A Holistic Model of the Performance and the Energy-Efficiency of Hypervisors in an HPC Environment

Mateusz Guzek[1], Sébastien Varrette[2], Valentin Plugaru[2], Johnatan E. Pecero[2] and Pascal Bouvry[2]

[1]Interdisciplinary Centre for Security Reliability and Trust
[2]Computer Science and Communications (CSC) Research Unit
University of Luxembourg, Luxembourg
Emails: `{Firstname.Name@uni.lu}`

Abstract. With a growing concern on the considerable energy consumed by HPC platforms and data centers, research efforts are targeting toward green approaches with higher energy efficiency. In particular, virtualization is emerging as the prominent approach to mutualize the energy consumed by a single server running multiple Virtual Machines (VMs) instances. However, little understanding has been obtained about the potential overhead in energy consumption and the throughput reduction for virtualized servers and/or computing resources, nor if it simply suits an environment as high-demanding as an High Performance Computing (HPC) platform. In this paper, a novel holistic model for the power of HPC node and its eventual virtualization layer is proposed. More importantly, we create and validate an instance of the proposed model using concrete measures taken on the Grid5000 platform. In particular, we use three widespread virtualization frameworks, namely Xen, KVM, and VMware ESXi and compare them with a *baseline* environment running in native mode. The conducted experiments were performed on top of benchmarking tools that reflect an HPC usage, *i.e.* HPCC, IOZone and Bonnie++. To abstract from the specifics of a single architecture, the benchmarks were run using two different hardware configurations, based on Intel and AMD processors. The benchmark scores are presented for all configurations to highlight their varying performance. The measured data is used to create a statistical holistic model of power of a machine that takes into account impacts of its components utilization metrics, as well as used application, virtualization, and hardware. The purpose of the model is to enable estimation of energy consumption of HPC platforms in areas such as simulation, scheduling or accounting.

Keywords: Energy-efficiency, HPCC, IOZone, Bonnie++, Xen, KVM, ESXi

1 Introduction

With the advent of the Cloud Computing (Cloud Computing (CC)) paradigm, more and more workloads are being moved to virtual environments. Yet the question of whether CC is suitable for High Performance Computing (HPC) workload remain unclear. With a growing concern on the considerable energy consumed by HPC platforms and data centers, having a clear answer to this question becomes more and more crucial.

In this paper, we evaluate and model the overhead induced by several virtualization environments (often called *hypervisors*) at the heart of most if not all CC middlewares. In particular, we analyze in this study the performance and the energy profile of three widespread virtualization frameworks, namely Xen, KVM, and VMware ESXi, running a single VM instance and compare them with a *baseline* environment running in native mode. It is worth to notice that it is quite hard to find in the literature fair comparisons of `all` these hypervisors. For instance, in the few cases where the VMWare suite is involved, the study is generally carried on by the company itself. The experiments presented in this paper were performed on top of benchmarking tools that reflect an HPC usage, *i.e.* the HPC Challenge (HPCC), IOZone and Bonnie++. They helped to refine a novel holistic model for the power consumption of HPC components which is proposed in this article. Moreover, to abstract from the specifics of a single architecture, the benchmarks were run using two different hardware configurations, based on Intel and AMD processors. In this context, the Grid'5000 platform helped to deploy in a flexible way such heterogeneous configuration and provide a unique environment as close as possible to a real HPC system. To this extent, the work presented in this paper offers an interesting complement to precedent studies, which targeted a similar evaluation, yet limited the analysis to a subset of hypervisors (generally excluding VMWare products) and a fewer number of benchmarks. Our experimental settings reflect the increasing complexity of HPC systems' analysis and management. From the green computing perspective it is crucial to be able to estimate and predict the power and consequently the energy of a data center. Such prediction could be used to optimize the system by assisting scheduling or hypervisor choice, to simulate systems' future behavior or even to account the consumed energy in case of insufficient physical infrastructure. The hardware, configuration and type of processing, has impact on the power consumption of a machine, which is reflected in the novel model. First, the model redefines the structure of computing in a virtualized data center. The classical Task and Machine models are extended by the Configuration layer that represents the chosen middleware. Then, we propose a power model that combines multiple factors, either directly measured utilization metrics or classifiers such as used node, hypervisor or application, consequently calling the model *holistic*. The power modelling is lightweight in terms of creation and usage, as it is based on multiple linear regression. The purpose of power modelling is to validate the theoretical assumption that increasing the amount of information about the node enables better power estimation.
This article is organized as follows: Section 2 presents the background of this study and reviews related work. Then, our novel holistic model is detailed in Section 3. Section 4 describes the experimental setup used within this study, in particular we will present the different cutting-edge platforms we compare, together with the benchmark workflow applied to operate these comparisons. Then, Section 5 details the experimental results obtained. Finally, Section 6 concludes the paper and provides the future directions.

2 Context & Motivations

In essence, Cloud middleware exploit virtualization frameworks that authorize the management and deployment of Virtual Machines (VMs). Whereas our general goal is to

Hypervisor:	**Xen 4.0**	**KVM 0.12**	**ESXi 5.1**
Host architecture	x86, x86-64, ARM	x86, x86-64	x86-64
VT-x/AMD-v	Yes	Yes	Yes
Max Guest CPU	128	64	32
Max. Host memory	1TB	-	2TB
Max. Guest memory	1TB	-	1TB
3D-acceleration	Yes (HVM Guests)	No	Yes
License	GPL	GPL/LGPL	Proprietary

Table 1. Overview of the considered hypervisors characteristics.

model Cloud systems in an HPC context, we present here the first step toward this global modelization focusing on the underlying hypervisor or Virtual Machine Manager. Subsequently, a VM running under a given hypervisor will be called a guest machine. There exist two types of hypervisors (either *native* or *hosted*) yet only the first class (also named bare-metal) presents an interest for the HPC context. This category of hypervisor runs directly on the host's hardware to control the hardware and to manage guest operating systems. A guest operating system thus runs on another level above the hypervisor. Among the many potential approach of this type available today, the virtualization technology of choice for most open platforms over the past 7 years has been the Xen hypervisor [6]. More recently, the Kernel-based Virtual Machine (KVM) [15] and VMWare ESXi [5] have also known a widespread deployment within the HPC community such that we limited our study to those three competitors and decided to place the other frameworks available (such as Microsoft's Hyper-V or OpenVZ) out of the scope of this paper. Table 1 provides a short comparison chart between Xen, KVM and VMWWare ESXi.

