EVALUATING AND MODELING VIRTUALIZATION PERFORMANCE OVERHEAD FOR CLOUD ENVIRONMENTS

Nikolaus Huber, Marcel von Quast
Chair for Software Design and Quality, Karlsruhe Institute of Technology, Am Fasanengarten 5, Karlsruhe, Germany
nikolaus.huber@kit.edu, marcel.quast@student.kit.edu

Michael Hauck
Forschungzentrum Informatik FZI, Karlsruhe, Germany
hauck@fzi.de

Samuel Kounev
Chair for Software Design and Quality, Karlsruhe Institute of Technology, Am Fasanengarten 5, Karlsruhe, Germany
samuel.kounev@kit.edu

Keywords: Virtualization, Modeling, Benchmarking, Performance

Abstract: Due to trends like Cloud Computing and Green IT, virtualization technologies are gaining increasing importance. They promise energy and cost savings by sharing physical resources, thus making resource usage more efficient. However, resource sharing and other factors have direct effects on system performance, which are not yet well-understood. Hence, performance prediction and performance management of services deployed in virtualized environments like public and private Clouds is a challenging task. Because of the large variety of virtualization solutions, a generic approach to predict the performance overhead of services running on virtualization platforms is highly desirable. In this paper, we present experimental results on two popular state-of-the-art virtualization platforms, Citrix XenServer 5.5 and VMware ESX 4.0, as representatives of the two major hypervisor architectures. Based on these results, we propose a basic, generic performance prediction model for the two different types of hypervisor architectures. The target is to predict the performance overhead for executing services on virtualized platforms.

1 INTRODUCTION

In recent years, due to trends like Cloud Computing, Green IT and server consolidation, virtualization technologies are gaining increasing importance. Formerly used to multiplex scarce resources such as mainframes (Rosenblum and Garfinkel, 2005), nowadays virtualization is again used to run multiple virtual servers on a single shared infrastructure, thus increasing resource utilization, flexibility and centralized administration. Because this technology also allows sharing server resources on-demand, it promises cost savings and creates new business opportunities by providing new delivery models, e.g., Infrastructure as a Service or Software as a Service.

According to the International Data Corporation (IDC), 18% of all new servers shipped in the fourth quarter of 2009 were virtualized (IDC, 2010) and the server virtualization market is expected to grow 30% a year through 2013 (IT world, The IDG Network, 2008). However, the adoption of server virtualization comes at the cost of increased system complexity and dynamics. The increased complexity is caused by the introduction of virtual resources and the resulting gap between logical and physical resource allocations. The increased dynamics is caused by the lack of direct control over the underlying physical hardware and by the complex interactions between the applications and workloads sharing the physical infrastructure introducing new challenges in systems management.

Hosting enterprise services on virtualized platforms like Cloud environments requires an efficient performance management strategy at the application level. Service-Level Agreements (SLAs), e.g., performance guarantees such as service response time objectives, need to be respected. On the other hand, the target is to utilize server resources efficiently in order to save administration and energy costs. Thus, providers of virtualized platforms are faced with questions such as: What performance would a new service deployed on the virtualized infrastructure exhibit and how much resources should be allocated to it? How should the system configuration be adapted to avoid performance problems arising from changing customer workloads? In turn, customers using virtualized resources are interested in a service's performance behavior when, e.g., moving it to a Cloud Computing environment or when migrating it from one platform to another.

Answering such questions for distributed, non-virtualized execution environments is already a complex task (Menascé et al., 1994). In virtualized environments, this task is even more complicated because resources are shared. Moreover, since changes in the usage profiles of services may affect the entire infrastructure, capacity planning has to be performed continuously during operation. Proactive performance management, i.e., avoiding penalties by acting *before* performance SLAs are violated, requires predictions of the application-level performance under varying service workloads. Given that computation details are abstracted by an increasingly deep virtualization layer, the following research questions arise: i) What is the performance overhead when virtualizing execution environments? ii) Which are the most relevant factors that affect the performance of a virtual machine? iii) What are the differences in performance overhead on different virtualization platforms? iv) Can the performance-influencing factors be abstracted in a generic performance model?

Previous work on performance evaluation of virtualization platforms focuses mainly on comparisons of specific virtualization solutions and techniques, e.g., container-based virtualization versus full virtualization (Barham et al., 2003; Padala et al., 2007; Soltesz et al., 2007; Quétier et al., 2007). Other work like (Apparao et al., 2008; Tickoo et al., 2009; Iyer et al., 2009) investigates core and cache contention effects. (Koh et al., 2007) predict the performance inference of virtualized workloads by running benchmarks manually. (Huber et al., 2010) propose an approach on a systematic and automated experimental analysis of the performance-influencing factors of virtualization platforms applied to the Citrix XenServer 5.5.

In this paper, we use the automated experimental analysis approach from (Huber et al., 2010) as a basis. We extend this approach and evaluate its applicability to VMware ESX 4.0, another industry-standard platform with a different hypervisor architecture (Salsburg, 2007). The main goal is to build a generic model which enables the prediction of performance overheads on different virtualization platforms. To this end, we evaluate various performance-influencing factors like scheduling parameters, different workload types and their mutual influences, and scalability and overcommitment scenarios on the two different types of hypervisor architectures. At the same time, we evaluate the portability of automated experimental analysis approach to other platforms. Finally, we summarize the results of both case studies and formulate a basic generic model of the influences of various parameters on the performance of virtualized applications. This model shall provide the means for estimating the performance of a native application when migrated to a virtualized platform or between platforms of different hypervisor architectures. In addition, this model can be used for capacity planning, e.g., by Cloud providers, to estimate the number of virtual machines (VMs) which can be hosted.

