Evaluating the Impact of Virtualization on Performance and Power Dissipation

Francisco J. Clemente-Castelló, Sonia Cervera, Rafael Mayo and Enrique S. Quintana-Ortí
Department of Engineering and Computer Science, Jaume I University (UJI), Castellón, Spain

Keywords: Virtualization, Cloud Computing, KVM, Energy Consumption, Server Consolidation.

Abstract: In this paper we assess the impact of virtualization in both performance-oriented environments, like high performance computing facilities, and throughput-oriented systems, like data processing centers, e.g., for web search and data serving. In particular, our work-in-progress analyzes the power consumption required to dynamically migrate virtual machines at runtime, a technique that is crucial to consolidate underutilized servers, reducing energy costs while maintaining service level agreements. Preliminary experimental results are reported for two different applications, using the KVM virtualization solution for Linux, on an Intel Xeon-based cluster.

1 INTRODUCTION

Cloud computing is now a mainstream approach to accelerate application deployment and execution, with lower maintenance and total costs of ownership (TCO), increased manageability, and higher flexibility to rapidly adjust resources to a fluctuating demand. Cloud computing was originally conceived for applications and setups very different from those usually found in high-performance scientific computing. However, due to the recent advances in virtualization design and its smooth unfolding, cloud solutions are being also adopted for resource management in the high performance computing (HPC) arena.

Virtualization is the essential technology underlying cloud computing, which has resulted in the renaissance of a research line that was initiated in the 70s (Goldberg, 1974). Two key aspects that have further nourished the spur of virtualization are the development of production-level tools (e.g., KVM (Kivity et al., 2007) and Xen (Barham et al., 2003)) and the accommodation of hardware to support this technology (hardware-assisted virtualization) in a significant fraction of current processors from large architecture companies such as AMD, IBM or Intel.

Most virtualization efforts address the consolidation of underutilized dedicated physical servers and the implementation of tools for cloud computing. On the other hand, research & development projects that adopt virtualization as a tool for HPC are significantly more scarce. While there exist sound reasons for these primary studies, a needful prerequisite before embracing virtualization in HPC environments is an analysis of the impact of this technology on performance.

In this paper we present an experimental study of a virtual machine (VM) hypervisor, from the points of view of performance, productivity and, especially, power consumption. Specifically, our study examines the problems arising from live relocation of VMs, inspecting the actual effect on power of such strategy, for two well-known applications running on a three-node Intel Xeon-based cluster. For our study, we selected KVM (Kernel-based Virtual Machine), due to its low overhead, fair adoption, and its integration with the open source cloud computing platform OpenStack (Ope, 2013).

The rest of the paper is structured as follows. In section 2 we review several basic concepts, advantages and benefits of virtualization, and its connection with energy saving strategies via consolidation. In section 3 we briefly discuss KVM, the technology employed in our experimental evaluation. The main contribution of this paper is in section 4, where we present the experimental setup and preliminary results from our work-in-progress. Finally, in section 5 we close the paper with a few concluding remarks and a discussion of future work.

Clemente-Castelló F., Cervera S., Mayo R. and Quintana-Ortí E..
Evaluating the Impact of Virtualization on Performance and Power Dissipation.
DOI: 10.5220/0004837205130518
In *Proceedings of the 4th International Conference on Cloud Computing and Services Science* (CLOSER-2014), pages 513-518
ISBN: 978-989-758-019-2

2 VIRTUALIZATION AND CONSOLIDATION

System (or hardware) virtualization creates an abstraction layer on top of the physical machine (host) which allows one or more VMs (guests) to run separately from the underlying hardware resources. Current virtualization solutions rely on the concept of virtual machine monitor (VMM), or *hypervisor*, that is in charge of virtualizing the hardware and executing the VMs, acting as a firewall between these two components.

There exist two alternatives for virtualization (Goldberg, 1974). In one of these, the hypervisors interact directly with the hardware. The hypervisor is in this case a kernel-mode driver (or module) of the host operating system (OS). In the alternative one, both the hypervisor and the VMs execute on a top of a standard OS which offers access to the host resources, including its devices. The advantage of the first option is clear: since it provides direct access to the hardware, avoiding interaction through multiple software layers, in principle its peak performance can be close to that attained from a native execution. Xen, KVM and VMWare (Nieh and Leonard, 2007) are examples of the first type of virtualization, while QEMU (qem, 2013) and VirtualBox (vir, 2013) follow the nonnative alternative.

