Journal of Applied Computing Research, 2(2):75-82
July-December 2012

Monitoring and analysis of performance impact in virtualized environments

Pedro Freire Popiolek, Odorico Machado Mendizabal
Universidade Federal do Rio Grande. Av. Itália, km 8, 96203-900, Rio Grande, RS, Brazil.
p.f.popiolek@furg.br, odoricomendizabal@furg.br

Abstract. Recently, virtualization is becoming increasingly popular due to its wide adoption in cloud computing. This success stems from the fact that the infrastructure scales better and configuration and recovery tasks become simpler. Despite the benefits, the performance impact due to virtualization represents a major challenge for the establishment of accurate Service Level Agreements (SLA). This work presents an approach for monitoring and analysis of performance in virtualized environments. Experiments with generation of synthetic workloads in our monitored infrastructure allowed us to identify the main causes of the observed degradation of performance. Further, some specific behaviors were noticed for different load profiles.

Key words: performance analysis, performance monitoring, virtualization.

Introduction

Most of modern computational systems use largely distributed infrastructures, which allow the development of collaborative applications and sharing of remote resources. Such systems have as their main goal increasing performance, scalability, and availability of delivered services, as well as providing better resources usage.

In this context, cloud computing, aligned to virtualization, allows a set of physical servers to host dozens or hundreds of virtual machines (VM). Thus, the system scalability is increased, maximizing resources usage. A great challenge for those infrastructures, though, is to handle service requests without overloading host machines. Once a physical machine runs out of its limits, graceful degradation and instability could affect the performance of hosted VMs.

This paper describes the first steps towards development of mechanisms capable to monitor and proactively detect performance bottlenecks in cloud computing infrastructures. These mechanisms would be able to predict instability and performance degradation in advance, allowing adoption of efficient policies for dynamic management of resources.

As preliminary research, it is proposed a study on the use of native techniques for monitoring resources in Linux and Windows operating systems. Based on experiments, this work identifies a minimal set of performance counters useful to identify performance bottlenecks for such kind of infrastructures. Experiments measure the performance of guest Windows VMs on the KVM/QEMU virtualization environment.

The present paper extends the results presented in (Popiolek and Mendizabal, 2012). With respect to the original version, we detailed the use of performance counters, their meaning and equivalences between Windows and Linux metrics. Moreover, additional experiments were carried out with the aim of analyzing memory management in virtualized environments.

The experiments presented in this paper show the benefits of using native performance counters to monitor VMs performance. It is possible to observe memory usage profiles, such as throughput and memory allocation. We also observed that I/O operations represent a severe penalty on performance. We find that the idle time for waiting I/O operations can be more than double of the processing time of a VM.

The remaining of this paper is organized as follows. The next section introduces the concept of virtual machines. The section "Monitoring techniques" presents native monitoring tools for Windows and Linux. The executed experiments are presented in the section "Experiments". Results and related work are discussed in the section "Discussion". Conclusions about this work and future directions are presented in the section "Conclusion".

Virtualized environment

This section provides a short introduction to server virtualization. The main concepts applied to virtual machines are presented. Further, the impact of virtualized environments in the overall system performance is discussed.

Virtual machines

The massive use of virtual machines (VM) is one of the main characteristics of cloud computing infrastructures. Virtualization technique allows the partitioning of physical resources among virtual servers that execute concurrently in a same physical machine. Among the advantages of this approach there are the increase in infrastructure scalability and improvement of resources usage.

Virtualized environments are composed by a host operating system (OS) running over the physical hardware, guest OS instances that run over virtual hardware profiles, and there is a Virtual Machine Monitor (VMM), also known as hypervisor, that coordinates the VM instances. More precisely, the VMM is responsible for the management of the shared resources in the physical machine (for instance, CPU, memory and I/O devices). Among other tasks, the VMM is also responsible for the scheduling of VMs and handling of interruptions generated by VMs. In other words, it provides transparency of shared resources to guests OS (Smith and Nair, 2003). Figure 1 shows a potential arrangement of the described elements.