The contributions of this paper are: i) an in-depth experimental analysis of the the state-of-the-art Citrix XenServer 5.5 virtualization platform covering performance-influencing factors like scheduling parameter, mutual influences of workload types etc., ii) an evaluation of these results on VMware ESX 4.0, another representative virtualization platform with a different hypervisor architecture, iii) an experience report on the migration of virtual machines and the automated administration of virtualization platforms, iv) a basic model capturing the general performance-influencing factors we have identified.

The remainder of this paper is organized as follows. Section 2 provides an overview of the automated experimental analysis we use. Additionally, it presents further experimental results on the Citrix XenServer 5.5. An evaluation of our results based on repeated experiments on VMware ESX 4.0 is given in Section 3. In Section 4, we present our performance prediction model. Section 5 discusses related work, followed by a conclusion and an outlook on future work in Section 6.

2 AUTOMATED EXPERIMENTAL ANALYSIS

Because virtualization introduces dynamics and increases flexibility, a variety of additional factors can influence the performance of virtualized systems. Therefore we need to automate the experiments and performance analysis as much as possible. In this section we give a brief summary of the generic approach to automated experimental analysis of virtualized platforms presented in (Huber et al., 2010). We briefly explain the experimental setup as well as the general process that is followed. Furthermore, we explain the experiment types in more detail because they are used as a basis for our evaluation. We also summarize the previous results of (Huber et al., 2010) and enrich them with more more fine-grained results and analyses in Section 2.5.

2.1 Experimental Setup

The experimental setup basically consists of a *MasterVM* and a *controller*. From a static point of view, the MasterVM serves as a template for creating multiple VM clones executing a benchmark of choice (see Section 2.4). It contains all desired benchmarks together with a set of scripts to control the benchmark execution (e.g., to schedule benchmark runs). A second major part is the controller which runs on a machine separated from the system under test. From a dynamic point of view, the controller clones, deletes, starts, and stops VMs via the virtualization layer's API. Furthermore, it is responsible for collecting, processing and visualizing the results. It also adjusts the configuration (e.g., the amount of virtual CPUs) of the MasterVM and the created clones as required by the considered type of experiment.

2.2 Experiment Types

Several types of experiments are executed, targeted at the following categories of influencing factors: (a) virtualization type, (b) resource management configuration, and (c) workload profile.

For category (a), an initial set of experiments is executed to quantify the performance overhead of the virtualization platform. The number of VMs and other resource management-related factors like core affinity or CPU scheduling parameters are part of category (b). The influence of these factors is investigated in two different scenarios, focused on *scalability* (in terms of number of co-located VMs), and *overcommitment* (in terms of allocating more resources than are actually available). For scalability, one increases the number of VMs until all available physical resources are used. For overcommitment, the number of VMs is increased beyond the amount of available resources. Finally, for category (c) a set of benchmarks is executed focusing on the different types of workloads. For a more detailed description of the experiment types as well as a benchmark evaluation, we refer to (Huber et al., 2010).

2.3 Experimental Environment

We conducted our experimental analysis in two different hardware environments described below. In each considered scenario Windows 2003 Server was the native and guest OS hosting the benchmark application, unless stated otherwise.

Environment 1: This environment is a standard desktop *HP Compaq dc5750* machine with an Athlon64 dual-core 4600+, 2.4 GHz. It has 4 GB DDR2-5300 of main memory, a 250 GB SATA HDD and a 10/100/1000-BaseT-Ethernet connection. The purpose of this environment was to conduct initial experiments for evaluating the overhead of the virtualization layer. This hardware was also used to run experiments on a single core of the CPU by deactivating the second core in the OS.

Environment 2: To evaluate the performance when scaling the number of VMs, a *SunFire* X4440 x64 Server was used. It has 4*2.4 GHz AMD Opteron 6 core processors with 3MB L2, 6MB L3 cache each, 128 GB DDR2-667 main memory, 8*300 GB of serial attached SCSI storage and 4*10/100/1000-BaseT-Ethernet connections.

2.4 Benchmark Selection

Basically, any type of benchmark can be used in the automated experimental analysis. Only the scripts to start and stop the benchmark and to extract the results must be provided. For CPU and memory-intensive workloads, two alternative benchmarks have been discussed in (Huber et al., 2010): Passmark PerformanceTest v7.0[1] (a benchmark used by VMware (VMware, 2007)) and SPEC CPU2006[2] (an industry standard CPU benchmark). Both benchmarks have a similar structure consisting of sub-benchmarks to calculate an overall metric. Benchmark evaluation results in (Huber et al., 2010) showed that both Passmark and SPEC CPU show similar results in terms of virtualization overhead. However, a SPEC CPU benchmark run can take several hours or even to complete. Since passmark has much shorter runs, the authors use Passmark in their experiments and repeat each benchmark run 200 times to obtain a more confident overall rating and to gain a picture of the variability of the results. In addition to Passmark, the *Iperf* benchmark[3] is used to measure the network performance. It is based on a client-server model and supports the throughput measurement of TCP and UDP data connections between both end-points.