There are several important aspects that have to be considered when applying virtualization (Younge et al., 2011; Younge et al., 2010), independently of whether the target is a data processing center, where throughput is the fundamental driving force, or an HPC facility, which may be willing to trade-off workload productivity for application performance. One particular aspect asks for an assessment of the balance between the costs (i.e., negative performance impact) and the benefits of accommodating virtualization. Fortunately, many processor architectures nowadays feature hardware support to amend possible penalties resulting from virtualization. Furthermore, there is a continuous research to address these overheads also from the software viewpoint (e.g., paravirtualization that offers virtualized memory addresses; previrtualization software to adopt a certain hypervisor while, simultaneously, maintaining compatibility with the physical hardware; etc.)

On the other hand, applications can also benefit from embracing virtualization. For example, an OS can be tuned to improve application performance by letting the hypervisor control allocation of resources among applications such as a fixed memory space, a certain fraction of the processor cycles or, in VMs running with real-time constraints, the maximum latency for interrupt handling. In most of these situations, as a single VM runs in a virtual environment isolated from those of other VMs, an application failure will only affect the causing VM. Thus, in case the VM cannot recover, all its resources can be reallocated by the hypervisor to a different VM.

An additional advantage of virtualization is derived from the possibility of running a collection of the logical nodes of a virtual *cluster* concurrently on a smaller number of physical nodes of an actual cluster. Under certain conditions (workload characteristics, service level agreements, etc.), this in turn enables the deployment of a virtualization-aware energy-saving strategy where servers are consolidated on a reduced number of physical machines, which may render a reduction of energy consumption, both by the processing equipment and the infrastructure (e.g., cooling, UPS, etc.). A step further in this line is to adopt a dynamic strategy for consolidation that adaptively selects the number of active physical servers depending on the workload, migrating VMs to allow consolidation, and turning on/off unused nodes (Kusic et al., 2009).

Live migration of VMs is crucial to leverage the energy savings potentially yielded by server consolidation. In this process, a VM that is running on a physical server A is migrated to an alternative server B, transparently to the user that is running his applications in the VM. For this purpose, i) all the memory in use by the VM has to be migrated from A to B; next ii) those memory pages that were modified by the VM on A since the migration started are copied to B; and finally iii) the process is completed with the transfer of the current processor state for the VM from A to B.

3 KVM

KVM is an open source software for full virtualization of x86 hardware. From the implementation point of view, it is a Linux kernel module that operates as a type-I hypervisor, providing the functionality that is needed to run VMs on the host platform. The integration of KVM into Linux offers two major advantages: first, all enhancements to Linux can be automatically leveraged from KVM; and second, KVM developers only need to tackle with the optimization of the applications running on the VMs being thus isolated from the underlying software layer (OS).

The main characteristics of KVM are:

Scheduling, Resource Control, and Memory Management. VMs run in KVM as regular Linux processes. Therefore, all the kernel-level man-

agement tools in the OS are also applicable to VMs. This includes scheduling, resource control and memory management, among others. Thus, for example, improvements to Linux such as process priority, CFS (*Completely Fair Scheduler*) and group control, which allow a fine-grain control of scheduling, can be easily leveraged to guarantee a certain quality of service (QoS) in VM management.

Storage. Images of VMs are treated in KVM as any other Linux regular file. VMs are thus stored using the standard mechanisms and media available in Linux, which includes local disks as well as high performance network storage systems.

Support for New Hardware Features. KVM inherits the entire Linux system so that all devices supported by Linux are supported by KVM as well. This permits the interoperation with systems that comprise a large number of CPUs and large amounts of RAM.

Security. VMs run as Linux system processes in KVM. Therefore, any single VM is protected from malicious execution of other VMs. Even more important is that the hypervisor is protected from such malicious behaviour.

4 EXPERIMENTS

In this section we present the experiments that were performed to assess the impact that the adoption of virtualization exerts on the performance and power dissipation of two applications with quite different properties. One particular aspect that is analyzed is the overhead incurred during live migration of VMs since, as argued at the end of section 2, this is crucial to consolidate physical servers and thus attain energy savings.

