Analysis of the Performance-Influencing Factors of Virtualization Platforms

Nikolaus Huber, Marcel von Quast, Fabian Brosig, Samuel Kounev

Karlsruhe Institute of Technology
Chair for Software Design and Quality
Am Fasanengarten 5
76131 Karlsruhe, Germany
{nikolaus.huber,fabian.brosig,samuel.kounev}@kit.edu
marcel.quast@student.kit.edu
http://descartes.ipd.kit.edu

Abstract. Nowadays, virtualization solutions are gaining increasing importance. By enabling the sharing of physical resources, thus making resource usage more efficient, they promise energy and cost savings. Additionally, virtualization is the key enabling technology for Cloud Computing and server consolidation. However, the effects of sharing resources on system performance are not yet well-understood. This makes performance prediction and performance management of services deployed in such dynamic systems very challenging. Because of the large variety of virtualization solutions, a generic approach to predict the performance influences of virtualization platforms is highly desirable. In this paper, we present a hierarchical model capturing the major performance-relevant factors of virtualization platforms. We then propose a general methodology to quantify the influence of the identified factors based on an empirical approach using benchmarks. Finally, we present a case study of Citrix XenServer 5.5, a state-of-the-art virtualization platform.

Keywords: Virtualization, Modeling, Benchmarking, Performance

1 Introduction

In recent years, advances in virtualization technologies promise cost and energy savings for enterprise data centers. Server consolidation, i.e., running multiple virtual servers on a single shared infrastructure, increases resource utilization, centralizes administration, and introduces flexibility. Virtualization allows sharing server resources on-demand, thus creating new business opportunities by providing a new delivery model for a broad set of enterprise services. Therefore, it can be considered as a key technology enabler for Cloud Computing. According to the International Data Corporation (IDC), 18% of all new servers shipped in the fourth quarter of 2009 were virtualized, an increase from 15% compared to 2008 [6]. The server virtualization market is expected to grow 30% a year through 2013 [7]. 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 requires an efficient performance management at the application level. Service-Level Agreements (SLAs), e.g., performance guarantees such as service response time objectives, have to be respected. On the other hand, it is important to use server resources efficiently in order to save administration and energy costs. Thus, service providers 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? Answering such questions for distributed, non-virtualized execution environments is already a complex task [11]. In virtualized environments, this task is complicated by the sharing of resources. 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) How can the performance influence of the identified factors be quantified?

Related work concerning the characterization of virtualization platforms focuses mainly on comparisons of specific virtualization solutions and techniques, e.g., container-based virtualization versus full virtualization [3, 12, 16, 14]. Other work like [2, 17, 8] investigates core and cache contention effects, but the focus there is on reducing the virtualization overhead by introducing shared caches. To the best of our knowledge, an explicit characterization of the major performance-relevant factors of virtualization platforms does not exist.

In this paper, we classify and evaluate the major factors that affect the performance of virtualization platforms. We capture those factors that have to be considered for performance prediction at the application level, i.e., which have an impact on the application-level performance. We abstract from all possible configuration options of the currently available virtualization solutions with the goal to provide a compact hierarchical model capturing the most important performance-relevant factors and their dependencies. In addition, we propose a general methodology to quantify the performance influence of the identified factors. The methodology is based on an empirical approach using benchmarks executed in an automated manner in several predefined configuration scenarios. We applied the methodology to Citrix XenServer 5.5, the most popular freely

available and state-of-the-art solution [6]. The conducted experiments involved a series of benchmark runs on different hardware platforms. We evaluated the overhead of full virtualization for CPU-intensive and memory-intensive workloads, respectively. Following this, we evaluated how different core affinity properties affect the performance of individual virtual machines (VMs). Furthermore, we evaluated how different orders of overcommitment (when the logical resources allocated to all VMs exceed the available physical resources) influence the performance of the overall system and the individual VMs. The contributions of this paper are: 1) a generic model of the most relevant performance-influencing factors of virtualization platforms, 2) a benchmark-based methodology for quantifying the effect of the identified factors and 3) a case study applying the methodology to the state-of-the-art Citrix XenServer 5.5 virtualization platform.

The remainder of this paper is organized as follows. Section 2 provides an overview and classification of current virtualization technologies. A model characterizing the major performance-relevant factors of virtualization platforms is introduced in Section 3. In Section 4, we present our automated approach to quantify the effect of the identified factors. Section 5 presents the case study of Citrix XenServer 5.5 on two representative hardware platforms. Section 6 discusses related work, followed by a conclusion and an outlook on future work.

2 Background

Virtualization technology enables consolidating multiple systems on a shared infrastructure by running multiple *virtual machines* (VMs) on a single physical machine. Each VM is completely separated from the other VMs and hence, it can be moved to other machines. This simplifies load balancing, dealing with hardware failures and eases system scaling. In addition, sharing resources promises a more efficient usage of the available hardware. However, as all VMs share the same physical resources, they also mutually influence each others performance.