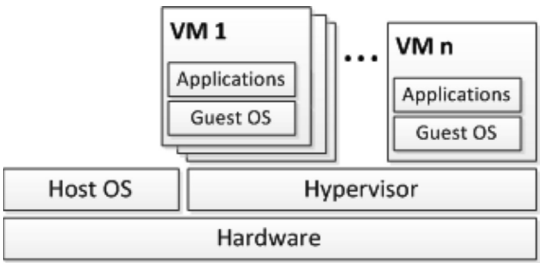

Figure 1. Hybrid virtualization.

Performance

It is not unusual to have heavy workload being generated into virtualized environments, assuming the same computer architecture can be configured either with virtual machines or with physical machines only. Whether both infrastructures are exposed to similar workloads, the former setup will not perform as well as the latter, but the intensity of performance decrease depends on the workload generated by applications. In short, the additional software layers responsible for emulation, virtualization of devices and the concurrency level among the guests result in performance cost (McDougall and Anderson, 2010).

Another important factor responsible by the decrease on performance in virtualized environments, is the scheduling performed by the VMM. Usually, the scheduling policy assigns higher priority to partitioning of processor resources among active VMs than scheduling of I/O operations. For this reason, I/O-bound processes have worst performance than those CPU-bound (Pu *et al.*, 2010).

In order to reduce the performance cost occasioned by virtualization, some cloud computing providers have their own hypervisors implemented. For example, Amazon has a customized version of Xen Hypervisor and Windows Azure uses the Wazure Hypervisor, a non-commercial hypervisor, developed by the company. They benefit from a total knowledge about the infrastructure and platform where the cloud is deployed. Although it allows better resources usage, the virtualization could still introduce undesirable performance costs for the infrastructure in certain situations. Therefore, monitoring of infrastructure at run time would be needed to detect these situations.

Monitoring techniques

This section describes how to observe system performance using native tools from operating systems. Performance metrics are acquired by monitoring tools that collect data regarding hardware and OS resources usage. The analysis of sample data, collected by regular intervals, allows one to identify system behaviors, detect performance bottlenecks and figure out the root cause for a particular behavior.

However, for an accurate analysis based on resources monitoring, a solid knowledge about the available metrics is required. Common metrics provided by performance counters are related to processors, main memory, hard disk and network. Specific metrics would be required for analysis of application with specific purposes (e.g. IIS server's counters for analysis of a web application server). HP presents a detailed study about monitoring metrics for Windows operating systems (HP, 2009).

Windows operating systems offer a native monitoring tool, called Performance Monitor. In this tool metrics are represented by objects and each available object may have one or multiple instances. Instances are used to differentiate non-unitary elements of the system (e.g. instances may differentiate processors' cores or multiple hard drives in the system).

In Linux operating systems it is possible to extract monitoring data of system resources from virtual files existent in/proc/folder. However, user-friendly tools presented in the sysstat package automates this task. The main monitoring tools for Linux systems are: Sar, Iostat e Pidstat.

Although the collected metrics may differ from an operating system to another, usually they have a similar meaning. Also, it is possible an operating system to have exclusive metrics, but with proper correlations, these metrics may have equivalent meaning to metrics of another operation system.

Tables 1 and 2 present performance counters for monitoring CPU and disk usage, respectively. The equivalent metrics for Windows and Linux are described, as well as their definition.

Experiments

In this section, we analyze performance of virtualized environments using performance counters monitoring. Some characteristics of the resource management performed by the system are discussed. Synthetic workloads were applied into virtual machines in order to reproduce specific usage profiles, such as those represented by CPU-bound and I/O-bound processes.