2.1 Considered HPC platforms

To reflect a traditional HPC environment, yet with a high degree of flexibility as regards the deployment process and the fair access to heterogeneous resources, the experiments presented in this paper were carried on the Grid'5000 platform [2]. Grid'5000 is a scientific instrument for the study of large scale parallel and distributed systems. It aims at providing a highly reconfigurable, controllable and monitorable experimental platform to its users. One of the unique features offered by this infrastructure compared to a production cluster is the possibility to provision on demand the Operating System (OS) running on the computing nodes. Designed for scalability and a fast deployment, the underlying software (named Kadeploy) supports a broad range of systems (Linux, Xen, *BSD, etc.) and manages a large catalog of images, most of them user-defined, that can be deployed on any of the reserved nodes of the platform. As we will detail in Section 4, we have defined a set of common images and environments that were deployed in two distinct hardware architectures (based on Intel or AMD) on sites that offer the measurement of Power distribution units (PDUs).

2.2 Considered benchmarks

Several benchmarks that reflect a true HPC usage were selected to compare all of the considered configurations. For reproducibility reasons, all of them are open source and we based our choice on a previous study operated in the context of the FutureGrid[1] platform [17], and a better focus on I/O operation that we consider as under-estimated in too many studies involving virtualization evaluation. We thus arrived to the three benchmarks:

The HPC Challenge (HPCC) [13], an industry standard suite used to stress the performance of multiple aspects of an HPC system, from the pure computing power to the disk/RAM usage or the network interface efficiency. It also provides reproducible results, at the heart of the ranking proposed in the Top500 project.
HPCC basically consists of seven tests: (1) `HPL` (the High-Performance Linpack benchmark), which measures the floating point rate of execution for solving a linear system of equations. (2) `DGEMM` - measures the floating point rate of execution of double precision real matrix-matrix multiplication. (3) `STREAM` - a simple synthetic benchmark program that measures sustainable memory bandwidth (in GB/s) and the corresponding computation rate for simple vector kernel. (4) `PTRANS` (parallel matrix transpose) - exercises the communications where pairs of processors communicate with each other simultaneously. It is a useful test of the total communications capacity of the network. (5) `RandomAccess` - measures the rate of integer random updates of memory (GUPS). (6) `FFT` - measures the floating point rate of execution of double precision complex one-dimensional Discrete Fourier Transform (DFT). (7) Communication bandwidth and latency - a set of tests to measure latency and bandwidth of a number of simultaneous communication patterns.

Bonnie++ [1], a file system benchmarking suite that is aimed at performing a number of simple tests of hard drive and file system performance.

IOZone [3], a more complete cross-platform suite that generates and measures a variety of file operations. Iozone is useful for performing a broad filesystem analysis of a given computing platform, covering tests for file I/O performances for many operations (Read, write, re-read, re-write, read backwards/strided, mmap etc.)

The results that are obtained from these benchmarks provide an unbiased performance analysis of the hypervisors and thus provide a valid reference for the holistic model proposed in this article.

2.3 Related Work

Different studies describe or apply models for power draw for data centers and HPC centers subsystems. Some modelling research work indicate that server power varies roughly linearly in CPU utilization [10]. Economu et al. [9] study the component-level power breakdown and variation, as well as temporal workload-specific power consumption of a blade server. The authors suggest to consider beside CPU and disk utilization also design properties of the server. Kansal et al. [14] proposed a power meter model for virtual machines, called *Joulemeter*. The model makes use of power models of individual hardware resources; at runtime software components monitor the resource usage of VMs and they convert it to energy usage using the available model. Bohra et al. in [7]

[1] See `https://portal.futuregrid.org/`.

proposed *vMeter*, a power modelling technique. The authors observed a correlation between the total system's power consumption and component utilization. They proposed a four-dimensional linear weighted power model for the total power consumed. The components of the model are: performance parameters for CPU, cache, DRAM and disk. The weights of the model are calculated per workflow. The authors refunded the power model by separating the contribution of each active domain in a node, either a VM or domain. Chen et al. [8] present a study in profiling virtual machines with respect to three power metrics: power, power efficiency and energy, under different high performance computing workloads. The authors proposed a linear power model that represents the behavior of a single work node and includes the contribution from individual components. Liu et al. [16] proposed a GreenCloud architecture that utilizes live migration of VMs based on power information of the physical nodes reducing energy consumption for applications running in clouds, specifically for online gaming.
At the level of the pure hypervisor performance evaluation, many studies can be found in the literature that attempt to quantify the overhead induced by the virtualization layer. Yet the focus on HPC workloads is recent as it implies several challenges, from a small system footprint to efficient I/O mechanisms. A first quantitative study was proposed in 2007 by A. Gavrilovska et al. in [4]. While the claimed objective was to present opportunities for HPC platforms and applications to benefit from system virtualization, the practical experimentation identified the two main limitations to be addressed by the hypervisors to be of interest for HPC: I/O operations and adaptation to multi-core systems. While the second point is now circumvented on the considered hypervisor systems, the first one remains challenging. Another study that used to guide not only our benchmarking strategy but also our experimental setup is the evaluation mentioned in the previous section that was performed on the FutureGrid platform [17]. The targeted hypervisors were Xen, KVM, and Virtual Box and a serious performance analysis is proposed, with the conclusion that KVM is the best overall choice for use within HPC Cloud environment. Compared to the above mentioned studies, our contribution in this paper can be summarized in the following elements: (1) a novel holistic model for the power consumption of HPC components, eventually running on top of three widespread virtualization frameworks (Xen, KVM and VMware ESXi), is proposed; (2) the model is refined using the hypervisors in a concrete HPC environment, comparing also the two leading concurrent hardware architecture (Intel and AMD – 84% of the represented processor technologies in the latest Top500 list). Our performance evaluation involves industrial reference benchmarks and does not ignore a measure of the impact on I/O operations, too often ignored in the literature; (3) it is one of the few independent study that takes into consideration not only open-source hypervisors (Xen and KVM) but also a proprietary solution from the leading vendor in the domain *i.e.* VMWare; (4) the energy-efficiency of the considered configuration is properly modelled and quantified with exact measures offered by the Grid5000 platform.