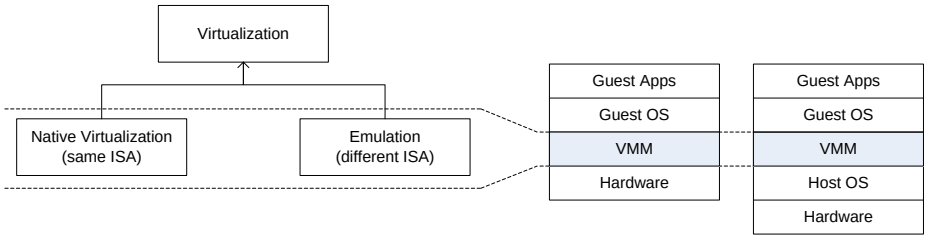


Fig. 1: Native Virtualization vs. Emulation and the VMM as an abstraction layer between hardware and guest (type-I) and between Host OS and guest (type-II).

Over the last years, different types of virtualization were introduced and sometimes terms related to virtualization are used with a different meaning. In this paper, virtualization always refers to *native virtualization* (or system virtualization, see Figure 1). In this case, the virtualization layer provides the native

instruction set architecture (ISA) to its guests. This is in contrast to *emulation* (or non-native virtualization) where the virtualization layer can provide a different ISA. The latter allows software applications or operating systems (OS) written for a special purpose computer processor architecture to be executed on a different platform. The rest of this paper focuses on system virtualization. The core of each virtualization platform is the *virtual machine monitor* (VMM, also called hypervisor). Basically, a VMM is an abstraction layer added on top of the bare metal hardware [10]. It provides a uniform interface to access the underlying hardware. A VM is an execution environment created by a VMM and is similar to the underlying physical machine, but usually with different or reduced hardware resource configuration (e.g., less memory).

Vallee et al. distinguish between two different types of system virtualization [18]. The first is *type-I virtualization* where the VMM runs directly on the physical hardware (see Figure 1), e.g., the Xen hypervisor. On the other hand, if the VMM runs within a host operating system, it is called *type-II virtualization.* An example is the VMware Server,

The VMM provides an abstraction of the underlying machine's hardware and transparent hardware access to all VMs. This implies that software, e.g., an operating system, can be executed without changes or adjustments. An example of such a VMM is the z/VM hypervisor [13].

Unfortunately, not all architectures were designed to support virtualization, e.g., the x86 architecture [10, 15]. Not all instructions of the standard x86 architecture can be virtualized and hence, standard x86 processors do not support direct execution. There are several approaches to address this issue. *Para-virtualization* (PV) is a software solution that addresses the above mentioned problem. Here, the VMM provides an "almost" identical abstraction of the underlying ISA. Any operating system running in a para-virtualized VM must be adapted to support the changed instruction set which limits the set of possible guest OSs. On the other hand, para-virtualization provides better performance since guest systems can be further optimized for their virtualized execution. An example of a VMM that uses para-virtualization is the Xen hypervisor. Another software solution is direct execution with *binary translation*, introduced by VMware [1]. The advantage of binary translation is that any unmodified x86 OS can be executed in VMware's virtual machines. Binary translation basically translates kernel code by replacing non-virtualizable instructions with new sequences of instructions that have the intended effect on the virtualized hardware. Recently, a hardware approach to address the challenge of virtualizing the x86 architecture has been developed by Intel and AMD, enabling *full virtualization* (FV) and the execution of unmodified guest operating systems.

3 Modeling Performance-Influencing Factors of Virtualization Platforms

Ongoing trends show that virtualization technologies are gaining increasing importance. VMware ESX continues to be the most popular virtualization platform

followed by VMware Server and Microsoft Hyper-V. Citrix XenServer showed year-over-year growth of 290% and hence is among the top 5 [6]. Table 1 gives a quick overview of the most common, currently available virtualization solutions. This overview is not complete but shows the diversity of existing solutions and their maturity. It shows several open source implementations like Xen, KVM, VirtualBox and OpenVZ covering all types of virtualization. Other commercial but also free alternatives are, e.g., VMware Server or VMware ESXi. Also important to note is that parts of the open source solutions like Xen or KVM are also used in commercial products like Citrix XenServer or RedHat Enterprise Server, respectively.

Name	Type	Executed As	License	Since	Latest Ver. (06-2010)
Xen	PV/FV	type-I	GPL	10-2003	4.0.0
KVM	FV	type-I	(L)GPL (v2+)	02-2007	kvm-88
VirtualBox	FV	type-II	GPL	02-2007	3.2.4
VMware ESXi	FV/PV	type-I	comm. (free)	12-2007	4
VMware Server	FV/PV	type-II	comm. (free)	07-2006	2
OpenVZ	container-based		GPL v2	12-2005	vzctl_3.0.23
Linux-VServer	container-based		GNU FDL 1.2	2003	2.2stable

Table 1: Common virtualization platforms.

In this section, we summarize 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. To this end, we abstract from all possible configurations of the currently available virtualization platforms presented in Table 1. 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*. Feature models, used in the context of the engineering of software product lines [4], capture variabilities of software products. They define all valid combinations of the software product's property values, called features. One feature defines a certain set of options in the considered domain. 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 reflect 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 or type of hardware (with or without hardware support).

We now discuss the different influencing factors included in the feature model depicted in Figure 2. 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.

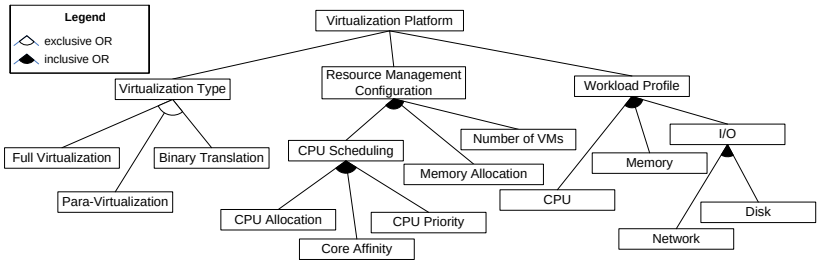


Fig. 2: Major performance-influencing factors of virtualization platforms.

