# Performance Models for Virtualized Applications

Fabrício Benevenuto[1], César Fernandes[1], Matheus Santos[1], Virgílio Almeida[1], Jussara Almeida[1], G.(John) Janakiraman[2], José Renato Santos[2]

[1] Computer Science Department
Federal University of Minas Gerais - Brazil
{fabricio, cesar, mtcs, virgilio, jussara}@dcc.ufmg.br
[2] HP Labs
Palo Alto - USA
{john.janakiraman, joserenato.santos}@hp.com

**Abstract.** This paper develops a series of performance models for predicting performance of applications on virtualized systems. It introduces the main ideas of performance modeling and presents a complete case study of an application running on Linux that is migrated to a virtualized environment consisting of Linux and Xen. The paper describes the models, the process of obtaining measurements for the models and calculates performance metrics for the two environments. A validation of the results is also discussed in the paper.

## 1 Introduction

Virtualization can be understood as a technique to partition resources of a machine in multiple environments, creating a new level of indirection between physical resources and applications. Recently, virtualization technologies are experiencing a renewed interest as a way to improve system security, reliability, and availability, reduce costs, and provide flexibility. Particularly, such benefits are gaining popularity with Virtual Machines Monitors (VMM), providing software-based solutions for building shared hardware infrastructures.

Virtual machines provide a suitable environment to consolidate multiple services into few physical machines. Nevertheless, in order to migrate applications from physical machines to virtualized consolidated platforms, one needs to be able to estimate the performance these applications will achieve on the new environment. *Will migrated applications run with competitive performance as they run on their current environment? How many servers will be needed to create a virtual environment able to support the execution of the services provided, with acceptable performance? What is the best configuration of resources on the virtual environment for a certain application?* Therefore, there is a current need for new tools for predicting performance, providing information for resource allocation, and determining optimal system configuration. This work gives the first step in this direction. We propose simple queuing models for predicting the performance that applications, currently running on Linux system, will achieve if migrated to a Xen virtual system, with same hardware configuration. We further validate these models with experimental results.

* This work was developed in collaboration with HP Brazil P&D.

The rest of the paper is organized as follows. The next Section presents related work. Section 3 discuss background on performance modeling. Section 4 reviews the main architectural aspects of Xen, describes the tools and the experimental environment used for the performance evaluation. Section 5 discuss experimental results, whereas Section 6 presents analytic equations for virtual systems and compares analytical and experimental results. Finally, Section 7 concludes the paper and presents directions for future works.

## 2 Related Work

Three popular virtual systems are VMware [11], Denali [12], and Xen [1]. VMware is responsible for several products using different virtualization strategies. Denali was projected to support a large number of VMs, where each VM is able to execute only one application. Unlike Denali, Xen was designed to support VMs able to execute an entire operating system and thus, more than one application. The focus of our model and analysis is the Xen VMM, which is briefly described in Section 4.1.

The first studies and applications of virtualization emerged in 1960, when virtualization was described as an evolution of the study of time sharing and multiprogramming [13]. One of the most popular virtual environments at that time was the IBM VM/370. Bard et. al proposed analytical models for VM/370 to estimate performance metrics such as CPU utilization, and developed a performance predictor tool for this environment [14].

More recently, queuing network models for performance prediction of virtual environments were proposed in [7, 3]. However, the proposed models were not validated for any specific virtual architecture. In this work, we propose specific analytic models for the Xen architecture, validating these models with experimental results.

Recently, Cherkasova et al. [2] proposed a methodology to measure separately the CPU overhead on the IDD due to I/O operations of a guest VM. The idea is to count the cost and the number of page exchanges between a certain guest virtual machine and the IDD to estimate the amount of CPU usage of the IDD consumed in behalf of a certain virtual machine. Other studies also provided performance evaluation of applications running on Xen [8, 4]. In common, all these efforts are based solely on experimental evaluation of Xen and its applications.

