount of CPU allocation is kept under the total CPU capacity (100%). From the result we can see that the CPU cap of VM2 affects the application performance on VM1 significantly, e.g., the performance of VM1's application drops up to 32% as the amount of CPU resource given to the other VM increases up to 60% even if its own CPU allocation is fixed at 40%.

In the second example, two TPC-H based database workloads are running on two separate VMs that are sharing the same disk I/O buffer. Similarly, we control the I/O resources by varying the maximum I/O bandwidth available to individual VMs. Figure 2 demonstrates the performance of one VM affected by the I/O allocations set to the other VM. The result indicates that the contention on non-partitionable I/O resources can also lead to complex interference behaviors between the VMs that are competing for it. For example, the performance of VM1 at a low I/O cap (5MB/s) is almost not affected by the I/O allocations to VM2; whereas, the performance degradation can be up to 34% when the two I/O caps are with both high values.

Furthermore, the results from both CPU and I/O interference imply that the behaviors of VMs can be highly complicated when such resource contention is involved; and the dynamics in interference resulted from ever-changing degree of contention intensity cannot be described by any linear pattern.

3. APPROACH

3.1 Fuzzy MIMO Model

Based on the preliminary work [18] of fuzzy-logic-based single VM performance model which implicitly shows that a VM's performance can vary substantially when the level of cache contention changes from the co-hosted VMs, we propose to apply new multi-input-multi-output (MIMO) fuzzy model to simultaneously capture the more dynamic behaviors of multiple co-hosted VMs and their complex resource contention.

Consider a resource provider that hosts multiple applications by multiplexing multiple types of resources among them via VMs, a general *MIMO* model to build the time-varying relationship between resource allocations and application performance can be described in the following equation,

$$\boldsymbol{y}(t) = \Phi(\boldsymbol{u}(t), \ldots, \boldsymbol{u}(t-m), \boldsymbol{y}(t-1), \ldots, \boldsymbol{y}(t-n))$$

where the input vector $\mathbf{u}(t) = [u_1(t), u_2(t), \ldots, u_N(t)]^T$ represents the allocation of p types of controllable resources to the q applications' VMs at time step t ($N = pq$), and the output vector $\mathbf{y}(t) = [y_1(t), y_2(t), \ldots y_q(t)]^T$ is referred to as the predicted performance of q applications at time step t. For example, if there are two applications whose performance relies on two types of resources, i.e., CPU and disk I/O, then $\mathbf{u}(t)$ is a 4-dimensional vector, $[u_{CPU1}(t), u_{CPU2}(t), u_{IO1}(t), u_{IO2}(t)]^T$. Hence, function $\Phi(.)$ captures the mapping between the VM resource allocations to the application performance. m and n which indicate the dependence on the previous inputs and outputs to current estimation, are usually set to small values (e.g., m=0, n=1) in order to reduce the complexity of the model.

Traditionally, linear models are usually adopted to approximate the nonlinear behaviors around the current operating point. The general form of the model then can be described as $\boldsymbol{y}(t) = \boldsymbol{a}\mathbf{u}(t) + \mathbf{b}\boldsymbol{y}(t\text{-}1)$, where parameter matrices $\boldsymbol{a}$ and $\boldsymbol{b}$ can be trained using simple regression. In our proposed fuzzy MIMO model, the general function from the control inputs to the system outputs can

be instantiated by a collection of Takagi-Sugeno fuzzy rules [2] in the form of:

$$\boldsymbol{R}^i: If\ \boldsymbol{u}(t)\ is\ \boldsymbol{A}^i\ and\ \ \boldsymbol{y}(t-1) is\ \boldsymbol{B}^i,$$
$$then\ \boldsymbol{y}^i(t) = \boldsymbol{a_i u}(t) + \boldsymbol{b_i y}(t-1) \quad (1)$$

In the premise $\boldsymbol{A^i}$ and $\boldsymbol{B^i}$ are fuzzy sets associated with the fuzzy rule $\boldsymbol{R^i}$. Their corresponding membership functions μ_{Ai} and μ_{Bi} determine the membership grades of the control input vectors $\boldsymbol{u}(t)$ and $\boldsymbol{y}(t\text{-}1)$, respectively, which indicate the degree that they belong to the fuzzy sets. In the consequence, the output $\boldsymbol{y}(t)$ can be any function of the inputs. In most cases, it takes a linear form with trainable parameter matrices $\boldsymbol{a_i}$ and $\boldsymbol{b_i}$.