Several influencing factors are grouped under *resource management configuration*. First, CPU scheduling has a significant influence on the virtualization platform's performance. In turn, it 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 [2]. Hence, the second factor that has a significant performance influence is core affinity, specifying if a virtual CPU of a VM is assigned to a dedicated physical core (core-pinning). The third parameter reflects the capability of assigning different CPU priorities to the VMs. For example, the Xen hypervisor's weight and cap parameters or VMware's limits and fixed reservations parameters are represented by these CPU priority configurations. Finally, the memory allocation and the number of VMs influence the resource management configuration, too. Managing 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 *type of workload* 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 consisting of all three workload types.

4 Automated Experimental Analysis

We now present a generic approach based on an automated experimental analysis to quantify the performance-influence of the factors captured in our feature model. First, we give an overview of the experimental setup and describe the general process that is followed when conducting experiments. We then describe the different sets of experiments and how to structure them to assess the performance influence of a given factor. We assume that in each set of experiments a

selected benchmark is executed multiple times in different scenarios to characterize the impact of the considered factor. The process is completely automated using a set of scripts for each experiment type. In Section 4.3, we provide an overview of several benchmarks that provide a basis for evaluating the various influence factors.

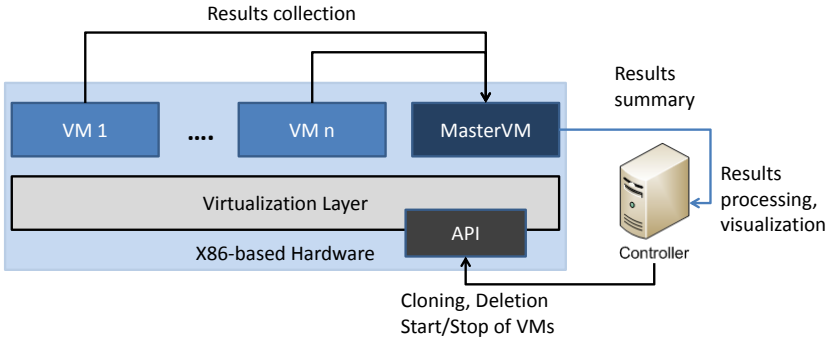


Fig. 3: Static view of the experimental setup.

4.1 Experimental Setup

Static view: As a first step, the virtualization platform to be evaluated is installed on the target hardware platform. In case of type-II virtualization, one would have to install a host OS first. Then, we create a *MasterVM* (see Figure 3) which serves as a template for creating multiple VM clones executing the selected benchmark as part of the considered set of experiments described in Section 4.2. To this end, the respective benchmark is installed on the MasterVM together with scripts to control the benchmark execution (e.g., to schedule benchmark runs). The MasterVM is the only VM with an external network connection. All other VMs and the MasterVM are connected via an internal network. The second major part of our framework is the *controller* which runs on a separate machine. It adjusts the configuration (e.g., amount of virtual CPUs) of the MasterVM and the created clones as required by the considered type of experiments. The controller also clones, deletes, starts, and stops VMs via the virtualization layer's API. Furthermore, it is responsible for collecting, processing and visualizing the results. In this generic approach, the benchmark choice is left open and one can use any available benchmark stressing the considered influencing factor.

Dynamic view: Figure 4 shows the process of automated execution of experiments from the controller's point of view. At first, the controller starts and configures the MasterVM by configuring the benchmark to be executed and scheduling the experiment runs. After that, the MasterVM is replicated according to the requirements of the respective set of experiments described in Section 4.2. After the VM cloning, the controller undertakes further VM-specific configuration for each created clone as required, e.g., assigning the VMs' virtual

CPUs to physical cores. Finally, the VMs are started and the benchmarks are executed at the scheduled starting time. The controller is responsible to detect the end of the benchmark runs and after the experiments are finished, it triggers the MasterVM to collect the results of all VMs. This is done by the MasterVM because it is the only connection between the VM subnet and the controller. If there are further experiments to be executed, the MasterVM is reconfigured and the whole process starts from the beginning, continuing until all experiments are completed. Finally, the controller processes and stores the results from the experiments. The grey boxes of Figure 4 depict the parts of the process where configuration is applied depending on the specific set of experiments considered.

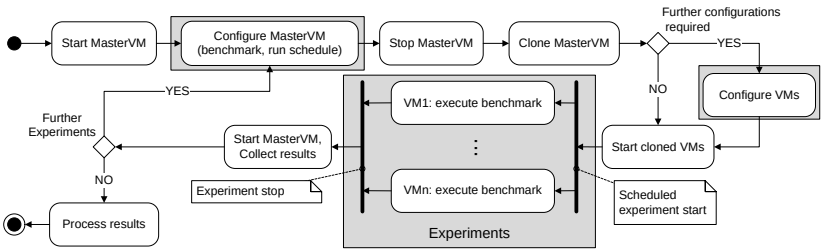


Fig. 4: Automated execution of experiments from the controller's point of view.