3 The Holistic System Model

In this section, we introduce a novel model for the performance and energy-efficiency analysis of HPC or CC components. The model is holistic, i.e. it includes all elements

important for the performance and power of distributed computing systems, and it relies on classical scheduling models with machines and tasks. Our first contribution is the addition of a machine configuration layer that corresponds to the used middleware. The bottom layer, the *Machine* layer, describes the physical characteristics of the host. Modelling that layer assumes the derivation of the energy model of a machine which is based on the measured utilization of its components or environmental factors (e.g. temperature, supply voltage). In this work we take into account utilization metrics of CPU, Memory, Disk and Networking of a node. This layer describes each machine separately, taking into account their heterogeneity.
The middle *Configuration* layer corresponds to the overhead induced by the software that is used to process tasks. The basic element of this layer is *Container* that represents the used OS, including a potential virtualization technology. In case of virtualized systems there can exist multiple, possibly heterogenous, containers on a single machine.
The top *Task* layer represents the computation or work performed by applications. A *Task* represents a workload processed by an application and its corresponding data. Multiple tasks can be executed simultaneously in a single Container, with the performance depending on the availability of resources.
The second contribution of the proposed model is an extension of the classical definition of resources. Instead of representing resources as discrete entities, the holistic model represents each resource by a resource vector: $res = (typ_1, \dots, typ_z)$. The vector is composed of *resource types* that represents distinct resource types, e.g. CPU, Memory, Disk etc. Each resource type is further expressed as a vector of *resource supplies*: $typ_i = (sup_{i1}, \dots, sup_{iy})$ that represents the exact implementation and number of resource supplies, i.e. hardware components. Finally, each resource supply is defined as $sup_{ij} = (arch_{ij}, cap_{ij})$, where $arch_{ij}$ represents the component architecture (that determines the components characteristics) and cap_{ij} represents the component capacity (e.g. MIPS, RAM or disk size). The architectures can create a partial order, based on the relation of the strict superiority (in terms of performance) of $arch_i$ over $arch_k$, denoted as $arch_k \prec arch_i$. A sample graphical resource vector of a real node is presented on the top of Figure 1. The node is composed of four resource types. Each of the types is composed of a single supply. In this case these are: CPU with 4 symmetric cores (with capacity expressed in e.g. corresponding number of MIPS), Memory and Storage with single capacity, and Network card wit 2 interfaces. The architecture ordering could be based on the comparison of architectures of this node supplies with other architectures in the same data centre. The resource allocation in a holistic model is represented by resource provisions and resource demands. A *Resource provision* is the representation of the resource offered by a lower layer to the higher layer. Resource providers are the resources of machines and the resources offered by containers to tasks. The *Resource demand* is the representation of the resources consumed by higher layer entities and it corresponds to the resources reserved by a container on a machine, or the resources requested by a task from a container. Figure 1 presents also a simple allocation example, where two architecture types ($A = \{1, 2\}$ and $1 \prec 2$) are defined for two nodes, each having various capacity of provisioned resources P. U is the aggregated resource utilization at the machine level, D is the resource demand. The difference between P and D of a container represents its overhead. The colour of VMs and Tasks represents

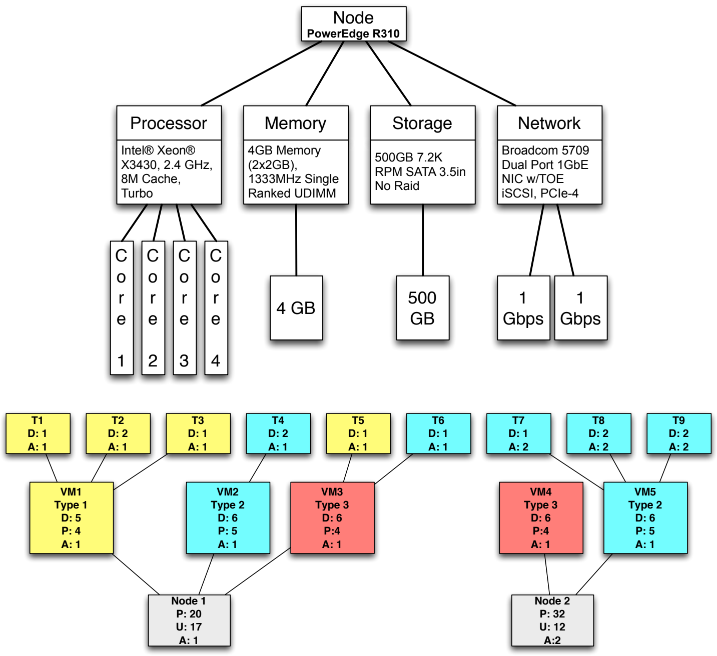


Fig. 1. Example of a representation of a computing node within an HPC cluster

their type: blue tasks can be executed on blue or red VMs, while yellow tasks can be executed on yellow or red VMs. The main issue in such a holistic model is to relate the resource utilization with the obtained performance and power. In this paper, we present a lightweight approach for power modelling. Multiple linear regression is the tool used to accurately predict the impact of the used Machine, Configuration, and Task on the system power. As it is of prime importance to correctly derive the parameters of this model, the best approach requires the collection of concrete observations in a real situation, featuring the virtualization technologies we try to characterize. The next section details the experimental setup performed to reach this goal.

4 Experimental Setup

Two sites of Grid5000, Lyon and Reims were selected for the benchmarking process, as they host two modern HPC clusters: Taurus and StRemi, with diverse hardware architectures and support for Power distribution unit (PDU) measurements. An overview of the selected systems we compare in this article is provided in Table 2.

The benchmarking workflow, as presented in Figure 2, has been described in the following paragraphs. The baseline benchmark uses a customized version of the Grid'5000 squeeze-x64-base image, which contains the benchmark application suite comprised of OpenMPI 1.6.2, GotoBLAS2 1.13, HPCC 1.4.1, Bonnie++ 1.96 and IOzone 3.308. This image is deployed on the target node with the kadeploy3 Grid'5000 utility from the sites' frontend and then the *hypervisor-benchmark-baseline* script is used to launch the benchmark process. This script mounts on the host the site's NFS shared homes, launches in background the dstat utility which is being used to collect resource usage statistics from the node, then starts the *benchmark* script. The *benchmark* script runs HPCC, Bonnie++ and IOzone with cluster-specific values, logging the progress of these applications and archiving their results at the end, along with the output from the dstat utility. The archive is placed directly in the user's NFS shared home, in a directory which reflects the site name, cluster name, and the job ID of the OAR reservation the user holds. The benchmarking workflow for KVM and XEN is identical, although it is based on different scripts customized to work with these hypervisors. The KVM deployment image has been created from the baseline image, while the XEN image is based on squeeze-x64-xen. Both host images contain VM image files which incorporate the same benchmark suite as the baseline image. After the deployment of the appropriate host image the corresponding *hypervisor-benchmark-{kvm,xen}* launcher is started, which contains user-configured parameters that specify how many virtual machines will be configured, the number of virtual cores and memory available to each. The launcher starts the appropriate *prepare-{kvm,xen}* script, which in turn connects to the host node, copies and resizes the virtual image (to more than twice the configured VM RAM size, as needed by the Bonnie++ benchmark), pushes the *benchmark* script to it, then starts the VM, pinning the virtual cores to host cores one-to-one. In the next step, a VM-controller *runbench-{kvm,xen}* script is started in the background on the frontend, which waits for the VM to become available on the network, launches the dstat utility in the background on the host and in the VM, then starts the benchmark. When the *benchmark* script has finished, the results archive (containing also the dstat statistics) is retrieved from the VM, along with the host statistics, and the results are placed on the site frontend, in the user's home directory following the same pattern as for the baseline test. The workflow for the ESXi benchmark requires that the target host be booted (through the Grid'5000 kareboot application) with a specific PXE profile and configuration files so that the ESXi installer boots and configures the host according to a cluster-specific kickstart automated installation script. After the installation is done, the host automatically reboots and manual user intervention is required in order to ensure that the host will boot from the local drive by having an *ESXi-install* script reboot the host with another PXE profile that chainloads the newly installed MBR. The ESXi installer has been forced to use a MBR type partitioning scheme, as opposed to its default GPT in order to not interfere with the operation of the Grid'5000 platform. When the ESXi hypervisor has booted, the *hypervisor-benchmark-esxi* user-customized launcher starts the *prepare-esxi* script which then creates an appropriate ESXi VM configuration file and copies it along with the VM image to the host. The script registers the VM and adds a second disk to it as the VM image itself cannot be resized on the host as was