2.5 Experiment Results

In (Huber et al., 2010), several benchmarks (CPU, memory, network I/O) were automatically executed

[1] Passmark PerformanceTest: http://www.passmark.com/products/pt.htm

[2] SPEC CPU2006: http://www.spec.org/cpu2006/

[3] Iperf: http://iperf.sourceforge.net/

to analyze the performance of native and virtualized systems. The results of a case study with Citrix XenServer 5.5 showed that the performance overhead for CPU virtualization is below 5% due to the hardware support. However, memory and network I/O virtualization overhead amounts up to 40% and 30%, respectively. Further experiments examined the performance overhead in scalability and overcommitment scenarios. The results for both the scalability and overcommitment experiments showed that the performance behavior of Citrix XenServer 5.5 meets the expectations and scales very well even for more than 100 VMs. Moreover, the measurements showed that performance loss can be reduced if one assigns the virtual CPUs to physical cores (called core pinning or core affinity). For a detailed discussion of the previous results, we refer to (Huber et al., 2010).

These previous findings are now extended by our more recent in-depth measurement results. At first, we compare the performance overhead of virtualization for different workload types. Second, we present performance overheads in scaled-up and overcommitment scenarios. Finally, we investigated the impact of network I/O in more detail.

Overhead of Virtualization In these experiments, we investigate the performance degradation for CPU, memory and I/O in more detail by looking at the sub-benchmark results of each benchmark metric. The new measurement results of Figure 1 depict the fine-grained sub-benchmark results for the Passmark CPU Mark results normalized to the native execution. The results demonstrate that floating point operations are more expensive (up to 20% performance drop for the `Physics` sub-benchmark) than the other sub-benchmark results. However, the overall performance drops are still in the range of 3% to 5%.

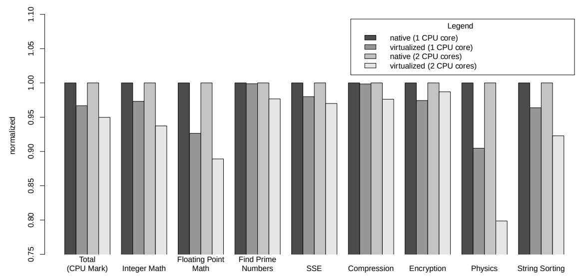

Figure 1: Sub-benchmark results of the Passmark CPU Mark metric.

Looking at the fine-grained memory benchmark results depicted in Figure 2, one can see that for the memory-intensive workloads the main cause for the overall performance drop stems from the allocation of large memory areas. For the `Large RAM` sub-benchmark, performance overhead is almost 97%. The problem was, that to replicate our VM template in the CPU overcommitment scenarios, we could only assign 256MB main memory to each VM because memory overcommitment is currently not supported by Citrix XenServer 5.5. However, when increasing this size to 3GB main memory in separate, independent experiment with one VM at a time, the performance overhead for large memory accesses is only 65% instead of 97%, which also improves the overall memory benchmark results slightly. Hence, increasing memory allocation can significantly improve performance for memory-intensive workloads, especially if expensive swapping can be avoided.

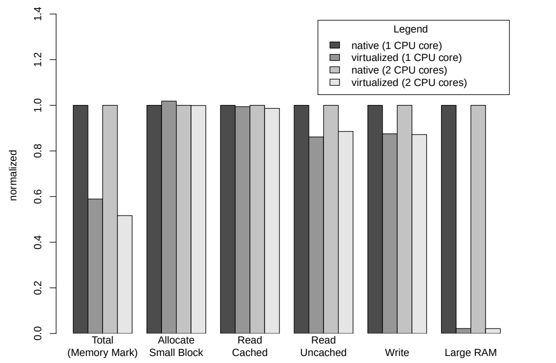

Figure 2: Sub-benchmark results of the Passmark Memory Mark metric.

Finally, Figure 3 shows our results for disk I/O intensive workloads . With the Passmark Disk mark benchmark, we measured a performance overhead of up to 28%. A more detailed look at the benchmark results shows that most of the performance overhead is caused by sequential `read` requests, which achieve only 60% of the native performance, whereas for `write` request's the performance overhead does not exceed 20%.

The performance increase for the random seek benchmark can be explained by the structure of the virtual block device, a concept used in Citrix XenServer 5.5 for block oriented read and write, minimizing administration overhead and thus decreasing access times.

Scalability and Overcommitment Previous experimental results showed that the performance overhead can be reduced for CPU workload when using core affinity. The following more detailed measurements give more insights and explain why core affinity improves performance. Figure 4(a) shows the boxplot of 24 VMs simultaneously executing the CPU bench-

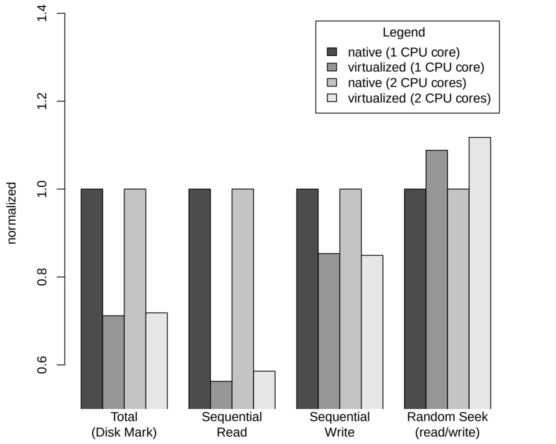

Figure 3: Sub-benchmark results for local disk access of the Passmark Disk Mark metric.

