Discovering hypervisor overheads using micro and macrobenchmarks

Andrea Bastoni, Daniel P. Bovet, Marco Cesati, and Paolo Palana

System Programming Research Group
University of Rome "Tor Vergata"
Roma, Italy
{bastoni,bovet,cesati,palana}@sprg.uniroma2.it

Abstract. This paper illustrates an approach for evaluating the performance of open-source virtualization solutions based on the well-known SPEC Web macrobenchmark.
Our tests particularly target the virtualization of 64-bit guests over a 64-bit host and the use of hardware virtualization support. Our analysis covers both virtualization overhead of a single virtual machine as well as the overhead of concurrently executing virtual machines.
We then integrate the macrobenchmark results with microbenchmark-based investigation of the impact of virtualization on network, disk and CPU.
The results emerging from micro and macrobenchmark analysis show the need for a deeper investigation of virtualization solutions. Current profiling and instrumentation techniques cannot provide a lightweight monitoring of virtualized systems and are therefore unsuitable for on-line use under heavy workload.
We conclude by proposing a virtualization-aware hardware performance counter infrastructure that can help dealing with this problem.

1 Introduction and Motivation

In the last years, some performance studies that investigate the impact of virtualization on server consolidation have been proposed [16] [10] [4].

We believe that these studies makes use of too many variables (e.g., enterely different workloads, heavy stress of different hardware components. . .) in the performance evaluation and may impair the analysis of virtualization solutions, preventing the discovery of undesirable overhead and scalability problems in the hypervisor.

Other studies, mainly based on microbenchmarks [8] [2] [18] [19], tend to focus on single components of the system (e.g., CPU, memory, disk, network) and are thus unable, in our opinion, to reveal the complexity of the interactions between these virtualized components [9].

In this paper we analyze the effectiveness of three open-source x86 virtualization solutions with hardware support. Our performance evaluation use the *SPECweb2005* [13] macrobenchmark to heavily stress all hardware and software components of a virtualized system at the same time.

The SPECweb macrobencmark generate a demanding workload that is uniform in all virtual machines concurrently executing on the same hypervisor; thus, it can be used to discover hypervisor related problems in single and multiple guest virtualization.

To get a better understanding of the impact of virtualization on network, disk and CPU subsystems, we also run a series of microbenchmarks that separately test each subsystem.

We limit our performance analysis to open-source virtualization solutions with hardware support as their user licences allow us to publish the results without any restriction. In particular, we analyze the *performance* and *scalability* of *KVM*, *Xen*, and *Virtual Box*. The focus of our analysis is on the virtualization of 64-bit guests over 64-bit hosts.

The comparison of micro and macrobenchmark results indicate that benchmarks are not able to fully underline the sources of overhead of virtualization solutions. Therefore, a more detailed analysis and monitoring of the virtual machine and hypervisor execution is needed.

Currently, monitoring and profiling techniques (e.g., [12] [14]) are not suitable for on-line monitoring of virtual machines and hypervisor under heavy workload. Furthermore, profiling techniques for full virtualization solutions with hardware support are still in their infancy and we believe that more support from hardware is needed in order to perform this analysis.

The remainder of this paper is organized as follows. In section 2 we present a brief overview of full virtualization technology. In section 3 we describe related works in virtualization performance evaluation. In section 4 we illustrate the methodology adopted in our experiments. In section 5 we detail the experimental setup of our performance evaluation tests, while in section 6 we report and discuss the results gathered from the tests and we propose a new virtualization-aware hardware performance counter infrastructure. Finally, in section 7 we draw some conclusions.

2 Overview of full virtualization technologies

Informally, the term "virtualization" describes the separation of a service request from the underlying physical delivery of that service. Virtualization is not a new concept ([1][17]) but has recently gained renewed attention, expecially in server consolidation and server isolation.

In full virtualization solutions, unmodified guest kernels (and guest applications) are unaware of the virtualization layer which decouples them from the physical hardware. Guest kernels *sensitive* instructions (those which access kernel data structures or system state) are *trapped* or *binary-translated* by the Virtual Machine Monitor (VMM or hypervisor) to safely execute on the physical CPU.

On x86 architecture, trapping of sensitive instructions works by running guest kernels and applications at a lower privilege level (*ring*) than the hypervisor.

Ring deprivileging is the major source of architectural problems in supporting full virtualization.

