2013 7th International Conference on Software Security and Reliability

A Performance Mapping Model for Physical-to-Virtual Migration

Borting Chen, Chien-Yu Lai, Yu-Lun Huang
Institute of Electrical and Control Engineering
National Chiao Tung University, Taiwan
Email: {bortingchen.ece98g, fronter.ece00g}@nctu.edu.tw, ylhuang@cn.nctu.edu.tw

***Abstract*—**
With the increasing popularity of cloud computing, hosting services on a remote virtual machine becomes a trend due to its ease of use and cost saving. When migrating services to a cloud, users need to submit a resource plan to ensure service's performance. However, it is difficult to estimate the resources required by a virtual machine for running arbitrary services. To address the problem, we propose an estimation model for a resource plan. The model takes the performance characteristics of a physical machine as input and estimates virtual machine configurations. Using the estimated configurations, a user can obtain a virtual machine having similar performance characteristics to that of the physical machine. In this paper, we derive a model to estimate the CPU capability for creating a virtual machine on Xen. Our experiments show that our model can provide a resource plan for a virtual machine, which has the minimum performance difference compared with a designated physical machine.

I. Introduction

Cloud computing is a new business model that enables users to rent computing resources at remote. It provides a shared pool of configurable computing resources that can be provisioned on demand and host a variety of services. According to NIST's definition [1], three fundamental service models are defined for cloud computing, which are Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). Among these service models, IaaS has been widely accepted due to its ease of use and cost saving. Many service providers, such as Amazon [2] and Microsoft [3], have deployed a public cloud serving IaaS model and bill on a virtual computing unit, called virtual machine.

A virtual machine offers fundamental computing resources as a physical machine, including CPU, memory, storage and network. Users are able to run arbitrary OSs and applications on a virtual machine to provide their services, such as web servers, virtual desktop [4] and network experiments [5]. In an IaaS cloud, users rent a virtual machine from the service provider, install OS and applications on the virtual machine, move data hosted on their original physical servers to the new platform and start their services. In order to run services properly, users are also required to submit a resource plan to the service provider to establish the computing resources they need. The service provider parses the resource plan and allocates resources for virtual machines. To simplify the resource allocation, most virtualization software provides interfaces for virtual machine configuration. For example, Xen [6] provides an interface to configure CPU capability of a virtual machine. The CPU capability informs CPU scheduler the amount of CPU cycles a virtual machine can consume. This helps an IaaS cloud service provider to allocate CPU resource for virtual machines and meet user's resource plan.

To accelerate the deployment of a virtual machine, some service providers offer physical-to-virtual tools (P2V) [7], [8] for data migration. The P2V tools decouple OSs, applications and data of a physical machine and convert them to a virtual machine hosted on a virtualization platform. The P2V tools simplify the process of conversion from a physical machine to a virtual machine; however, neither the hardware specification of the physical machine nor the resource requirements of applications are converted during the migration. Users need to manually set up a resource plan to ensure applications running on the virtual machine can obtain a desired performance.

Much recent work [9], [10], [11], [12], [13] addresses this problem by modeling the performance of applications running on a virtual machine. These work aims to find the relationship between resource allocation and the performance of applications such that users can predict the resource requirements and guarantee execution performance. Modeling the performance of applications is useful for the virtual machine running regular services, such as a web server. But, for the virtual machine assigned to execute arbitrary applications, such as a virtual desktop, it is hard to model the performance and set up a resource plan. Hence, some other research [14], [15] profiles user behaviors on a physical machine and transforms the profiling results to a resource plan. Such a transformation helps a user set up a virtual desktop which has a similar performance to that of the original platform. Nevertheless, the customized virtual machine loses its generality when running different applications or serving other users.

An alternative resource planning method is to make a virtual machine obtaining similar performance characteristics to that of a designated physical machine. Performance char-

978-0-7695-5021-3/13 $26.00 © 2013 IEEE
DOI 10.1109/SERE.2013.29

IEEE computer society

acteristics are measured by benchmark tools, which assess the performance of a particular hardware feature of a computer. The benchmark results are usually used for performance comparison or regarded as a reference when estimating the performance of applications running on a new platform. When two machines have similar performance characteristics, the applications running on the machines would obtain similar performance. We apply this concept to set up a resource plan for a virtual machine by taking a physical machine as a reference. We allocate resources for a virtual machine and guarantee its performance characteristics similar to a designated physical machine. Therefore, both machines would have similar performance on executing applications. Such a method enables a user to take his/her physical servers as a reference to set up a resource plan for a virtual machine. The virtual machine is able to run arbitrary services, and the execution performance can be estimated by running same services on user's physical machine.

In this paper, we propose an estimation model, called performance mapping model, to provide a resource plan suggestion when migrating services from a physical machine to a virtual machine. The model maps the performance characteristics of a physical machine to virtual machine configurations. Using the estimated configurations to create a virtual machine, the virtual machine can obtain similar performance characteristics to that of the designated physical machine; therefore, applications running on the virtual machine can obtain similar performance as running on the physical machine. Referring the estimation result to set up a resource plan, users can provide same service quality in the virtualized environment as well as in their physical servers. The contribution of this paper includes the following:

- We propose a performance mapping model to estimate the relationship between performance characteristics and virtual machine configurations. We use multiple linear regression to derive the model and adopt stepwise selection to find statistically significant variables for the model.
- We derive a model to plan CPU resources for a virtual machine. The model takes the performance characteristics measured by CPU benchmarks as input and estimates the CPU capability of a hardware-assisted virtual machine running on a Xen-based virtualization platform [6].
- We conduct several experiments to evaluate the accuracy of our model. The results show that our model helps find an optimal capability for creating a virtual machine which has the minimum performance difference compared with a designated physical machine.
- We also demonstrate that our performance mapping model can be adopted when mapping performance characteristics between virtual machines running on different virtualization platforms.