Name	**Site**	**Cluster**	**#cpus/n**	**#RAM**	**Processor**	**R_{peak}**
`Intel`	Lyon	`taurus`	2	32GB	Intel Xeon E5-2630@2.3GHz 6C	110,4 GFlops
`AMD`	Reims	`stremi`	2	48GB	AMD Opteron 6164 HE@1.7GHz 12C	163.2 GFlops

Table 2. Overview of the two types of computing nodes used in this study.

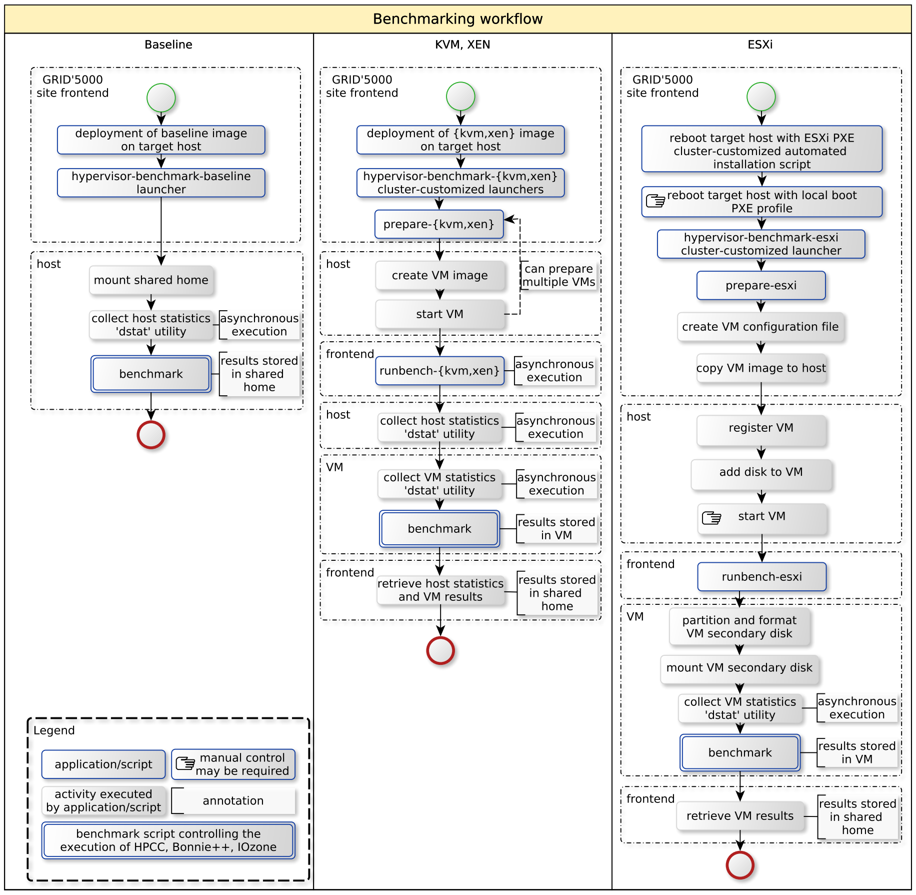

Benchmarking workflow
Baseline
KVM, XEN
ESXi
GRID'5000 site frontend
deployment of baseline image on target host
hypervisor-benchmark-baseline launcher
host
mount shared home
collect host statistics 'dstat' utility
asynchronous execution
benchmark
results stored in shared home
deployment of {kvm,xen} image on target host
hypervisor-benchmark-{kvm,xen} cluster-customized launchers
prepare-{kvm,xen}
create VM image
start VM
can prepare multiple VMs
frontend
runbench-{kvm,xen}
collect VM statistics 'dstat' utility
VM
results stored in VM
retrieve host statistics and VM results
reboot target host with ESXi PXE cluster-customized automated installation script
reboot target host with local boot PXE profile
hypervisor-benchmark-esxi cluster-customized launcher
prepare-esxi
create VM configuration file
copy VM image to host
register VM
add disk to VM
runbench-esxi
partition and format VM secondary disk
mount VM secondary disk
retrieve VM results
Legend
application/script
manual control may be required
activity executed by application/script
annotation
benchmark script controlling the execution of HPCC, Bonnie++, IOzone

Fig. 2. Benchmarking workflow.

the case for KVM and XEN, then starts the VM. This last step may require manual user control, as in some cases the VM is not successfully started automatically. The *runbench-esxi* script is then used to control the VM, in which it partitions, formats and mounts the new disk that was added (which will hold both temporary files from the benchmark and the results), then launches the dstat tool in background and starts the benchmark script. After the benchmark ends, the results archive (with the dstat statistics) is retrieved and stored on the Grid'5000 site's frontend in the same way was as for the baseline, KVM and XEN tests. The number of experimental runs for combinations of environments and nodes are presented in Table 3. Due to the need of manual interventions and special preparation of cluster, the ESXi environment was tested only on nodes 7 and 10 in taurus cluster and nodes 30 and 31 in stremi cluster. The baseline environment was tested only 4 times on taurus-8 due to technical problems with that node that appeared at the end of the sequence of experiments.