4.1 Hardware Setup

All the experiments were performed on a cluster composed of three HP Proliant DL120 G6 servers, with a single Intel Xeon X3430 processor (four cores operating at 2.67 GHz with a power dissipation of 95 Watts), 4 GB of RAM, and two Fast Ethernet cards per server. All nodes operated under 64-bit Linux Ubuntu 10.04LTS, and were connected via a Fast Ethernet switch.

The hypervisor was KVM version 0.12.3, with support for live migration of VM. Created instances of VMs featured a single-core `x86_64` CPU with 512 MB of RAM, running 64-bit Linux Ubuntu 11.10.

Power was measured using an external powermeter *Watts up? .Net* with a sampling rate of 1 Hz. The measures were collected for the full execution of the test, and the average power was multiplied by the execution time to obtain the energy consumption in Watts-hour.

4.2 Application Benchmarks

We employed the following two benchmarks for the evaluation:

AB. A significant part of today's servers perform data and web searches. We therefore selected an Apache server (Fielding and Kaiser, 1997) to evaluate the effect of virtualization on this type of services. Performance was measured for this test in terms of average time to serve requests (response time) from the clients.

MVB. Many scientific applications running on HPC platforms occupy a relevant fraction of CPU and memory resources of the target system. Performance, in terms of time-to-solution, is the key factor for these applications. Therefore, our goal here is to measure the impact of virtualization and live migration on the execution time of this type of computations. For that purpose, we chose a simple linear algebra operation, the matrix-vector product, which is representative of many memory-bounded computations that underlie more complex scientific computations. The actual performance is reported as the rate of GFLOPS (billions of floating-point arithmetic operations per second), which is inversely proportional to the time-to-solution.

When reporting results from both benchmarks, we also indicate the size of the VM that needs to be migrated.

4.3 Evaluation of Benchmark AB

In the experiments in this subsection, we evaluate the impact of live migration on performance by comparing two different scenarios. In one of them, (labelled as "VM" in the following four figures,) the Apache server runs in a VM placed on node A of the system, serving the web queries sent by a number of clients (users) running on node B. In the alternative scenario, (labelled as "VM+Migration",) the Apache server is migrated from node A to node C, while requests from the users on node B are simultaneously received. To avoid interferences, in this second scenario the migration occurs over a network different from that used to send queries and responses between

server and clients. Thus, the differences observed between both scenarios, for the duration of the VM relocation, are a clear indicator of the overhead introduced by the migration process.

Figure 1 reports the results with the benchmark **AB** with 20 and 120 users, that concurrently send requests to the server, for the period of time that comprises the VM migration plus some additional margins before the relocation commences and after it is completed. The traces show several relevant details. When there are only 20 users, the average response time is 0.889 s, while this value grows to 5.375 s for 120 users. It is also worth pointing out the consequences that the workload intensity has on the time that is necessary to migrate the VM. With 20 users (left-hand side plot), the VM that runs the server employs only 316 MB of memory, which results in 25 s to transfer it over the network and a total migration time of 31 s. The difference in this case, 6 s, is due to the time required to send the actual state of the VM. When the number of users grows to 120 (right-hand side plot), the size of the VM becomes larger, 441 MB, but the impact on the migration time is even more acute: 35 s for the network transfer and a total migration time of 64 s. One more aspect to note from the results in both figures is that, during the period of time in which the state of the VM is being migrated, there appears a much longer delay in the average response time. The reason is that, during this period, certain parts of the VM memory are modified while, simultaneously, others are being transferred and, as a consequence, the synchronizations necessary to maintain the consistency of the information have a relevant negative impact on the productivity.

Figure 2 shows the results from the same tests, but now with the option `keep alive` active that enables to use persistent connections to the server. Although the average response times are similar to those reported in the previous experiment, we note the increase of the memory utilized by the VM, for example, 508 MB for 120 users. The net outcome is a remarkable growth of the time required for the migration, which now requires 155 s.