The paper is organized as follows. We first review previous work on resource planning in Section II. We discuss the challenges and the approach to constructing a performance mapping model in Section III. We derive the model using multiple linear regression in Section IV and evaluate the accuracy of our model in Section V. Section VI compares our work with other performance mapping methods, and discusses the scope of application as well as the limitation of our model. Finally, we conclude the paper in Section VII.

II. Related Work

Previous research on planning resources for a virtual machine can be classified into two categories: (1) modeling the performance of virtualized applications, and (2) analyzing user behaviors for a virtual desktop.

The first category, "modeling the performance of virtualized applications," aims to find the relationship between resource allocation and performance of applications running on a virtual machine. Wood et al. [9] designed a model to predict the resource usage of an application running in a virtualized environment. They adopted various benchmark tools to profile resource usages of applications, and applied multiple linear regression to derive the model from profiling results. In 2011, Huber et al. [10] proposed a model to predict the overheads for services running on a virtualization platform. They calculated the relative deviation values between a virtualization platform and a physical platform to estimate the computing resource requirements against virtualization overheads. Sudevalyam et al. [11] proposed a model to estimate the CPU usage of virtualized applications. They also considered the co-location problem to avoid virtual machines running on the same host affecting the performance of each other. Chan et al. [12] focused on modeling the relationship between response time and CPU workload. They collected CPU usage of a virtualization platform to predict the response times of applications. In 2012, Kundu et al. [13] used artificial neural network and support vector machine to derive the relationship between resource allocation and performance of virtualized applications. In order to handle non-linear data, they also proposed a sub-modeling technique to enhance accuracy.

The second category of resource planning methods analyzes user behaviors on a physical platform to plan resources for a virtual desktop. Beaty et al. [14] proposed a desktop-to-cloud transformation planning. They developed a profiling tool [16] to record the resource utilization of the processes executed on user's desktop. The utilization traces are replay in a virtualized environment to calculate the transformation ratio for capacity provisioning. Calyam et al. [15] proposed a model to minimize the resource usage of a cloud serving virtual desktops. They developed a virtual desktop performance benchmarking toolkit, called VDBench [17], to collect the resource usage of user's applications. The profiling results are then used to decide the creation of user desktop pools and determine an optimal resource allocation.

The above methods require users to profile the resource usage of a particular target in advance. However, for the use case that a virtual machine is assigned to run arbitrary services, neither modeling the performance of applications nor analyzing user behaviors could be done. An alternative approach is required to determine a resource plan for such a virtual machine, and provide a reference of application performance when running on that virtual machine.

III. Challenges and Approach

Given a physical machine, we aim to estimate the virtual machine configurations to make both machines have similar performance characteristics. Such an estimation faces two challenges:

- First, hardware differences between a physical machine and a virtualization platform may complicate the configuration estimation. Since the hardware features (such as CPU instruction sets, pipeline depth, and memory architecture) are diverse, the physical machine and the virtualization platform may have different performance on executing an operation. For example, a multiply-accumulate operation may take two clock cycles for addition and another one cycle for multiplication on one processor, where as another processor may complete the same work in two cycles. It is hard to find a performance relationship between different hardware features.
- Second, virtualization overheads increase the difficulty to estimate virtual machine configurations. Virtualization overheads are resulted from the implementation of virtual machine monitor or hypervisor. Some operations of a virtual machine are modified in order to run multiple virtual machines on a single host. For example, some privilege instructions are trapped and handled specially in the implementation of hardware-assisted virtualization. These instructions take a more complex execution path and cause a longer execution time compared with that on a non-virtualized platform. Neglecting virtualization overheads during estimation may cause a virtual machine obtain insufficient resources and result in a downgraded performance.

To counter these challenges, we propose a performance mapping model to estimate configurations of a virtual machine. The model takes performance characteristics of a physical machine as input. Each performance characteristic represents a particular hardware feature of the physical machine such that the features of different platforms become comparable. The output of the model is virtual machine configurations. Using the estimated configurations to create a virtual machine, the virtual machine would obtain similar performance characteristics to that of the designated physical machine. Users can refer to the estimation results to set up their resource plan.

We use multiple linear regression to derive the relationship between performance characteristics and virtual machine configurations. The data used in derivation are gathered from the virtual machines in order to consider the effect of virtualization overheads. The model construction includes the following steps:

1) On a given virtualization platform, we create several virtual machines and set each of them with different configurations. Then, we execute benchmarks on each virtual machine to measure the performance characteristics.
2) We collect the result pairs (performance characteristics versus configurations) as observed data and adopt stepwise selection to find statistically significant variables for the performance mapping model.
3) According to the selection result, we remove some benchmarks in the observed data and apply multiple linear regression to derive the relationship between performance characteristics and virtual machine configurations.

After model construction, we use the same set of benchmarks to measure the performance characteristics of a designated physical machine. The benchmark results are then applied to the model to estimate virtual machine configurations.