## 3 Performance Models

A *model* is a representation of a system. Performance models [7] are useful for predicting the values of performance measures of a system from a set of values of workload, operating system, and hardware parameters. Performance prediction is the process of estimating performance measures of a computer system for a given set of parameters. Typical performance measures include response time, throughput, resource utilization, and resource queue length. The input parameters to such a model fall into one of three categories: workload, basic software, and hardware parameters. The workload parameters describe the load imposed on the system of interest by the applications, i.e., the transactions

submitted to it. The software parameters describe features of the basic software, such as the Xen virtual machine monitor overhead. Examples of such parameters are virtualization overhead, CPU dispatching priority, etc. Examples of hardware performance parameters include the components of the servers that supports a Xen system, such as processor speeds, disk latencies and transfer rates, and local area network speed. The output of a performance model is a set of performance measures, such as response times, throughput, and resource utilizations.

The emphasis of this paper is on how to build performance models for virtualized applications. We start out with simple models that provide bounds on some performance metrics. For example, one common question in the analysis of virtualized systems is "what is the maximum theoretical value of the arrival rate $\lambda$ for a given virtualized system?" This question has an easy answer that depends solely on the service demands of all resources. Note that the service demand $D_i$, the utilization $U_i$, and the arrival rate $\lambda$ are related by $\lambda = U_i/D_i$ for all resources $i$. Because the utilization of any resource cannot exceed 100%, we have that $\lambda \leq 1/D_i$ for all $i$'s. The maximum value of $\lambda$ is limited by the resource with the highest value of the service demand, called the bottleneck resource.Thus,

$$\lambda \leq \frac{1}{\max_{i=1}^{K} D_i} \tag{1}$$

In order to estimate more accurate performance metrics, models have to consider contention for resources and the queues that arise at each system resource—CPUs, disks, memory and communication devices. Queues also arise for software resources, such as threads, database locks, and protocol ports. The various queues that represent a virtualized system are interconnected, giving rise to a network of queues, called a queuing network (QN). A common question that could be answered by a queuing network model of a virtualized system is: "what will the average response time of application Y be when it is migrated from a pure Linux to a Linux/Xen environment that has the same system configuration?" To answer this question, we need models that present details of the system. The level of detail at which resources are depicted in a QN model of a virtualized system depends on the reasons to build the model and the availability of detailed information about the operation and availability of detailed parameters of specific resources. In other words, models depend on data collected by measurement tools in virtualized systems. The basic input parameters that we use in our performance models are service demands and arrival rates. Service demand is the sum of the service time at a resource (e.g. processor, disk, network) over all visits to that resource during the execution of a transaction or request. In a queuing network model, not all requests that flow through the resources of a system are similar in terms of the resources used and the time spent at each resource. Therefore, Workload may be broken down into several workload components, which are represented in a QN model by a *class* of requests. Different classes may have different service demand parameters and different workload intensity parameters. Classes of requests may be classified as *open* or *closed* depending on whether the number of requests in the QN is unbounded or fixed, respec-

tively. Open classes allow requests to arrive, go through the various resources, and leave the system. In this paper we initially present a simple model that estimates bounds on the performance of a virtualized system and then we develop an open-class queuing network model that represents applications running on Linux and Xen.

## 4 Case Study

In order to demonstrate the causes of virtualization overhead on the Xen VMM, we provide a performance evaluation of three benchmarks running on Xen and Linux. Then, we use a simple case study of a Web server to validate the performance models. The goal is to predict the performance a Web server application, currently running on a non-virtual system, will achieve if migrated to a Xen virtual machine. Our research strategy consists of a performance study of a simple Web server which provides only static content. In addition to this case study, we presented some results to discuss which components of the Xen environment need to be considered in a model. Next, we briefly describe the Xen virtual machine monitor and Xen I/O model. Our discussion focuses on aspects which are more relevant for the performance evaluation presented. Then, we present the tools and the hardware platform used.