4.2 Experiment Types

We distinguish between the following categories of influencing factors according to Section 3: (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, possibly varying the hardware environment and/or the virtualization type if multiple alternatives are supported . This initial set of experiments quantifies the overhead of the hypervisor but does not consider the influence of the number co-located VMs.

The number of VMs and other resource management-related factors like core affinity or CPU scheduling parameters are part of category (b). We investigate the influence of these factors in two different scenarios. The first one focuses on *scalability* (in terms of number of co-located VMs), the second focuses on *overcommitment* (in terms of allocating more resources than are actually available). For scalability, we increase the number of VMs until the all available physical resources are used. For overcommitment, the number of VMs is increased beyond the amount of available resources. The process is illustrated in Figure 5. As the example resource type, we use the number of available physical cores c. In the first case, the number of VMs is increased step-by-step up to c, whereas in the second case the number of VMs is increased by a factor $x \in \{1, \ldots, n\}$ multiplied

with the number of cores c. As an example, to determine the influence of core affinity on scalability and overcommitment, the experiment series depicted in Figure 5 is executed one time with and one time without core affinity comparing the results. In the latter case, each virtual core is automatically pinned to a dedicated physical core.

Finally, for category (c) we execute a set of benchmarks focusing on the different types of workloads as described in the next section.

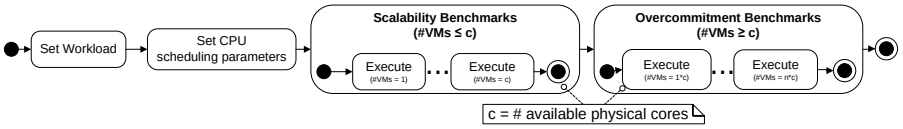


Fig. 5: Benchmark execution in scalability and overcommitment scenarios.

4.3 Benchmark Selection

We now briefly discuss a set of benchmarks representing different types of workloads that can be used in the experiments described in the previous section. For CPU and memory-intensive workloads, we recommend two alternative benchmarks: Passmark PerformanceTest v7.0[1] (a benchmark used by VMware [19]) and SPEC CPU2006[2] (an industry standard CPU benchmark).

Passmark PerformanceTest is a benchmark focused on CPU and memory performance. The benchmark rating is a weighted average of several single benchmark categories (CPU, memory, I/O, etc.). In this paper, we focus on the CPU mark and memory mark rating and are not interested in the overall Passmark rating. SPEC CPU2006 is an industry standard benchmark for evaluating CPU performance. It is structured in a similar fashion and consists of CINT2006 integer benchmarks and CFP2006 floating point benchmarks. However, unlike Passmark, SPEC CPU2006 does not distinguish between CPU and memory performance.

The Passmark benchmark has the advantage that it explicitly distinguishes between CPU and memory, and its benchmark runs are much shorter. Given the short benchmark runs, in our experiments with Passmark we 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 the scalability scenario, multiple instances of the benchmark are executed in separate identical VMs simultaneously. In the end, the results of all benchmark instances are aggregated into one set to compute the overall mean rating. For example, when executing 200 benchmark runs on one machine, we would get 4800 separate benchmark results when scaling up to 24 VMs.

In addition to the above, the Passmark benchmark offers a separate workload focusing on I/O performance, however, for I/O intensive workloads we recom-

[1] Passmark PerformanceTest: `http://www.passmark.com/products/pt.htm`
[2] SPEC CPU2006: `http://www.spec.org/cpu2006/`

mend to use the *Iometer* benchmark[3] which measures the performance of disk and network controllers as well as system-level hard drive performance. Furthermore, for network performance measurements, the *Iperf* benchmark[4] can be used. It is based on a client-server model and supports the throughput measurement of TCP and UDP data connections between both endpoints.

Finally, further workloads that can be used in our experiments are provided by SPEC standard benchmarks such as SPECjbb2005 (stressing CPU and memory performance), SPECmail2009 (stressing I/O performance) and SPECjEnterprise2010 (emulating a complex three tier e-business application). These benchmarks are partly used together with others in the new SPECvirt benchmark which is currently under development. However, this benchmark is out-of-scope of this work as it calculates an overall metric to compare servers and different virtualization option. It is not designed to analyze the influence of specific factors on the system performance.

5 Case Study: Citrix XenServer 5.5

We now present a case study with the Citrix XenServer 5.5 virtualization platform. We apply our automated experimental analysis to evaluate the influence of the major performance-influencing factors. We chose Citrix XenServer 5.5 as the representative virtualization platform because it is a free solution with a significant market share and implementing current virtualization technologies based on the open source hypervisor Xen. We consider full virtualization because it is the most common type used in practice. As workload types, we investigate CPU, memory and network intensive workloads. Concerning the resource management configuration, we investigate the influences of the memory management and the credit-based CPU scheduler implemented in the Xen hypervisor. We also put a special focus on varying the number of co-located VMs.

5.1 Experimental Environment

We conducted experiments in two different hardware environments described below. In each considered scenario, unless stated otherwise, we used Windows 2003 Server as the native and guest OS hosting the benchmark application.

Environment 1: The purpose of the initial experiments was to evaluate the overhead of the virtualization layer. To this end, we used 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. We also used this hardware 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, we used a *SunFire* X4440 x64 Server. It has 4*2.4 GHz AMD Opteron

[3] Iometer: `http://www.iometer.org/`
[4] Iperf: `http://iperf.sourceforge.net/`

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.