In this paper, we focus on mapping the CPU performance characteristics to the CPU configuration of a Xen-based virtual machine. For simplicity, both the physical machine and the virtual machine we discussed only have a single-core processor. The configuration we desired to estimate is CPU capability, which informs CPU scheduler how many CPU cycles a virtual machine is able to consume. CPU capability is expressed in percentage of one CPU core. If CPU capability is set to 50, a virtual machine can get half execution cycles of a CPU core. In the next section, we discuss how to derive the relationship between CPU performance characteristics and CPU capability.

IV. Performance Mapping Model

We derive the performance mapping model using multiple linear regression. Given a virtualization platform, we run *N* benchmarks on a virtual machine to collect *M* observed data $\{ (P_1^1, P_2^1, ..., P_N^1, C^1), (P_1^2, P_2^2, ..., P_N^2, C^2), ..., (P_1^M, P_2^M, ..., P_N^M, C^M)\}$, where C^i denotes the CPU capability configured in the i^{th} test ($1 \leq i \leq M$) and P_j^i denotes the result of the j^{th} benchmark ($1 \leq j \leq N$) in the i^{th} test. Using the observed data, we can form a set of linear equations as following:

$$\begin{aligned} C^1 &= b_0 + b_1 \times P_1^1 + b_2 \times P_2^1 + ... + b_N \times P_N^1 + \varepsilon^1 \\ C^2 &= b_0 + b_1 \times P_1^2 + b_2 \times P_2^2 + ... + b_N \times P_N^2 + \varepsilon^2 \\ &... \\ C^M &= b_0 + b_1 \times P_1^M + b_2 \times P_2^M + ... + b_N \times P_N^M + \varepsilon^M \end{aligned} \quad (1)$$

The coefficient set, b_0, b_1, ..., b_N, describes the estimation from CPU performance characteristics to CPU capability. The ε^i is error term.

Let β_0, β_1, ..., β_N denote the approximate solution for Eq. 1. We can solve β_j using "Least Square Error" to minimize the total sum of squares errors. Hence, the estimated capability $\hat{C}$ can be derived using Eq. 2:

$$\begin{aligned}\hat{C} &= \beta_0 + \beta_1 \times P_1 + \beta_2 \times P_2 + ... + \beta_N \times P_N \\ &= \beta_0 + \sum_{j=1}^{N} \beta_j \times P_j \end{aligned} \quad (2)$$

In order to avoid data overfitting, we adopt stepwise selection to find the statistically significant variables from the benchmark sets. (The detail steps of stepwise selection are explained in Section V-B.) Hence, the final performance mapping model is shown in Eq. 3:

$$\begin{aligned}\hat{C} &= \beta_0 + \beta_1 \times P'_1 + \beta_2 \times P'_2 + ... + \beta_N \times P'_{N'} \\ &= \beta_0 + \sum_{j=1}^{N'} \beta_j \times P'_j \end{aligned} \quad (3)$$

where $\bar{P'} = \{P'_1, P'_2, ..., P'_{N'}\}$ is a subset of $\bar{P} = \{P_1, P_2, ..., P_N\}$ and $0 < N' \leq N$. Only the variables chosen in stepwise selection are included in the final model. Note that stepwise selection may choose different variables for different virtualization platform since each platform has its special hardware features.

V. Experiments

In this section, we first introduce the CPU benchmarks for performance characteristic measurement. We then describe the virtualization platforms for conducting experiments and follow the steps described in Section IV to derive a performance mapping model. To evaluate model accuracy, we examine the performance difference between a physical machine and the virtual machine set with an estimated capability. We also evaluate the effectiveness of our models on mapping performance characteristics between virtual machines running on different virtualization platforms.

A. CPU Benchmarks

CPU benchmarks are usually classified into two categories: integer benchmarks and floating-point benchmarks. The former performs a variety of processor-intensive operations to measure integer performance, while the later executes bunches of floating-point operations to stress-test the floating-point unit (FPU) of a processor. We choose five integer benchmarks and two floating-point benchmarks to measure the CPU performance characteristics of a machine.

The five integer benchmarks are:

- **Sysbench** [18] (abbreviated as **Sys**) evaluates the OS parameters of a system running a database under intensive load. We execute the CPU test mode of **Sysbench** to measure the execution time of prime search.
- **Blowfish** (abbreviated as **Blow**) benchmark runs the Blowfish symmetric-key algorithm. It measures the execution time for encrypting and decrypting a text file.
- **Cryptohash** (abbreviated as **Crypt**) benchmark measures the performance of running cryptographic hash functions. It hashes a text file using MD5 and SHA1 algorithms and reports the execution time.
- **Fibonacci** (abbreviated as **Fibo**) benchmark measures the execution time of Fibonacci number calculation. It calculates the Fibonacci number by executing a recursive function rather than solving the Binet's Fibonacci number formula.
- **N-Queens** (abbreviated as **NQ**) benchmark reports the execution time for solving an N-Queen problem. It adopts a brute-force algorithm to find the solution.

and the two floating-point benchmarks are:

- **FFT** benchmark solves a linear equation using LUP decomposition and reports the execution time. The LUP decomposition executes a lot of basic floating-point operations, such as addition, multiplication and division.
- **Raytracing** (abbreviated as **Ray**) benchmark measures the execution time for solving optical ray tracing problem. It tests library performance as well as floating-point hardware support for trigonometric functions and square root.

All benchmarks, except **Sysbench**, are retrieved from the "hardinfo" benchmark package [19]. None of the benchmarks perform disk or network I/O during the measurement in order to isolate the measurement done to just the processor. We also change benchmarks' output format to inverse of time for unification.