Settings

In our experiments the test environment is composed by 1 host machine with AMD FX-6100 processor (6 cores), 8 GB of RAM, 500 GB of hard disk, operating system Ubuntu 12.04-64 bits and KVM hypervisor. The guests, or hosted virtual machines, were configured with QEMU 64 processor, 512MB and 1GB of RAM, 20 GB of virtual hard disk pre-allocated, and operating system Windows XP Professional SP3 32 bits.

The scenarios executed differ by the number of active VMs, varying from 1 up to 6 VMs running concurrently, and by the kind of workload. The modeled workload profiles represent CPU-bound processes, I/O-bound processes and memory-bound processes. In all, eighteen different scenarios are demonstrated and for all scenarios, each active VM was exposed to exactly the same workload.

CPU - Bound

In order to reproduce CPU-bound workload it was used the WPrime Benchmark v2.09 (wPrime, 2012). We set up processing option as "1024M" during the tests, i.e. the square root of the first 1024 million of numbers is calculated by the benchmark. Before starting the analysis of resources usage in virtualized environments, observe the average processing time for executions of this benchmark in Table 3.

As the number of concurrent VMs in execution increases, the benchmark takes longer to finish. The VMs scheduling performed by VMM and greater competition for resources shared by the host are the main causes for the additional time to finish each individual benchmark execution.

According to performance monitors observation, the more VMs execute concurrently, the percentage of CPU time allocated to each VM decreases, and the percentage of time for host execution in privileged mode increases. Figure 2 depicts percentage of CPU execution in guest mode (VMs), privileged mode (OS operations), and idle time.

Observe that the load of the host machine increases linearly (it can be better visualized through the linear decay of % idle CPU). However, due to a costly VMs management performed by VMM, the effective use of CPU by VMs does not increase linearly (see the percentage of CPU execution in guest mode).

I/O - Bound

In order to represent processes predominantly I/O-bound, we used the CrystalDiskMark 3.0.1 benchmark (CrystalMark, 2012).

Table 1. Main CPU performance counters for Windows and Linux.

Windows	Linux	Description
Performance monitor	**Sar**	
% Processor Time	%user + %nice + %system	Average percentage of CPU utilization.
% Privilege Time	%system	Average percentage of CPU usage in Kernel mode.
% Interrupt Time	%irq + %soft	Average percentage of CPU utilization receiving and serving interruption.
Context Switches/sec	cswch/s	Average rate of context switches per second.
Processor Queue Length	runq-sz	Number of processes queued in the ready state.
% DPC Time	-	Average percentage of CPU utilization receiving and serving DPCs.
% User Time	%user	Average percentage of CPU utilization in user mode.
-	%usr	Average percentage of CPU utilization in user mode. Does not include time spent on virtual processors.
-	%nice	Average percentage of CPU utilization in user mode with positive nice priority.
-	%steal	Average percentage of virtual CPU involuntary waiting to be answered by the Hypervisor.
-	%irq	Average percentage of CPU utilization serving hardware interrupts.
-	%soft	Average percentage CPU utilization serving software interrupts.
-	%iowait	Average percentage of idle CPU waiting for pending I/O disk request.
-	%guest	Average percentage of CPU utilization running virtual CPU.
% Idle Time	%idle	Average percentage of idle CPU.
DPCs Queued/sec	-	Average queuing DPCs per second.
Interrupts/sec	intr/s	Average rate of interrupts per second.

Table 2. Main disk performance counters for Windows and Linux.

Windows	Linux		Description
Performance monitor	**iostat**	**df**	
% Idle Time	-	-	Percentage of time the disk stays idle.
(Disk Bytes/sec) / 1024	(rKB/s) + (wKB/s)	-	Number of Kilobytes write/read per second.
Disk Transfers/sec	(r/s) + (w/s)	-	Number of disk requests per second completed.
Split IO/Sec	-	-	Number of requests per second that were divided into multiple requests.
Free Megabytes	-	Available	Megabytes available for use in the storage unit.
Avg. Disk sec/Transfer	Await	-	Average time to complete a request (queue time + service time).
Avg. Disk Queue Length	avgqu-sz	-	Average size of the queue of requests waiting for the hard drive.