### 4.1 Xen Architecture

Xen is a free and open-source virtual machine monitor (VMM) which allows multiple (guest) operating system (OS) instances to run concurrently on a single physical machine. It uses paravirtualization, where the VMM exposes a virtual machine abstraction slightly different from the underlying hardware. The Xen system has multiple layers. The lowest and most privileged is called VMM. Each guest OS runs on a separate virtual machine called domain. The domains are scheduled by the VMM to make effective use of the available physical CPUs. Each application, running on a guest OS, accesses hardware devices via a special privileged virtual machine called *isolated driver domain (IDD)*, which owns specific hardware devices and run their I/O device drivers. All other domains (*guest domains* in Xen terminology) run a simple device driver which communicates with the driver domain to access the real hardware devices. Figure 4.1 provides an overview of Xen I/O model. The IDD can access directly the hardware devices it owns. However, a guest domain accesses the hardware devices indirectly through a virtual device connected to the IDD. The IDD maps through bridges or routing the physical interface to its virtual interface which communicates with the guest virtual interface. The guest domain exchanges requests and responses with the IDD over an I/O channel. In order to avoid copying, references to page-sized buffers are transferred over the I/O channel instead of the actual I/O data.

### 4.2 Monitoring Framework

In order to develop performance models, we need to be able to measure the virtualized system. We developed an application called *Xencpu* to measure CPU

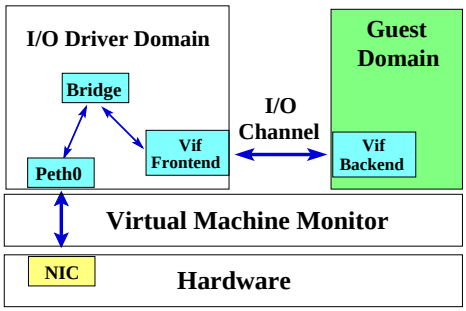



**Fig. 1.** Overview of Xen I/O model

busy time on Xen. This tool is based on the source code of *xm top* tool, provided with Xen, and was designed aiming at the automatic execution of our scripts. The CPU busy time on the Linux system is obtained based on information from */proc* directory. Disk busy time, on both Xen and Linux, was also obtained from the */proc* directory. Other parameters such as experiment duration and number of processed requests are obtained with scripts or from the benchmarks used.

### 4.3 Workload

The workload used is generated by the following benchmarks:
**- Web server**: we used httperf [9] as clients and Apache [10] version 2.0.55 as the Web server. The httperf is a tool which allows generating several HTTP workloads and measuring the performance of the Web server from the point of view of the clients. We run httperf on a client machine, sending requests with rate $\lambda$ to the server, measuring throughput and server response time of the requests. The workload used is part of a set of workloads from SPECWeb99 [7] and it does not contain dynamic content.
**- Disk intensive application**: This benchmark consists of copying a 2 GB file from a directory to another, on the same partition. We use this benchmark to analyze the impact of disk activity on the virtual environment overhead.
**- CPU intensive application** It consists on a kernel compilation, which evocates several functions, stressing system CPU.

### 4.4 Experimental Setup

For all experiments, we use a two 64-bit CPU 3.2 GHz Intel Xeon server, with 2 GB of RAM, one disk with 7200 RPM and 8 MB of cache, and two Broadcom Realtek gigabit Ethernet card. The server is directly connected to a client machine, an one-CPU AMD Athlon64 3 GHz, with 2 GB of RAM, and two Realtek gigabit Ethernet card. We configure Xen to use i386 architecture. We use Xen version 3.0 and the virtual machine runs XenoLinux derived from Linux Debian Etch i386 distribution, kernel 2.6.12. The client machine also uses the same Linux distribution and kernel. The virtual machine is configured to use 512 MB of RAM as well as the IDD. The *Borrowed Virtual Time* (BVT) is used as the Xen scheduler. To provide a fair comparison between Linux and the Xen VM, we also use 512 MB of RAM on Linux.

## 5 Performance Evaluation

This Section presents an experimental performance study of applications running on the Xen VMM. Unlike previous work [2, 8, 4], the focus of our analysis is on collecting metrics to support the design and validation (next Section) of models for performance prediction. We configure the virtual environment with one IDD and one guest, each one using a different CPU. The Linux system uses two SMP CPUs. Each result is an average of 20 runs. With a confidence level of 90%, the results differ from the mean by 10% at maximum.