B. Model Construction

We adopt two virtualization platforms in our experiments:

- *VP1*: IBM System X3200 M3 with 1 Intel Xeon X3430 2.40 GHz processor and 12 GB memory
- *VP2*: IBM System X3200 M3 with 1 Intel Xeon X3450 2.67 GHz processor and 10 GB memory

Both platforms support hardware-assisted virtualization and are installed with Linux kernel 3.2 and Xen 4.1.2.

To gather performance characteristics, we create a hardware-assisted virtual machine on *VP1* and *VP2*, and execute CPU benchmarks on the virtual machine. The virtual machine is configured with one virtual processor (vcpu) and 1 GB memory, and its OS is Linux kernel 3.2. We set CPU capability of the virtual machine to 25 in the beginning and execute the benchmarks. The capability of the virtual machine is incremented by 1 after executing each benchmark once until the capability reaches 100.

After collecting performance characteristics, we adopt stepwise selection to find statistically significant variables for *VP1's* and *VP2's* performance mapping models. The stepwise selection begins with no variables in the model and proceeds by adding or removing one variable step by step:

1) At the first step, we choose the variable with the smallest p-value into the model.

TABLE I
THE EXECUTION STEPS OF STEPWISE SELECTION

Virtualization Platform	Total Steps	Execution Steps						
		1	2	3	4	5	6	7
VP1	7	**+Blow**	**+Sys**	**+FFT**	**+Ray**	**+Crypt**	**+Fibo**	**–Ray**
VP2	5	**+Sys**	**+FFT**	**+Crypt**	**+Fibo**	**+NQ**	X	X

+: Add variable into the model
–: Remove variable out of the model

2) We calculate the p-value for each variable out of the model. For variables whose p-value smaller than a predefined threshold, called Significant Level to Enter (SLE), we add the variable with the smallest p-value into the model.
3) Then, we calculate the p-value for each variable in the model. For variables whose p-value larger than a predefined threshold, called Significant Level to Stay (SLS), we remove the variable with the largest p-value out of the model.
4) Repeat step 2 and step 3 until no variables can be added or removed.

We set both SLE and SLS to 0.05. The execution steps of stepwise selection are shown in Table I. We choose five benchmarks, **Blow**, **Sys**, **FFT**, **Crypt** and **Fibo** for *VP1's* model. The **Ray** benchmark is selected in the 4^{th} step, but removed at the last step due to its p-value becomes larger than SLS after adding **Fibo** benchmark to the model. For *VP2*, **Sys**, **FFT**, **Crypt**, **Fibo** and **NQ** are adopted to derive a performance mapping model.

C. Mapping Evaluation: Physical Machine

Now, we evaluate the effectiveness of our model on mapping CPU performance characteristics of a physical machine to CPU capability. Two experiments are conducted in our evaluation: (1) evaluating the effectiveness of stepwise selection, and (2) evaluating the accuracy of our models. We conduct the evaluation by comparing the performance difference between a physical machine and the virtual machine set with an estimated capability. We adopt two physical machines for evaluation:

- *PM1*: 1 Intel Core Solo U1500 1.33 GHz processor and 1 GB memory
- *PM2*: 1 Intel Pentium M 1.5 GHz processor and 1 GB memory

Both machines are installed with Linux kernel 3.2. On the physical machines, we run seven benchmarks mentioned in previous section to get a set of CPU performance characteristics, $\tilde{P} = \{\tilde{P}_1, \tilde{P}_2, ..., \tilde{P}_7\}$, and apply the benchmark results to *VP1's* and *VP2's* models to estimate CPU capability. We use the estimation results to create a virtual machine on *VP1* and *VP2*, and run benchmarks on the virtual machine to get a new set of performance characteristics, $\hat{P} = \{\hat{P}_1, \hat{P}_2, ..., \hat{P}_7\}$. The performance difference between a physical machine and a virtual machine is expressed using *Root Mean Square Percentage Error*, *RMSPE*, as shown in Eq 4:

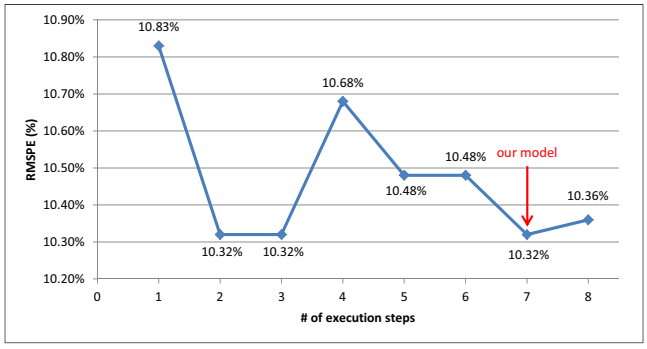
$$RMSPE = \frac{1}{7}\sqrt{\sum_{i=1}^{7}|\frac{\tilde{P}_i - \hat{P}_i}{\tilde{P}_i}|^2} \quad (4)$$

All benchmarks are included in the calculation in order to estimate the worst case of performance difference. A smaller RMSPE value means that the virtual machine has a smaller performance difference compared with the physical machine.