Table 3. Average duration of CPU-bound experiments.

VMs running	Average processing time (minutes)
1	34.98
2	34.04
3	34.82
4	39.70
5	43.58
6	48.54

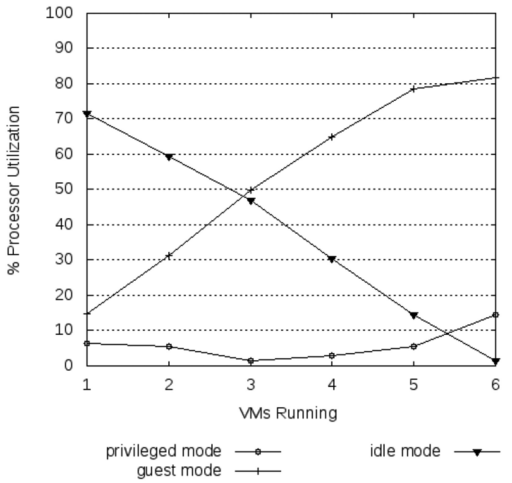

Figure 2. Monitoring CPU for CPU-Bound experiments.

For our test scenarios, the modeled workload processes 2000 MB of sequential read/write operations. Figures 3 and 4 shows, respectively, read and write rates for each experiment[1].

Although the same workload has been applied individually to each VM, the read and write rates perceived by them are distinct. Analogously to the previous experiment, it is possible to observe that as the number of concurrent VMs increases, the overall performance decreases. Observe an expressive decrease in the average read and write rates in Figures 3 and 4.

Differently from CPU-bound processes, Figure 5 shows that the increase of CPU execution in guest mode is not proportional to the number of VMs in execution. This fact can be explained by the percentage increasing of I/O operations waiting for the hard drive.

Even though each VM has its own virtual disk, all of them share the same hard disk. Thus, an expressive number of read and write requests to this storage device incurs in per-

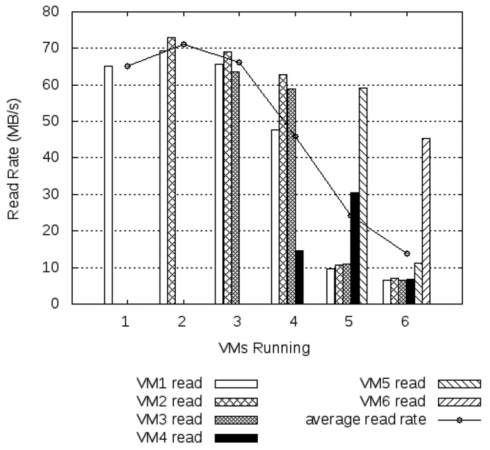

Figure 3. Memory read rate of I/O-bound experiments.

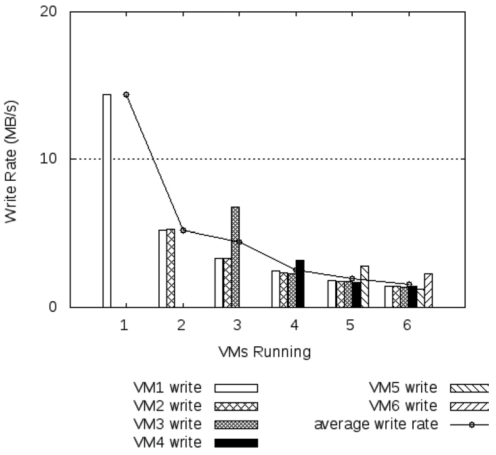

Figure 4. Memory write rate of I/O-bound experiments.

