Integrating I/Os in Cloudsim for Performance and Energy Estimation

Hamza Ouarnoughi
b<>com Research Institute of Technology and Univ. Bretagne Occidentale
UMR 6285, Lab-STICC
hamza.ouarnoughi@b-com.com

Jalil Boukhobza
b<>com Research Institute of Technology and Univ. Bretagne Occidentale
UMR 6285, Lab-STICC
boukhobza@univ-brest.fr

Frank Singhoff
b<>com Research Institute of Technology and Univ.Bretagne Occidentale
UMR 6285, Lab-STICC
singhoff@univ-brest.fr

Stéphane Rubini
Univ. Bretagne Occidentale
UMR 6285, Lab-STICC
rubini@univ-brest.fr

ABSTRACT

This article presents an extension of the IaaS Cloud simulator *CloudSim*. This extension takes into account the processing of I/O workload generated by virtual machines within a data center, and evaluates the overall performance and energy consumption. Indeed, according to state-of-the-art studies, storage systems energy consumption may account for as much as 40% in a data center. So, we modified the time computation model of *CloudSim* to consider I/O operations. Additionally, we designed several models of storage system devices including Hard Disk Drives and Solid-State Drives. We also modeled CPU utilization to compute the energy consumptions related to I/O request processing. This was achieved through machine learning techniques. Our storage system extensions have been evaluated using video encoding traces. The simulation results show that a significant amount of energy, around 25%, is consumed due to I/O workload execution. This corroborates the soundness of our *CloudSim* extensions.

Keywords

Cloud Computing, CloudSim, Storage, I/O, Energy.

1. INTRODUCTION

IaaS Cloud Computing is an emerging technology supporting a new way of using hardware infrastructures. Cloud providers offer these infrastructures as virtualized hardware (i.e. CPU, storage, and network), managed by suitable software (i.e. virtualization technologies). They propose their services under the form of Virtual Machines (VMs), ready to be used on demand by Cloud customers.

Mastering the utilization costs of Cloud infrastructure represents a real challenge for Cloud providers. Power consumption represents a significant part of a data center cost [17]. Several approaches have been proposed in order to minimize data center energy consumption. One of the most commonly used is VM placement optimization which aims to find the optimal allocation of physical machines of a data center (i.e. host servers) to customers' virtual machines to reduce energy consumption. Most state-of-art VM placement optimization methods are based on CPU utilization [23]. They consider that power consumption of a given host depends exclusively on its CPU load. However, several studies, among which [10], have reported that other system components may greatly contribute to the overall host power consumption while others [20] emphasize that the power consumed by a storage system may represent up to 40% of the overall data center power consumption. Therefore, we believe that it is necessary to consider storage systems and associated workload in VM placement optimization.

The work presented in this article is part of an energy-aware VM placement optimization project that considers storage systems and I/O workload execution. For performance evaluation sake, we used one of the most popular approach [9] as a comparison baseline. In this previous work, authors used the Cloud simulator *CloudSim*[11]. Unfortunately, *CloudSim* does not consider I/O processing related time and energy consumption.

Even if *CloudSim* considers transfer time of Cloud customer binary files to the Cloud storage system, it does not take into account I/Os related to VM image creation and I/O workload execution. Many state-of-the-art studies aimed to provide storage system support in *CloudSim* [22][30][15][21]. To the best of our knowledge, none considered I/O workload processing or CPU load generated by I/O operations.

This article presents an extension of *CloudSim*. The proposed extension aims to take into account I/O workload processing in the overall simulation results. We then extended *CloudSim* as follows:

- We modeled the VM I/O workload execution in the time computation model of *CloudSim*,

WOPSSS '16 June, 2016, Frankfurt am Main, Germany

This work is based on an earlier work: "Considering I/O Processing in CloudSim for Performance and Energy Evaluation", in LNCS 9945: ISC High Performance 2016 International Workshops. ©2016 Springer. http://dx.doi.org/10.1007/978-3-319-46079-6_40

- We extended the *CloudSim* storage device entities to express both Hard Disk Drive (HDD) and Solid State Drive (SSD) devices performance and energy models,
- We designed and integrated a CPU correlation model to predict the CPU load related to the I/O workload processing.

The remainder of this article is organized as follows. Section 2 presents the related work. Section 3 describes the contribution of this paper. Section 4 presents the extension evaluation and section 5 concludes the article.

2. BACKGROUND AND RELATED WORK

In this section, we summarize state-of-the-art work about I/O and storage integration in *CloudSim*. There are other work that aim to consider storage capabilities in other simulators used in Cloud context such as *SimGrid* in [19] but as they do not target VM-based concepts nor storage system energy consumption, they fall out of the scope of this article.

CloudSim is a discrete event simulator that enables modeling and simulation of Cloud computing systems and application provisioning environments [11]. To the best of our knowledge, there are four state-of-the-art studies that dealt with storage in *CloudSim*. In [22], the authors implemented *CloudSimDisk*, a *CloudSim* extension based on an analytical energy consumption model for three hard drives. This extension considers transaction time and energy consumption related to adding and retrieving binary files executed by VMs. In [30], Sturm *et al.* target the simulation of a STaaS Cloud (STorage as a Service). This approach focuses on a pricing model of object storage in *CloudSim* [24], and gets around the usual use of VM execution concept in *CloudSim*. The closest work to ours introduced several extensions for *CloudSim* in order to overcome resource over-utilization, and then minimize costs [15]. To achieve such a purpose, Grozev *et al.* focused on load balancing algorithms by considering all resources, including the storage parts. Finally, Long and Zhao [21] proposed an extension of *CloudSim* storage system in order to maximize system performance. This approach targets the integration of file replication over storage devices and data centers.

Approaches in [22], [30], [15], and [21] exhibit three main drawbacks:

- VM I/O workload processing by storage system or CPU is not considered in the time and energy models of *CloudSim*;
- CPU utilization generated by I/O operations is not modeled or considered;
- when used in *CloudSim*, the storage system is limited to a shared SAN which only relies on one model of HDD.

In this work, we answer each of the three above mentioned issues. We propose an extension to simulate more accurate and realistic scenarios by considering I/O workloads and storage system performances. The next section details our extension for *CloudSim*. First, we present the concept of VM I/O workload execution and its related time and energy consumption. Second, we give an overview of our modeling of different classes of storage device in the *CloudSim* system.