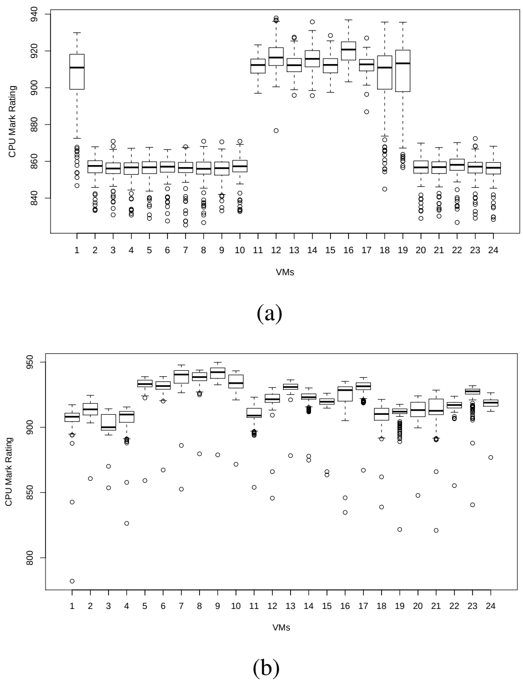

Figure 4: CPU benchmark results for 24 VMs executed *without* (a) and *with* (b) core affinity.

mark without core affinity. There are clearly two categories of VMs, one category (1, 11-19) performing significantly different from the other (2-10, 20-24). This behavior is not observed with core affinity (see Figure 4(b)), when each VM is executed on a separate physical core. This indicates that XenServer's scheduler does not distribute the VMs on the cores equally. In general, it demonstrates that multicore environments still are a challenge for hypervisor schedulers. In Section 3, we investigate if this effect is observable on other hypervisor architectures, too.

With core affinity enabled, for scalability and overcommitment the performance drops linear with the scaled amount of VMs and inversely proportional to the overcommitment factor, respectively. For example, assume c is the amount of physical cores. If provisioning $x \cdot c$ amount of virtual CPUs, performance roughly drops by $\frac{1}{x}$ in each of our experimental environments (single core, dual core, 24 cores).

Network I/O We conducted further experiments with the network I/O benchmark Iperf to gain more insight on the performance overhead of XenServer's credit-based scheduler and its performance isolation. More precisely, the goal of these experiments was to demonstrate how the additional overhead introduced by the hypervisor to handle I/O requests is distributed among the running VMs. To this end, we executed four VMs, two VMs running CPU Mark and two VMs with *Iperf*. We pinned them pairwise on two physical cores, i.e., core c_0 executed a pair of the CPU VM and Iperf VM and a different available core c_x the other pair. The CPU benchmark was executed on both VMs, simultaneously and the network I/O benchmark was started separately on one VM. This symmetric setup allows us to compare the results of VMs executed on c_0 (where the Dom0 is executed) with the results on the different $core_x$.

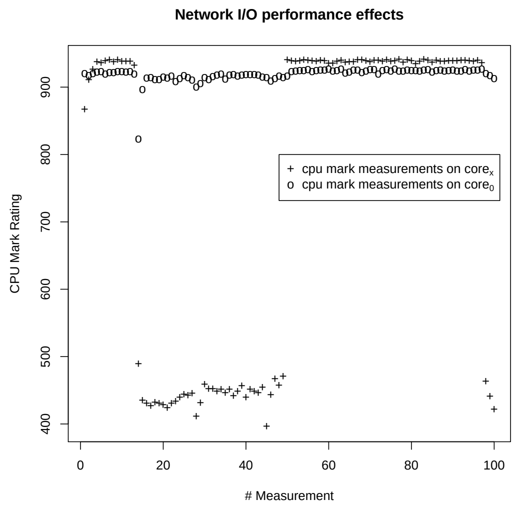

Figure 5: CPU benchmark results of two VMs executed on $core_0$ and $core_x$ when network I/O is received by the VM running on $core_x$.

One would expect that there is no performance effect on the VMs running on c_0 when the Iperf VM on

VM_A	CPU	CPU	Mem	CPU	Mem	Disk	CPU	Mem	Disk
VM_B	CPU	Mem	Mem	Disk	Disk	Disk	Net	Net	Net
r_A	46.71%	50.64%	50.33%	23.35%	24.82%	31.16%	52.88%	52.85%	3.07%
r_B	52.44%	45.93%	49.04%	1.49%	-0.09%	45.99%	40.46%	42.18%	33.31%

Table 1: Mutual performance degradation for different workload types on Citrix XenServer 5.5.

c_x executes network I/O. However, the results show that the performance of the VM running the CPU benchmark on c_0 drops up to 13% when the VM on c_x is executing the network I/O benchmark. Figure 5 depicts the CPU benchmark results of VMs executed on $core_0$ (o) and $core_x$ (+). The second VM on $core_x$ receives the network load, hence the benchmark rating of the VM sharing this core drops significantly. But also the benchmark rating of the VM on $core_0$ drops, although its paired VM is idle. Because VMs executed on other cores than c_0 did not exhibit this behavior, this indicates that Dom0 mainly uses c_0 to handle I/O. This causes a slight performance drop for VMs simultaneously executed on c_0, i.e., about 1% on average. However, this drop could further increase if further machines on other cores receive network load. We did not execute these scalability experiments as this would require further network interfaces to generate network load on further Iperf VMs. However, this is an interesting question for future work as well as the question if this effect can be observed on other hypervisors too, which will be discussed in the following section.

Mutual Influences of Workload Types Target of these experiments is to identify the mutual influences of VMs sharing their resources and serving different workload types. To this end, we pinned two VMs VM_A and VM_B on the same physical core other then $core_0$ to avoid interferences with Dom0. Then, we ran an experiment for each possible combination of benchmark types. As a result, we calculate the relative performance drop as $r = 1 - (r_i/r_s)$, where r_i is the interference result and r_s the result measured when executing the benchmark on an isolated VM. Table 1 summarizes the results for all combinations of workload types. Note that we did not run network vs. network experiments because a different hardware environment with additional hardware would have been required.