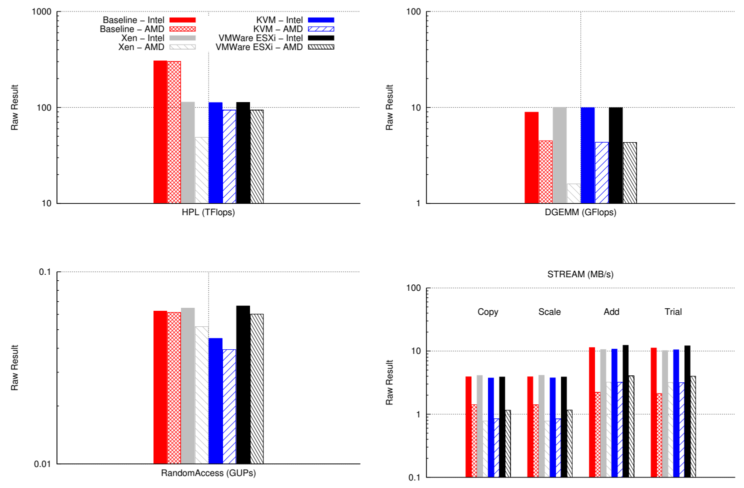
config:	**baseline**	**KVM**	**Xen**	**VMWare ESXi**	**Observation No.**
stremi-3	5	5	5	0	10916
stremi-6	5	5	5	0	10907
stremi-30	5	5	5	5	13706
stremi-31	5	5	5	5	14026
taurus-7	5	5	5	5	6516
taurus-8	4	5	5	0	4769
taurus-9	5	5	5	0	5085
taurus-10	5	5	5	5	6545

Table 3. Number of runs for environment and node.

Monitoring of the performance metrics and Data processing. To accurately estimate the status of the analyzed system at a given period of time, a set of performance and power metrics have to be collected. This data was gathered using two monitoring tools: *Ganglia* and *dstat*. Ganglia is a monitoring tool that works at the grid level and is used in this work to gather power readings of the monitored nodes: in the `stremi` cluster, instantaneous power readings are available every 3s using SNMP (Raritan) with an accuracy of 7W, while in the `taurus` cluster, power is recorded using OmegaWatt power meters that return average power each second with an accuracy of 1W. In both cases, Ganglia aggregates measured values over 15s periods. In order to ensure persistency of values recorded using Ganglia, they are accessed using Grid5000 API and stored in an external database. The utilization metrics of CPU, memory, disk IO and networking IO are recorded using dstat with a frequency of 1s. The recorded metrics and corresponding units are: (1) CPU – user, system, idle, wio (%) (2) Memory – used, buffered, cached, free (B) (3) Disk – read, write (B) (4) Network –received, send (B). Due to the specifics of dstat, missing or duplicated readings are possible. In order to prepare the data for modelling, the dstat values are aggregated (summed for flow values such as IO, averaged for utilization metrics such as *cpu_user*) over 15s periods corresponding to the Ganglia monitoring readings. In case of missing dstat values for periods longer than 15s, the data from Ganglia is discarded from modelling (the amount of data removed in this way is 1.2%, removed observations do not follow any obvious pattern). Such granularity of measurements corresponds to the monitoring utilities used in production systems and diminishes the impact of measuring infrastructure on the system performance. Additionally, each observation has supplemental information about the cluster, node_uid, hypervisor and benchmark phase. The data was preprocessed and statistically analysed using *R* statistical software with the packages *zoo* for data series processing. As a result, 72470 observations were measured, as presented in detail in Table 3.

5 Experimental Results

Performance analysis. While the heart of this study does not reside in the pure performance evaluation of the considered virtualization technology, we present here the average raw performance results obtained over the multiple runs of the HPCC benchmark in each considered configuration. In an attempt to improve the readability of the



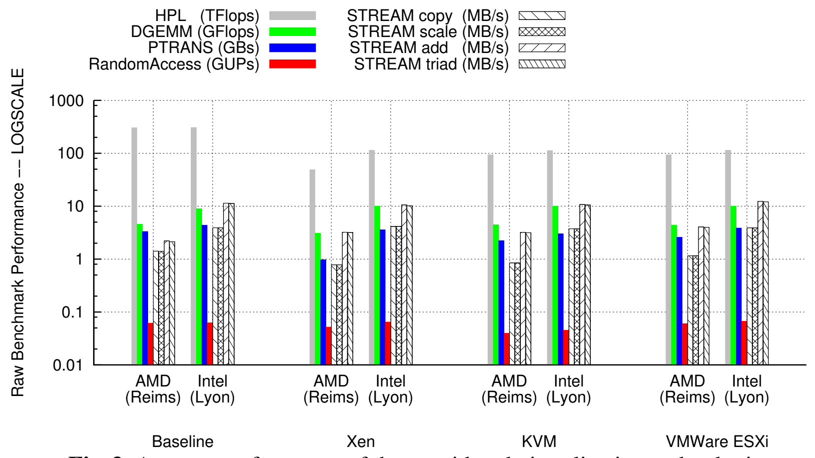


Fig. 3. Average performance of the considered virtualization technologies.

article, we limit on purpose the number of displayed test results to the ones of HPL, DGEMM, PTRANS and STREAM. First of all, a synthetic view indexed over the hardware architecture is proposed in Figure 3. We can see that in every single case, the Intel-based configuration outperforms its AMD counterpart, despite a presumed lower peak performance T_{peak}. Then, we illustrate the performances for each test to extract a trend in the relative overhead induced by the considered virtualization technologies. It appears from these plots that the three considered hypervisors offer a similar overhead as regards the computing benchmarks (between 10 to 15%).

One inherent limit to the usage of virtualization in an HPC environment obviously resides in the huge overhead induced on I/O operations. Thus, we present the results of the IOZone benchmark in Figure 5. If a significant degradation of the performances is observed as soon as a hypervisor is present, we can see a surprising element as regards the `rewrite` and `random_write` test on the ESXi environment which perform better than the bare-metal system. This is probably due to a better cache strategy on the file system deployed by this environment.