3. APPROACH

3.1 Storage in CloudSim and Motivations

CloudSim is a discrete event simulator for IaaS Cloud developed in Java. It is composed of a set of entities: `Datacenter`, `DatacenterBroker`, `CloudInformationService` and `CloudsimShutdown`. Those entities model the main architectural elements of a data center. They communicate using predefined events (e.g. `VM_CREATE`, `VM_MIGRATE`, `VM_DESTROY`, etc). Events can be external (i.e. between different entities) or internal (i.e. sent and received by the same entity). When received by an entity, each event is handled by the receiving entity before sending an acknowledgment (e.g. `VM_CREATE_ACK`, `VM_MIGRATE_ACK`, `VM_DESTROY_ACK`, etc). A *CloudSim* simulation is based on the execution of a set of *Cloudlets*. A *Cloudlet* models a process with a CPU workload ran by a VM.

CloudSim allows users to simulate IaaS Cloud scenarios with different infrastructure architectures. It is usually used to experiment VM placement optimization strategies with a focus on energy efficiency or load balancing. From the storage system perspective, the latest *CloudSim* version uses a shared SAN (Storage Area Network), which is a set of similar hard drives [13], connected by a LAN (Local Area Network). From the I/O workload perspective, *CloudSim* mainly considers the time to store an input *Cloudlet* file during its submission. This time is obtained using hard drives performance metrics (latency and throughput), and the LAN throughput, but the related energy consumption is not accounted. In addition, the VM image and its I/O workload management are not considered.

The main objective of our contribution is to consider I/O workload execution of VM in an IaaS Cloud context. I/O workload impact on performance and energy consumption can be decomposed in two main elements, 1) the execution of the I/O workload on the storage devices containing the requested data, and 2) the execution of the I/O software stack for workload processing on host's CPU and RAM. Thus, executing I/O workload induces time and energy latencies on storage system devices, and also on CPU and RAM. Our contribution can be summarized as follows:

Time and energy computations for I/O processing: In order to consider I/O workload execution, we have updated both VM time and energy computation models to take into account I/O processing. We have chosen to define I/O workload at the VM level. Indeed, on one hand defining workloads at the VM granularity corresponds to the targeted objective (placement of VM), and on the other hand, it is more convenient as in real platforms, tracing is achieved at the hypervisor level [18].

Storage device support: Our extension updates the model of storage system devices already used in *Cloudsim*, especially the `Storage` interface and the class `HardDriveStorage`. The additional material targets mainly performance and power related characteristics.

I/O workload and CPU correlation: As discussed earlier, during I/O workload execution, a given VM does not only solicit the storage system, but also uses the CPU and RAM for I/O requests processing. In *CloudSim*, RAM utilization is driven by CPU usage. Based on performance measurement, we developed empirical correlation models in order to evaluate CPU usage according to synchronous I/Os. The designed model depends on I/O workload and storage device type.

3.2 Time and energy computations for I/O processing

3.2.1 Time computation

Simulated execution time in *CloudSim* depends on the time between sending an event by an entity and receiving the acknowledgment. Among the events that impact simulation time, we distinguish *Cloudlets* and VMs processing events (create, move, pause, resume, etc). Figure 1 shows a simple scenario for the execution of a *Cloudlet* on a VM. The first phase is the creation of a *Cloudlet* on the storage device (`CLOUDLET_SUBMIT` event). This event is produced once for each *Cloudlet* in the beginning of the simulation. Its duration depends on *Cloudlet* file size and storage system performance. The second phase is to execute *Cloudlet* instruction. Its duration depends on the *Cloudlet* length (i.e. number of instructions), and CPU performances of both VM and host. There is another event that implies I/O which is the *Cloudlet* relocation (`CLOUDLET_MOVE` event).

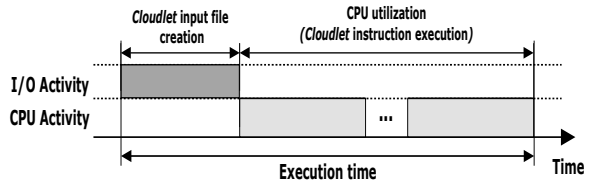

Figure 1: Time model before the extension

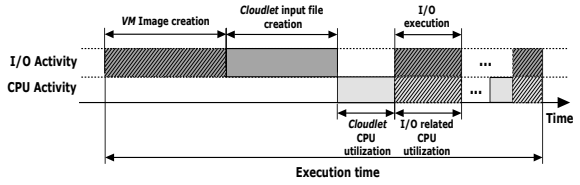

Figure 2: Time model after the extension

Figure 2 shows the event sequence that affects simulation time after our extension integration. Our contribution related to I/O time can be outlined as follows:

- VM *image creation:* VM creation is a data center internal event. We extended VM creation time by storing VM image on the storage device. This time varies depending on VM image size and storage device characteristics.

- I/O *execution:* we have introduced I/O execution time concept in order to simulate I/O workload execution on the storage device and related energy consumption. VM I/O execution is a data center internal event. I/O execution time depends on characteristics of the VM I/O workload and the storage device on which the VM image is stored. An I/O workload is characterized by: 1) read rate, 2) sequential rate, 3) I/O request size, 4) I/O request arrival rate, and 5) the total amount of processed data. This characterization is widely used in state-of-art work, and specially in virtualized environment [16].

3.2.2 Energy consumption computation

I/O workload execution does not affect only execution time, but also the system energy consumption. On the previous version of *CloudSim*, energy consumption depended solely on hosts CPU load. Equation (1) shows how *CloudSim* gets the host power consumption $P(u)$, depending on its CPU utilization u:

$$P(u) = k \cdot P_{max} + (1 - k) \cdot P_{max} \cdot u \quad (1)$$

P_{max} denotes the maximum power of the host machine (i.e. when CPU utilization is equal to 100%). k is the ratio between the maximum power and the idle power (i.e. when CPU is not used). In [9], the authors consider that a host machine consumes 70% of the maximum power in idle mode, which means that $k = 0.7$.

The input parameter of the power function P presented in equation (1) represents the host CPU utilization (noted u). This utilization depends on VM CPU workload and the host CPU performance. The CPU workload of a given VM is the sum of all *Cloudlets* CPU workloads given in MI (Million Instructions). *CloudSim* uses the MIPS (Million Instructions Per Second) as the only CPU performance metric.

Figure 3 is an illustrative figure that shows the power consumption with and without considering the storage system. The top timeline is obtained by the initial power model used in *CloudSim* that computes the host power consumption (CPU and RAM) only from the CPU utilization. P_{max} and k values are obtained from experimentations presented in [9]. This diagram illustrates three power phases for a given host. The first and the last phase show the power consumption during idle mode which represents here 0% of CPU utilization. During the second phase, the host executes VMs CPU workload.