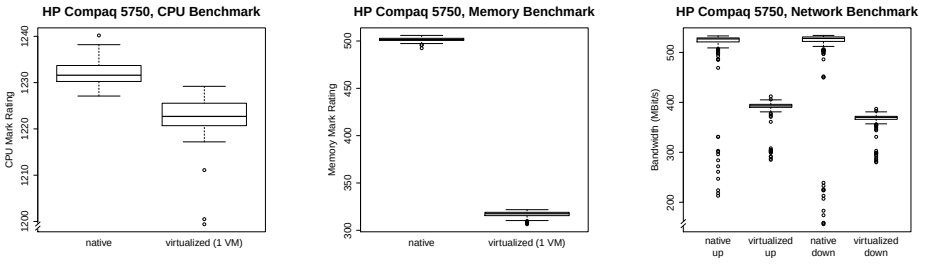
5.2 Experimental Results

We now describe the different experiments we conducted. We consider three different scenarios. The target of the first scenario is to quantify the performance overhead of the Xen hypervisor for CPU and memory intensive workloads. The second scenario addresses the influences of core affinity. The third scenario analyses the influence of the number of VMs and specifically addresses the scalability of the virtualization platform.

CPU Benchmark Ratings	**native**	**virtualized**	**delta (abs.)**	**delta (rel.)**
Passmark CPU, 1 core	639.3	634.0	5.3	0.83%
Passmark CPU, 2 cores	1232.0	1223.0	9.0	0.97%
SPECint(R)_base2006				3.61%
SPECfp(R)_base2006				3.15%
Memory Benchmark Ratings	**native**	**virtualized**	**delta (abs.)**	**delta (rel.)**
Passmark Memory, 1 core	492.9	297.0	195.9	39.74%
Passmark Memory, 2 cores	501.7	317.5	184.2	36.72%
Network Benchmark Ratings	**native**	**virtualized**	**delta (abs.)**	**delta (rel.)**
Iperf, Client to Server	527.0	393.0	134.0	25.43%
Iperf, Server to Client	528.0	370.0	158.0	29.92%

Table 2: CPU, memory and network benchmark ratings for native and virtualized system on the HP Compaq dc5750.

Scenario 1 – Native vs. Virtualization: The purpose of this scenario is to compare the performance of the native system with a virtualized platform for CPU, memory and network intensive workloads. To this end, we executed Passmark, SPEC CPU2006 and Iperf benchmarks in native and virtualized environments. In the virtualized environment, we used only one VM executing the benchmark. The results of these experiments are depicted in Table 2. The relative delta is the ratio of absolute delta and native system performance. As one can see, the performance overhead of CPU intensive workloads is almost negligible. For both the Passmark and SPEC CPU2006 benchmark, the performance degradation when switching from a native to a virtualized system remains below 4%. The results from the two benchmarks are very similar in terms of the measured overhead and lead to the same conclusion. Moreover, the boxplots of the Passmark measurements in Figure 6 show a relatively low scattering. Please note the different y-axis scales in the sub-figures. Also consider that for the SPEC benchmark results we can only publish the relative delta because of licensing reasons.



(a) (b) (c)

Fig. 6: Native vs. virtualized Passmark CPU and memory mark results and the Iperf benchmark results

When comparing the performance of a memory-intensive workload (Table 2), one can observe a much higher performance degradation, i.e., about 40%. The reason for this is the fact that CPU virtualization is well-understood and hardware supported, whereas memory virtualization is still rather immature and currently lacks hardware support [15].

Table 2 and Figure 6c depict the results of the network performance measurements with Iperf. In our experiment setup, client and server were connected with a DLink 1Gbit Switch and we used Windows2003 Server for both machines. We observe a performance decrease for TCP connections in both directions, upstream (Client to Server) 25% and downstream (Server to Client) 30%. This shows that like for memory virtualization there is still a relatively high performance loss because of lacking hardware support. Also interesting is that performance degradation of up- and downstream differs by 5%. Hence, it is important to consider the type of network traffic (up- or downstream)when modeling the workload.

As a preliminary result, we consider the performance overhead of CPU, memory and I/O virtualization to be 5% and 40% and 30%, respectively (in the case of full virtualization).

Scenario 2 – Core Affinity and Scalability: In a virtualized environment, the way VMs are scheduled to cores has a significant influence on the VMs' performance [2]. For example, imagine a machine with 24 cores, each core having its own cache. If a VM is re-scheduled from one core to another, its performance will suffer from cache misses because the benefit of a warm cache is lost. To avoid this, current virtualization platforms provide means to assign cores to VMs. In this scenario, we quantify the performance influence of core affinity considering the effect of scaling the number of co-located VMs. Core affinity denotes the assignment of virtual CPU(s) of VMs to dedicated physical cores, also called core pinning. In this case, the VM is executed only on the assigned core(s) which in turn has a significant influence on the cache and core contention and hence

on performance. We also investigate the performance overhead when scaling the number of VMs up to the limit of available resources, i.e., CPU cores.