Both the model structure (the number of rules) and parameters (the parameters of each rule) are automatically trained from observed data. Each rule in a fuzzy model characterizes a certain aspect of the system using a simple linear model whereas with all the rules aggregated together the model can effectively capture nonlinear behaviors. More details discussed in the following subsections.

3.2 Model Creation

To build a concise rule base with a small number of fuzzy rules that can effectively represent the VMs' behaviors, one of the common algorithms adopted is subtractive clustering [6], an efficient one-pass clustering algorithm. Without pre-defining the number of rules and the fuzzy sets in each rule needed to model the system, it automatically exemplifies representative system characteristics by clustering the training data. Consequently, the number of clusters decides the size of rule base where each cluster is associated with the fuzzy set for input in the premise of the rule.

Such a model is created and updated online in a way that at the end of every interval, the current performance measurement $\boldsymbol{y}(t)$ with the corresponding resource allocations $\boldsymbol{u}(t)$ are collected and used as part of the training dataset for learning the fuzzy model for the next interval. Owing to the low computation overhead of subtractive clustering, the updating of the model can be completed quickly within a small control interval (e.g., 10s). Since the clustering technique is not restricted by the size of the training set, the model can be flexibly initialized based on only a few data points which represent a limited input-output space. For example, a model with only one fuzzy rule can be learned based on two data points, which represents a simple linear surface. However, the model graduadlly evolves into a more accurate one with more rules that reflect finer-grained system behaviors as the data size grows over time. During this process, model inaccuracy is quickly corrected as the data reflecting the system's actual behavior is used to immediately update the model.

3.3 Model Prediction

A well-trained fuzzy MIMO model is used to estimate the consequence performance output $\boldsymbol{y}(t)$ for a given control input $< \boldsymbol{u}(t),\ \boldsymbol{y}(t\text{-}1) >$. *Fuzzy inference* is applied to produce the prediction based on existing fuzzy rule base with S fuzzy rules. It entails the following steps: 1) Evaluation of antecedents: the input variables are *fuzzified* to the degree, $\boldsymbol{\delta^i}$, to which they belong to each of the fuzzy sets via the corresponding membership functions for each fuzzy rule $\boldsymbol{R^i}$;2) Implication to consequents: implication is performed on each fuzzy rule by computing $\boldsymbol{y^i}(t)$ based on the equation in the consequent of the rule; 3) Aggregation of consequents: the final prediction is performed as $\boldsymbol{y}(t) = \sum_{i=1}^{S} \boldsymbol{\delta^i y^i}(t)$, where the outputs $\boldsymbol{y^i}(t)$ of all the fuzzy rules are aggregated into a single numeric value based on their corresponding membership grades $\boldsymbol{\delta^i}$.

In a typical resource management system, the performance model is usually employed by controller to search for the best allocation schema that achieves certain global objective, e.g., optimizing the overall VM performance. In such a process, the model prediction is invoked iteratively by all possible allocation candidates in order to find the optimum across the multi-input space.

3.4 Interference Modeling

With the global knowledge of resource allocation solution to all VMs in the system, a MIMO model is able to capture the coupling between all the co-hosted VMs' resource allocations and performance when the contentions on both the partitionable and non-partitionable resources exist. The former can be well-addressed by keeping the sum of the required allocations under available resource capacity. The latter is not obvious but can still be implicitly reflected in a MIMO model. For example, if a linear model is used in a 2-VM system with only one type of resource involved, then it can be described in the following:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \quad (2)$$

If there is no contention on non-partitionable resources, there should be no coupling across different VMs in the MIMO model, and the non-diagonal elements of matrix $\boldsymbol{a}$ should have non-zero values only on the diagonal elements $\boldsymbol{a}_{ij}$= 0 where $i \neq j$. Otherwise, there will be non-zero values on the non-diagonal elements and their magnitude should represent the level of such interference. However, in such a linear MIMO model, the interference level can only be quantified by a constant factor $\boldsymbol{a}_{ij}$ which means the application performance is linearly affected by both its own allocation share and the others.

