Computer Networks 94 (2016) 1–13

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

A joint CPU-RAM energy efficient and SLA-compliant approach for cloud data centers

Pedro H.P. Castro [a], Vívian L. Barreto [a], Sand Luz Corrêa [a,*], Lisandro Zambenedetti Granville [b], Kleber Vieira Cardoso [a]

[a] Federal University of Goiás, Goiânia, Brazil
[b] Federal University of Rio Grande do Sul, Porto Alegre, Brazil

ARTICLE INFO

Article history:
Received 20 April 2015
Revised 14 October 2015
Accepted 27 November 2015
Available online 2 December 2015

Keywords:
Cloud computing
Energy efficiency
Resource management
Virtualization
Dynamic consolidation

ABSTRACT

Cloud computing is a new paradigm that offers computing resources in a virtualized way with unprecedented levels of flexibility, reliability, and scalability. The benefits of cloud computing, however, come at a high cost in terms of energy consumption, mainly because of one of the cloud's core enablers, the data center. There are a number of proposals that seek to enhance the energy efficiency of data centers. Still, most of them focus on the energy consumed by CPU and ignore other important hardware components, *e.g.*, RAM. In this paper, we show the considerable impact that RAM can have on the total energy consumption, particularly in servers with large amounts of this memory. We then propose two new approaches for dynamic consolidation of virtual machines in cloud data centers that take into account both CPU and RAM usage. We have implemented and evaluated our proposals in the CloudSim simulator using real-world traces and compared the results with other state-of-the-art solutions. By adopting a wider view of the system, our proposals can reduce not only energy consumption but also service level agreement (SLA) violations, thus providing a better service at a lower cost.

1. Introduction

In the cloud computing paradigm, computing resources are offered as a service to consumers. That leads to the establishment of a new computing model in which such resources are provisioned and allocated differently than in traditional scenarios. By connecting to cloud infrastructures, it is possible to access remotely a collection of virtualized resources that include processing, memory, storage, and data communication. Computing resources are dynamically provided, and their use is expected to comply with Service Level Agreements (SLAs) brokered between infrastructure providers and consumers [1]. This computing model has shown to serve properly a wide variety of consumers: from the personal user who needs a small virtual disk, to the enterprise user who outsources processing and storage on a large-scale.

Cloud computing elevates distributed systems to a new level, offering computing resources in a virtualized way with unprecedented levels of flexibility, reliability, and scalability. However, attention must be paid to the energy consumed by cloud infrastructures. Cloud computing environments are deployed in data centers that can contain hundreds of thousands of servers. In addition to hardware resources (*e.g.*, servers, storages, and switches), data centers are equipped with HVAC (heating, ventilating, and air conditioning) control systems that are responsible for nearly half of all the energy consumed by the infrastructure [2]. In this context, reducing the amount of energy consumed by computing resources not only results in a reduction in the operating costs of data centers, but it also has a positive effect on the environment. Since data centers are large-scale computing

* Corresponding author. Tel.: +55(62)3521-1181.
E-mail addresses: pedrocastro@inf.ufg.br (P.H.P. Castro), vivianbarreto@inf.ufg.br (V.L. Barreto), sand@inf.ufg.br (S.L. Corrêa), granville@inf.ufrgs.br (L.Z. Granville), kleber@inf.ufg.br (K.V. Cardoso).

http://dx.doi.org/10.1016/j.comnet.2015.11.026
1389-1286/

infrastructures, this effect is significant. As reported by Greenpeace [3], cloud infrastructures and the use of Internet have produced a collective electricity demand that would currently rank in the top six in the world, and this demand is expected to increase by 60% or more by 2020.

The issue of energy consumption in data centers can be divided into two key factors. First, there is a hardware problem regarding the amount of computing resources and the power inefficiency of their physical components. Studies have shown that even when resources are idle, current servers still consume about 70% of their peak power [4]. Second, there is a software issue concerning the inefficient usage of computing resources. It has been reported that, most of the time, data center servers operate at less than half of their total capacity [5], thus wasting energy and causing unnecessary heat.

While the energy inefficiency of physical components has been tackled by solutions in hardware and firmware (adopting approaches such as dynamic component deactivation [6,7] and dynamic voltage and frequency scaling (DVFS) [8–10]), the software problem of underutilization of computing resources can be addressed by employing diverse techniques based, for example, on self-organization/dynamic reconfiguration [11] and on virtualization [12–17]. In this paper, we focus on virtualization because virtualization is core to modern data centers. Basically, that consists in creating virtual machines (VMs) to meet the demands of the data center and consolidating them through a minimum number of servers. This minimum must also be controlled as part of the solution to ensure full compliance with the SLA established between the cloud provider and its consumers, which means that idle servers can be switched off or set to sleep mode, thus reducing energy consumption. As the demand for resources in cloud infrastructures is elastic [1], VM consolidation must be accomplished dynamically to meet current needs. VM dynamic consolidation is carried out by using live migration [18] of VMs running on overloaded or underloaded servers.