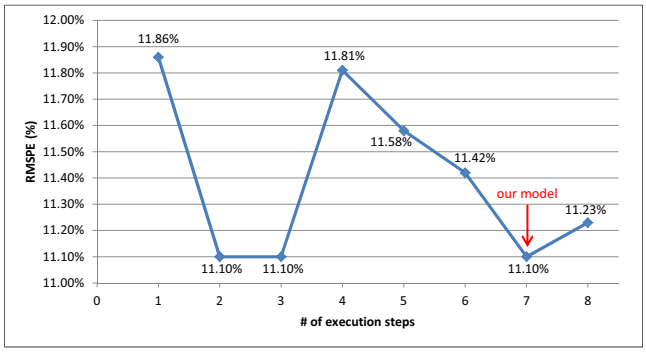
1) Effectiveness of Stepwise Selection: We first evaluate whether stepwise selection chooses the best set of variables for a performance mapping model. Following the execution steps shown in Table I, we derive a model for the variable sets selected in each step and use the models to estimate CPU capability for *PM1* and *PM2*. Besides, we also execute one more step when selecting variables for *VP1's* model and two more steps for *VP2's* model in order to evaluate the effect of adding more variables to our models. For *VP1's* model, since stepwise selection removes **Ray** benchmark at the 7^{th} step, we add both **Ray** and **NQ** benchmarks to the model at the 8^{th} step. For *VP2's* model, we add the variable with smaller p-value (**Blow** benchmark) to the model at the 6^{th} step and choose the rest (**Ray** benchmark) at the 7^{th} step. We use the estimated capabilities to create virtual machines and calculate the performance differences compared with *PM1* and *PM2*. The results are shown in Figure 1 and Figure 2.

From Figure 1, we observed that three models could be selected as *VP1's* model, which are models generated from 2^{nd} step, 3^{rd} step, and 7^{th} step (our model). Using the capabilities estimated by these models to create a virtual machine, the virtual machine can have the minimum performance difference compared with the physical machine. In the case of *VP2*, however, the virtual machine can obtain the minimum performance difference only when applying our model. As shown in Figure 2, using the capabilities estimated by other models could result in a larger performance difference. Such a result demonstrates that stepwise selection can effectively find the statistically significant variables for a performance mapping model.

2) Model Accuracy: In the second experiment, we examine the accuracy of our models by evaluating whether the

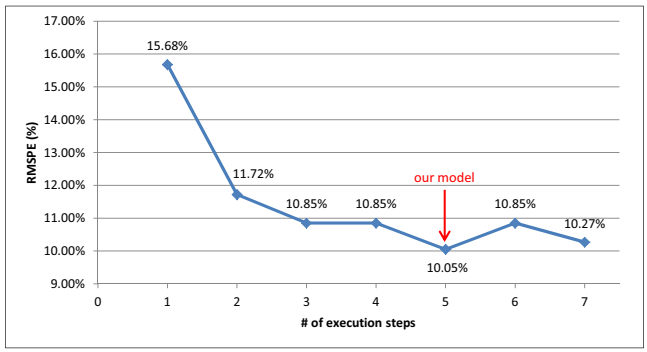


(a) Performance difference compared with *PM1*

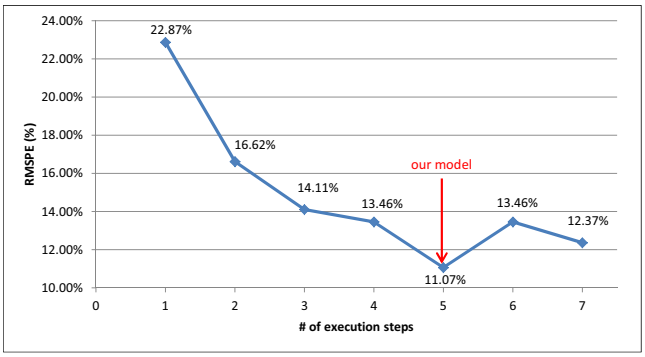


(b) Performance difference compared with *PM2*

Fig. 1. Effectiveness of stepwise selection for *VP1's* model



(a) Performance difference compared with *PM1*



(b) Performance difference compared with *PM2*

Fig. 2. Effectiveness of stepwise selection for *VP2's* model

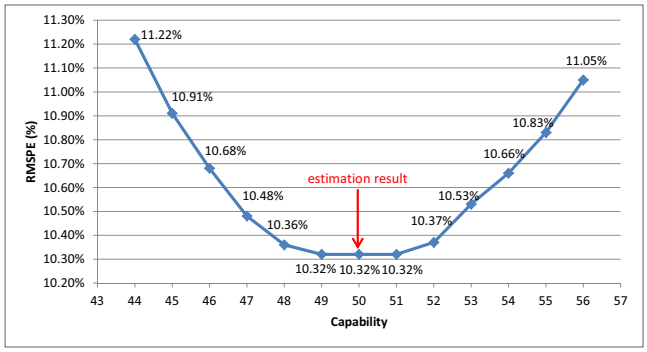
estimated capability is an optimal solution. A virtual machine with the optimal capability has the minimum performance difference compared with a designated physical machine. In this experiment, we first create a virtual machine and set its capability to the estimation result. Secondly, we choose some capabilities closed to the estimation result and use them to create virtual machines. Then, we compare the performance differences of these virtual machines and evaluate the optimal capability.

Figure 3 illustrates the performance differences of the virtual machines created on *VP1*. Using the capability estimated for *PM1*, the virtual machine created on *VP1* has a 10.32% performance difference compared with *PM1*. When mapping *PM2's* performance characteristics to *VP1*, the performance difference is about 11.10%. The capabilities estimated by *VP1's* model are the optimal solution for both physical machines.