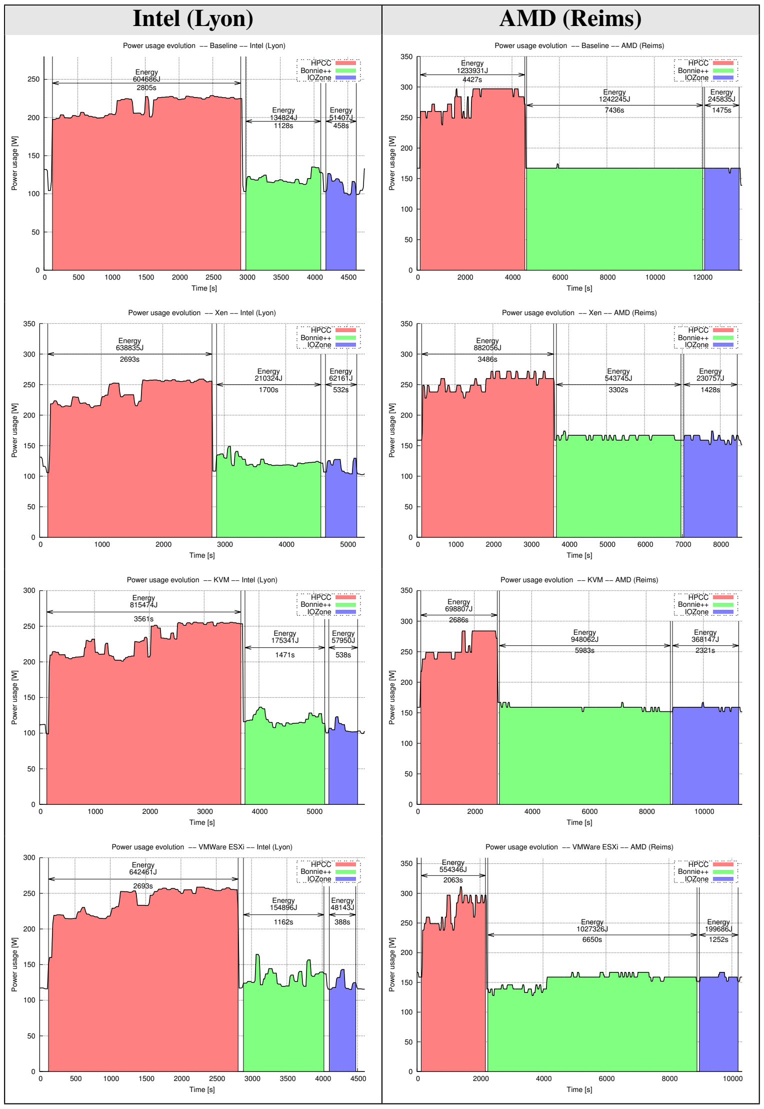


Fig. 4. Power profile of selected runs in each configuration.

Energy-efficiency analysis and Power modelling by Multiple Regression. For each considered configuration we have measured the energy consumed to run the different benchmarks. As an illustration of the many runs performed, we provide in Figure 4 traces of selected runs. The analysis of the traces left of each run on the selected configuration permitted to refine the parameters of the holistic model presented in Section 3. We now detail the statistical approach operated in this context.

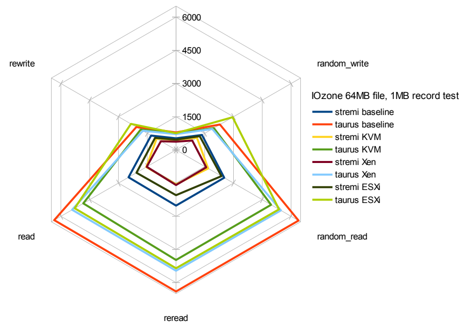


Fig. 5. IOZone results for each configuration.

The presented approach to model power using utilisation metrics as predictors is multiple regression. The advantage of this method is low computational complexity, no need for parameters and deterministic results. The final model is presented as a linear function of predictors [11]:

$$E(y|x_1, \ldots, x_k) = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k, \tag{1}$$

where $E(y|x_1, \ldots, x_k)$ is the expected value of response y given fixed values of regressors $x_1, \ldots, x_k$. The coefficient β_0 is referred to as *intercept* and the other coefficients $(\beta_1, \ldots, \beta_k)$ are called *slopes*. In the context of this work two categories of predictors can be distinguished. *Numerical* predictors are based on the data gathered by monitoring tools. *Categorical* predictors are additional data that group the observations. The aim of modelling at the node level is twofold: to analyse the operation of a system and to predict its behavior. The results of multiple regressions are presented according to concepts of *complete-pooling* and *no-pooling* [12]. In the former case all observations are taken into account disregarding groups specifics, while in the latter case different groups are modelled separately. The following models, varying by sets of predictors, are proposed:

1. *Basic* – all possible predictors taken into account
2. *Refined* – Basic processed by backward stepwise algorithm based on AIC (using default *step* function in R). In all cases it resulted removing only *disk read*.
3. *No Phases* – all possible predictors taken into account but no explicit information about the workload type
4. *CPU Hom.* – only *cpu user* and *cpu idle* taken into account for a simplest homogeneous bottom-line model
5. *CPU Het.* – only *cpu user*, *cpu idle* and *node uid* taken into account for a simplest heterogenous model
6. *No Group* – all possible numerical predictors, no categorical predictors
7. *Cluster-wise* – all possible numerical predictors and cluster predictor, test of homogenous hardware hypothesis (only in Complete-pooling setting)
8. *Group Only* – all possible group predictors, no numerical predictors

The results of Complete-pooling analysis are presented in Table 4. The tables in this section are based on three main quality indicators of a model: R^2 value, distribution of residuals including standard error, minimal and maximal values, first and third quartile, and median, and finally the average values of absolute prediction error presented as total error in Watts or relative error in percents. The absolute prediction error was calculated for each observation in the data set as an absolute value of the difference between predicted and observed values of node power, thus the under and over. The models that take

Model	R^2	Residuals						Error	
		St.er.	Min	1Q	Median	3Q	Max	W	%
Basic	0.959	10.4	-116	-3.54	0.8	5.07	117	6.67	3.8%
Refined	0.959	10.4	-116	-3.54	0.806	5.07	117	6.67	3.8%
No Phases	0.941	12.4	-127	-4.13	1.04	4.88	125	8.18	4.4%
CPU Hom.	0.814	22	-147	-13.1	3.9	13.6	160	16.7	9.6%
CPU Het.	0.922	14.2	-129	-5.12	-0.0472	4.89	129	9.73	5.0%
Only phases	0.922	14.3	-114	-3.94	2.06	5.51	72.8	8.75	4.9%
No group	0.856	19.4	-142	-8.45	2.96	11.3	129	14.1	8.0%
Clusterwise	0.928	13.7	-122	-7.22	0.876	7.02	131	9.97	5.3%
Group only	0.924	14.1	-113	-4.29	2.52	5.88	69.9	8.77	4.9%

Table 4. Complete-pooling models quality

No-pooling `stremi` cluster models quality

Model	R^2	Residuals						Error	
		St.er.	Min	1Q	Median	3Q	Max	W	%
Basic	0.968	8.74	-105	-2.9	0.398	3.52	106	5.28	2.8%
Refined	0.968	8.74	-105	-2.9	0.396	3.52	106	5.28	2.8%
No Phases	0.955	10.4	-114	-3.24	0.785	3.51	121	6.15	3.1%
CPU Hom.	0.925	13.3	-116	-7.07	0.0893	7.98	128	9.44	4.7%
CPU Het.	0.938	12.2	-120	-4.2	1.04	4	125	7.8	3.7%
Only phases	0.956	10.2	-104	-3.69	1.3	4.31	112	6.3	3.2%
No group	0.942	11.7	-120	-6.33	1.05	6.78	126	8.19	4.3%
Group only	0.96	9.77	-104	-3.07	0.694	4.11	112	5.87	3.1%