The results show that there are no significant mutual influences of CPU and memory intensive workloads. The performance drop for both benchmarks is reasonably equal and the drop also fits the expectation that each VM receives only half of its performance compared to isolated execution. Explanations are the similarity of both workload types in terms of the used resources (memory benchmarks require CPU as well) and the hardware support for CPU virtualization. An interesting observation is that the Disk benchmark is not influenced by other workload types except of when executed vs. the disk benchmark. This indicates that on Citrix XenServer 5.5, disk intensive workloads do not compete for resources of CPU and memory intensive workloads. This can be explained with the similar reason as for the virtualization overhead of the Disk mark result: the concept used in Citrix XenServer 5.5 for block oriented read and write to minimize administration overhead. With this concept, disk workload can be passed through without requiring major hypervisor intervention.

3 EVALUATION

To evaluate the validity of the conclusions from our analysis of XenServer on other hypervisor architectures, we conducted the same experiments on another popular industry standard platform, VMware ESX 4.0, which has a different type of hypervisor architecture. This section compares the experiment results of both platforms. We also provide a short experience report on the portability of the automated experimental analysis approach, which is of general interest when migrating from one virtualization platform to another as well as for automatic administration of virtualization platforms.

3.1 Platform Comparison

The following discussion and comparison of the results of our measurements on VMware ESX 4.0 has a similar structure as Section 2.5. However, because of unavailable driver support for the HP Compaq machine (see Section 3.2), we were only able to install VMware ESX 4.0 and to conduct our experiments on the SunFire machine.

Overhead of Virtualization After repeating the experiments on VMware ESX 4.0, we calculate the relative delta between the two platforms as

$\frac{VMwareESX\ 4.0 - CitrixXenServer\ 5.5}{VMwareESX\ 4.0}$. The results in Table 2 show almost identical results for the CPU and memory benchmarks because both virtualization platforms use the hardware virtualization support. However, for the I/O benchmarks, VMware ESX 4.0 performs better. The reason for this is that in Citrix XenServer 5.5, all I/O workload is handled by the separate driver domain Dom0, which is less efficient monolithic architecture of VMware ESX 4.0. Hence, it is important to distinguish these architectural differences when generalizing the results for the I/O performance overhead.

Benchmark	rel. Delta
CPU Mark	0.15%
Memory Mark	0.19%
Disk Mark	19.14%
Iperf, outgoing	13.91%
Iperf, incoming	15.94%

Table 2: Relative deviation of CPU, memory, disk I/O and network I/O benchmark results.

Scalability and Overcommitment Concerning the performance behavior of VMware ESX 4.0 when scaling up and overcommitting, respectively, we observe a similar trend to the one on Citrix XenServer 5.5. Figure 6 shows this trend for the overcommitment scenario. As one can see, both platforms behave similarly. The results for scalability are similar with VMware ESX 4.0 performing slightly better. Another observation was that on VMware ESX 4.0, using core affinity did not result in any performance improvements. This indicates an improved hypervisor scheduling strategy which takes care of multicore environments and the cache and core effects observed in Section 2.5.

Network I/O Analyzing the results of the network I/O experiments repeated on VMware ESX 4.0 shows some further advantages of the monolithic architecture and that the concept of a separate management VM (Dom0) has a slight performance drawback. For example, we did not observe the effect of the Dom0 discussed in Section 3.1 (Network I/O). Hence, on VMware ESX 4.0, the additional overhead for I/O virtualization is distributed more evenly than in Citrix XenServer 5.5.

Mutual Influences of Workload Types We repeated the same experiments to determine the mutual influences of workload types on VMware ESX 4.0. Table 3 lists the results. For CPU and memory intensive workloads, the observations are comparable

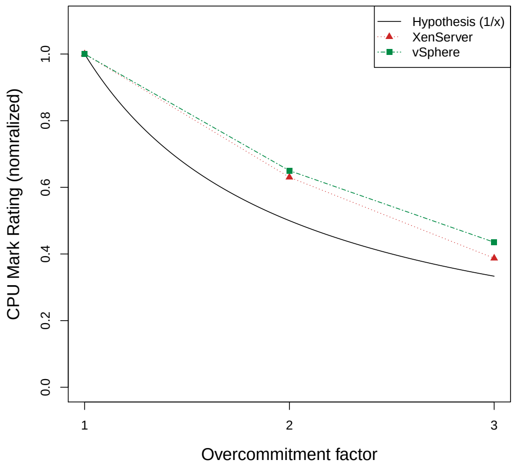

Figure 6: Performance behavior for CPU overcommitment for Citrix XenServer 5.5 and VMware ESX 4.0.

to the ones for Citrix XenServer 5.5: both workload types have a similar effect on each other caused by their similarities.

However, there is a big difference to Citrix XenServer 5.5 for disk intensive workload. For VMware ESX 4.0, we observe a high performance degradation of the disk workload independent of the other workload type. For example, if the disk benchmark is executed with CPU or memory benchmark, disk benchmark results drop almost 50%, whereas CPU and memory benchmark results suffer from only 10% and 20% performance loss, respectively. One explanation is that VMware's virtual disk concept is different from Xen and in this concept both VMs compete for CPU time assigned by the hypervisor, thus confirming the differences in both hypervisor architectures. However, CPU and memory suffer from less performance degradation when running against disk workload than in Citrix XenServer 5.5.