4.4 Evaluation of Benchmark MVB

We have performed three types of experiments with benchmark **MVB**, to analyze the impact on both performance and power dissipation. In the first experiment, we simply evaluate the impact that the usage of VMs and live migration exerts on the GFLOPS rate of the operation. For that purpose, we consider three scenarios, depending on how/where the operation is run: i) directly executed on (one single core) of a node (label "Node"); ii) on a VM running on (one core) of a node (label "VM"); and iii) on a VM while it is migrated between two nodes (label "VM+Migration").

Figure 3 compares the results for this first experiment. As the matrix-vector product is a memory-bounded operation, the differences observed between the performances attained for the first two scenarios are quite small: 1.75 GFLOPS when the operation is run directly on the node vs. 1.70 GFLOPS if instead is executed by the VM. Furthermore, the migration only exerts a visible impact on the GFLOPS rate while the control is being transferred between the nodes, while the overhead is negligible for the rest of the relocation period.

The next experiments examine on the energy costs of virtualization and relocation. For reference, we initially analyze the power dissipated by a node that executes benchmark **MVB** using 1, 2, 3 and 4 cores as well as when the node is in different states/phases: shut off waiting for a Wake-on-LAN (WoL) signal, starting up, idle, and shutting down; see Figure 4. The corresponding values are captured numerically in Table 1. These results reveal that the power that is dissipated per core is gradually diminished as the the number of cores executing the workload is increased. For instance, the execution of the benchmark by a single core yields an increase of 22 W with respect to the idle state; on the other hand, the increase from 3 to 4 cores executing the workload results in an increment of power of only 10 W. A rapid conclusion from this study is that consolidation, when possible, can actually yield the desired energy-savings for this particular benchmark.

Table 1: Power dissipation and energy consumption of different states.

State	Duration	Power or Energy
Shut off/WoL	–	9 W
Starting up	75 s	1.42 Wh
Idle	–	50 W
Running **MVB** - 1 core	–	72 W
Running **MVB** - 2 core	–	90 W
Running **MVB** - 3 core	–	100 W
Running **MVB** - 4 core	–	110 W
Shutting down	35 s	0.48 Wh

The following experiment reports the power overhead incurred by virtualization of benchmark **MVB**. Specifically, in Figure5 we show two scenarios: one with the benchmark being executed directly on the node (label "Node") and an alternative with the benchmark running on a VM (label "VM"). In both cases, up to 4 cores/VMs are involved in the experi-

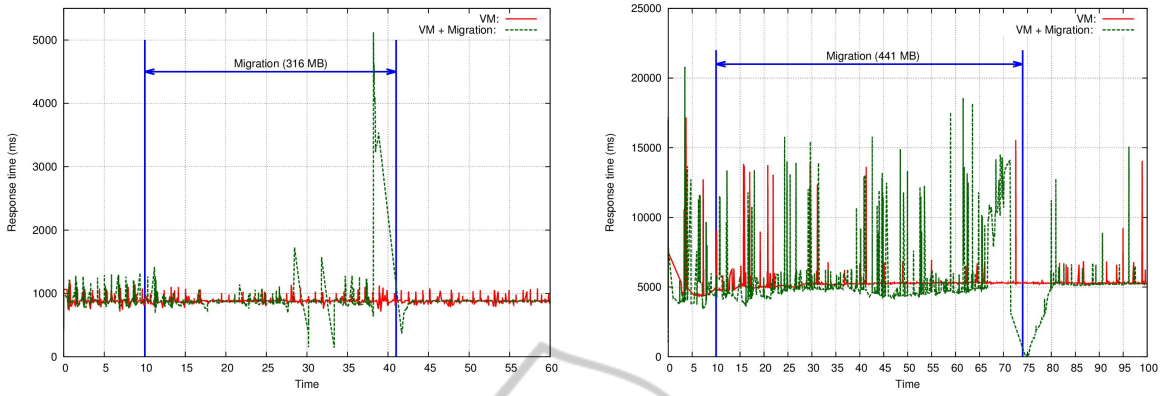

Figure 1: Response time for benchmark **AB** with 20 and 120 users (left and right, respectively).