No-pooling `taurus` cluster models quality

Model	R^2	Residuals						Error	
		St.er.	Min	1Q	Median	3Q	Max	W	%
Basic	0.957	11.6	-117	-6.16	0.0846	5.97	132	7.62	4.7%
Refined	0.957	11.6	-117	-6.16	0.08	5.97	132	7.62	4.7%
No Phases	0.925	15.2	-127	-9.04	0.967	10.8	142	11.2	6.3%
CPU Hom.	0.896	17.9	-128	-10.6	-1.49	12.2	137	13.5	7.4%
CPU Het.	0.898	17.7	-131	-10.7	-2.18	12.3	133	13.4	7.4%
Only phases	0.884	18.9	-114	-6.75	1.33	9.28	57.4	11.9	7.3%
No group	0.918	15.9	-127	-9.05	0.0862	11.9	141	11.8	6.6%
Group only	0.902	17.4	-112	-5.95	0.883	7.59	56.4	10.2	6.6%

Table 5. No-pooling models quality

into account all predictors (*Basic* and *Refined*) have the highest R^2 values, the smallest residual standard error and average prediction error. The *No Phases* model presents similar scores, arguing that knowledge about applications specifics is not necessary if utilisation metrics and environment information are available. The *CPU Het.* model has high R^2 value and acceptable values for the statistics of residuals, contrary to the *CPU Hom.* that is the worst of investigated models. Similarly, the *No Group* model has a low quality, worse than the *Group Only* model that estimates the best value for categorical predictors and has in effect only several possible response values. The comparison of *Cluster-wise* and *CPU Het.* is interesting: the former model has slightly better R^2 values and residual standard error, but the distribution of residuals is better for the latter model, as well as mean absolute prediction errors, pointing out that the nodes in each cluster have heterogenous power consumption, despite their homogenous hardware configuration. Reassuming, the Basic and Refined models are the most accurate and they include all elements of holistic model, confirming the necessity of detailed information for accurate system modelling .

The quality of No-pooling models is presented in Table 5 for clusters stremi and taurus. The No-pooling methodology divides the set of data used in the Complete-pooling scenario into two disjoint subset, one for each of the clusters. As a result, the obtained

stremi-3	stremi-30	stremi-31	stremi-6	taurus-10	taurus-7	taurus-8	taurus-9
0	-1.8	-14	-4.7	-45	-40	-46	-44

Bonnie	DGEMM	FFT	HPL	IOZONE	PTRANS	RandomAccess	STREAM	idle
0	5.7	6.5	16	0.012	-6.1	-11	3.1	6.1

ESXi	KVM	Xen	baseline
0	-3.6	-5.4	-19

Intercept	cpu user	cpu system	cpu idle	cpu wio	mem used
316	-0.78	-1.3	-1.7	-1.8	1.1E-9

mem buffers	mem cached	mem free	disk write	bytes rec.	bytes send
-3.1E-08	8.0E-10	8.3E-10	-1.8E-10	4.6E-05	-1.1E-04

Table 6. The Complete-pooling model coefficients for nodes, phases and hypervisors, together with the model numerical coefficients.

model have distinct slopes of coefficients for two clusters, which is rational considering the differences in the underlying hardware. As a result, No-pooling models have better quality than corresponding Complete-pooling models. The quality of No-pooling modelling is apparently worse than the complete-pooling model, but it must be taken into account that because of the longer running time of experiments on stremi nodes, there are more observations from this cluster, that bias the Complete-pooling model results and accuracy towards this cluster results. The more interesting fact is the worse prediction results for the taurus cluster, which has more accurate power measurement infrastructure. The sample results of using power prediction by the Refined non-pooling models, identified as the best ones, are presented in Figure 6. The figure presents the predicted by Refined model power consumption against the observed values for sample runs for each combination of hardware and hypervisor. The less accurate prediction of ESXi can be explained by distinct hypervisor engine or limited amount of samples for this hypervisor. Despite that, the model is able to accurately follow the power consumption pattern for each configuration.

The results of models can be used to create an instance of holistic model. In such case, given the used, node, hypervisor, and utilisation levels of hardware components of a selected machine, one can predict the final power output. The model can be further refined by observation of the phases of computation. General utilisation of holistic model for decision making is presented in this section. Table 6 presents the difference between nodes in the two clusters as well as between nodes with the same hardware configuration. In this case, the difference between clusters is approximately 40W. The difference between nodes in stremi cluster is up to 14W, while in Taurus node it is up to 6W. The phases have large impact on the final power consumption in derived model, as presented in Table 6. The presented values are *adjustments* to the amount based on the utilization metrics. Therefore, there is no sense in straight comparison of these values (e.g. nodes do not consume approximately 3W *more* in idle state than during the STREAM phase). However, these values present that knowledge about running application characteristics can add valuable information to power predictions. The hypervisor

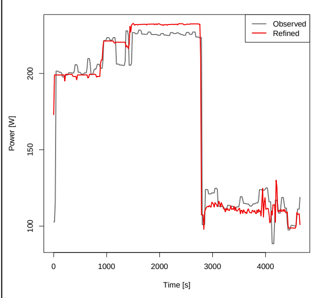
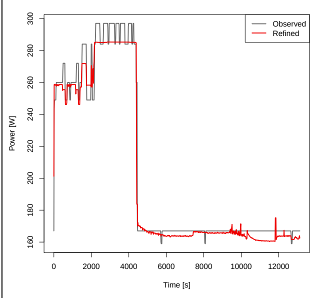
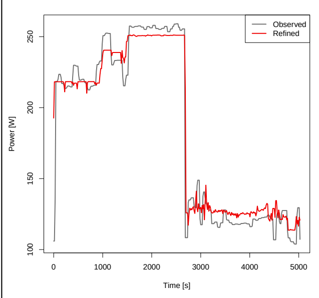
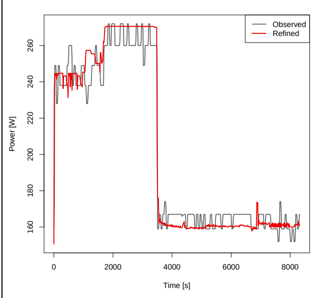
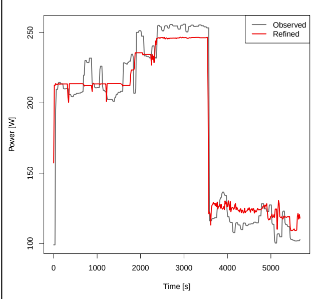
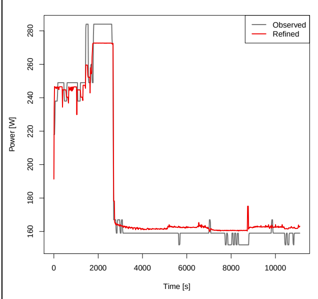
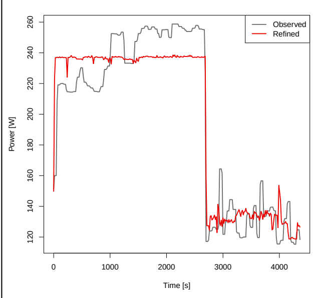
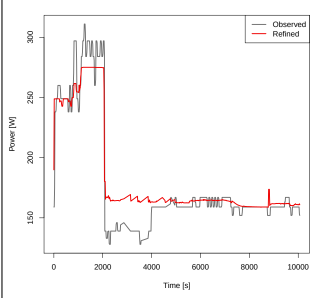
Baseline – Intel (Lyon)	Baseline – AMD (Reims)
	