3.2 Portability of the Automated Analysis

The first problem when applying the automated experimental analysis approach to VMware ESX 4.0 was that we could not install it on the HP Compaq machine because of the lack of driver support. Supporting only commodity hardware keeps the monolithic hypervisor's footprint low. Because of Citrix XenServer 5.5's architecture, further drivers can be easily implemented in the Dom0 domain while still keeping the hypervisors footprint small. Hence, we could not repeat the measurements for VMware ESX 4.0 on the HP Compaq machine. The next chal-

VM_A	CPU	CPU	Mem	CPU	Mem	Disk	CPU	Mem	Disk
VM_B	CPU	Mem	Mem	Disk	Disk	Disk	Net	Net	Net
r_A	47.03%	46.64%	49.23%	10.02%	17.21%	44.53%	9.95%	35.32%	14.87%
r_B	48.21%	40.29%	51.34%	49.56%	45.53%	44.82%	65.02%	54.56%	32.74%

Table 3: Mutual performance degradation for different workload types on VMware ESX 4.0.

lenge we faced was that the VMs of VMware ESX 4.0 are usually intended to be managed via graphical external tools, which hinders an automated approach. Fortunately, a special command line interface, which must be activated separately, can be used for automation.

As both platforms support the *Open Virtualization Format (OVF)* for virtual machines in theroy, porting the MasterVM should be easy. Although this standardized XML schema for OVF is in fact implemented on both platforms, they use a different XML tag semantic to describe the VM geometry (like its partitions etc.). Hence, the export of a VM from Citrix XenServer 5.5 to VMware ESX 4.0 works in theory, but practically only with additional tools and workarounds, involving manual XML editing.

Once migrated, one can reuse the concept of automated experimental analysis and experiment types, but one has to adapt the scripts to the new API. For example, the credit-based scheduler parameter *capacity* is named differently on VMware ESX 4.0.

In summary, the concept itself is portable but requires some manual adjustments. This mainly stems from the fact that currently there is no standardized and working virtual machine migration mechanism across different virtualization platforms.

3.3 Summary

By migrating and repeating our automated analysis to VMware ESX 4.0, we were able to confirm the results for CPU and memory intensive workloads as well as the observed trends in the scalability and overcommitment scenarios. However, the experiments also showed that there are differences when handling I/O intensive workloads. In these scenarios, VMware ESX 4.0's performance behavior and performance isolation is better. However, this product has high licensing costs. Moreover, it is intended for graphical administration which makes automized administration more difficult.

4 MODELING THE PERFORMANCE-INFLUENCING FACTORS

Having analyzed two major representative virtualization platforms, we now structure the performance-influencing factors and capture them in a basic mathematical performance model allowing one to predict the performance impacts of virtualized environments.

4.1 Categorizing the Performance-Influencing Factors

This section categorizes the performance-influencing factors of the presented virtualization platforms. The goal is to provide a compact hierarchical model of performance-relevant properties and their dependencies. We capture those factors that have to be considered for performance predictions at the application level, i.e., that have a considerable impact on the virtualization platform's performance, and we structure them in a so-called *feature model* (Czarnecki and Eisenecker, 2000). In our context, a feature corresponds to a performance-relevant property or a configuration option of a virtualization platform. The goal of the feature model is to capture the options that have an influence on the performance of the virtualization platform in a hierarchical structure. The feature model should also consider external influencing factors such as workload profile or type of hardware. The model we propose is depicted in Figure 7.

The first performance-influencing factor is the *virtualization type*. Different techniques might cause different performance overhead, e.g., full virtualization performs better than other alternatives because of the hardware support. In our feature model, we distinguish between the three types of virtualization: i) full virtualization, ii) para-virtualization and iii) binary translation. Furthermore, our experiments showed, that another important performance-influencing factor is the *hypervisor's architecture*. For example, a monolithic architecture exhibited better performance isolation.

Several influencing factors are grouped under *resource management configuration*. First, the CPU

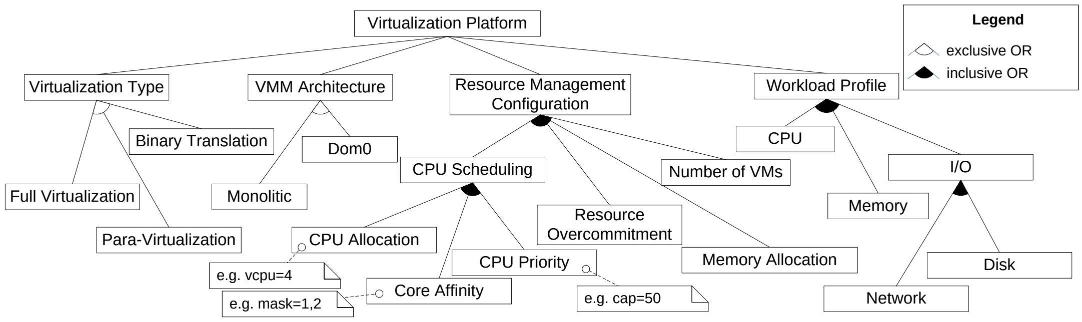

Figure 7: Major performance-influencing factors of virtualization platforms.

scheduling configuration has a significant influence on the virtualization platform's performance and is influenced by several factors. The first factor CPU allocation reflects the number of virtual CPUs allocated to a VM. Most of the performance loss of CPU intensive workloads comes from core and cache inferences (Apparao et al., 2008). Hence, the second factor is core affinity, specifying if virtual CPUs of VMs are assigned to dedicated physical cores (core-pinning). The third factor reflects the capability of assigning different CPU priorities to the VMs. For example, the Xen hypervisor's cap parameter or VMware's limits and fixed reservations parameters are CPU priority configurations. In addition, the level of resource overcommitment influences the performance due to contention effects caused by resource sharing. Finally, the memory allocation and the number of VMs influence the resource management configuration, too. Managing virtual memory requires an additional management layer in the hypervisor. The number of VMs has a direct effect on how the available resources are shared among all VMs.