Energy-aware VM consolidation can be formulated as a computing resource allocation problem where the objective is to minimize energy consumption while delivering the quality of service (QoS) agreed with consumers. Traditionally, the common strategy consists in reducing the energy consumption of a single type of resource, *i.e.*, CPU. However, the growth of multi-core architectures and virtualization itself have forced servers to have large amounts of random-access memory (RAM)[1], leading to the situation where the power consumed by RAM has become non-negligible. In 2009, Lim et al. [19] had already observed that the energy consumption incurred by RAM utilization could correspond to up to 25% of the total consumption of a server.

In this paper, we introduce two novel strategies to improve energy-aware VM consolidation in data centers. We call our proposals *CPU and RAM Energy-aWare* (CREW) and *Underload Detection* (UD). CREW improves VM placement by employing a power model that considers the energy consumed by both CPU and RAM. UD, in turn, improves the detection of underloaded servers by taking into account the history of CPU utilization of hosts to establish a minimum threshold for turning servers off. To evaluate our proposed strategies, we used a workload based on the traces of a Google cluster [20]. Google traces allow a comprehensive assessment of our proposals because they represent one of the largest Infrastructure as a Service (IaaS) cloud providers in the market. Results show that our strategies reduce energy consumption and improve QoS assurance.

The rest of this paper is organized as follows. In Section 2, we discuss related work. In Section 3, we introduce the system model employed in the development of our proposal and present the energy model that quantifies CPU and RAM power. In Section 4, we provide a detailed description of the strategies we have developed for energy-aware VM consolidation. We provide a description and evaluation of results in Section 5. Finally, in Section 6, we summarize our achievements and discuss directions for future work.

2. Related work

There is a large body of literature on energy management of data centers that provide virtualized resources. One of the first work in this context was carried out by Nathuji and Schwan [15]. The authors have proposed an architecture where local and global managers are combined to provide an energy-aware resource allocation system. The local manager resides on each physical server and employs operating system-level policies to control power consumption. Periodically, the global manager obtains information on resource usage from the local managers and dynamically adjusts VM placement using live migration. Still, the authors do not provide any particular strategy for global consolidation.

Verma et al. [16] have approached energy-aware VM placement as a continual optimization problem: periodically, active VMs are migrated from one physical server to another, so that resources can be offloaded and placed into power saving states. The authors have addressed the problem using heuristics for the bin-packing problem with differently sized bins, where: bins represent physical servers, items are the VMs that have to be allocated, bin sizes are available CPU capacity of servers, and costs represent the power drawn by servers. However, the authors' proposed solution does not support the definition of SLAs, and the VM consolidation algorithm can indeed degrade the performance of applications.

Kusic et al. [14] have posed the energy-aware VM consolidation problem as one of sequential optimization under uncertainty, to then solve the problem using Limited Lookahead Control (LLC). In their model, SLA is defined as request processing rates; customers pay for the negotiated infrastructure and are refunded in case of SLA violation. The goal is to maximize the provider's profits by minimizing power consumption and reducing SLA violations. Still, the proposed model requires a training phase to adjust to the application-specific features. This factor limits the generality of the authors' approach.

Beloglazov and Buyya [12] have divided the problem of energy-aware VM consolidation into four subproblems: server overload detection, server underload detection, VM selection, and VM placement. To solve the first subproblem, the authors have proposed different adaptive utilization

[1] In this work, we will use RAM as a synonym for primary memory of servers, independently of the technology used for its manufacture.

thresholds, which are based on statistical analysis of historical data from CPU utilization. To identify underloaded servers, the system finds hosts with the minimum CPU utilization. The VM selection problem is tackled using live migration of VMs that require the minimum time to complete a migration while the VM placement problem is solved using heuristics for the bin-packing problem. The target system consists of heterogeneous physical servers, and SLA is defined by a workload-independent QoS metric. More recently, Cao and Dong [21] have improved over Beloglazov and Buyya framework by distinguishing overloaded hosts without SLA violation from those with SLA violation.

The common limitation of the aforementioned work [12,14–16,21] is that the solutions for VM consolidation only focus on CPU as the source of power consumption. However, as mentioned before, RAM accounts for up to 25% of the total energy consumption of a server. In contrast with the previous work, we seek to improve VM consolidation solutions by adding the RAM dimension to the problem.

Other research efforts have indeed addressed the problem of energy-aware VM consolidation by considering more than the CPU dimension. For example, Feller *et al.* [13] have proposed an energy-aware VM management framework that considers CPU, memory, and network as sources of power consumption. For each resource, the authors apply static utilization thresholds to determine whether each resource is overloaded or underloaded, whereas the VM placement problem is addressed by employing a variant of the multidimensional bin-packing problem. However, static utilization thresholds are not suitable for IaaS environments with dynamic and unpredictable workloads [12]. Another limitation of the work of Feller et al. [13] is the absence of an energy model for RAM, since the RAM consumption was directly measured from the servers of the testbed during the evaluation.