In 2006 the two major chip makers AMD and Intel started developing architectural changes in their CPUs (AMD-V [3], Intel VT-x [15]) in order to natively support standard virtualization on x86.

When virtualization support is enabled on the CPU, two operating modes called *root mode* and *guest mode* are available, each supporting all four x86 protection rings. VMM runs in root mode, while all guest OSes run in guest mode in their original privilege levels. VMM can control guest execution though control bits of hardware-defined structures.

Both KVM, Xen 3.3, and Virtual Box are able to exploit the hardware CPU support.

3 Related Work

A performance evaluation of the paravirtualized version of Xen can be found in [7] and [11]. These evaluation was performed using both microbenchmarks (lmbench suite, SPEC CPU2000, dbench) and macrobenchmarks (SPEC WEB99 and OSDB). None of these papers discusses in depth scalability issues of the Xen product.

Quetier *et al.* [8] used microbenchmarks (CPU load, dd, Netperf) to evaluate performance of Linux-VServer, UML, Xen, VMware Workstation, and VMware GSX in the context of Grid Computing.

In [2] Adams and Agesen used microbenchmarks and nanobenchmarks as well as SPEC Java Server Benchmark jbb2005 to analyze differences in VMware software and hardware-assisted VMMs. The authors identified several stress conditions where hardware virtualization support fails to provide a clear performance advantage over binary translation.

In 2007 VMware, Inc. and XenSource, Inc. published two technical reports [18][19] comparing performance obtained by VMware ESX Server, Xen, and XenEnterprise under both microbenchmarks load (SPEC CPU, Passmark, Netperf) and more complete load (SPEC jbb2005).

Padala *et al.* [16] used an application level workload to perform a test which stresses several components of a virtualized system. In particular, they analyzed the performance of Xen and OpenVZ in the context of server consolidation. The authors measured the performance and provided detailed overhead analysis of RUBiS web servers in a multi-tier (two-tier) server consolidation context, exploiting up to four virtual containers and two physical machines.

Camargos *et al.* [10] analyzed KQEMU, KVM, Linux-VServer, OpenVZ, VirtualBox, and Xen using recent Linux kernels (mostly 2.6.22) under microbenchmarks load. The authors also focused on the *scalability* of the chosen solutions.

Apparao *et al.* investigated Xen performance using *vConsolidate* [4], a benchmark developed by Intel that specifically target server consolidation. The authors also performed detailed profiling on cache usage and other architectural characteristics (e.g., cycles per instruction) and showed the execution profile of server consolidation workload.

4 Methodology and Workload

In the first part of our analysis, we use the SPEC Web macrobenchmark [13] to measure the *overall performance* and the *scalability* of Xen, KVM, and Virtual Box. In the second part of the paper we make use of some well-known ([10]) microbenchmark (*Netperf*, *Bzip*, *dd*) to detail network, CPU, and disk overheads introduced by each virtualization solution.

The use of *SPECweb2005* as application level workload has been motivated by two factors: SPEC web is a *de-facto* standard for web server performance evaluation and can be effectively used to simulate real-world web server workloads; the benchmark stresses *at the same time* CPU, memory, disks, and network, thus allowing us to evaluate the effects of virtualization on the interaction between them.

As our main objective is to stress different virtual machines running concurrently on the same physical machine, we did not run the full SPEC web benchmark; rather, we just performed a complete run (three iterations) of the E-commerce workload, which generates an heavy CPU, disk, and network load that can effectively stress the whole system.

Our main macrobenchmark performance metric is the number of *SPEC simultaneous sessions* which represent "the number of simultaneous user sessions the System Under Test was able to support while meeting the quality-of-service (QoS) requirements of the benchmark" [13]. QoS requirements are defined using two parameters (*Time_Good* and *Time_Tolerable*) which identify the maximum aggregate response time allowed for each page request. In our tests we required 97% of the page requests to be returned within Time_Good and 99% of the requests within Time_Tolerable.

In the *Netperf*, *Bzip*, *dd* microbenchmark, the perfomance metrics are those of each benchmark (Mb/s, seconds, and MB/s respectively).

5 Setup

The virtualization solutions considered are KVM 75 (KQEMU 0.9.1), VirtualBox 2.0.6, and Xen 3.3.0. In all solutions we use hardware virtualization support to virtualize *64-bit* guests over a *64-bit* host.