Our model completes the initial power consumption model in *CloudSim*, by adding the energy consumption related to VMs I/O workload execution.

The bottom time-line in figure 3 illustrates the result after we updated *CloudSim*. Notice that a storage device in idle mode can consume power (e.g. up to 60% of the power during operational mode [29]), and as consequence, the power in idle mode is also increased. During the second phase, the host alternates between I/O and CPU workloads execution. Storage system and I/O workload affect the consumed energy in terms of time and power.

3.3 Storage device support

We extended the storage system of *CloudSim* by implementing two storage device types: HDD and SSD. We also added several class (see figure 5). We classified the added attributes and methods in two main classes: 1) performance-related, and 2) power-related ones. For each storage device, performances depends on I/O workload characteristics (e.g. read/write, sequential/random, I/O size, etc.), and device performances properties: latency, average seek time, data transfer rate (for sequential access), and IOPS (for random access).

The energy consumed by a storage device depends on: its performance, I/O workload characteristics, and power properties (i.e. power for random/sequential and read/write operations, idle power, standby power). We also implemented a storage system energy consumption model that we have already presented in previous work [26].

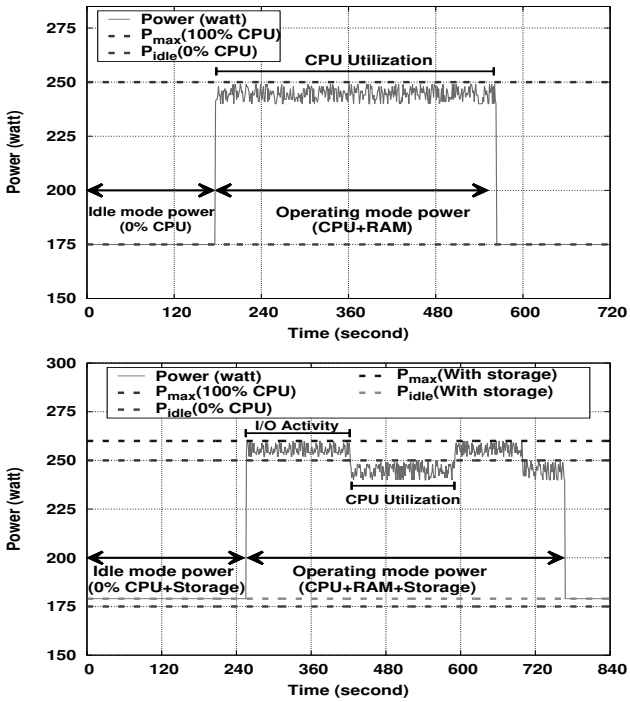

Figure 3: Power model before and after our extension for HDD and SSD (with idle and operational power values variation but a similar behavior).

3.4 I/O Workload and CPU correlation

Simulating I/O workload execution does not only imply storage system activities, but also CPU utilization due to I/O software stack execution. In order to quantify and implement the CPU involvement in synchronous I/O processing, we have established an empirical correlation model that gives CPU load depending on I/O workload characteristics. To obtain a CPU correlation function that depends on our I/O workload and storage system models, we follow two steps:

1. Benchmarking step
2. Modeling step

Figure 4 shows the followed steps to model the CPU utilization due to I/O processing.

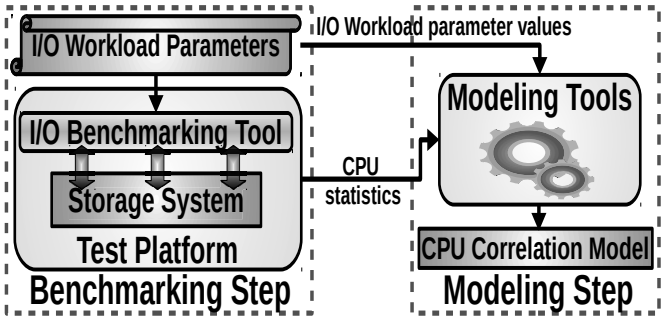

Figure 4: CPU Correlation modeling steps

The next subsections detail the benchmarking and the modeling step.

3.4.1 Benchmarking Step

The benchmarking step consists in running an I/O benchmark with different scenarios. Each scenario represents a combination of benchmarking input parameter values. This step targets to get the system CPU load, depending on varied input parameters. Benchmark input parameters are divided into two subsets: a) I/O workload related parameters, and b) storage system related parameters.

a) I/O Workload Parameters

The workload associated to a VM is modeled by a set of parameters [26]. We have used five parameters to accurately represent an I/O workload executed by a given VM: 1) I/O requests pattern in term random rate, 2) I/O request type in term of write rate, 3) average I/O size, 4) I/O requests arrival rate, and 5) total amount of data. The first four parameters have been varied in order to observe the related CPU load variation, and identify the more impacting ones. The last parameter (i.e. total amount of data) have been fixed, as we have observed during experimentation that it has no impact on the CPU load, but only on the execution time. Parameter value interval are presented in the evaluation section (section 5).

b) Storage System Parameters

As mentioned in section 3.3, our extension adds the support of two storage device classes: 1) hard disk drives, and 2) solid state drives. Thus, the varied parameter in this case is the storage device type (i.e. HDD or SSD). In the evaluation part, we take as a case study two storage devices with different performance specifications, in order to illustrate the CPU load variation depending on the storage device type.

3.4.2 Modeling Step

Table 1 gives an overview of the notations used all along the CPU correlation modeling process.