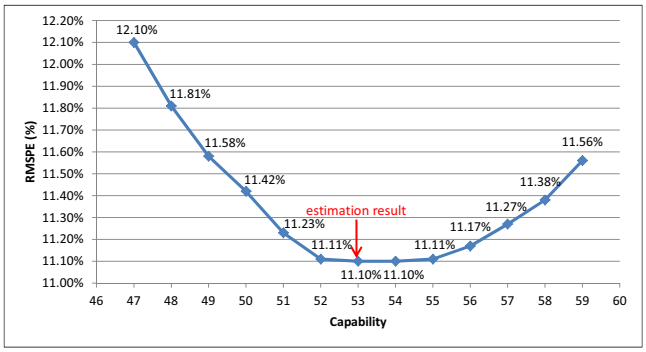
Investigating the cause of performance difference, we observe that **Ray** benchmark contributes the most to such a difference. Using the capability estimated for *PM1*, the virtual machine runs **Ray** benchmark 1.52 times faster than *PM1*. Setting capability to the estimation result, the virtual machine also runs **Ray** benchmark 1.36 times faster than *PM2*. Even we set the virtual machine with a relative low capability, e.g. 25, **Ray** benchmark still obtains a better performance. We argue that such a performance difference is resulted from the improvement of FPU design. The new FPU design accelerates the calculation of trigonometric functions so that **Ray** benchmark performs better on the virtual machine. In fact, without counting **Ray** benchmark, the performance differences compared with *PM1* and *PM2* are decreased to 6.63% and 8.69%. We conclude that *VP1's* model can map CPU performance characteristics of a physical machine to an optimal CPU capability. The virtual machine with the suggested capability has the minimum performance difference compared with the physical machine.

The performance difference between the physical machines and the virtual machines created on *VP2* is shown in Figure 4. Using our model to estimate the capability for *PM1*, the performance difference between *PM1* and the virtual machine is the minimum, which is about 10.05%. Same as the virtual machine created on *VP1*, the performance difference is mainly resulted from the improvement of FPU design. Without counting **Ray** benchmark, the performance difference is decreased to 6.42%.

When mapping *PM2's* performance characteristics to *VP2*,

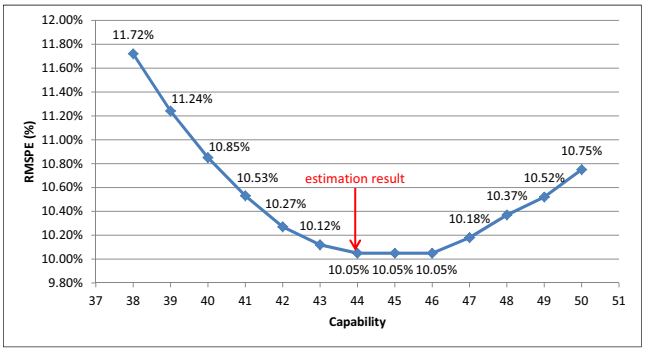


(a) Performance difference compared with *PM1*

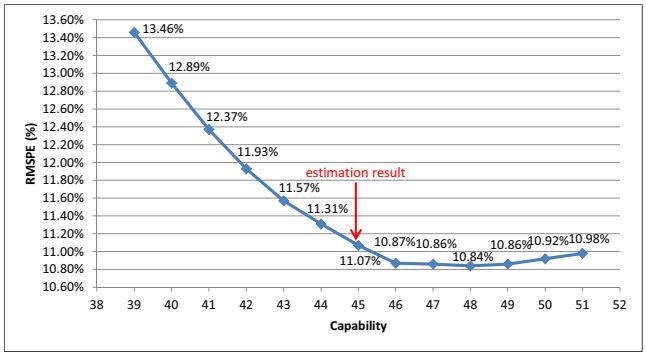


(b) Performance difference compared with *PM2*

Fig. 3. Accuracy of *VP1's* model



(a) Performance difference compared with *PM1*



(b) Performance difference compared with *PM2*

Fig. 4. Accuracy of *VP2's* model

the estimated capability, nevertheless, is not the optimal choice. The estimation error is mainly caused by the significant architecture difference between the processor of *PM2* and *VP2*. The processor of *PM2* is launched in 2003, but the processor of *VP2* is launched in 2009. The pipeline architecture and CPU instruction sets have changed a lot during these years. Although a large architecture difference existed between the processors, our model still finds a suboptimal solution for *PM2*. The difference of RMSPE between the virtual machine with the estimated capability and the virtual machine with the optimal capability is only 0.23%. Therefore, we still claim that: using *VP2's* model to estimate CPU capability for a physical machine, the virtual machine can obtain similar performance characteristics to that of the physical machine.

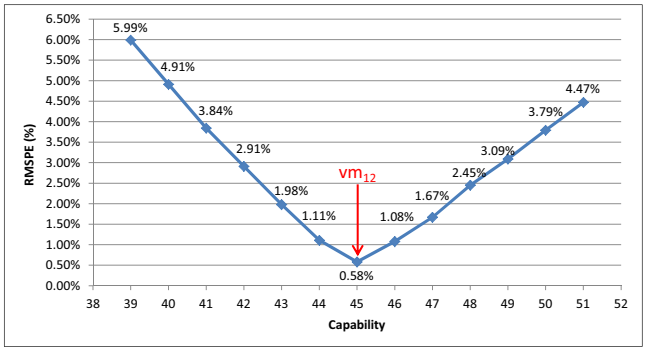
D. Mapping Evaluation: Virtual Machine

In addition to mapping performance characteristics of a physical machine, we evaluate the effectiveness of our model to map the performance characteristics of a virtual machine to the CPU capability of another virtual machine running on different virtualization platform. We create one virtual machine on *VP1* (called vm_{11}) and set its capability to 50. Then, we benchmark the performance characteristics of vm_{11} and apply the result to *VP2's* model to estimate CPU capability. The estimation result is used to create a virtual machine on *VP2*, called vm_{12}. We also create a virtual machine on *VP2* with capability set to 50 (called vm_{22}) and map $vm_{22}{}'s$ performance characteristics to *VP1* (the resulting virtual machine is called vm_{21}).