The *System Under Testing (SUT)* is an AMD Opteron 8212 (NUMA, 4 dual-core processor, 2GHz, 1MB L2 cache per core), 16 GB of DDR2 RAM, one 300 GB SCSI disk, and two Intel Gigabit network interfaces. Host and guests run Gentoo with vanilla Linux 64-bit kernel 2.6.21.7, while the HTTP server used is Apache version 2.2.9.

The complete setup of a SPECweb2005 benchmark requires at least three different machines: the SUT machine, the "database" backend machine (*beSim*), and one or more client machines, which can provide more than one SPEC client. One client machine and the *beSim* are Apple Xserve (Intel Xeon 2.6 GHz, 4GB RAM), and the other two client machines are a PowerMac G5 and a Dual Core AMD Athlon 64, both with 4 GB RAM. All these machines run vanilla Linux 2.6.24 64-bit kernel.

We analyzed virtual machine with one and two virtual CPUs (*VCPU*) each with 1.5 GB of RAM. To get comparable results, Linux native performance have been obtained by limiting the number of usable physical CPUs and the amount of usable RAM.

We performed microbenchmark analysis running three iterations of a standard ten-seconds Netperf test, a dd raw copy of a 742 MB file (Gentoo amd64 livecd), and a bzip compression of the same file.

6 Results and discussion

In this section, we first compare the performance of a system running a single virtual machine with that of a system without virtualization. We then measure the scalability of the three virtualization solutions. Here, each virtualization technology is compared with the others by measuring the performance obtained when increasing the number of concurrently executing virtual machines. Finally, we underline a few performance results which are hard to interpret and we compare them with the results coming from microbenchmark analysis.

6.1 Comparison with the non-virtualized system

We compare SPEC performance obtained by a single VM with those obtained by a non-virtualized system; the results of this test are shown in Figure 1. The value on top of each bar denotes the maximum number of SPEC simultaneous sessions obtainable by the corresponding solution[1]. The height of each bar is proportional to the ratio between the performance of the considered solution and that of the non-virtualized one.

As can be seen in the figure, KVM performs best with a score of 0.26 (the number of maximum SPEC simultaneous sessions is only 26% of the number obtainable by a non-virtualized system). Virtual Box's score of 0.2 is the worst, with a performance degradation of 80%.

Figure 2 reports the results of KVM and Xen HVM in the 2 VCPUs configuration, as well as the non-virtualized Linux using two physical CPUs and 1.5 GB of RAM. Virtual Box cannot emulate 2 virtual CPUs on one VM and is therefore not shown. The results are similar to those in the 1 VCPU configuration. However, adding a virtual CPU (in KVM and Xen) introduces additional overheads which prevent the virtualization solutions from reaching the same performance increase experimented by non-virtualized Linux. In fact, SPEC results in KVM

[1] This value is an average of the values obtained in the three iterations of the test. The standard deviation associated with the measure is low and the values are close to each other.

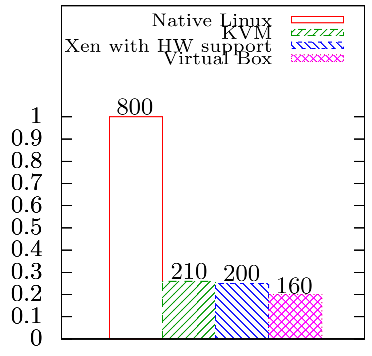

Fig. 1. SPEC perf. with 1 VCPU

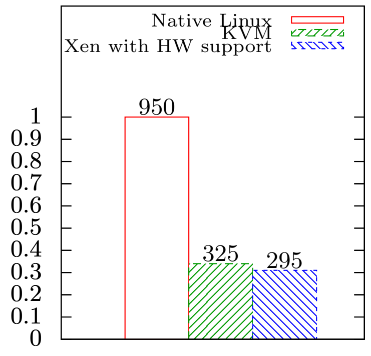

Fig. 2. SPEC perf. with 2 VCPUs

and Xen only increase by around 100 sessions without saturating 1.5 GB RAM, while native Linux result, if the constraint of 1.5 GB of RAM is relaxed to 3 GB, grows to 1250 sessions (with a gain of 450 sessions).

KVM and Xen's additional overheads could be related to the NUMA architecture of machine used as SUT. In particular, the VMM in both KVM and Xen 3.3 is unable to export the NUMA node layout to the guest virtual machine (thus preventing guest to properly exploit the NUMA architecture). Memory allocation and virtual machine scheduling decision are therefore suboptimal.