When using the proposed fuzzy modeling, it can describe complex interference behaviors more accurately. Since the MIMO model is defined by a set of fuzzy rules, where each rule $\boldsymbol{R^i}$ can define a sub linear model representing a constant coupling relationship using a matrix $\boldsymbol{a^i}$. The final matrix $\boldsymbol{a}$ is the weighted aggregation of all individual $\boldsymbol{a^i}$ from the rules base where the weights are the membership degrees of the input to the rules. For a given $\boldsymbol{u}(t)$,

$$\boldsymbol{a} = \sum_{i=1}^{S} \boldsymbol{\delta^i a^i}(\boldsymbol{u}(\boldsymbol{t})) \quad (3)$$

As $\boldsymbol{u}(t)$ changes, the degree of interference may also vary, which captures that as the allocations of partitionable resources changes, the interference from non-partitionable resources can vary. For example, when CPU is highly utilized, there can be more contention on shared last-level CPU cache; when disk I/O is intensive, there can be more contention on the shared disk buffer (as demonstrated in our motivation examples in Section 2.2). Therefore, the fuzzy model can not only identify such contention but also well capture its variable behavior, whereas the linear model cannot.

4. EVALUATION

4.1 Setup

This section evaluates our proposed FMPC-based modeling of VM interference using representative benchmarks hosted on a typical VM environment. The testbed is an Intel Core i7 physical machine, which has quad 3.4GHz CPUs, 4GB RAM, 8MB LLC and 500GB SATA disk storage. Each i7 core is also hyperthreaded, presenting two separate logical cores to the system software. The processor private and shared cache sizes are

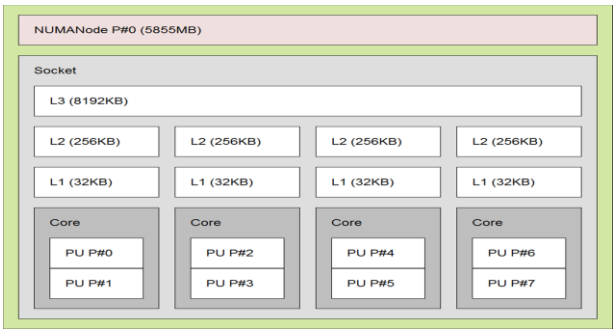

Figure 3 The architecture of Core i7-2600

illustrated in Figure 3. Xen 3.4 is installed to provide the VMs, where the operating system for both Dom0 and DomU VMs is CentOS Linux 5.4 with paravirtualized kernel 2.6.18. Each DomU VM is configured with 1 or 2 virtual CPUs and 1G RAM. The management of CPU allocation is done by setting CPU caps to VMs using Xen's Credit CPU scheduler [12] and the I/O allocation is done by Linux's dm-ioband controller [19].

Two benchmarks are used in our experiment. The RUBiS benchmark models a multi-tier online auction site that supports the core functionalities such as browsing, selling, and bidding [7]. To generate collocated workloads with high CPU contention, the multiple tiers of one RUBiS instance are deployed on the same DomU VM, including a web-tier of Apache Tomcat 4.1.40, a database-tier of MySQL 5.0 and 30 separate client sessions, each of which emulates the browsing behaviors of a single user on the website. The TPC-H benchmark which provides representative queries to compute complex business logic and involves the processing of large volumes of data is used to create I/O intensive workloads and create I/O contention on the shared physical disk.

Here we consider the case when the predicted performance is only dependent on the current resource allocation. So a general fuzzy rule of the performance model is revised as $\boldsymbol{R^i}$: $If\ \boldsymbol{u}(t)\ is\ \boldsymbol{A}_i$, $then\ \boldsymbol{y^i}(t) = \boldsymbol{a_i u}(t) + \boldsymbol{b_i}$.

A linear MIMO performance model, $\boldsymbol{y}(t) = \boldsymbol{a u}(t) + \boldsymbol{b}$, is built using linear regression based on the same set of training data used to train the fuzzy model. It is used as a baseline to compare to our fuzzy-modeling-based. The parameters for the linear model are estimated using the least squares method [8]. The fuzzy model is constructed by subtractive clustering where the parameters $\boldsymbol{a_i}$ and $\boldsymbol{b_i}$ in each fuzzy rule are trained by neuro-adaptive learning [9].