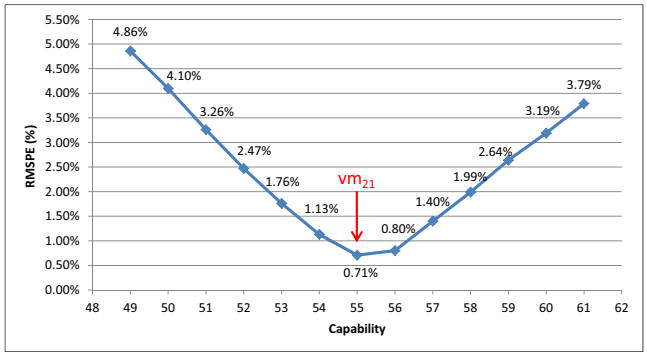
Figure 5 (a) plots the performance differences compared with vm_{11}. The results show that vm_{12} has the minimum performance difference (0.7%). A similar result is also observed when mapping $vm_{22}{}'s$ performance characteristics to *VP1*. Figure 5 (b) shows that the performance difference between vm_{22} and vm_{21} is also the minimum (0.58%). We attribute the small performance difference to the little architecture difference between the processor of *VP1* and *VP2*. We conclude that our models are also useful for mapping performance characteristics between virtual machines running on different virtualization platforms.

VI. Discussion

In this section, we make a comparison with other performance mapping methods to explain the reasons for using performance characteristics and multiple linear regression to



(a) Performance difference compared with vm_{11}



(b) Performance difference compared with vm_{22}

Fig. 5. Accuracy of mapping performance characteristics between virtual machines running on different virtualization platforms

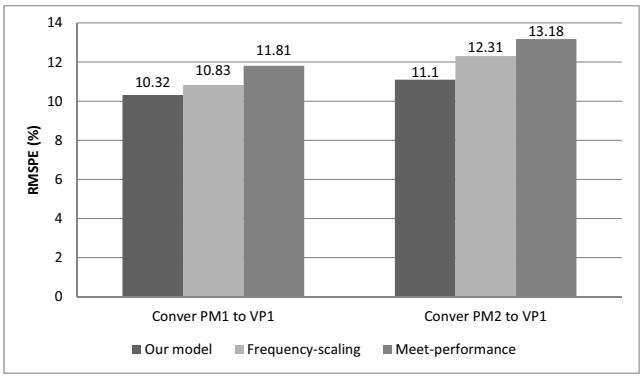


Fig. 6. Comparison of different performance mapping methods

derive the model. We also discuss the scope of application as well as the limitation of our model.

We compare our work with two intuitive performance mapping methods:

- The "meet-performance method" selects a sufficient capability to create a virtual machine such that all performance characteristics of the virtual machine can outperform that of a designated physical machine.
- The "frequency-scaling method" uses the ratio of CPU clock rate to determine capability. The capability can be derived by $100 \times \frac{f_{pm}}{f_{vp}}$, where f_{pm} and f_{vp} are the CPU clock rates of the physical machine and the virtualization platform, respectively.

Figure 6 illustrates the performance difference compared with *PM1* and *PM2* when creating a virtual machine on *VP1* using our model and the performance mapping methods mentioned above. Since different benchmarks require different capabilities, the meet-performance method has the largest performance difference compared with the others. Especially when mapping the performance characteristics of an old machine, the capability requirements are highly dependent on the CPU architecture of a virtualization platform. For example, some benchmarks (e.g. **Ray** benchmark) running on a virtual machine with a smaller capability can get better performance than those running on *PM2*. However, others, which get little benefit from the new processor architecture, require a larger capability to obtain same performance as those running on *PM2*. Meeting the highest capability requirement enlarges the performance difference between the virtual machine and the physical machine.

Using the frequency-scaling method to estimate capability also results in a larger performance difference compared with our method. Such an examination demystifies the myth that one can use the ratio of CPU clock rate to determine capability. The CPU clock rate does not dominate the performance of a processor. The pipeline depth, instruction sets and other hardware features which hidden from the CPU specification do actually influence the performance of a processor. Since the impacts of these hardware features are neglected, it is hard to find an optimal solution by applying the frequency-scaling method.

Our method adopts benchmarks to reveal the performance impacts of these hidden hardware features. The hardware features are transformed to numeric value so that we can use multiple linear regression to minimize the performance difference between a physical machine and a virtual machine. Although multiple benchmarks may assess the same hardware feature, we can use stepwise selection to eliminate the effect of duplicate measurement. In addition, virtualization overheads are considered during model construction. In order to mitigate the impact of virtualization overheads, the multiple linear regression uses the performance characteristics gathered from virtual machines to derive the model.

Our model helps users determine a resource plan when the goal is to make a virtual machine obtain similar performance to a physical machine. Such a method saves the time of virtual machine configuration from a trial-and-error process. Although our model may have a small estimation error when physical machine's processor is several generations earlier than that of the virtualization platform, our experiments show

that the estimation results still provide a good suggestion for users to set up a resource plan. Especially for the use case that the virtual machine is assigned to run arbitrary applications, such as a virtual desktop, our estimation can give a reference of application performance. Applications running on the virtual machine can obtain a similar performance as running on a designated physical machine.