Last but not least, an important influencing factor is the *workload profile* executed on the virtualization platform. Virtualizing different types of resources causes different performance overheads. For example, CPU virtualization is supported very well whereas I/O and memory virtualization currently suffer from significant performance overheads. In our model we distinguish CPU, memory and I/O intensive workloads. In the case of I/O workload, we further distinguish between disk and network intensive I/O workloads. Of course, one can also imagine a workload mixture as a combination of the basic workload types.

4.2 Performance Model

Based on the results of Section 2.5 and Section 3 we now propose a basic mathematical performance prediction model (e.g., based on linear regression). We focus on the performance-influencing factors for which similar results were observed on the two virtualization platforms considered. These are the overhead for CPU and memory virtualization, the performance behavior in scalability scenarios and the performance behavior when overcommitting CPU resources. Our model is intended to reflect the performance influences of the factors presented in the previous section. It can be used to predict the performance overhead for services to be deployed in, e.g., Cloud Computing or virtualized environments in general.

In our experiments, performance is measured as the amount of benchmark operations processed per unit of time, i.e., the throughput of the system. This is not directly transferable to system utilization or response times, as the benchmarks always try to fully utilize the available resources. Therefore, in the following, we refer to throughput as the system performance. We calculate a performance overhead factor o which can be used to predict the performance $p_{virtualized} = o \cdot p_{native}$, where o can be replaced by a formula of one of the following sections.

Overhead of Virtualization The following basic equations allow to predict the overhead introduced when migrating a native system to a virtualized platform. These equations assume that there are no influences by other virtual machines, which we consider later below.

For CPU and memory virtualization, we calculate the overhead factors o_{cpu} and o_{mem} as $1 - \frac{relative\_deviation}{100}$ using the measured relative deviation values. One can use our automated approach to determine these factors for any other virtualization platform to derive more specific overhead factors.

For I/O overhead, we recommend to measure the performance overhead for each specific virtualization platform using our automated approach because the evaluation showed that there are significant differences between different virtualization platforms and their implementations, respectively.

Scalability To model the performance-influence of scaling-up CPU resources, we use a linear equation. The performance overhead is defined as $o_{scal} = a + b \cdot c_{virt}$, where c_{virt} is the number of virtual cores. The coefficients a and b are given in Table 4. We distinguish between scenarios without core affinity and scenarios, where the virtual CPUs are pinned to the physical cores in an equal distribution. These equations give an approximation of the performance degradation when scaling-up which is independent of the virtualization platform. However, this approximation is only valid until you reach the amount of physical cores available. The overcommitment scenario is modeled in the next section. Moreover, the coefficients of determination show that the linear trend fits very well, except for the CPU with affinity scenario.

Scenario	**a**	**b**	R^2
CPU	1.008	-0.0055	0.9957
Memory	1.007	-0.0179	0.9924
CPU (w. affinity)	1.003	-0.0018	0.7851
Memory (w. affinity)	1.002	-0.0120	0.9842

Table 4: Coefficients a, b for the linear equations for CPU and memory performance when scaling-up and the corresponding coefficient of determination.

Overcommitment When considering a scenario with overcommitted CPU resources, we can approximate the performance overhead as $o_{overc} = \frac{1}{x}$, where x is the overcommitment factor. The overcommitment factor is determined by $\frac{c_{virt}}{c_{phy}}$, the ratio of the provisioned virtual cores c_{virt} and available physical cores c_{phy}. Note that for CPU overcommitment this dependency between the performance overhead and the overcommitment factor is independent of the virtualization platform and the amount of executed VMs. Our experiments on two leading industry standard virtualization platforms demonstrated that the performance overhead simply depends on the ratio of virtual and physical cores. This dependency is valid at the core level, i.e., if you pin two VMs with one virtual core each on a single physical core, you experience the same performance drop.

5 RELATED WORK

There are two groups of existing work related to the work presented in this paper. The first group deals with benchmarking and performance analysis of virtualization platforms and solutions. The second group is related to this work in terms of modeling the performance-influencing factors.

(Barham et al., 2003) present the Xen hypervisor and compare its performance to a native system, the VMware workstation 3.2 and a User-Mode Linux at a high level of abstraction. They show that the performance is practically equivalent to a native Linux system and state that the Xen hypervisor is very scalable. (Quétier et al., 2007; Soltesz et al., 2007; Padala et al., 2007) follow similar approaches by benchmarking, analyzing and comparing the properties of Linux-VServer 1.29, Xen 2.0, User-Mode Linux kernel 2.6.7, VMware Workstation 3.2. and OpenVZ, another container-based virtualization solution. (Apparao et al., 2008) analyze the performance characteristic of a server consolidation workload. Their results show that most of the performance loss of CPU intensive workloads is caused by cache and core interferences. However, since the publication of these results, the considered virtualization platforms have changed a lot (e.g., hardware support was introduced) which renders the results outdated. Hence, the results of these works must be revised especially to evaluate the influences of, e.g., hardware support. Moreover, the previous work mentioned above does not come up with a model of the performance-influencing factors nor does it propose a systematic approach to quantify their impact automatically. Such a generic framework to conduct performance analyses is presented in (Westermann et al., 2010). This framework allows adding adapters to benchmark, monitor, and analyze the performance of a system. The framework has been applied to the performance analysis of message-oriented middleware, however, the adapters currently do not support the analysis of performance properties of virtualization platforms or Cloud Computing environments.