4.2 CPU Interference

The hyperthreaded Intel i7 processor allows us to create three different interesting CPU contention cases: 1) The VMs are hosted on the same logical core and compete for the shared resources (pipeline, private caches and LLC) of a logical core; 2) The VMs are hosted on a pair of logical cores presented by the same physical core, so they compete for the shared resources of a physical core; 3) The VMs are hosted on a pair of logical cores from different physical cores and compete for only the shared LLC (and memory bandwidth). In the subsection, we evaluate the level of contention for all these three cases.

In the first experiment, we evaluate the effectiveness of fuzzy modeling in describing the VMs' performance interference from CPU contentions. The impact of sharing CPU LLC among VMs on hosting applications' performance is studied by running 2

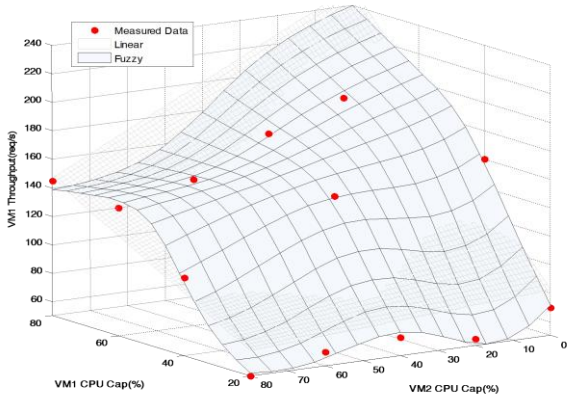

Figure 4 VM1's performance with contention on logical core

RUBiS VMs with equal workload intensity. Both VMs are pinned on the same logical CPU core and share for a total amount of 100% CPU. To create different degrees of contention, we keep the workloads concurrently running with constant intensity on both VMs but vary the CPU cap value set to each VM. For each setting, we keep them running for 15 control intervals (300s) and measure the throughput as performance metric in every interval. We collect the training data points for building the performance model, which is a set of control input and measured output pairs [u, y]. The input is the two cap values u = [cap1 cap2]^T chosen from a set of evenly distributed values in the range from 30% to 80%. The sum of CPU caps is kept under 100% in order to prevent from CPU oversubscribing so that we can observe the impact of contention on non-partitionable resources. The performance model is trained based on a total of 200 data points.

Figure 4 compares two 2-input-1-output performance models for one of the VM trained by fuzzy modeling and linear regression. From the results we can see the linear surface with a fitting error of 14.28% does not capture all data points as accurately as the fuzzy model which has a much smaller fitting error of 5.9%.

In the second experiment, we launch 3 VMs concurrently and let them compete for the same pair of logical cores. Two different settings are considered: the two logical cores sharing the same physical core, denoted as *same_core*; the two cores associated with separate physical cores, denoted as *diff_core*. Similarly, in both cases, we vary the CPU caps to all 3 VMs but keep the total CPU caps less than 200%. The control inputs for a performance model are the CPU allocations to all three VMs u = [u1 u2 u3]^T and the outputs are measured corresponding average throughputs collected from each VM at every control period y = [y1 y2 y3]^T.

To better understand the different contention level on non-partitionable CPU cache, we compare two linear performance models between the *same_core* case and the *diff_core* case. To better visualize their difference, we use two 3-D surfaces to compare the performance impact on VM1 from cap1 and cap3 in both cases. A similar comparison on the impact from cap1 and cap2 is omitted due to lack of space. It is observed that the performance of VM1 in the *diff_core* case has less dependence than in *same_core* on the allocations to other VMs. The results imply that the VMs running on the same physical core exhibit higher degree of coupling than the ones running on separate cores.