6.2 Scalability of full virtualization solutions with hardware support

Figure 3 shows the cumulative SPEC performance for KVM (two VCPUs), Xen (two VCPUs), and Virtual Box (one VCPU) as the number of VMs increases. Figure 4 shows the same performance metrics when KVM and Xen are given a single virtual CPU. Non virtualized Linux performance is not reported in these charts as its value is much higher (2750 SPEC session on average) than that of virtualization solutions.

KVM gets the best results when the number of virtual machines is less than four (approximately 20% better than Xen HVM). Each virtual machine sustains less than 350 SPEC sessions, that is, nearly 30% of those handled by non-virtualized Linux (900 SPEC session). The best cumulative result scored by KVM is 1200 SPEC sessions – less than 50% of those handled by native Linux.

It is interesting to note that the cumulative performance of KVM does not increase anymore when the number of VMs rises above four; Xen HVM shows a similar behaviour with 6 virtual machines. In these cases, the VMs must be multiplexed on the eight available physical CPUs thus causing additional overhead. Virtual Box behaves differently when the number of VMs increases. This is mainly because for a given number of VMs Virtual Box implements half of the VCPUs with respect to the other solutions, thus requiring less physical CPU multiplexing and limiting the impact of overhead due to multiplexing.

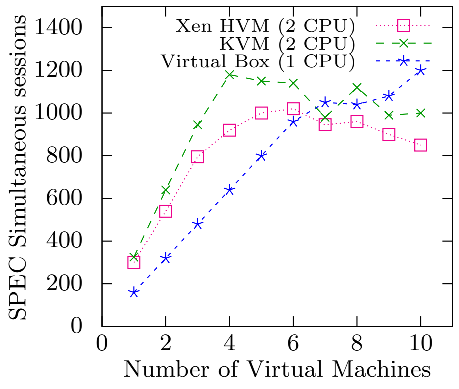

Fig. 3. Cumul. SPEC perf.

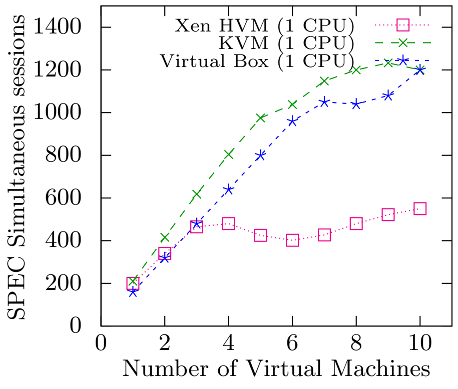

Fig. 4. Cumul. SPEC perf.(1 CPU)

6.3 Microbenchmark analysis

Though macrobenchmark analysis can reveal the complex interaction among components in the system, it cannot address specific subsystem issues. For example, the results shown in figure 3 and 4 can only underline the poor performance experimented by KVM with more than 6 VMs or the very poor scores of Xen in the 1 VCPU configuration, but these results can hardly be used to understand the causes of this behaviour.

To partially address this problem we used *bzip*, *netperf*, and *dd* microbenchmarks to get a better understanding of CPU, network, and disk subsystem performances. These microbenchmarks are intrisically serial and therefore each guest is assigned one virtual CPU only.

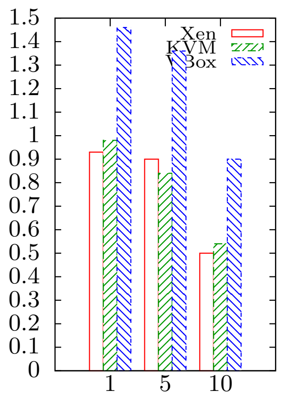

Fig. 5. *bzip2*

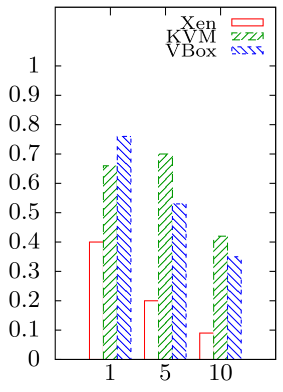

Fig. 6. *netperf*

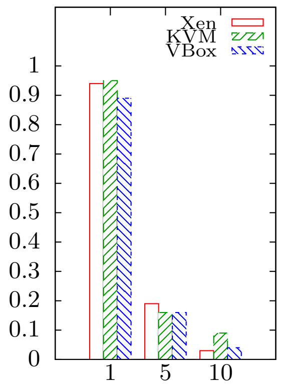

Fig. 7. *dd*

Figures 5, 6, 7 report the results of the bzip, netperf, and dd respectively for an increasing number of concurrently executing virtual machines (for the sake of brevity only results for 1, 5, 10 VMs are shown; results are normalized to native Linux performance).