Parameter	Description
$Load()$	Function that models the CPU load
D	Storage device, where $D \in \{HDD, SSD\}$
W	I/O workload executed
cpu_{load}	CPU load during the I/O workload execution
$rate_{rnd}$	Rate of random I/O request
$rate_{wrt}$	Rate of write I/O request
$rate_{req}$	I/O request arrival rate
$size_{I/O}$	Average I/O request size

Table 1: Correlation model parameter notations

The global view of the correlation model my be formalized as follows:

$$Load(D, W) = cpu_{load} \quad (2)$$

Equation (2) shows that the input of the correlation model function (noted $Load$), is a storage device (noted D) and an I/O workload (noted W). The output of the correlation model function is the CPU load (noted cpu_{load}) generated during the I/O workload execution. The correlation model is designed to support two storage classes, which are hard disk drives and solid state drives (noted HDD and SSD respectively):

$$D \in \{HDD, SSD\} \quad (3)$$

The second input parameter of the model function represents the VM associated I/O workload. In our study, we model an I/O workload using four parameters:

$$W =< rate_{rnd}, rate_{wrt}, rate_{req}, size_{I/O} > \quad (4)$$

By combining equations (4), (3), and (2), we note that the CPU load depends on at least five variables. We used machine learning approaches to produce a correlation model for CPU load prediction according to I/O workload parameters. The used approaches are *Ordinary Least Squares (OLS)* and *Multivariate Adaptive Regression Splines (MARS)*, which are two multivariate linear regression methods. The justification of such a choice is the following:

- *OLS* is the most simple and frequently used linear regression model (e.g. the default model of `scikit-learn` tools [7], MS Excel [6], Libreoffice Calc [5]).
- *MARS* model chooses during modeling process only impacting parameters and designs the CPU load model around this parameters, where OLS model takes all input parameters.

The rest of this section gives an overview of each one of the used linear regression methods.

a) Modeling by Ordinary Least Squares (OLS): *Ordinary Least Squares* is a widely used linear regression method. For a given experimentation, *OLS* aims to find the relationship between input variables and observed results. Specifically, this method targets to minimize the error between real observed results and the produced prediction model. In our case and for a given storage device, the CPU load is modeled depending on I/O workload parameters as follows:

$$cpu_{load} = \alpha_0 + \alpha_1 rate_{rnd} + \alpha_2 rate_{wrt} + \alpha_3 rate_{req} + \alpha_4 size_{I/O} \quad (5)$$

Based on equation (5), *OLS* regression model targets to calculate the coefficients α_0, α_1, α_2, α_3, and α_4, while minimizing the error between the calculated CPU load and the real one.

b) Modeling by Multivariate Adaptive Regression Splines (MARS): To deal with the non-linearity of the CPU load variation while taking into account several variables [12]. MARS models are mainly based on hinge loss functions [27]. Hinge functions is a function type frequently used in machine learning algorithms. The equation shows the definition of a hinge function h:

$$h(x,t) = [x-t]_+ = \begin{cases} x-t & if\ x > t \\ 0 & if\ x \leq t \end{cases} \quad (6)$$

where x is one input model variable and t is a constant defined by MARS algorithm. This algorithm builds the model following two main steps: 1) forward pass, and 2) backward/pruning pass. During the forward pass, MARS searches firstly the intercept term (i.e. α_0 in the equation (5) when all input parameters are equal to zero). After getting the intercept term, the forward pass adds hinge functions for input parameters while minimizing the error rate between the model output and the real one (extracted from experimentation). In our case, a simple example of forward pass output may be given as follows:

$$cpu_{load} = \alpha_0 + \alpha_1 h(rate_{rnd} - t_1) + \alpha_2 h(rate_{wrt} - t_2) + \alpha_3 h(rate_{req} - t_3) + \alpha_4 (size_{I/O} - t_0) \quad (7)$$

The backward/pruning pass takes the model resulting from the forward pass, then tries to refine it by dropping least impacting terms using the *Generalized Cross Validation* GCV [14]. As pruning pass results, the final model may contain several hinge functions for one or more input variables, and no hinge function for some others. The evaluation section 5 shows the obtained model and its validation metrics.

4. IMPLEMENTATION DETAILS

We extended already existing *CloudSim* classes without any modification of the previous official implementation. Figure 5 shows the UML diagram of our proposed storage extension. The diagram presents only added and mainly extended classes that impact storage system and I/O execution. All classes and interfaces with the dark color have been added in our extension. Methods and attributes presented in this diagram (from the initial implementation) have been overridden by our extension.

4.1 Time and energy for I/O processing

As shown previously (see subsection 3.2), the I/O processing related time impacts the total simulated execution time. This impact is implemented as an internal event (`VM_IO`) within a Data Center. The event sending, handling, and receiving is implemented in the class `IoDatacenter`. The I/O processing time depends mainly on the storage device storing the VM files (implemented in the class `IoVm`), and the VM I/O workload (implemented in the class `IoWorkloadModel`).

The energy consumption related to the I/O workload execution is calculated using the cost model [26] implemented in the class `StorageEnergyModel`. There is an additional energy related to the CPU utilization related to I/O execution, which is calculated by the native CPU power model of *CloudSim*. In order to implement the storage device diversity (HDD and SSD), we extended the existing storage interface named `Storage`.

4.2 Storage Device Support

Our extension proposes two storage device type support: 1) hard disk drive implemented by the class (`HardDriveStorage`), and 2) solid state drive implemented by the class (`SolidStateStorage`). These two classes implement the existing interface named `Storage`. The storage devices may be attached on hosts and used as direct attached storage, and/or used as a shared storage (i.e. NAS storage).

4.3 I/O Workload and CPU Correlation

In addition to different storage device class support, we extended *CloudSim* by considering the CPU utilization generated by the I/O execution process. The proposed CPU correlation model studied in this article (see subsection 3.4), has been implemented by the class `IoCpuCorrelationModel`. The CPU correlation model depends on a given host. More precisely, such a model depends on performances of the CPU and storage devices attached to a given host.