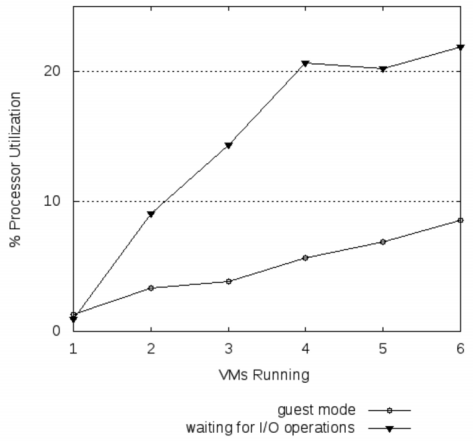

Figure 5. Monitoring CPU for I/O-Bound experiments.

[1] Once the acquired results present high deviations for each active VM, we prefer to exhibit measured value for each VM individually.

formance reduction for every active VM requesting for I/O. Observe that I/O operations represent a severe penalty on performance. The idle time for waiting I/O operations is more than double of the processing time of a VM.

This performance issue can be confirmed in the host machine by the average disk queue length (see Figure 6). This competition for resources in the physical machine explains the non-linear increase on CPU usage, as well as the decay of processing usage by VMs.

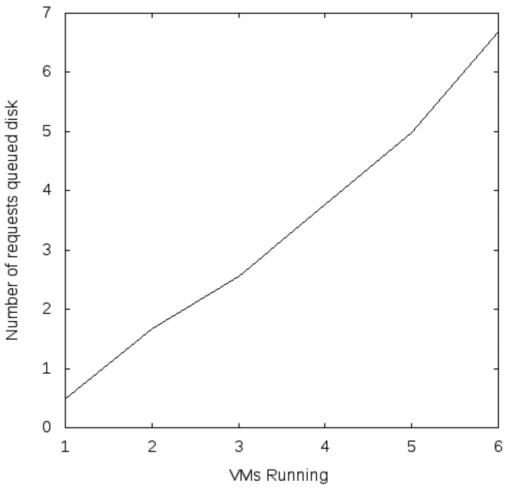

Figure 6. Monitoring disk queue.

Memory consumption

Memory access performance was analyzed in our experiments with support of RAMspeed benchmark (RAMspeed, 2012). The component FLOATmen executes uninterruptedly a set of operations: Copy (A = B), Scale (A = m*B), Add (A = B + C) and Triad (A = m*B + C). In our test scenario, 1 VM is started every 5 minutes, and 2 minutes after the VM initiates the benchmark is started.

Figure 7 depicts the amount of free memory measured in the host. VMs were configured with 512 MB of RAM. The VMM allocates a portion of memory with the total amount of memory requested by VM right after the VM initialization.

The similar experiment was conducted using VMs with 1 GB of RAM. As expected, the same memory management behavior was observed (see Figure 8). As soon as the VM is finished, the total amount of memory it was using will be disposed.

This experiment identifies a memory threshold given by the free memory counter. No additional VMs should be initiated if free memory is insufficient.

Another metric acquired in this test is the memory access throughput. We observed how much memory is accessed by VMs during the experiment. Table 4 shows the average rate of read and write operations, and Figure 9 shows the total throughput. As the number of VMs increases, the individual throughput reduces. However, observe that the overall throughput of the host is increased with more VMs running concurrently.

The total throughput measured in the server host, depicted in Figure 9, shows the maximum throughput for memory operations is reached with 3 or more VMs running at the same time. The RAM size of VMs does not affect the memory operation's rate.

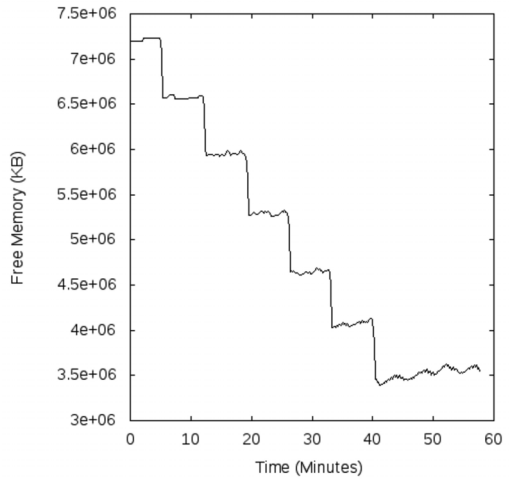

Figure 7. Memory consumption on host OS with 512MB RAM VMs.