The fuzzy performance model is built based on the same training data, which consists of 14 fuzzy rules. For better illustrate the

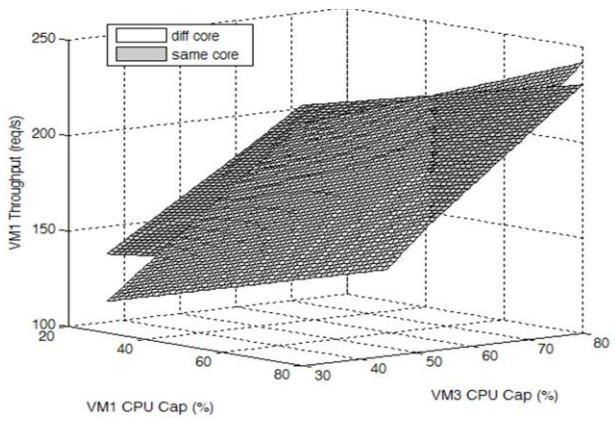

Figure 5 Performance comparison on interference from VM3

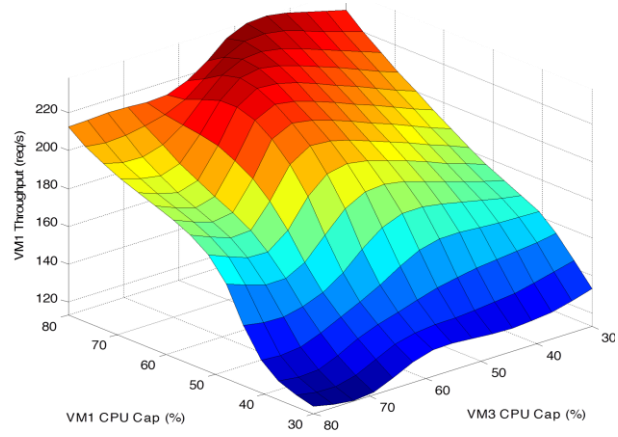

Figure 6. VM1's performance with contention on physical core when VM2's cap fixed

resulting fuzzy model, we extract a 3-D surface for VM1 in the *same_core* case, showing the behaviors of VM1 under the interference from VM3, as illustrated in Figure 6. A similar surface is observed for the interference from VM2 and obmitted here. From the non-linear surfaces, we can see that the performance of VM1 is not only dependent on its own CPU allocation but also affected by the CPU caps set to the other VM. With the same value of cap set to one VM, its application's performance will drop as the cap value of the other VM increases. It also shows that the degree of interference can vary as the allocations fall into different input regions. Similar surfaces generated from *diff_core* case are omitted due to the space limitation. Figure 7 extracts from a fuzzy model the degrees of VM1's performance interference from VM3 by calculating the weighted sum of degree of interference from all rules given a specific input, as shown in Eq. 3. Although the generated fuzzy models for *diff_core* and *same_core* share the same input space and produce the same number of fuzzy rules, the results in the figure demonstrate that the variation in interference behaviors can be also well reflected as the input falls into different region. In contrast, the linear model can only represent the interference by a constant factor, specifically, the interference from VM3 to VM1 is 0.19 and 0.63 for *diff_core* and *same_core* respectively.

4.3 I/O Interference

In this section, we study the performance interference from co-hosted VMs that are competing for disk I/O. We use two VMs each running a TPC-H instance. In order to control the disk I/O bandwidth of these VMs separately, each VM is launched from an