In order to demonstrate the causes of virtualization overhead on the Xen VMM, we compare the three benchmarks described in Section 4.3 running on Linux and Xen systems. Figure 2 shows a kiviat representation which compares CPU and disk utilization on Linux and Xen systems for the three benchmarks. In a kiviat graph each radial line, which starts on the central point 0, represents one metric with maximum value 1 [5]. We plot six metrics on this graph, namely, the CPU utilization on the two Linux CPUs, on the IDD and on the VM, and disk utilization on both systems. Each curve represents a benchmark. Observing the disk-bound benchmark curve, we see that disk utilization is 1 (i.e., 100%) for both systems, and the VM CPU utilization is slightly higher than the sum of the utilizations of the two CPUs on Linux. Note that there is a significant load on the IDD CPU, since it works as interface for the hardware to the VM. For the kernel compilation benchmark, the VM CPU utilization is 1, which is also the sum of the CPU utilizations of the two CPUs running Linux. Since this benchmark does not execute a representative number of I/O operations, the CPU utilization on the IDD is almost 0. Note that the disk activity is negligible for the Web server, but there is a considerable CPU processing on the IDD due to network I/O operations.

Based on these observations, we can conclude that the assignment of CPU resources to VMs and IDDs can affect critically system performance, since the IDD processing is significant for workloads which stress I/O operations. We discuss how to predict the performance of each of these components separately in the next Section. We focus the rest of our analysis on the Web server benchmark. Figure 2 shows that disk utilization is almost zero for this benchmark. Thus, we do not consider disk residence time in our experiments and models. Our Web server provides only static content and uses a hardware platform with two CPUs and one disk. The two processors work in parallel on the Linux system. On the other hand, in a Xen environment the two CPUs are used to process each request. In the virtual environment, we assign one CPU for the IDD and one CPU for the guest VM. The requests are processed by the Web server running on the guest virtual machine. The VM is not able to access directly network and disk, and thus, it uses the IDD to access them. Figure 5 summarizes the experimental and analytical results. The analytical results will be explained on the next Section.

Figures 5(a) and 5(b) show the measured CPU utilization and demand as a function of request rate. Note that CPU utilization increases linearly with request rate for both IDD and VM. Moreover, the CPU utilization consumed by the IDD represents a significant overhead and must be considered by our mod-

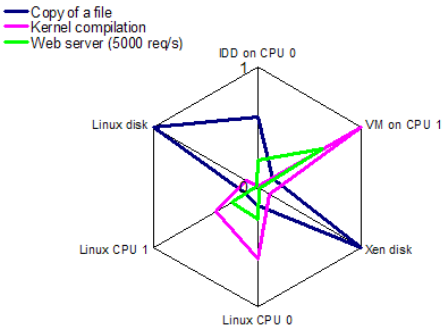



**Fig. 2.** Kiviat graph: comparison of CPU and disk utilization on Xen and Linux

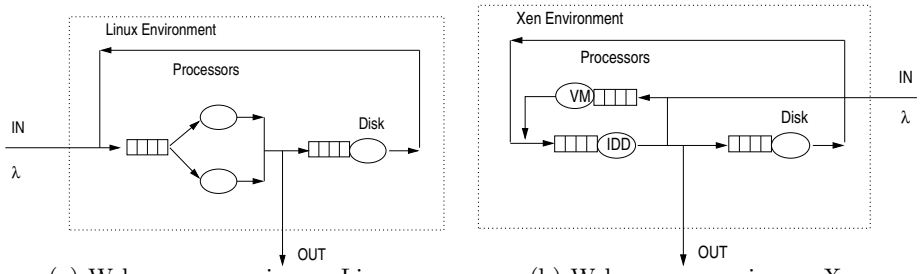
els. As we will further discuss, the ratio between the CPU demands (and thus CPU utilizations) for the IDD and the VM is fairly constant over the range of request rates we experiment with. Figures 5(c) shows average response time as a function of request rate. As one might expect, average response time is significantly longer at the virtual environment for request rates greater than 7000 requests per second, when the server starts to become overloaded. Clearly, the VM is the bottleneck in the Xen system, and performance degrades significantly with $\lambda \geq 7800$ requests/sec. From this point on, the overloaded server experiences significant delays in accepting connections from the client, which causes httperf not to send requests on the ratio we desire, limiting our experiment to this ratio. The Xen virtual environment is able to provide the same throughput of the Linux system, at the cost of a higher CPU utilization.