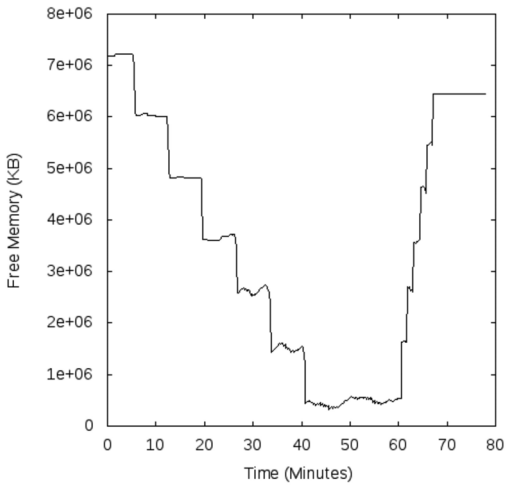

Figure 8. Memory consumption on host OS with 1024MB RAM VMs.

Table 4. Average rate of memory operations in FLOATmem experiments.

VMs	Average rate of operations (MB/s)		Maximum throughput (VMs * Average)	
	(512 MB)	(1024 MB)	(512 MB)	(1024 MB)
1	5980.0	6012.8	5980.0	6012.8
2	4474.6	4417.5	8949.2	8835.0
3	3337.4	3317.1	10012.2	9951.3
4	2537.6	2518.7	10150.4	10074.8
5	2022.7	2025.8	10113.5	10129.0
6	1666.6	1704.9	9999.6	10229.4

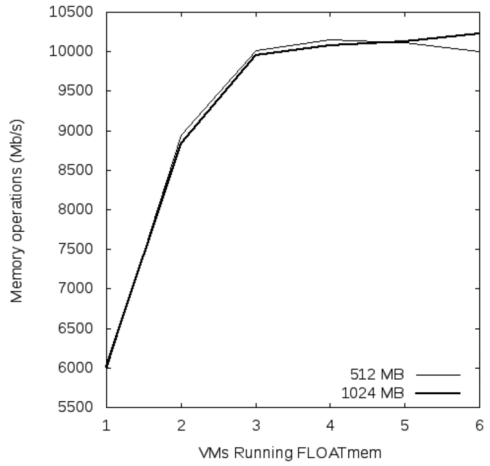

Figure 9. Memory access throughput.

Discussion

The results presented in this paper reaffirm that virtualization of servers add some cost to the system's performance. However, with careful observation of metrics acquired by performance counters it is possible to disclose the root causes and associated effects responsible for performance degradation of the system. Besides the set of performance counters presented in this paper, other counters were also collected. There are evidences of correlation among observed metrics, but this aspect has not been yet analyzed thoroughly in our research.

Statistical analysis over sample data will help us to determine existent correlations among metrics in future experiments. Furthermore, we expect to establish thresholds in which the system can operate without meaningful loss of performance. Other metrics would also be studied, such as performance metrics for network interfaces and metrics collected by multiple storage interfaces.

A similar study was presented by Pu *et al*. (2010). Their paper show workloads applied to one VM that affects the performance of another one. They also used application profiles with intensive I/O operations, but their study was focused on network performance. Regola and Ducom (2010), analyzed the virtualization impact for high performance computing. In their paper is presented both I/O operations performance, for disk and network.

Our research is restricted to CPU, disk and memory analysis. The use of native tools for monitoring the system performance is less intrusive. That is an important requirement once we intend to detect performance issues at run time.

Conclusion

Although resource management in cloud computing paradigm benefits from a dynamic infrastructure, impacts on performance due to virtualization cannot be easily detected. Thus, ensuring rigid SLAs for services running in those infrastructures constitutes a hot topic for researchers.