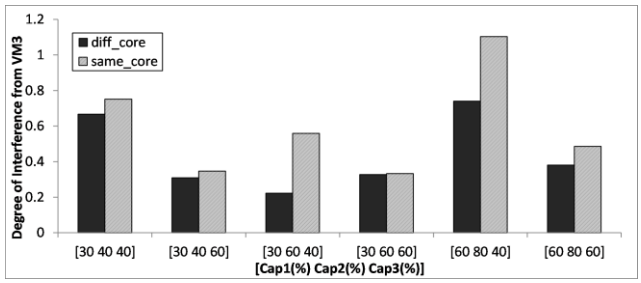

Figure 7 Interference degrees from CPU contention

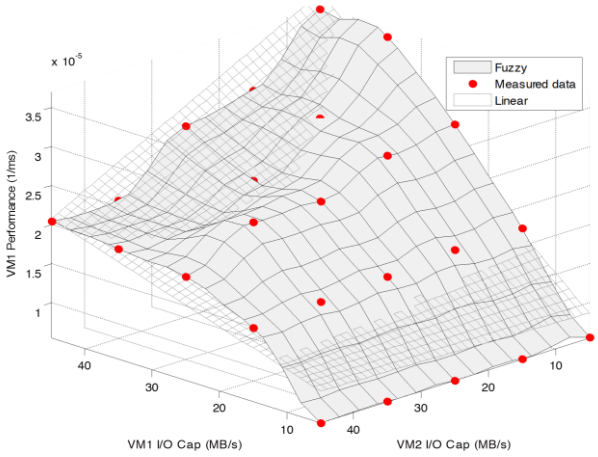

Figure 8. VM1's performance with I/O contention from VM2

image that is stored on a separate partition of the same physical disk. We run an I/O-intensive workload on both VMs using the TPC-H Q6 query which accesses a 200MB database table. Similarly, we keep the two I/O-intensive workloads concurrently running with constant intensity on these two VMs while varying the I/O bandwidth available to both VMs. Each run is performed with a different pair of the I/O caps set to both VMs ranging from 5MB/s to 45MB/s. We measure the execution time e_i of the workload on VM*i*. The input vector for training the performance model is the I/O allocations to both VMs, i.e., u = [cap1 cap2]^T, while the outputs y is a vector of $[1/e_1\ 1/e_2]^T$.

Based on a total of 50 training data points, a 2-input-2-output fuzzy model which includes 7 fuzzy rules is constructed with a small fitting error of 0.04%. Figure 8 illustrates the fuzzy model using a 3-D surface generated for one of the VM. From the resulting model, we can see that the input I/O allocations to VM1 and VM2 affect the VM1's performance in different manners. On one hand, VM1's performance is substantially improved as its own I/O allocation increases; on the other hand, the higher the I/O allocation to VM2, the more VM1's performance is affected, especially at higher values of both I/O caps where the contention on shared disk buffers is more intense. As a comparison, the linear model trained from the same dataset in Figure 8 can only reflect the rough trend in interference, but poorly capture the entire input data set with a fitting error of 22.6%.

4.4 Interference Prediction

In this section, we study how to apply the fuzzy model to predict the performance interference in order to guide VM placement for performance improvement. Assuming VM migration is worthwhile only if there is substantial performance improvement for the remaining VMs. In order to decide which VM and when to migrate, it is important to predict how much of the performance

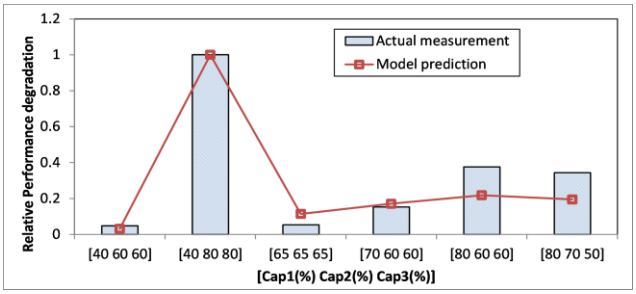

Figure 9 Relative performance degradation of VM1 when co-located with VM3 vs. with VM2

for a specific VM is affected by each of the other VMs. Then the migration strategy can be to move the VM whose interference degree to others is the highest or exceeds a given threshold.

We build a fuzzy model for 3 VMs initially running on the same host and competing for the same CPU, both VM1 and VM2 run the RUBiS workload, while VM3 runs a CPU-bound program that is not cache intensive. As another CPU becomes available, two placement schemes are considered: 1) *VM1-VM2*: co-locating the two RUBiS VMs while migrating VM3 out; 2) *VM1-VM3*: co-locating the two different VMs while migrating VM2 out. The performance impacts on VM1 using these two schemes are predicted by calculating the interference degrees of VM2 and VM3 to VM1 separately using the model (Eq. 3).

First, the model prediction shows that VM2 has larger interference to VM1 compared to VM3 for all given inputs because it causes more cache contention. We confirm this by comparing the actual performance of VM1 from the two schemes. Second, the predicted interference degree is also able to correctly quantify the level of performance impact. As illustrated in Figure 9, the degree of interference of VM3 to VM1 vs. VM2 to VM1 obtained from the model is consistent with the amount of actual performance degradation caused by co-locating VM3 with VM1 vs. VM2 with VM1. Hence, this knowledge is valuable to guide VM placement in order to minimize interference and optimize performance.