## 6 Performance Models for Xen-based Applications

This Section presents simple analytic models based on queueing network theory for applications running on the Xen virtual environment. Figures 3(a) and 3(b) present our model representations for the Linux system and the virtual environment using Xen VMM, respectively. These two systems are not equivalent, since in the Xen environment, the guest VM needs to use the IDD to access the hardware components. Note that Figure 3(b) represents the configuration of the XEN system used in *our experiments*, with IDD and VM running on separate independent CPUs. Nevertheless, we note that alternative system configurations, where either IDD or VM receives only a fraction of a CPU, would lead to different model representations. In the rest of this Section, we discuss asymptotic bounds and average value equations for open class queuing network models, validating these bounds with the experimental results discussed in Section 5.

### 6.1 Asymptotic Bounds

Asymptotic bound analysis can be quite useful to determine the maximum load a system can support while still providing reasonable response times (i.e., before saturation). The maximum rate, $\lambda_{max}$, the system can support without saturation is given by equation 1. As discussed previously, the maximum request




(a) Web server running on Linux (b) Web server running on Xen

**Fig. 3.** Queue network model representation for Linux and Xen environments

rate reached in our experiments is $\lambda = 7800$. Considering $D_{CPU}^{VM}$ as the CPU demand for the VM, equation 1 yields a $\lambda_{max} \leq \frac{1}{D_{CPU}^{VM}} = \frac{1}{0.000127} = 7874$, which is very close to the measured value.

### 6.2 Queuing Network Model

This Section presents an analytical model for an application on the Xen VMM. The strategy for building the performance model is to define a *slowdown factor* [7, 3] (i.e., $S_v$) of an application running on a virtual machine which is a function of the number of privileged instructions executed by the guest VM and of the number of instructions needed to emulate a privileged instruction. However, the fraction of privileged instructions executed by a VM and the average number of instructions required by the VMM to emulate a privileged instruction are not easy to be measured on a real system. We propose an equation to capture the overhead of virtualization of an application. The slowdown of a given application can be computed as the busy time of resource $k$ on the virtual environment, $B_k^{Virt}$, divided by $B_k$ the busy time of resource $k$ to execute the same application on an equivalent non-virtual system.

$$S_v = \frac{B_k^{Virt}}{B_k} \tag{2}$$

The slowdown can be interpreted as the overhead introduced by virtualization to execute a certain application. It means that applications running on virtual machines will see their CPU time increased $S_v$ times. $S_v$ will be used to calculate the input parameters (i.e., service demands) for the virtualized performance model. Queuing models will be used to predict the performance of applications migrating from non-virtual environments to virtualized environments.

As described in Section 4.1, the Xen architecture is divided in two different kinds of components: the IDD and the VMs. The Xen I/O model, also used by other virtual systems [12], use a page exchange mechanism between the IDD and the guest VM. In this context, the CPU time required by the IDD to process a package of 36 and 1448 bytes is the same if the number of page exchanges to deliver these packages are the same [2]. In order to compute the IDD CPU overhead (i.e., the CPU time demanded by the IDD) due to an application running on a guest VM we define $Cost_{VM_i}^{IDD}$ as:

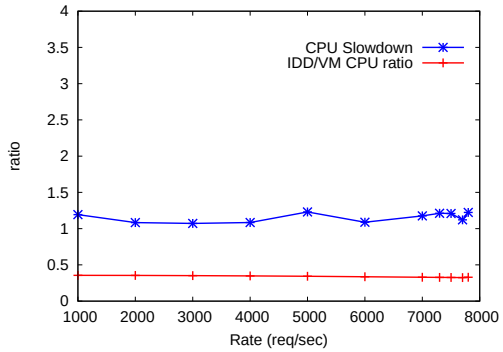



**Fig. 4.** Slowdown and $Cost_{VM_i}^{IDD}$ for the Web server benchmark

$$Cost_{VM_i}^{IDD} = \frac{B_{CPU}^{IDD}}{B_{CPU}^{VM_i}} \tag{3}$$

where $B_{CPU}^{IDD}$ is the portion of CPU time consumed on the IDD on behalf of the application running on the guest $VM_i$ and $B_{CPU}^{VM_i}$ is the CPU time consumed by the guest $VM_i$. In [2], it is discussed how to isolate the CPU cost of the IDD of a certain VM, even when running multiple VMs simultaneously. Since the virtualization overhead is divided in these two kinds of components (IDD and VMs), we consider $S_v$ as the overhead factor relative to the VM part of the virtualization slowdown. We estimate the IDD CPU utilization due to I/O operations of a certain VM, considering that $Cost_{VM_i}^{IDD}$ is constant relative to request rate, as supported by the experimental results shown in Figure 4.

Figure 4 displays the VM CPU slowdown and $Cost_{VM_i}^{IDD}$ as a function of $\lambda$. We compute $S_v$ based on busy time measured on the VM and the sum of the busy time of the two CPUs on Linux. The slowdown obtained is around 1.2. For practical use, a table of slowdown factor per class of applications can be built from experimental measurements. Note that we are comparing the sum of busy time of the two CPUs on Linux with only one CPU on the VM, and the slowdown factor does not consider the IDD CPU cost. The slowdown factor stems from several factors such as the emulation of TCP checksum, which is done by hardware on Linux. Note that the $Cost_{VM_i}^{IDD}$ for the system under study is also constant and around 0.34. Considering this evidence, we assume $Cost_{VM_i}^{IDD}$ is also a constant in our analysis. In Section 7, we discuss how to generalize the model to represent this part of the overhead for any type of application.

Based on equations (2) and (3) we can calculate the service demand for the virtual environment. Let $VM_1, VM_2, ..., VM_n$ be virtual machines that share the same hardware platform. The service demand at resource $k$ for the new environment, $D_k^{VM_i}$, is given by:

$$D_k^{VM_i} = D_k \frac{S_v}{P_{VM_i}} \tag{4}$$

where $D_k$ stands for the demand of resource $k$ for Linux running the same workload as $VM_i$, and $P_{VM_i}$ is the speedup of the hardware of virtual machine $VM_i$

compared to the non-virtualized hardware. Note that we do not use the same speedup for the IDD and VM, since these components can use different amounts of resources. For instance, we can configure a virtual environment assigning 1 CPU for the IDD and 3 CPUs for a certain VM. The utilization of resource $k$ on $VM_i$ can be obtained by the following equation:

$$U_k^{VM_i} = \lambda D_k^{VM_i} \tag{5}$$

The utilization can also be calculated based on the utilization measured on Linux, as we did for service demand. The $Cost_{VM_i}^{IDD}$ can be used to calculate the IDD demand for a certain class of applications as

$$D_{CPU}^{IDD} = D_{CPU}^{VM_i} \frac{Cost_{VM_i}^{IDD}}{P_{IDD}} \tag{6}$$

where $P_{IDD}$ is the speedup of the IDD relative to a baseline Xen configuration. Note that the speedup of the IDD is not relative to the non-virtual system. We introduced $P_{IDD}$ so that the model can predict resource allocation for the IDD.