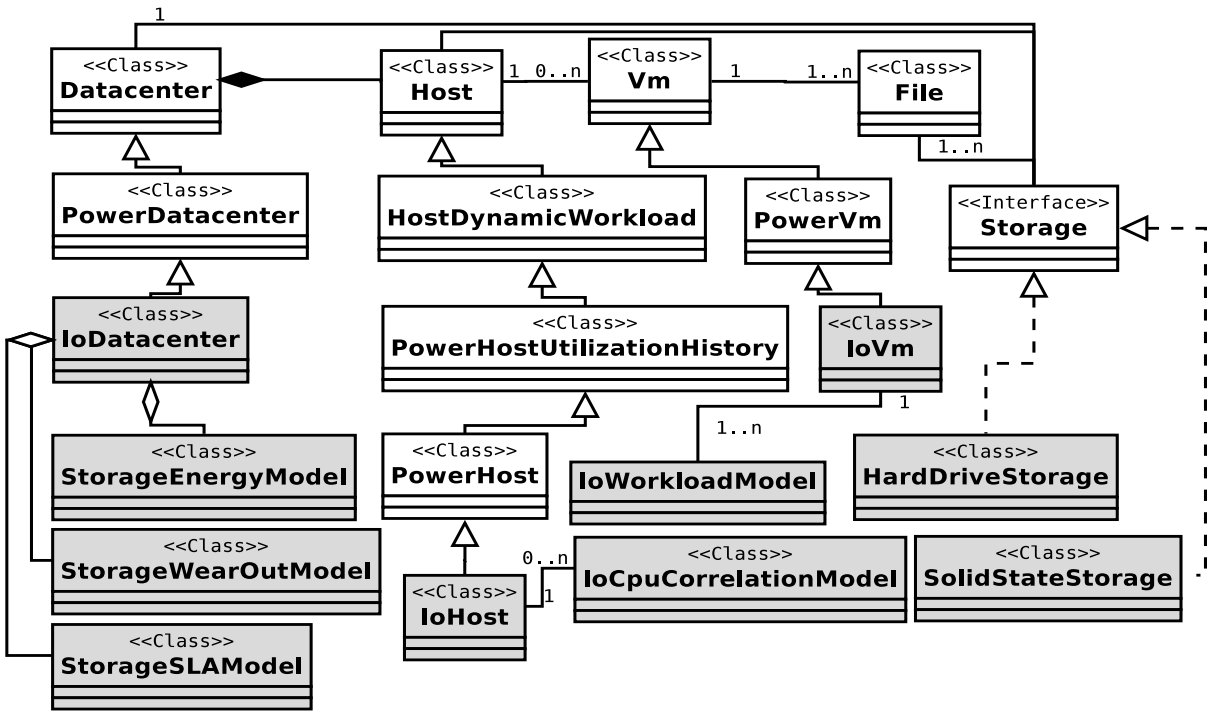

Figure 5: UML Class diagram of the main added and modified elements in *CloudSim*.

At this point, we have detailed the theoretical and implementation aspects of our proposed extension. The next section presents the evaluation methodology of our proposition, and obtained results.

5. EVALUATION

This section presents the evaluation of our contributions. The evaluation is divided into two parts. The First part shows the validation methodology and results of the CPU correlation model with the two regression models used (see 3.4.2). The second evaluation part targets to validate our proposed extension with different scenarios.

5.1 CPU Correlation Model Evaluation

In order to evaluate the CPU correlation model, we proceed following the two steps presented in section 3.4. The first step targets to measure real CPU utilization for different scenarios. The second step aims to obtain a CPU prediction model, based on varied parameters and real CPU utilization. To validate the obtained results, we compared the two models (*OLS* and *MARS*) using the coefficient of determination (also called R^2).

5.1.1 Evaluation Methodology

In the benchmarking step, we ran the I/O benchmark `fio` [1] while varying a set of input parameters. Table 2 shows all varied parameters.

The first varied parameter is the storage device type. We evaluated the CPU correlation model for two storage device classes; SSD and HDD. The rest of the input parameters are I/O workload related parameters. The random and write rates were varied to evaluate the impact of mixed random/sequential and read/write I/O requests. For the I/O request rate, `fio` uses all the available storage device bandwidth. The I/O request size was varied in order to quantify the impact of average I/O size on the CPU load.

Parameter	Values & Ranges
D	$D \in \{HDD, SSD\}$
$rate_{rnd}$	[0%,...,100%] by steps of 20%
$rate_{wrt}$	[0%,...,100%] by steps of 20%
$rate_{req}$	Available storage device performance
$size_{I/O}$	[4KB,...,1024KB] with logarithmic scale

Table 2: Model parameters values and ranges

For all benchmark scenarios, we configured `fio` to run only synchronous I/O requests. We aim by this configuration to minimize the impact of system caches and buffers, and then minimize the RAM interference. In order to keep the same initial start state, test partitions was formatted before each benchmark execution. Each scenario benchmark has been executed five times, and average output values was considered.

5.1.2 Evaluation Setup

For the hardware setup, experiments have been achieved on a computer with an Intel(R) Xeon E5-1620 3.70GHz CPU, and 16GB RAM. Concerning the storage system, we used two partitions with identical size, 128GB. We used two storage devices: a Seagate ST1000DM003 HDD, and a Samsung 840 PRO SSD.

For the software setup, the experimental computer runs a Linux 3.2.0 kernel. Test partitions was formatted with EXT4 file system. The used I/O benchmark is `fio` version 2.10-2 [1]. For modeling and analysis process, we used the Python machine learning tool kit named `scikit-learn`

[7], and the MARS open-source implementation [4] often referred to as Earth.

5.1.3 Evaluation Results

After getting the results of CPU utilization depending on the I/O workload and storage system parameters, we used them with the input parameter to build a CPU correlation model.

We first split the obtained results into two datasets. The first dataset represents 70% of the obtained results, and was used to model the CPU load (often called *training features* in machine learning algorithms). The second dataset represents the remaining 30% of the results and was used to validate the obtained model (often called *validation features* in machine learning algorithms).

In order to model the CPU load, we have used two regression approaches as presented in the section 3.4.2. Before In this subsection, we present and compare the obtained results using OLS and MARS approaches by using the coefficient of determination (often called R^2). The coefficient of determination is the most used statical metric, to evaluate the accuracy of predicted values using a regression model as compared to real obtained values. T Before applying the modeling approaches, we have normalized the input parameters values.

b) OLS Model Results: the obtained CPU correlation model when using the Ordinary Least Squares regression method is given by equation (8).

$$cpu_{load} = 0.0353 + (-0.0232 \cdot rate_{rnd}) + (0.0331 \cdot rate_{wrt}) + (-0.0689 \cdot size_{I/O}) + (-0.4346 \cdot \frac{1}{rate_{req}}) + (0.1378 \cdot D) \quad (8)$$

The obtained OLS equation (8) uses all input parameters to model the CPU correlation model. The coefficient of determination $R^2 = 0.596$ which represents an inaccurate model. Figures 6 and 7 show a comparison example of CPU load obtained from experimentation, and that obtained using models. They confirm the inaccuracy of the model.

c) MARS Model Results: the CPU correlation model obtained using the Multivariate Adaptive Regression Splines regression technique is given by equation (9).

$$\begin{aligned} cpu_{load} = {} & 0.1526 + [328.027 \cdot h(\frac{1}{rate_{req}}, 0.00024)] \\ & + [9.4753 \cdot h(0.0038, \frac{1}{rate_{req}})] \\ & + [-1154.85 \cdot h(\frac{1}{rate_{req}}, 7.1882 \cdot 10^{-5})] \\ & + [3771.77 \cdot h(7.1882 \cdot 10^{-5}, \frac{1}{rate_{req}})] \\ & + [100.137 \cdot h(\frac{1}{rate_{req}}, 0.00063)] \\ & + [726.126 \cdot h(\frac{1}{rate_{req}}, 0.00012)] \\ & + (0.0091 \cdot rate_{wrt}) + (0.0102 \cdot size_{I/O}) + (0.02397 \cdot D) \end{aligned} \quad (9)$$

Unlike the OLS model previously presented in equation (8), the MARS model 9 chooses the most impacting inputs to model the CPU load. We noticed that the most important parameter is the I/O request rate, which is used by all hinge functions. The other observation is that random rate parameter $rate_{rnd}$ has been eliminated by the MARS algorithm. The coefficient of determination $R^2 = 0.98$, which means a very accurate model.