5. RELATED WORK

Various approaches have been studied in the literature to address the challenges of system modeling in resource management for virtualized systems. In particular, linear modeling approaches have been extensively applied to deal with the most common scenarios. For example, a linear MIMO [3] model is employed in a feedback control management system to capture interactions and dependencies among multi-tier applications sharing the common pool of resources; its extended work [1] builds the online linear ARMA model for each application when the application tiers are hosted on VMs spanning across physical nodes. In a model predictive control system, Nathuji *et al* apply a general linear function to capture the last-level cache interference between concurrent VMs and compensate its performance impact [4].

Other machine learning techniques have also been considered to automatically learn the complex resource model for a virtualized system based on data observed from the system. For example, the VCONF project has studied using reinforcement learning to automatically tune the CPU and memory configurations of a VM in order to achieve good performance for its hosted application [13]; Kund *et al.* employ artificial neural networks to build performance models that consider both resource allocation to VMs and resource interference between VMs [14].

In comparison, this paper focuses on using fuzzy modeling approach to build MIMO model for performance interference between multiple VMs competing for non-partitionable resources, which is shown to be more accurate in quantifying the level of resource contention and more flexibly in evolving model structure without any *prior* knowledge from the system.

6. CONCLUSION

This paper presents a new fuzzy modeling approach to establish a multi-input-multi-output performance model for co-hosted VMs in order to capture the coupling among VMs on shared non-partitionable resources. Based on the model, the level of contention on the competing resources is detected and quantified by the model parameters. Experimental evaluation using TPC-H and RUBiS benchmarks demonstrate that this fuzzy modeling approach can detect the interference from contention on both CPU and I/O non-partitionable resources and successfully capture the variation of such contention compared to a linear model. In future work, we will apply this approach to resource management of virtualized systems by optimizing VM placement and resource allocations based on the knowledge of non-partitionable resources contention.

7. REFERENCES

[1] P. Padala et al., "Automated Control of Multiple Virtualized Resources," Proceedings of EuroSys, 2009.

[2] T. Takagi, M. Sugeno, "Fuzzy identification of systems and its application to modeling and control," *TSMC*, 1985.

[3] X. Liu et al, "Optimal Multivariate Control for Differentiated Services on a Shared Hosting Platform," CDC, 2007.

[4] R.Nathuji et al., "Q-Clouds: Managing Performance Interference Effects for QoS-Aware Clouds," EuroSys, 2010.

[5] Diwaker Gupta et al., "Enforcing Performance Isolation Across Virtual Machines in Xen," Middleware, 2006.

[6] S. Chiu, "Fuzzy Model Identification Based on Cluster Estimation," Journal of Intelligent and Fuzzy Systems, 1994.

[7] C. Amza et al., "Specification and Implementation of Dynamic Web Site Benchmarks," WWC, 2002.

[8] K. Astrom, B. Wittenmark, "Adaptive Control," 1995.

[9] Neuro-adaptive Learning, URL: http://www.mathworks.com/help/toolbox/fuzzy/fp715dup12.html.

[10] Jing Xu et al., "Autonomic Resource Management in Virtualized Data Centers Using Fuzzy-logic-based Control", Cluster Computing, 2008.

[11] L. Wang, et al., "Fuzzy Modeling based Resource Management for Virtualized Database Systems," MASCOTS, 2011.

[12] Credit-Based CPU Scheduler, URL: http://wiki.xensource.com/xenwiki/CreditScheduler.

[13] J. Rao et al., "VCONF: A Reinforcement Learning Approach to Virtual Machines Auto-configuration", ICAC, 2009.

[14] S. Kundu et al., "Application Performance Modeling in a Virtualized Environment," HPCA, 2010.

[15] 1998 World Cup Web Site Access Logs, URL: http://ita.ee.lbl.gov/html/contrib/WorldCup.html.

[16] D. Arteaga et al., "Cooperative Virtual Machine Scheduling on Multi-core Multi-threading Systems — A Feasibility Study," MASVDC, 2010.

[17] TPC-H Benchmark Specification, http://www.tcp.org.

[18] L. Wang et al., "Adaptive Virtual Resource Management with Fuzzy Model Predictive Control," FeBID 2011.

[19] Dm-ioband: http://sourceforge.net/apps/trac/ioband/wiki/dm-ioband