Based on equations 4 and 5, the estimated residence time $R_k$ at resource $k$ for a open class system is given by:

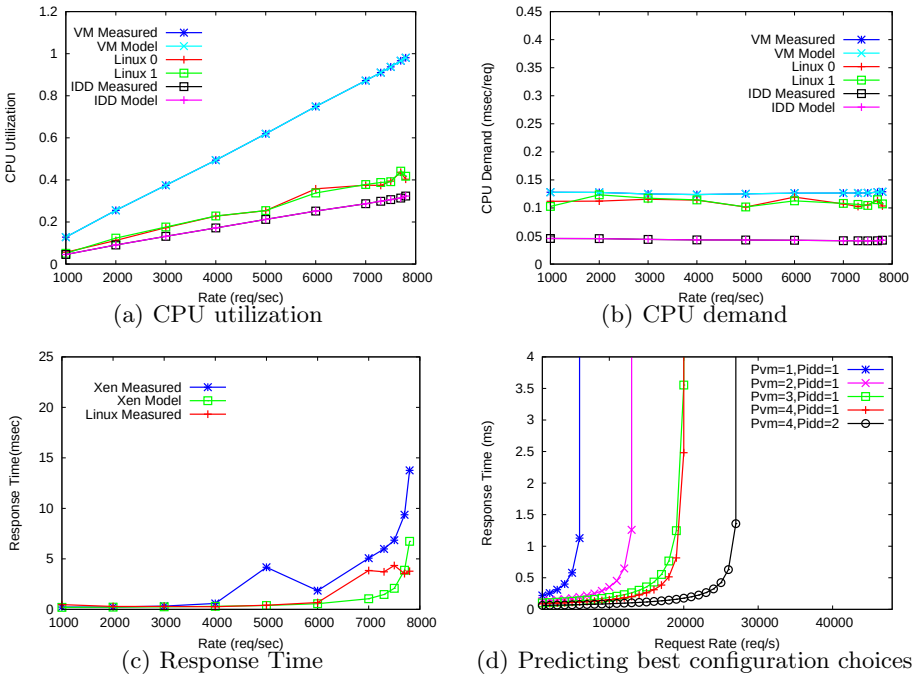
$$R_k = \frac{D_k^{Virt}}{1 - U_k^{Virt}} \tag{7}$$

The response time, $R^{Virt}$, on the virtual system can be obtained as the sum of the residence time at all resources:

$$R^{Virt} = R_{CPU}^{VM} + R_{CPU}^{IDD} + R_{Disk} \tag{8}$$

Figure 5 represents, on the same graphs, both the experimental and the analytical results. We use the values of $S_v$ and $Cost_{VM_i}^{IDD}$ presented on Figure 4 as parameters for the models. Both, CPU demand and utilization models for Xen IDD and VM are exactly the same as the ones obtained with experimental measurements since the $S_v$ and $Cost_{VM_i}^{IDD}$ factors were obtained with the same experiments. Figure 5(c) shows the calculated response time as a function of request ratio. When the request ratio is about to reach its upper bound, the response time start increasing quickly for both experimental and analytical results. The model response time is sub-estimating the measured response time, which was expected since there is a small network overhead and disk is not represented in the model. This result validates the proposed models and shows that they can be used to predict performance of an application running on a Xen VM.

In order to show another applicability of the models, consider a situation where one wants to know how powerful should be a hardware platform to support with quality of service high request ratios. Figure 5(d) exhibits the server response time as a function of the request rate for different speedups of the VM ($P_{VM}$), and different speedups of the IDD ($P_{IDD}$). We use the same slowdown factor and Linux demands of the experimental results for plotting the graph. As we increase the VM speedup the maximum request rate also increases. However, for $P_{VM} = 3$ and $P_{VM} = 4$ the maximal request rate is basically the same since the system bottleneck is the IDD. When the IDD speedup is increased, the maximum request rate is also increased. This example shows how models can




(a) CPU utilization (b) CPU demand

(c) Response Time (d) Predicting best configuration choices

**Fig. 5.** Analytical and experimental results for the Web server benchmark

be used to identify system bottleneck and consequently help to define the most adequate system configuration. Note that the virtual environment may need a high speedup compared to the non-virtual system to achieve the same performance. Increase in hardware cost may be counterbalanced by reduced costs in infrastructure management.