Xen – Intel (Lyon)	**Xen – AMD (Reims)**
	
KVM – Intel (Lyon)	**KVM – AMD (Reims)**
	
ESXi – Intel (Lyon)	**ESXi – AMD (Reims)**
	

Fig. 6. The predicted energy profiles for the best identified models.

type influence is also depicted. The presented values show a gap between hypervisors and baseline. However, it is important to remember that performance metrics were collected at the container (guest VM) level, therefore these differences may be adjustments for the resources consumed by the hypervisor, which were not used for modeling. Finally, we discuss the numerical predictors. The intercept has high positive value. The CPU utilization coefficients are negative. The most power consuming mode is cpu user, followed by cpu system. Cpu idle and cpu wio are the least power-consuming modes of operation. The memory in used, cached, and free states has significantly higher power consumption that in buffered state. The output activities generally decrease the power of the system, which is coherent with the cpu wio values and may be caused by entering cpu into lower power states during large output operations. Contrary to that, network receive state increases the system power, however less significantly than the decrease of network send.

6 Conclusion

In the paper we introduce and experimentally evaluate a holistic model based on its power estimation. The holistic power modelling is able to represent power consumption with an average relative error of 3.8%. The model is lightweight, as it is represented by a single equation, and its creation also has a low time complexity. It can be extended after creation by adding or modifying the coefficients, thus it can adapt to dynamic systems. The model could be further refined using Partial-pooling techniques with varying slopes, to benefit from an intermediary approach between Complete-pooling and No-pooling [12]. The parameters of this model have been refined by a set of concrete experiments on the Grid'5000 platform that permitted the careful analysis and modelling of three widespread virtualization, namely Xen, KVM and VMware ESXi on the two leading hardware architectures (AMD and Intel). When compared to a *baseline* environment running in native mode, we have seen that the usage of hypervisors in an HPC environment raises a limited overhead, and can be foreseen as soon as the I/O operations are correctly handled, as the computing performances are nearly identical between the considered virtualization technologies. In this sense, we confirm the results of preceding studies. The future work includes adding more elements to holistic models, such as environmental metrics (temperature, supplied voltage), examining the effect of multi tenancy and overhead of cloud management systems (e.g. OpenNebula, OpenStack), modelling the performance of configurations, considering network-intensive load and parallel tasks, and building such models based on the hardware component of node (directly using the resource vector concept to build the power model), rather than full hardware platform. All that work require further experimentation on a larger set of applications and machines.

Acknowledgments: The experiments presented in this paper were carried out using the Grid'5000 experimental testbed, being developed under the INRIA ALADDIN development action with support from CNRS, RENATER and several Universities as well as other funding bodies (see `https://www.grid5000.fr`). This work was completed with the support of the INTER/CNRS/11/03/Green@Cloud project and the COST action IC0804. M. Guzek acknowledges the support of the FNR in Luxembourg and Tri-ICT, with the AFR contract no. 1315254. The authors would like also to thank Hyacinthe Cartiaux and the technical committee of the Grid'5000 project for their help and advises in the difficult deployment of the ESXi environment.

References

1. Bonnie++. [online] `http://www.coker.com.au/bonnie++/`.
2. Grid'5000. [online] `http://grid5000.fr`.
3. Iozone filesystem benchmark. [online] `http://www.iozone.org/`.
4. A. Gavrilovska et al. High-Performance Hypervisor Architectures: Virtualization in HPC Systems. In *Proc. of HPCVirt 2007*, Portugal, Mar. 2007.
5. Q. Ali, V. Kiriansky, J. Simons, and P. Zaroo. Performance evaluation of hpc benchmarks on vmware's esxi server. In *Proceedings of the 2011 international conference on Parallel Processing*, Euro-Par'11, pages 213–222, Berlin, Heidelberg, 2012. Springer-Verlag.
6. 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 *Proceedings of the nineteenth ACM symposium on Operating systems principles*, SOSP '03, pages 164–177, New York, NY, USA, 2003. ACM.
7. A. Bohra and V. Chaudhary. Vmeter: Power modelling for virtualized clouds. In *Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*, pages 1 –8, 2010.
8. Q. Chen, P. Grosso, K. v. d. Veldt, C. d. Laat, R. Hofman, and H. Bal. Profiling energy consumption of vms for green cloud computing. In *Proceedings of the 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*, DASC '11, pages 768–775, Washington, DC, USA, 2011. IEEE Computer Society.
9. D. Economou, S. Rivoire, and C. Kozyrakis. Full-system power analysis and modeling for server environments. In *In Workshop on Modeling Benchmarking and Simulation (MOBS*, 2006.
10. X. Fan, W.-D. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. In *Proceedings of the 34th annual international symposium on Computer architecture*, ISCA '07, pages 13–23, New York, NY, USA, 2007. ACM.
11. J. Fox and S. Weisberg. *An R Companion to Applied Regression*. SAGE Publicatotions, Los Angeles, 2011.
12. A. Gelman and J. Hill. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge University Press, Cambridge, 2009.
13. P. L. J. J. Dongarra. Introduction to the hpcchallenge benchmark suite. Technical report, ICL, 2004.
14. A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya. Virtual machine power metering and provisioning. In *Proceedings of the 1st ACM symposium on Cloud computing*, SoCC '10, pages 39–50, New York, NY, USA, 2010. ACM.
15. A. Kivity and al. kvm: the Linux virtual machine monitor. In *Ottawa Linux Symposium*, pages 225–230, July 2007.
16. L. Liu, H. Wang, X. Liu, X. Jin, W. B. He, Q. B. Wang, and Y. Chen. Greencloud: a new architecture for green data center. In *Proceedings of the 6th international conference industry session on Autonomic computing and communications industry session*, ICAC-INDST '09, pages 29–38, New York, NY, USA, 2009. ACM.
17. A. J. Younge, R. Henschel, J. Brown, G. von Laszewski, J. Qiu, and G. C. Fox. Analysis of virtualization technologies for high performance computing environments. In *The 4th International Conference on Cloud Computing (IEEE CLOUD 2011)*, Washington, DC, 07/2011 2011. IEEE, IEEE.