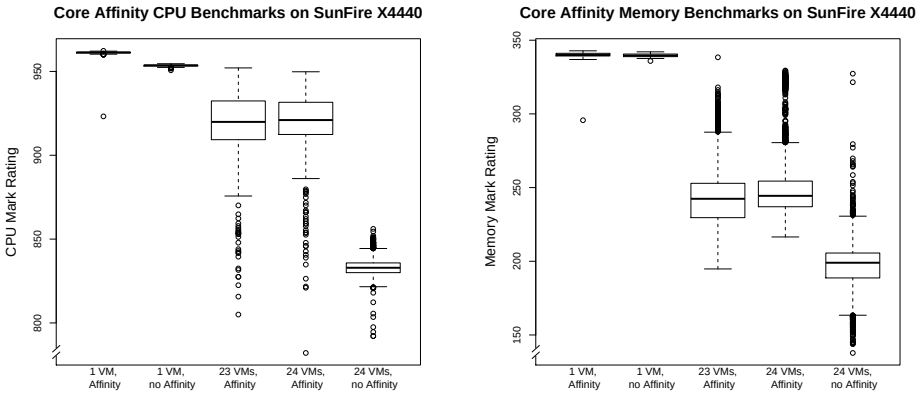


Fig. 7: Performance influence of core affinity in different experiments. The boxplots for multiple VMs contain the measurements over all VMs.

We tested the effect of core pinning using several experiments on the 24 core SunFire X4440 (see Table 3 and Figure 7). First, we compare the CPU and memory mark rating of one VM running with no affinity and one VM pinned to a dedicated core. In this case, performance changes about 0.80% for the CPU mark and 0.10% for the memory mark, so there is no measurable performance influence. However, when comparing the benchmark results of the same experiment for 24 VMs (each VM has one virtual CPU), performance increases by 88.1 (9.56%) and 46 (18.82%) for the CPU and memory mark, respectively.

	CPU Mark			Memory Mark		
	no affinity	with affinity	rel. delta	no affinity	with affinity	rel. delta
one VM	953.60	961.30	0.80%	339.95	339.60	0.10%
24 VMs	832.90	921.00	9.56%	198.40	244.40	18.82%
rel. delta	12.66%	4.19%	-	41.64%	28.03%	-

Table 3: CPU and memory benchmark results for different core affinity experiments. The values shown are the median over all benchmark runs (200 for one VM, $200 * 24$ for 24 VMs) on the SunFire X4440.

The performance loss comparing one VM without affinity and 24 VMs without affinity is 120.7 (12.66%) and 141.55 (41.64%) for CPU and memory mark, respectively. However, when increasing the amount of VMs from one VM with core affinity to 24 VMs with core affinity, performance drops only by 40.3 (4.19%) and 95.2 (28.03%), respectively. Hence, on average 8.47% of the performance penalty for the CPU mark and 13.61% for the memory mark can be avoided when using core pinning. Also interesting is that the variability of the measurements increases with the number of co-located VMs. This leads to performance degradations of up to 10% in the worst case. Nonetheless, the difference between the median and mean values is negligible.

Another interesting fact observable in Figure 7 is that there is little difference in performance between 23 VMs and 24 VMs both with affinity. In the former case one core is left free for the hypervisor. The median of both measurements deviates only by 0.12% for the CPU mark and 0.83% for the memory mark. Hence, leaving one core for the hypervisor has no significant effect on reducing the performance degradation introduced by virtualization.

From the above results we conclude that core affinity has a significant effect on virtualization platform's performance. We model this factor in such way that for predictions consider performance gains up to 20% relative to the ratio of executed virtual machines and available resources.

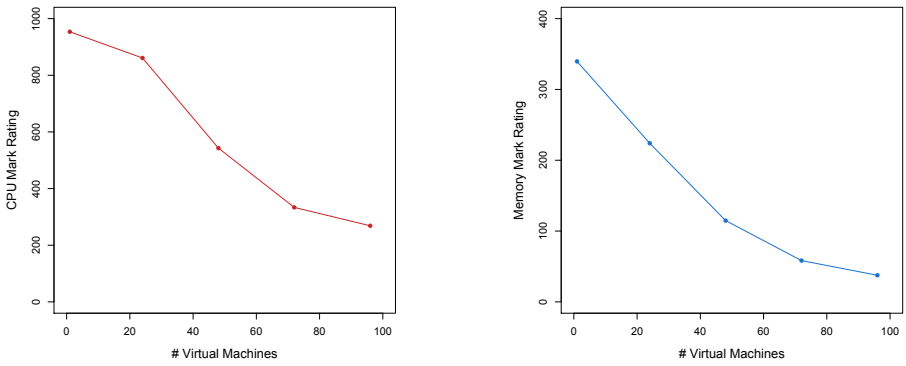


Fig. 8: Absolute scalability and overcommitment experiment results of the CPU and memory benchmark on SunFire X4440.

Scenario 3 – Overcommitment: In this scenario, we investigate the performance degradation when systematically overcommitting shared resources. In this scenario, we scale the amount of VMs (each VM is configured with one virtual CPU) above the amount of physically available CPUs by a factor x ranging between 1 and 4.