In all the microbenchmarks, KVM performs quite well on the average, and in particular it obtains good results virtualizing the network layer (10% better than Virtual Box which is the second place) and in virtualizing the disk (particularly with an high number of VMs), while it performs slightly worse than both Xen and Virtual Box in virtualizing the CPU (5% worse than Xen).

Virtual Box scores well in both disk and network benchmark, while it outperforms others in the CPU benchmark, even with a high number of VMs. Surprisingly, Virtual Box performs also better than non-virtualized Linux. A further analysis and monitoring of Virtual Box execution showed that VBox runs the thread performing the CPU intensive task at kernel privilege level (in close cooperation with the VBox driver), thus avoiding many of the checks normally performed by the kernel when dealing with the userspace. This has a huge impact on performance and Virtual Box therefore obtains very low compression times. This very good CPU virtualization is likely to influence the rising of Virtual Box performance in the macrobenchmark analysis (see figure 4) for an high number of VMs.

The Xen case is particularly interesting as it obtains comparable results with KVM in the bzip and dd benchmark, while it performs very poorly in the virtualization of the network subsystem. The poor network layer virtualization performed by Xen was already reported by Apparao *et al.*[5]. Although possible, it seems unlikey that the huge overhead in Xen network virtualization is the unique cause for the low Xen performance underlined by the SPEC web macrobenchmark.

The comparison of micro and macrobenchmark results shows that virtualization solutions present behaviours which are either unexpected or difficult to explain in an appropriate way. A deeper analysis is needed in order to pinpoint the sources of overheads and to explain such unexpected performance results. In non-virtualized systems this analysis is generally done through profiling or online monitoring of the system under testing (e.g., through OProfile, `top`, `sar`), but for virtualized systems only few tools are available (XenOprof [14], Xenmon [12]) and often they can be used only with *paravirtualized* solutions (e.g., Xen PV). Tools such as XenOprof works by monitoring platform performance counters and by accounting through software structure the correct performance values to guest domains. Guest domain kernels need to be *modified* in order to effectively exploit XenOprof information. To the best of our knowledge, none of the virtualization profiling tools support performance counter virtualization nor allow guest domains to access HW performance counters. Furthermore, accounting per-domain statistics using system wide performance counters has severe performance impact and is impractical when profiling paravirtualized domains with very demanding workloads.

6.4 Virtualization-aware hardware performance counters

To overcome these limitations, we propose a novel virtualization-aware hardware performance counter architecture. In this architecture, which share some similarities with the performance counter for virtualized NICs architecture [6], a configurable number of hardware performance counters (up to the physically available performance counters) can be registered for each guest by the hypervisor in the Virtual Machine Control Structure of the guest and exported through, for instance, the VCPU structure. Each virtualized performance counter is updated only when the controlling guest is executing (VMCS is `current` and `active`) and overflows are reported to the guest through the normal interrupts delivery flow. The guest is thus able to exploit non modified monitoring and profiling tools which can correctly reports statistics of the guest execution. Furthermore, gathering all performance counter statistics for guests and hypervisor gives a system-wide profiling of the machine.

7 Conclusion

In this paper we have evaluated different open-source full virtualization solutions by using a SPEC Web benchmark and three microbenchmarks. We have proposed a macrobenchmark-based approach to virtualization performance-evaluation that uses SPEC Web to generate a *similar*, simultaneous workload on all main virtualized components of a system; this particular workload helps in reducing the number of variables in the performance analyisis and in identifying overheads in the hypervisor. Macrobenchmark results have been integrated with microbenchmark tests which focus on single virtualized subsystems in order to get a better understanding of virtualization solutions performance.

KVM provides the best performance in both efficiency and scalability tests, with Xen (2 VCPUs) following closely. Virtual Box provides a very good CPU virtualization which allows it to outperform other solutions with a high number of concurrently executing VMs. Surprisingly, Xen (1-VCPU) performs very poorly with a number of VMs greater than 4.