This work presents a suitable approach to monitor performance of virtualized environments running Windows or Linux platforms. By execution of experiments, it was possible to highlight some collateral effects caused by virtualization. Metrics collected by performance monitors allowed us to identify the root cause of the perceived issues.

Our experiments exercised the target infrastructure through workloads generated by benchmarks. The load profiles modeled CPU-bound and I/O-bound processes.

With this approach we observe the behavior of virtualized infrastructures in extreme situations. Although those workloads, in terms of high intensity and invariability type, are unusual for the most applications, they are very useful for understanding performance issues regarding specific bottlenecks and identifying correlation of performance counters.

Hereafter, performance counter correlations will support the development of algorithms capable to detect whether performance of a given VM is going to be affected with reasonable precedence. Assertions about performance obtained by this kind of algorithms allow resource management policies to prevent performance degradation and, as a consequence,

infrastructure would be able to safely attend rigid SLAs.

Following this work, we plan another round of experiments with more realistic workload profiles. Once we have determined performance counters correlations and performance thresholds associated to a set of metrics, we would expand our analysis for the workload of other applications, such as those defined by TPC consortium (TPC, 2012).

With these and future works, we intend to build a prevention mechanism capable to detect performance degradation at run time. This will allow the preventive migration of virtual machines to healthier servers, just before critical usage resources have been detected in the original server.

References

CRYSTALMARK. 2012. CrystalMark Available at: http://crystalmark.info/software/CrystalDiskMark/indexe.html. Accessed on: August 10, 2012.

HP. 2009. Performance Engineering Best Practices. Available at: http://h30499.www3.hp.com/t5/HPLoadRunner-and-Performance/HP-Performance-Engineering-Best-Practices-Series/bap/2407627. Accessed on: August 10, 2012.

MCDOUGALL, R.; ANDERSON, J. 2010. Virtualization Performance: Perspectives and Challenges Ahead. *SIGOPS Operating Systems Review*, **44**:40-56. http://dx.doi.org/10.1145/1899928.1899933

POPIOLEK, P.F.; MENDIZABAL, O.M. 2012. Monitoramento e Análise do Impacto no Desempenho em Ambientes Virtualizados. *In:* ESCOLA REGIONAL DE REDES DE COMPUTADORES, 10, Pelotas, 2012. *Proceedings…* Pelotas, p. 11-14

PU, X.; LIU, L.; MEI, Y.; SIVATHANU, S.; KOH, Y.; PU, C. 2010. Understanding Performance Interference of I/O Workload in Virtualized Cloud Environments. *In:* IEEE INTERNATIONAL CONFERENCE ON CLOUD COMPUTING, 3, Miami, 2010. *Proceedings…* Miami, p. 51-58. http://dx.doi.org/10.1109/CLOUD.2010.65

RAMSPEED. 2002. RAMspeed, a cache and memory benchmarking tool. Available at: http://alasir.com/software/ramspeed/. Accessed on: October 1, 2012.

REGOLA, N.; DUCOM, J. 2010. Recommendations for Virtualization Technologies in High Performance Computing. *In:* IEEE INTERNATIONAL CONFERENCE ON CLOUD COMPUTING TECHNOLOGY AND SCIENCE, 2, Indianapolis, 2010. *Proceedings…* Indianapolis, p. 409-416. http://dx.doi.org/10.1109/CloudCom.2010.71

SMITH, J.E.; NAIR, R. 2003. An Overview of Virtual Machine Architectures. *Elsevier Science,* p. 1-21.

TPC. 2012. About the TPC. Available at: http://www.tpc.org/information/about/abouttpc.asp. Accessed on: August 10, 2012.

WPRIME. 2012. wPrime Multi-threaded Benchmark. Available at: http://www.wprime.net. Accessed on: August 10, 2012.

Submitted on October 18, 2012
Accepted on January 3, 2013