In case of the HP Compaq dc5750 environment, we increased the number of VMs to 2, 4, 6 and 8 and for the SunFire X4440 to 24, 48, 72 and 96. The

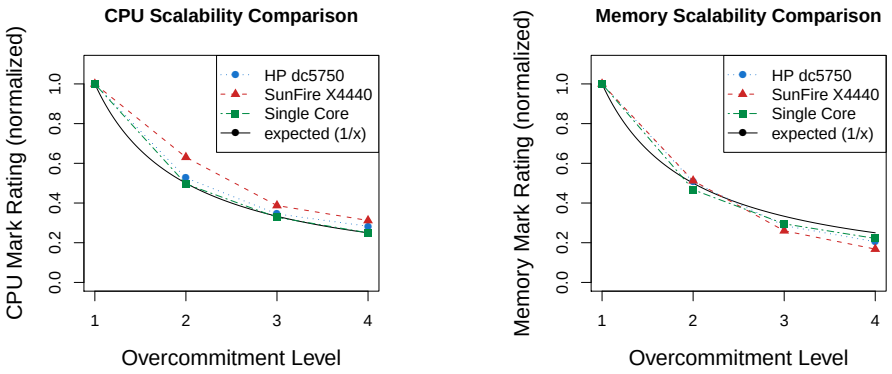


Fig. 9: Normalized scalability comparison of the HP dc5750, the SunFire X4440 and a single core for CPU and memory benchmark compared to the expected value of $1/x$.

absolute results for the CPU and memory benchmark in the SunFire X4440 environment are depicted in Figure 8. We also considered a single core scenario with the SunFire X4440 in which we deactivated all but one physical core and increased the number of VMs to 1, 2, 3 and 4. Figure 9 compares the normalized CPU rating of both hardware environments with the single core scenario. We observe that performance decreases roughly about $1/x$. Moreover, for the CPU benchmark, the measured performance is even slightly better than this expected theoretical value. The reason for this observation is that the single benchmark cannot utilize the CPU at completely 100%. Hence, there are unused CPU cycles which are utilized when executing multiple VMs in parallel. When increasing the number of VMs up to 72, we observed a CPU utilization of all cores at 100%. This effect however does not apply to the memory-intensive workload. Therefore the memory mark rating is slightly worse than the expected theoretical value. The single core performs better than the HP Compaq which in turn is better than the SunFire.

From the results we see that the performance degradation can be approximated by $1/x$ which is the expected theoretical value. Intuitively, one might assume that performance decreases faster or even suddenly drops when overcommitting system resources. Moreover, one can see that performance degradation is very similar in both hardware environments (max. 10% deviation). This is remarkable because one would intuitively assume that the SunFire would perform significantly better because of the more resources it has available. Additionally, the boxplots in Figure 10 show the CPU and memory mark measurements over all 96 VMs. Although there is a relative huge positive deviation of the outliers from the median, the scattering around the median is low, indicating a fair resource sharing and good performance isolation.

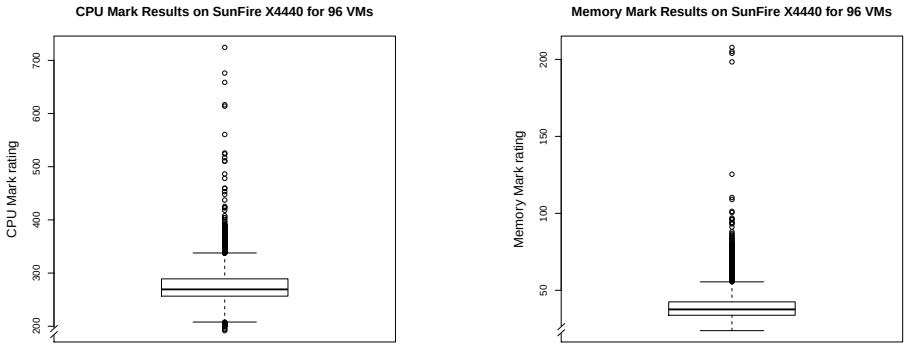


Fig. 10: Boxplots of CPU mark and memory mark results over all 96 VMs (overcommitment factor of 4).

As a result, we conclude that the performance degradation by overcommitting CPU resources through increasing number of VMs is proportional to the overcommitment factor with an upper limit of $1/x$. Furthermore, we observed that the hardware environment has almost no influence on the scalability and performance degradation and both CPU and memory workloads are affected in a similar way.

6 Related Work

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

In [3], Barham et al. present the Xen hypervisor and compare its performance to a native system, the VMware workstation 3.2 and a User-Mode Linux (comparable to container-based virtualization) on a high level of abstraction. They show that the performance is practically equivalent to a native Linux system and state that Xen is able to run up to 100 operating systems on a single server. Quétier et al. [14] follow a similar approach by benchmarking and analyzing the overhead, linearity and isolation for Linux-VServer 1.29, Xen 2.0, User-Mode Linux kernel 2.6.7 and VMware Workstation 3.2. Soltesz et al. propose an alternative to the Xen hypervisor, the container-based virtualization Linux-VServer [16]. In their work they evaluate the Linux-VServer with Xen3 performance with system benchmarks and compare performance in scalability scenarios. Similar is the work by Padala et al., where the authors compare Xen 3.0.3 unstable with OpenVZ, another container-based virtualization solution [12]. Both approaches

conclude that a container-based virtualization performs better than the hypervisor solution, especially for I/O intensive workloads. In [2], Apparao et al. analyze the performance characteristic of a server consolidation workload. They study the performance slowdown of each workload due to consolidation. Their results show that most of the performance loss of CPU intensive workloads is caused by cache and core interferences. They also show that core affinity can mitigate this problem. However, they do not consider the virtualization overhead due to consolidation. Moreover, the considered virtualization solutions have changed a lot since the above results were published (e.g., hardware support was introduced) which renders them outdated. For example, meanwhile Xen 4.0 has been released, introducing a lot of new features. Hence the results of these works must be revised especially to evaluate the influences of hardware support. Moreover, all the work presented do not come up with a model of the performance-influencing factors nor do they propose a systematic approach to quantify their impact.