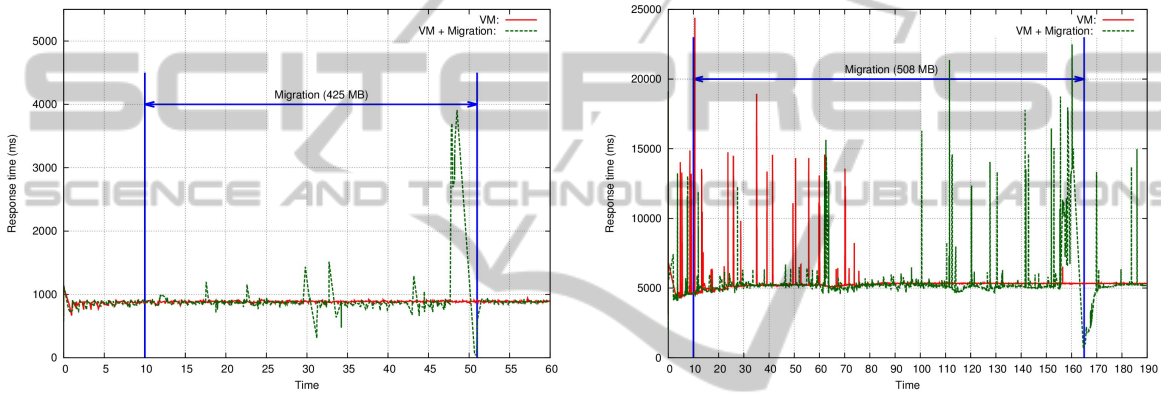

Figure 2: Response time for benchmark `AB` with 20 and 120 users (left and right, respectively) and *keep alive* active.

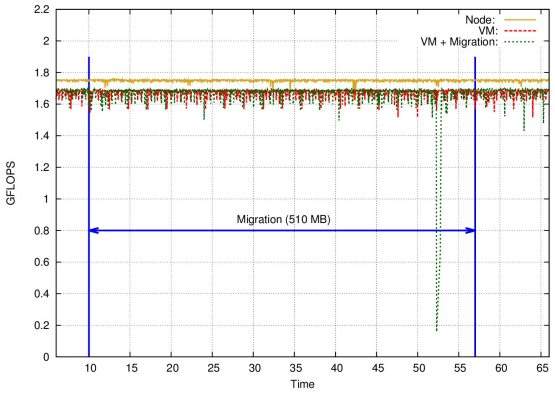

Figure 3: GFLOPS rate for benchmark **MVB**.

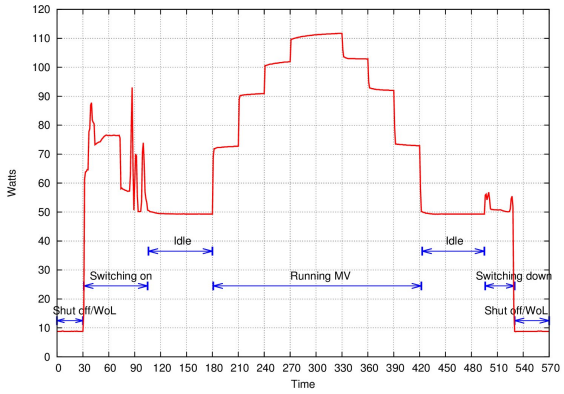

Figure 4: Power dissipation and energy consumption of different states.

ment. The trace reveals that the power differences between both scenarios are small, decreasing the number of active cores, and being in all cases lower than 2.5 W.

The final two experiments illustrate the overhead of VM relocation on power consumption. In the first one, we report the power dissipated by a node A during the relocation of an idle VM to a different node B, and (after 300 s) back from B to A; see the left-hand side plot of Figure 6. In the second test we repeat the experiment, this time while the VM is executing one instance of benchmark **MVB**; see the right-hand side plot of the same figure. Theses two traces clearly expose the overhead in both performance (time) and

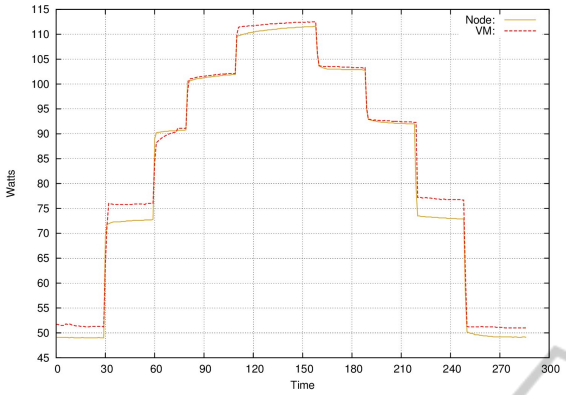

Figure 5: Power dissipation of a node running benchmark **MVB**.

power that the second case exhibits due to the presence of the workload.