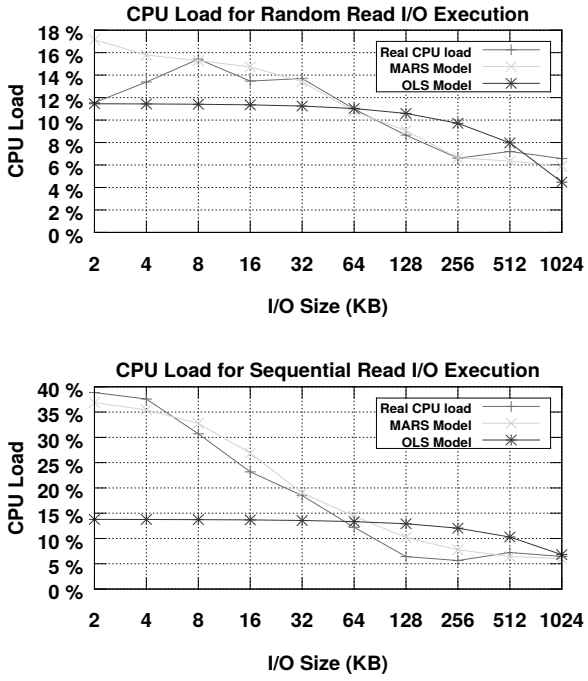

Figure 6: SSD Read I/O: Real vs Models

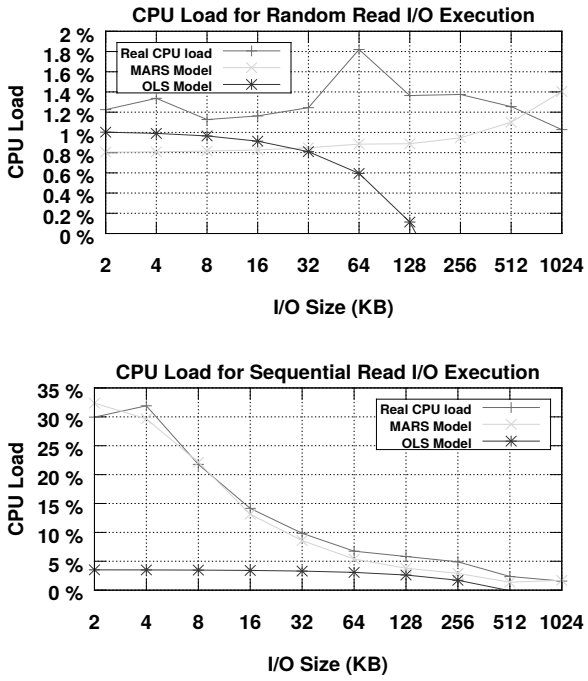

Figure 7: HDD Read I/O: Real vs Models

Figures 6 and 7 show examples of CPU load obtained using studied regression models OLS and MARS. We give examples of random and sequential read operations (i.e. $rate_{rnd}$ = 100% and $rate_{rnd}$ = 0%, while $rate_{wrt}$ = 0%), for SSD and HDD. Overall, we notice that the CPU load is the highest for sequential I/O and small I/O request sizes. The CPU load become less important for big size I/O requests. This trend is due to the I/O request processing by the CPU, which is higher for small size and sequential requests as compared to large size and random requests. The second observation is the difference between CPU load trend for sequential request as compared to random ones. For all cases, the MARS CPU model fits better with the real obtained results as compared to OLS model. This observation confirms the obtained coefficient of determination R^2. In the next section, we used the obtained correlation models to evaluate the impact of storage on the overall energy consumption of a computer system.

5.2 Extension Implementation Evaluation

This section presents the evaluation of our extension in two parts. The first one validates the claim that I/O workload processing contributes highly in the energy budget while the second part validates the CPU correlation model usage.

5.2.1 CPU, memory and I/O traces simulation

This first evaluation assesses the I/O processing energy consumption of a real use case. To do so, we proceeded in two steps, figure 8 summarizes the methodology:

1. *Real workload execution*: we ran experiments in a real environment and we collected a set of measures on the CPU, memory load, and I/O requests.
2. *Workload simulation*: from the collected measures and traces, we replayed the same scenario in *CloudSim* by turning ON and OFF the I/O processing including the storage system.

In this evaluation step, as the CPU load is obtained from traces, the CPU correlation model were used to quantify I/O processing related CPU load, among the total CPU load[1].

1) Real workload execution phase: as mentioned above, the first evaluation phase includes two steps: a) workload execution, and b) measures collection.

a) Workload execution: this step consists of running a workload in VMs (step 1 in figure 8). We chose a video transcoding benchmark as a use case of VMs running in a Cloud environment. Virtual machines encode video from `mov` format [8], to several videos in TS (Transport Stream) format [28]. This process is used for video streaming using HLS (HTTP Live Streaming) protocol [3]. We used H264 for video transcoding and HE-AAC for audio transcoding. Eight VMs were used in this experiment, four of which were stored in HDD and four in SSD. We performed 4 experiments in which we varied the number of VMs running on (HDD, SSD) as follows: (1, 1), (2, 2), (3, 3), and (4, 4). Each VM has a 20GB image and all VMs have the same computing resource configuration (1 VCPU, 1GB RAM).

[1]All benchmarking and data extraction tools are available in our github repository [2].

b) Trace collection: during workload execution, CPU and memory utilization were monitored, in addition to I/O. CPU and memory traces were aggregated to an average value related to one unit of time (defined to 5 mn in *CloudSim*). Concerning I/O traces, each VM had its own I/O trace file gathered at the I/O block level [25]. Each I/O trace file line includes five fields formatted as follows:
`<data_amount>,<read_rate>,<random_rate>,<io_size>,-<io_arrival_rate>`.