For cloud service providers, our model enables a new service scenario where a user can configure CPU capability for a virtual machine. Users can leverage the benchmark tools and the model provided by service provider to determine a resource plan. Besides, our model is also useful for a cloud service provider which serves multiple virtual instance types across heterogeneous virtualization platforms, such as Amazon. Since our model can map performance characteristics between virtual machines running on different virtualization platforms, it saves the time from manually estimating the proper configuration for serving virtual instance types on a new platform.

Our model has a limitation that capability has a upper bound on estimation, 100. Setting CPU capability larger than 100 may cause a virtual machine be scheduled by two physical cores simultaneously, and make the performance become unpredictable. We plan to improve our model to estimate a capability larger than 100 in the future.

VII. Conclusion

In this paper, we propose a performance mapping model to estimate a resource plan when migrating services from a physical server to a virtualized environment. The model maps the performance characteristics of a physical machine to virtual machine configurations. Using the estimated configurations to create a virtual machine, the virtual machine can obtain similar performance characteristics to that of the physical machine. Such a method helps a user determine a resource plan for a virtual machine and provide a reference of application performance. We use multiple linear regression to derive a performance mapping model and find the relationship between CPU performance characteristics and CPU capability. Although multiple linear regression has been used in estimating the resource requirements of virtualized applications, our study extensively uses the regression model to map the resources of a whole physical system, not just applications, to the configurations of a virtual machine. Our experiments show that our model can estimate an optimal CPU capability so that the virtual machine can obtain a small performance difference compared with a designated physical machine. We also demonstrate that our model is applicable when mapping performance characteristics between virtual machines running on different virtualization platforms. In our future work, we plan further consider storage and network I/O in our model, and provide a more precise resource plan. We are also working on the capability estimation for a multi-core machine. By enhancing the support of performance mapping model, we believe that we can improve the usage of physical-to-virtual migration.

Acknowledgements

This effort was partially supported by the Taiwan Information Security Center (TWISC) Projects and Taiwan National Science Council under Grants NSC 101-2219-E-009-016, NSC 101-2221-E-009-075 and NSC 101-2219-E-009-027.

References

[1] P. Mell and T. Grance, "The NIST definition of cloud computing," National Institute of Standard and Technology, Tech. Rep., 2011.
[2] Amazon, "Elastic Compute Cloud," http://aws.amazon.com/ec2/.
[3] Microsoft, "Windows Azure," http://www.windowsazure.com/.
[4] T. Petrović and K. Fertalj, "Demystifying desktop virtualization," in Proceedings of the 9th WSEAS international conference on Applied computer science, 2009, pp. 241–246.
[5] A. Nakao, R. Ozaki, and Y. Nishida, "CoreLab: an emerging network testbed employing hosted virtual machine monitor," in Proceedings of the 4th ACM International Conference on emerging Networking EXperiments and Technologies, 2008, pp. 73:1–73:6.
[6] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in Proceedings of the 19th ACM symposium on Operating systems principles, 2003, pp. 164–177.
[7] VMware, "VMware vCenter Converter," http://www.vmware.com/products/converter/.
[8] Microsoft, "Disk2vhd," http://technet.microsoft.com/en-us/sysinternals/ee656415.aspx.
[9] T. Wood, L. Cherkasova, K. Ozonat, and P. Shenoy, "Profiling and modeling resource usage of virtualized applications," in Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware, 2008, pp. 366–387.
[10] N. Huber, M. von Quast, M. Hauck, and S. Kounev, "Evaluating and modeling virtualization performance overhead for cloud environments," in Proceedings of the 1st International Conference on Cloud Computing and Service Science, 2011.
[11] S. Sudevalayam and P. Kulkarni, "Affinity-aware modeling of cpu usage for provisioning virtualized applications," in Proceedings of the 4th IEEE International Conference on Cloud Computing, 2011, pp. 139–146.
[12] L. Chan, "Modeling virtualized application performance from hypervisor counters," Ph.D. dissertation, Massachusetts Institute of Technology, 2011.
[13] S. Kundu, R. Rangaswami, A. Gulati, M. Zhao, and K. Dutta, "Modeling virtualized applications using machine learning techniques," ACM SIGPLAN Notices, vol. 47, pp. 3–14, 2012.
[14] K. Beaty, A. Kochut, and H. Shaikh, "Desktop to cloud transformation planning," in Proceedings of the 23rd IEEE International Symposium on Parallel & Distributed Processing, 2009, pp. 1–8.
[15] P. Calyam, R. Patali, A. Berryman, A. Lai, and R. Ramnath, "Utility-directed resource allocation in virtual desktop clouds," Computer Networks, vol. 55, pp. 4112–4130, 2011.
[16] J. Rhee, A. Kochut, and K. Beaty, "Deskbench: flexible virtual desktop benchmarking toolkit," in Proceeding of 11th IFIP/IEEE International Symposium on Integrated Network Management, 2009, pp. 622–629.
[17] A. Berryman, P. Calyam, M. Honigford, and A. Lai, "VDBench: A Benchmarking Toolkit for Thin-client based Virtual Desktop Environments," in Proceedings of the 2nd IEEE Conference on Cloud Computing Technology and Science, 2010, pp. 480–487.
[18] A. Kopytov, "Sysbench: a system performance benchmark," http://sysbench.sourceforge.net/.
[19] "Hardinfo: system profiler and benchmark," http://hardinfo.berlios.de/.