The second area of related work is the modeling of virtualization platforms or shared resources. Tickoo et al. identify three challenges of VM modeling, namely modeling the contention of the visible and invisible resources and the hypervisor [17]. In their consecutive work based on [2] and [17], Iyer et al. measure and model the influences of VM shared resources. The concept they present is a virtual platform architecture (VPA) with a transparent shared resource management [8]. They show the importance of shared resource contention on virtual machine performance. However, their model only focuses on cache and core effects and does not consider other performance-influencing factors. There is still a lot of work to do on measuring, benchmarking and analyzing different virtualization solutions and their performance-relevant factors. Particularly because virtualization is a technology improving and changing very quickly, it is difficult to keep benchmark results and models up-to-date.

7 Conclusion and Future Work

In this paper, we presented a generic feature model of the performance-influencing factors of virtualization platforms. We proposed a benchmark-based methodology for quantifying the effect of the identified factors and applied the methodology to analyze and quantify the performance-influencing factors and properties of the Citrix XenServer 5.5. The results showed that performance degradation for CPU intensive workloads on full virtualized platforms is below 5% and for network workloads below 30%, and memory intensive workloads suffer from performance degradation of up to 40%. We also showed that core affinity has a considerable influence on reducing the performance degradation. Our scalability experiments revealed that performance degradation is independent of the hardware environment and is roughly proportional to the overcommitment factor. Moreover, it is remarkable that performance converges to the reciprocal of the overcommitment-factor and does not suddenly drop when overcommitting resources.

As a next step, we plan to study further performance influencing factors considering other virtualization platforms such as VMware. In addition, we plan to use the results of this work as input in the Descartes research project to predict the performance of services deployed in dynamic virtualized environments, e.g., Cloud Computing [5, 9].

References

1. K. Adams and O. Agesen. A comparison of software and hardware techniques for x86 virtualization. In *Proceedings of ASPLOS*, 2006.
2. P. Apparao, R. Iyer, X. Zhang, D. Newell, and T. Adelmeyer. Characterization & Analysis of a Server Consolidation Benchmark. In *VEE '08: Proceedings of the 4th Int. Conference on Virtual Execution Environments*, 2008.
3. 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 *SOSP '03: Proceedings of the 19th Symposium on Operating Systems Principles*, 2003.
4. K. Czarnecki and U. W. Eisenecker. *Generative Programming.* Add.-Wesley, 2000.
5. Descartes Research Group. http://www.descartes-research.net, June 2010.
6. IDC. Virtualization Market Accelerates Out of the Recession as Users Adopt "Virtualize First" Mentality, According to IDC, April 2010.
7. IT world, The IDG Network. Gartner's data on energy consumption, virtualization, cloud. http://www.itworld.com/green-it/59328/gartners-data-energy-consumption-virtualization-cloud, 2008.
8. R. Iyer, R. Illikkal, O. Tickoo, L. Zhao, P. Apparao, and D. Newell. VM3: Measuring, modeling and managing VM shared resources. *Computer Networks*, 53(17):2873 – 2887, 2009.
9. S. Kounev, F. Brosig, N. Huber, and R. Reussner. Towards self-aware performance and resource management in modern service-oriented systems. In *Proceedings of the 7th IEEE International Conference on Services Computing*, 2010.
10. D. A. Menascé. Virtualization: Concepts, applications, and performance modeling. In *Int. CMG Conference*, pages 407–414, 2005.
11. D. A. Menascé, V. A. F. Almeida, and L. W. Dowdy. *Capacity Planning and Performance Modeling - From Mainframes to Client-Server Systems.* P.-H., 1994.
12. P. Padala, X. Zhu, Z. Wang, S. Singhal, and K. G. Shin. Performance evaluation of virtualization technologies for server consolidation. *HP Labs Tec. Report*, 2007.
13. L. Parziale, E. L. Alves, E. M. Dow, K. Egeler, J. J. Herne, C. Jordan, E. P. Naveen, M. S. Pattabhiraman, and K. Smith. *Introduction to the New Mainframe: z/VM Basics.* IBM Redbooks, 2007.
14. B. Quétier, V. Néri, and F. Cappello. Scalability Comparison of Four Host Virtualization Tools. *Jounal on Grid Computing*, 5(1):83–98, 2007.
15. M. Rosenblum and T. Garfinkel. Virtual machine monitors: current technology and future trends. *Computer*, 38(5):39–47, 2005.
16. S. Soltesz, H. Pötzl, M. E. Fiuczynski, A. Bavier, and L. Peterson. Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors. *SIGOPS Oper. Syst. Rev.*, 41(3):275–287, 2007.
17. O. Tickoo, R. Iyer, R. Illikkal, and D. Newell. Modeling virtual machine performance: Challenges and approaches. In *HotMetrics*, 2009.
18. G. Vallee, T. Naughton, C. E. H. Ong, and S. L. Scott. System-level virtualization for high performance computing. In *Proc. of PDP*, 2008.
19. VMware. *A Performance Comparison of Hypervisors*, 2007.