To analyze unexpected behaviours like this one, on-line monitoring tools and profiling are needed. We have underlined the current limits of available tools and we have proposed a sketch of a virtualization-aware hardware performance counter architecture which could help in overcoming those issues.

References

1. Adair, R., Bayles, R., Comeau, L., Creasy, R.: A virtual machine system for the 360/40. Tech. Rep. 320-2007, IBM Cambridge Scientific Center, Cambridge, MA (May 1966)
2. Adams, K., Agesen, O.: A comparison of software and hardware techniques for x86 virtualization. In: ASPLOS-XII: Proc of the 12th international conference on Architectural support for programming languages and operating systems. pp. 2–13. ACM, New York, NY, USA (2006)

3. AMD: AMD64 Architecture Programmer's Manual Volume 2: System Programming (September 2007)
4. Apparao, P., Iyer, R., Zhang, X., Newell, D., Adelmeyer, T.: Characterization & analysis of a server consolidation benchmark. In: VEE '08: Proc of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments. pp. 21–30. ACM, New York, NY, USA (2008)
5. Apparao, P., Makineni, S., Newell, D.: Characterization of network processing overheads in xen. In: VTDC '06: Proceedings of the 2nd International Workshop on Virtualization Technology in Distributed Computing. p. 2. IEEE Computer Society, Washington, DC, USA (2006)
6. Arndt, R.L., Craddok, D., Fuhs, R.E., Gregg, T.A., Schmidt, D.W., Walk, B.M.: Performance counter for virtualized network interface of communications network (June 2009), United States Patent 7548964
7. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A.: Xen and the art of virtualization. In: SOSP '03: Proc of the 19th ACM symposium on Operating systems principles. pp. 164–177. ACM, New York, NY, USA (2003)
8. Benjamin Quetier, Vincent Neri, F.C.: Selecting a virtualization system for grid/p2p large scale emulation. In: Proc of the Workshop on Experimental Grid testbeds for the assessment of large-scale distributed applications and tools (EXPGRID'06), Paris, France (2006)
9. Bershad, B., Draves, R., Forin, A.: Using microbenchmarks to evaluate system performance. Workstation Operating Systems, 1992. Proc of Third Workshop on pp. 148–153 (Apr 1992)
10. Camargos, F.L., Girard, G., Ligneris, B.D.: Virtualization of linux servers: a comparative study. In: 2008 Ottawa Linux Symsposium. pp. 63–76 (July 2008)
11. Clark, B., Deshane, T., Dow, E., Evanchik, S., Finlayson, M., Herne, J., Matthews, J.N.: Xen and the art of repeated research. In: ATEC '04: Proc of the annual conference on USENIX Annual Technical Conference. pp. 47–47. USENIX Association, Berkeley, CA, USA (2004)
12. Gupta, D., Gardner, R., Cherkasova, L.: Xenmon: Qos monitoring and performance profiling tool. Tech. Rep. HPL-2005-187, HP Laboratories Palo Alto (October 2005)
13. Standard Performance Evaluation Corporation SPECweb2005 v1.20: http://www.spec.org/web2005/
14. Menon, A., Santos, J.R., Turner, Y., Janakiraman, G.J., Zwaenepoel, W.: Diagnosing performance overheads in the xen virtual machine environment. In: VEE '05: Proc of the 1st ACM/USENIX international conference on Virtual execution environments. pp. 13–23. ACM, New York, NY, USA (2005)
15. Neiger, G., Santoni, A., Leung, F., Rodgers, D., Uhlig, R.: Intel(r) virtualization technology: Hardware support for efficient processor virtualization. Intel Technology Journal 10(3), 167 – 178 (August 2006)
16. Padala, P., Zhu, X., Wang, Z., Singhal, S., Shin, K.G.: Performance evaluation of virtualization technologies for server consolidation. Tech. Rep. HPL-2007-59, HP Laboratories Palo Alto (April 2007)
17. Popek, G.J., Goldberg, R.P.: Formal requirements for virtualizable third generation architectures. Commun. ACM 17(7), 412–421 (1974)
18. VMware: A performance comparison of hypervisors. Tech. rep., VMware, Inc. (2007)
19. XenSource: A performance comparison of commerical hypervisors. Tech. rep., XenSource, Inc. (2007)