The second area of related work is the modeling of virtualization platforms or shared resources. (Tickoo et al., 2009) identify the challenges of modeling the contention of the visible and invisible resources and the hypervisor. In their consecutive work based on (Apparao et al., 2008; Tickoo et al., 2009), (Iyer et al., 2009) measure and model the influences of VM shared resources. They show the importance of shared resource contention on virtual machine performance and model cache and core effects, but no other performance-influencing factors. Another interesting approach to determine the performance interference effects of virtualization based on benchmarking is (Koh et al., 2007). They explicitly consider different types of workloads (applications) and develop performance prediction mechanisms for different combinations of workloads. Unfortunately, this approach is neither automated nor evaluated on different platforms.

6 CONCLUSION AND OUTLOOK

In this paper, we conducted fine-grained experiments and in-depth analyses of the Citrix XenServer 5.5 based on the results of (Huber et al., 2010). We migrated this approach to VMware ESX 4.0 and evaluated the validity of the previous findings. In summary, the results showed that CPU and memory virtualization performance behavior is similar on both systems as well as CPU scalability and overcommitment. However, the results also indicated a deviation when it comes to I/O virtualization and scheduling. In these cases, VMware ESX 4.0 provides better performance and performance isolation than Citrix XenServer 5.5. We evaluated the portability of the automated experimental analysis approach. Finally, we presented a basic model allowing to predict the performance when migrating applications from native systems to virtualized environments, for scaling up and overcommitting CPU resources, or for migrating to a different virtualization platform. As a next step, we plan to study the performance overhead for mixed workload types and their mutual performance influence in more detail. In addition, we will use our model as a basis for future work in the Descartes research project (Descartes Research Group, 2010; Kounev et al., 2010). For example, we will integrate our results in a meta model for performance prediction of services deployed in dynamic virtualized environments, e.g., Cloud Computing.

ACKNOWLEDGMENTS

This work was funded by the German Research Foundation (DFG) under grant No. 3445/6-1.

REFERENCES

Apparao, P., Iyer, R., Zhang, X., Newell, D., and Adelmeyer, T. (2008). Characterization & Analysis of a Server Consolidation Benchmark. In *VEE '08*.

Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., and Warfield, A. (2003). Xen and the Art of Virtualization. In *Symposium on Operating Systems Principles*.

Czarnecki, K. and Eisenecker, U. W. (2000). *Generative Programming*. Addison-Wesley.

Descartes Research Group (2010). http://www.descartes-research.net.

Huber, N., von Quast, M., Brosig, F., and Kounev, S. (2010). Analysis of the Performance-Influencing Factors of Virtualization Platforms. In *DOA'10*.

IDC (2010). Virtualization Market Accelerates Out of the Recession as Users Adopt "Virtualize First" Mentality. http://www.idc.com/getdoc.jsp?containerId=prUS22316610.

IT world, The IDG Network (2008). Gartner's data on energy consumption, virtualization, cloud. http://www.itworld.com/green-it/59328/gartners-data-energy-consumption-virtualization-cloud.

Iyer, R., Illikkal, R., Tickoo, O., Zhao, L., Apparao, P., and Newell, D. (2009). VM3: Measuring, modeling and managing VM shared resources. *Computer Networks*, 53(17):2873 – 2887.

Koh, Y., Knauerhase, R. C., Brett, P., Bowman, M., Wen, Z., and Pu, C. (2007). An analysis of performance interference effects in virtual environments. In *ISPASS*.

Kounev, S., Brosig, F., Huber, N., and Reussner, R. (2010). Towards self-aware performance and resource management in modern service-oriented systems. In *Int. Conf. on Services Computing*.

Menascé, D. A., Almeida, V. A. F., and Dowdy, L. W. (1994). *Capacity Planning and Performance Modeling - From Mainframes to Client-Server Systems*.

Padala, P., Zhu, X., Wang, Z., Singhal, S., and Shin, K. G. (2007). Performance evaluation of virtualization technologies for server consolidation. *HP Labs Tec. Report*.

Quétier, B., Néri, V., and Cappello, F. (2007). Scalability Comparison of Four Host Virtualization Tools. *Jounal on Grid Computing*, 5(1):83–98.

Rosenblum, M. and Garfinkel, T. (2005). Virtual machine monitors: current technology and future trends. *Computer*, 38(5):39–47.

Salsburg, M. (2007). Beyond the Hypervisor Hype. In *CMG-CONFERENCE*, volume 2, page 487.

Soltesz, S., Pötzl, H., Fiuczynski, M. E., Bavier, A., and Peterson, L. (2007). Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors. *SIGOPS Oper. Syst. Rev.*, 41(3):275–287.

Tickoo, O., Iyer, R., Illikkal, R., and Newell, D. (2009). Modeling virtual machine performance: Challenges and approaches. In *HotMetrics*.

VMware (2007). A performance comparison of hypervisors. http://www.vmware.com/pdf/hypervisor_performance.pdf.

Westermann, D., Happe, J., Hauck, M., and Heupel, C. (2010). The Performance Cockpit Approach: A Framework for Systematic Performance Evaluations. In *SEAA'10*.