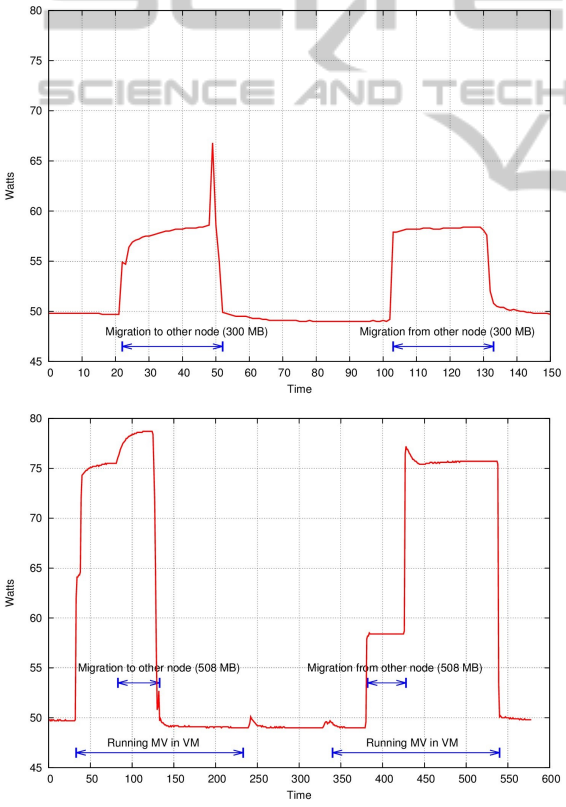

Figure 6: Power dissipation during the relocation of an idle VM and a VM running benchmark **MVB** on a single core (top and bottom, respectively).

5 CONCLUSIONS AND FUTURE WORK

In this paper we have investigated the benefits and costs of virtualization for two different applications, related to the common workloads running on datacenters and HPC facilities. The results from this work-in-progress link the response time and performance with the potential energy savings that consolidation of physical servers can yield. Further research is necessary to formally characterize the effects of VM relocation into the power dissipation and performance of the applications.

ACKNOWLEDGEMENTS

This work was supported by project TIN2011-23283 of the *Ministerio de Economía y Competitividad* and FEDER, and project P11B2013-21 of the Universidad Jaume I.

REFERENCES

(2013). Openstack open source cloud computing software. http://www.openstack.org.

(2013). QEMU. http://wiki.qemu.org.

(2013). VirtualBox. https://www.virtualbox.org/.

Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., and Warfield, A. (2003). Xen and the art of virtualization. *SIGOPS Oper. Syst. Rev.*, 37(5):164–177.

Fielding, R. and Kaiser, G. (1997). The Apache HTTP server project. *IEEE Internet Computing*, 1(4):88–90.

Goldberg, R. (1974). Survey of virtual machine research. *IEEE Computer*, 7(6):34–45.

Kivity, A., Lublin, U., and Liguori, A. (2007). KVM: the Linux virtual machine monitor. *Reading and Writing*, 1(1465-6914):225–230.

Kusic, D., Kephart, J. O., Hanson, J. E., Kandasamy, N., and Jiang, G. (2009). Power and performance management of virtualized computing environments via lookahead control. *Cluster Computing*, 12(1):1–15.

Nieh, J. and Leonard, O. C. (Aug. 2007). Examining VMware. *Dr. Dobb's Journal*.

Younge, A., Henschel, R., Brown, J., von Laszewski, G., Qiu, J., and Fox, G. (2011). Analysis of virtualization technologies for high performance computing environments. In *IEEE Int. Conf. on Cloud Computing 2011*, pages 9–16.

Younge, A. J., Brown, J., Henschel, R., Qiu, J., and Fox, G. (2010). Performance analysis of HPC virtualization technologies within FutureGrid. In *Conference CloudCom 2010*. poster.