## 7 Conclusions and Future Work

This work proposes and validates simple analytic models to predict how applications will perform on Xen VMs, based on the performance of applications running on non-virtual environments. We envision two directions towards which our work can evolve. First, our assumption that $Cost_{VM_i}^{IDD}$ is constant, verified to be true for the web server benchmark in our experiments, may be relaxed. In other words, our models can be generalized to represent the overhead of the IDD performed on behalf of application programs as a special workload class in a multi-class queuing model [6]. In the case that $Cost_{VM_i}^{IDD}$ is not constant, the service demands of the special overhead class are load dependent. Second, we believe that our models can support the design of performance predictor tools as well as self-adaptive virtual systems. Figure 6 represents the design of a performance predictor tool where a database of benchmarks and their $S_v$ (and other parameters) are used to predict the performance of applications. The main idea is to obtain metrics from the non-virtual environment and based on previously established information, such as slowdown, to estimate the performance

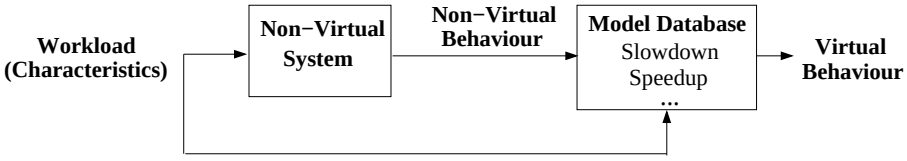



**Fig. 6.** Performance predictor tool architecture

of that application on the virtual environment. We plan to create such tool as a result of the consolidated models. Another interesting direction is to derive $S_v$ experimentally, capturing metrics from the non-virtualized environment.

## References

1. 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 *Proc. of 19th ACM Symposium on Operating Systems Principles*, Oct 2003.
2. L. Cherkasova and R. Gardner". "measuring CPU overhead for I/O processing in the Xen virtual machine monitor". In *Proc. of USENIX Annual Technical Conference*, Apr 2005.
3. D. Menascé. Virtualization: Concepts, Applications, and Performance. In *Proc. of The Computer Measurement Group's 2005 International Conference*, Orlando, FL, USA, Dec 2005.
4. D. Gupta, R. Gardner, and L. Cherkasova". XenMon: QoS Monitoring and Performance Profiling Tool. Technical Report HPL-2005-187, HP Labs, Oct 2005.
5. R. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley and Sons, INC, 1st edition, 1991.
6. D. Menasce, V. Almeida, and L. Dowdy. *Capacity Planning and Performance Modeling: From Mainframes to Client-Server Systems*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1994.
7. D. A. Menasce, L. W. Dowdy, and V. A. F. Almeida. *Performance by Design: Computer Capacity Planning By Example*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004.
8. A. Menon, J. R. Santos, Y. Turner, G. Janakiraman, and W. Zwaenepoel". Diagnosing Performance Overheads in the Xen Virtual Machine Environment. In *Proc. of First ACM/USENIX Conference on Virtual Execution Environments (VEE'05)*, Chicago, IL, Jun 2005.
9. D. Mosberger and T. Jin". httperf: A Tool for Measuring Web Server Performance. In *Proc. of First Workshop on Internet Server Performance*, pages 59—67, Madison, WI, Jun 1998.
10. A. W. Site. http://httpd.apache.org.
11. VMWare Web Site. http://www.vmware.com.
12. A. Whitaker, M. Shaw, and S. Gribble". Scale and Performance in the Denali Isolation Kernel. In *Proc. of Operating Systems Design and Implementation (OSDI)*, Dec 2002.
13. Y. Bard. Performance Analysis of Virtual Memory Time-Sharing Systems. *Proc. of IBM Systems Journal*, 14(4):366–384, 1975.
14. Y. Bard. An analytic Model of the VM/370 System. *Proc. of IBM Journal of Research and Development*, 22(5):498–508, Set 1978.