The first field (i.e. `<data_amount>`) represents the total read and written amount of data. Second and third fields (i.e. `<read_rate>` and `<random_rate>`) denote rates of read requests and access pattern randomness respectively. Fourth field represents I/O request size in bytes. The last field denotes request arrival rate during sample interval.

2) Workload simulation phase
Simulation phase includes two main steps: a) scenario simulation using CPU, memory, and I/O traces, and b) comparing simulation results by enabling and disabling I/O processing.

a) Simulation using real traces: in this step we used traces obtained from real workload execution phase. CPU and memory utilization are attached to hosts, while I/O traces are attached to VMs. Simulations were executed by varying the following parameter: 1) number of VMs per host, 2) storage system ON/OFF, and 3) storage device HDD/SDD (when storage system is ON). Configurations about the number of VMs were the same as for the real workload execution phase and we varied the number of hosts. The total number of VMs was set to 290 VMs as in [9]. The number of active hosts depends on the number of VMs per host (i.e. unused hosts are shutdown).

b) Simulation results comparison: I/O energy consumption is used in order to show and quantify the impact of the I/O processing and storage system. Section 5.2.2 shows the obtained results from the evaluation.

5.2.2 Evaluation results

This section shows the obtained results from the two parts of the evaluation. Two results are shown, first, the difference between simulations without I/O consideration and those with storage system and I/O processing enabled. This evaluation aims to show the impact of storage system and storage device types on the overall consumed energy. Second, the energy consumption related to I/O workload processing which validates the impact of each component contributing to the I/Os processing (i.e. storage devices and CPU).

Figure 9 shows the energy consumed when disabling and enabling storage system and I/O processing extension. The first observation is the trend of consumed energy which drops when increasing the number of VM by host. This trend is due to the number of active hosts that decreases by increasing the number of hosted VM per physical machine for a fixed total number of VMs. The second observation is the difference between the energy consumption with and without storage system and I/O processing enabled. We note that I/O processing and storage system energy may consume significant energy amount ($\sim$ 25% in the case of HDD and $\sim$ 17% in the case of SSD).

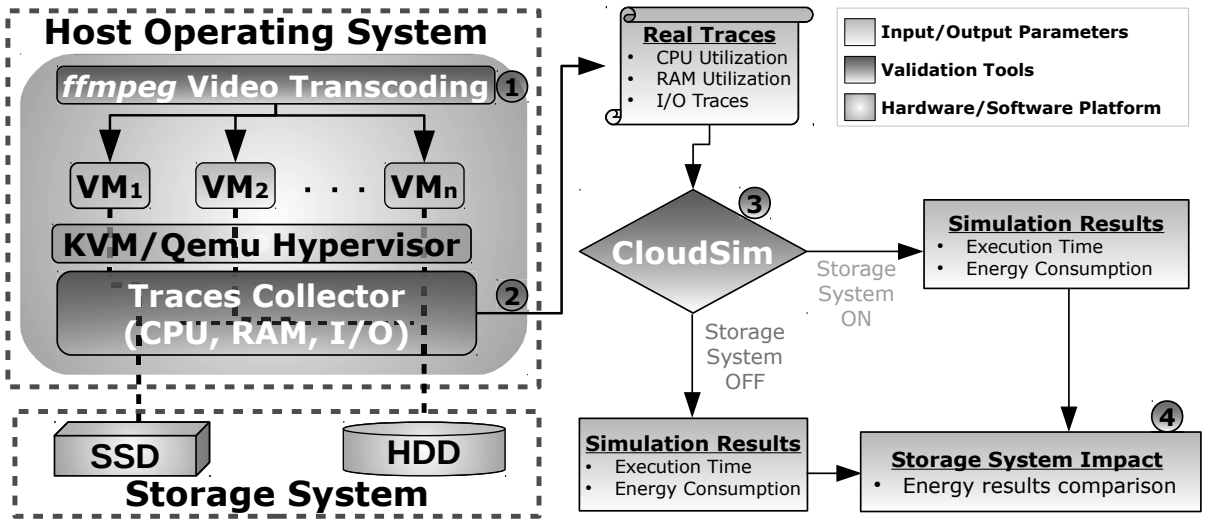

Figure 8: Evaluation methodology. The four steps are: (1) run benchmarks, (2) collect measures, (3) collect simulation results with and w/o I/O processing and storage system effects, (4) compare the results.

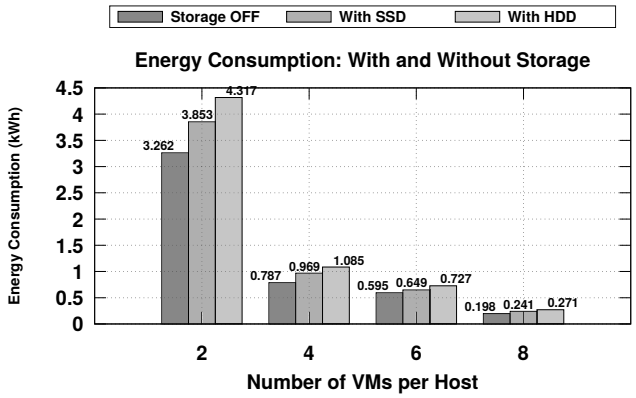

Figure 9: Energy with I/O processing ON and OFF

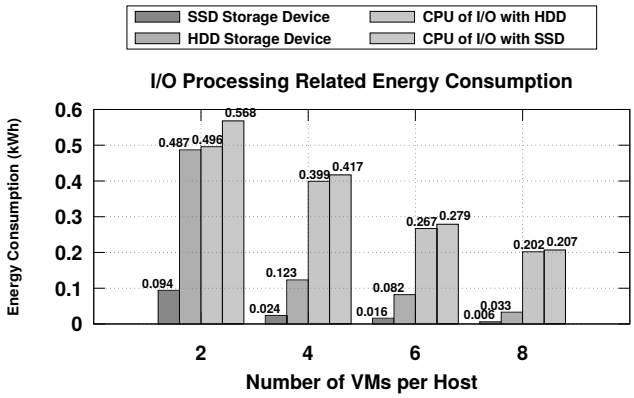

Figure 10: I/O processing related energy

Figure 10 presents the I/O processing and storage system related energy. It shows the energy consumed by storage devices (i.e. SSD and HDD), in addition to that consumed by the CPU during I/O processing on each storage device. We can observe that the most significant amount of energy is consumed by the CPU in most cases. The obtained results shown in the figure 10 confirms the high CPU load observed in the correlation model evaluation part, and specially in the case of SSD (see section 5.1.3). We note also that, the energy amount consumed by the CPU for both SSD and HDD is almost close (∼5% of average error). This is due to the sequential pattern and read rate exhibited by the I/O workload (∼88% of sequential rate and ∼82% of read rate), which produces a close CPU load for both SSD and HDD (see figures 7 and 6). Even if the most important part of energy is consumed by the CPU, one can observe that storage devices and especially HDD consume also energy. This observations must be considered and integrated to studies targeting energy efficient optimization in data centers.

6. CONCLUSION

Processing I/O requests does not only depend on the used storage system devices, especially when one considers the energy metric. Indeed, while performing I/O operations the CPU and RAM consumes a significant part of the energy while executing the I/O kernel software stack or waiting for I/O completion. This makes the overall energy related to I/O processing substantial enough to be considered.

This paper presents an extension of *CloudSim* to take into account I/O workload processing. Our extension considers I/O workload execution time and energy consumption by: 1) updating the time and energy computation model of *CloudSim*, 2) taking into account storage systems diversity (i.e. HDD and SSD), and 3) including a CPU correlation model depending on VM I/O workload and storage device type in order to represent CPU and RAM I/O processing time and energy. Simulations with video transcoding applications validated the impact of I/O processing on energy consumption to up to 25% of total energy consumption. As a perspective, this implementation will be used in a VM placement optimization approach that takes into account I/O processing cost. We also plan to study and integrate the problem of interference between VMs sharing the same storage device.

7. REFERENCES

[1] Aaron carroll: fio. http://linux.die.net/man/1/fio. Acces in Aug 2016.

[2] Hamza ouarnoughi: evaluation tools. https://github.com/Houarnoughi/sigops_osr_tools. Acces in Sep 2016.

[3] HTTP live streaming overview. https://developer.apple.com/library/mac/documentation/NetworkingInternet/Conceptual/StreamingMediaGuide/UsingHTTPLiveStreaming/UsingHTTPLiveStreaming.html. Accessed in Apr 2016.

[4] Jason rudy: py-earth project. https://github.com/jcrudy/py-earth. Accessed in Aug 2016.

[5] Libreoffice calc: Linest function. https://help.libreoffice.org/Calc/Array_Functions/fr#Other_LINEST_Results:. Acces in Aug 2016.

[6] Ms excel: Linest function. https://support.office.com/en-us/article/LINEST-function-84d7d0d9-6e50-4101-977a-fa7abf772b6d. Acces in Aug 2016.

[7] Scikit-learn. http://scikit-learn.org. Acces in Aug 2016.

[8] I. Apple Computer. Quicktime file format. Technical report, www.apple.com, 2001.

[9] A. Beloglazov, J. Abawajy, and R. Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer System*, 28, May 2012.

[10] R. Bianchini and R. Rajamony. Power and energy management for server systems. *Computer*, 37, Nov. 2004.

[11] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya. Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice & Experience*, 41, Jan. 2011.

[12] J. H. Friedman. Multivariate adaptive regression splines. *The annals of statistics*, pages 1–67, 1991.

[13] G. Gasior. Maxtor's diamondmax 10 hard drive. Technical report, Seagate, Accessed in Jan 2016.

[14] G. H. Golub, M. Heath, and G. Wahba. Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21(2):215–223, 1979.

[15] N. Grozev and R. Buyya. Multi-cloud provisioning and load distribution for three-tier applications. *ACM Transactions on Autonomous and Adaptive Systems*, 9, Oct. 2014.

[16] A. Gulati, C. Kumar, and I. Ahmad. Modeling workloads and devices for io load balancing in virtualized environments. *SIGMETRICS Perform. Eval. Rev.*, 37, Jan. 2010.

[17] J. Hamilton. Cost of power in large-scale data centers. Technical report, perspectives.mvdirona.com, Accessed in Apr 2008.

[18] A. Irfan. Easy and efficient disk i/o workload characterization in vmware esx server. In *IEEE 10th International Symposium on Workload Characterization*, Sept 2007.

[19] A. Lebre, A. Legrand, F. Suter, and P. Veyre. Adding Storage Simulation Capacities to the SimGrid Toolkit: Concepts, Models, and API. In *Proceedings of the 15th IEEE/ACM Symposium on Cluster, Cloud and Grid Computing*, 2015.

[20] Z. Li, K. M. Greenan, A. W. Leung, and E. Zadok. Power consumption in enterprise-scale backup storage systems. In *Proceedings of the Tenth USENIX Conference on File and Storage Technologies*, February 2012.

[21] S. Long and Y. Zhao. A toolkit for modeling and simulating cloud data storage: An extension to cloudsim. In *International Conference on Control Engineering and Communication Technology, Liaoning, China*, 2012.

[22] B. Louis, K. Mitra, S. Saguna, and C. Ahlund. Cloudsimdisk: Energy-aware storage simulation in cloudsim. In *IEEE/ACM International Conference on Utility and Cloud Computing*, 2015.

[23] Z. A. Mann. Allocation of virtual machines in cloud data centers—a survey of problem models and optimization algorithms. *ACM Computing Surveys*, 48, Aug. 2015.

[24] M. Mesnier, G. R. Ganger, and E. Riedel. Object-based storage. *IEEE Communications Magazine*, 41, Aug. 2003.

[25] H. Ouarnoughi, J. Boukhobza, F. Singhoff, and S. Rubini. A multi-level I/O tracer for timing and performance storage systems in iaas cloud. In *3rd IEEE International Workshop on Real-time and distributed computing in emerging applications*, 2014.

[26] H. Ouarnoughi, J. Boukhobza, F. Singhoff, and S. Rubini. A cost model for virtual machine storage in cloud iaas context. In *24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, 2016.

[27] L. Rosasco, E. De Vito, A. Caponnetto, M. Piana, and A. Verri. Are loss functions all the same? *Neural Computation*, 16(5):1063–1076, 2004.

[28] D. Ruiu. An overview of mpeg-2. Technical report, hewlett packard, 1997.

[29] Seagate. Barracuda st1000dm003. Technical report, http://www.seagate.com, Accessed in Mar 2016.

[30] T. Sturm, F. Jrad, and A. Streit. Storage cloudsim - A simulation environment for cloud object storage infrastructures. In *Proceedings of the 4th International Conference on Cloud Computing and Services Science*, 2014.