Zhang et al. [17] have proposed a heterogeneity-aware, multi-dimensional resource management system for cloud computing environments where the energy-aware VM consolidation problem is divided into three main parts. First, the system classifies the workload into task classes according to features such as priority, CPU and memory size. The classification is implemented using the k-means algorithm. Second, the future arrival rate of each task class is predicted using the ARIMA model. Finally, the system uses this information to estimate the minimum number of VMs required to support the workload for the next control period, as well as the minimum number of physical servers to host the VMs. The problem formulation also considers a multi-dimensional (CPU, memory, disk) power model where the total energy consumption of a physical server is estimated as a linear function of resource utilization. However, this model is unrealistic for RAM because this resource often draws power in a way that is disproportional to its load [22].

In this paper, we propose a novel approach to the problem of energy-aware VM consolidation. In a similar way to Beloglazov and Buyya [12], we divide the problem into four subproblems: (1) server overload detection, (2) server underload detection, (3) VM selection, and (4) VM placement. Different than Beloglazov and Buyya, however, we propose different solutions for the second and fourth subproblems. Another difference is that, in our proposal, VM placement is solved using a heuristic that considers the energy consumption incurred by both CPU and RAM utilization. To compute the RAM power, we build an analytical model inspired by the work of David et al. [22] on memory power management. This last work is focused on hardware and does not take VM consolidation into account. On the other hand, in contrast to Beloglazov and Buyya's work, where the solution for determining underloaded servers relies on estimating the instantaneous values of CPU utilization, we propose a strategy based on the historical data for solving the same problem.

3. System model

We focus on Infrastructure as a Service (IaaS) environments composed of large data centers with hundreds of physical servers. Servers are heterogeneous with different CPU speeds, memory sizes, and power consumption demands. In addition, servers only employ local disks for loading the operating system, while the storage of VMs and data is kept on a Network Attached Storage (NAS). The cloud computing system can run several thousands of VMs simultaneously. VMs may present heterogeneous QoS profiles, *i.e.*, each VM may generate a different CPU load, memory usage, and network transfer. Several types of consumers and applications can use the cloud computing system and no previous knowledge about workloads is needed. Consumers and the IaaS provider agree on QoS requirements by means of SLAs [23]. For example, an SLA can list as requirements the minimum network throughput and the maximum response time for an application. If the requirements are not fully met, there is an SLA violation. Such violation implies on a reward from the IaaS provider to the consumer. Although we do not describe how this reward is paid, we measure SLA violation so that the potential loss of the IaaS provider can be quantified.

As in Nathuji and Schwan's work [15], we consider a system comprised of two software layers: a global manager and multiple local managers. Each physical server runs a local manager that is responsible for monitoring CPU and RAM utilization. A local manager also decides when and which VMs must be migrated. In addition, the local manager is also responsible for changing the server power mode: either turning off or placing the server in sleep mode. Periodically, the global manager collects information from local managers to decide whether VM placement needs to be adapted. That adaptation is performed to minimize both energy consumption and SLA violations. Adaptation occurs at run-time and comprises the live migration of VMs along the available physical servers. In the following sections, we describe the energy model employed in this work for both CPU and RAM. We also present the metrics that quantify migration costs and SLA violations.

3.1. Energy model

We assume that all energy consumed by a physical server (E_{All}) is drained by CPU (E_{CPU}) and RAM (E_{RAM}), as shown in Eq. (1). In addition, we also assume that the energy consumed by the other two main hardware components, *i.e.*, disk and network interface, can be neglected. That is so because disk is hardly ever used, since only the operating system is stored locally, and a 10 Gbps network interface card (NIC)

consumes nearly 10% of a typical server, while a 1 Gbps NIC consumes less than 7% [24].

$$E_{All} = E_{CPU} + E_{RAM}. \quad (1)$$

We estimate CPU power consumption by employing real data for server power consumption from the SPECpower benchmark [25]. Table 1 summarizes the system power consumption as a function of the CPU utilization for the two servers used in this work: HP ProLiant ML110 G4 (Intel Xeon 3040, dual-core, 1860 MHz) and HP ProLiant ML110 G5 (Intel Xeon 3075, dual-core, 2660 MHz). In this benchmark, most of the power is drained by the CPU because, as part of the test methodology, both hard disk and display are turned off after one minute, network communications are very low, and memory pages of the stress application are locked in the physical RAM.

We estimate the RAM power consumption of our servers by employing an analytical model derived from the work of David et al. [22], in which two components are responsible for the RAM power consumption:

1. Background power, which does not depend on either the type or the number of commands run by the system, and
2. Operational power, which comes from read or write commands involving RAM.

Background power depends only on memory states and on the frequency of the operation. Although in [22] eight memory states are described, we employ only two states: Active Powerdown and Active Standby. These states have been selected because they represent a satisfactory tradeoff between energy consumption and latency. Active Standby is the state with the highest power consumption and no latency time. Active Powerdown state consumes nearly 39% less than the Active Standby, but at the cost of moderate latency. We assume that the RAM stays in Active Standby while CPU is in use, and changes to Active Powerdown when CPU is idle. The background energy consumption is defined by:

$$E_{Bg} = CPU_{\%} \times E_{Act\_Sb} + (1 - CPU_{\%}) \times E_{Act\_Pd}, \quad (2)$$

where $CPU_{\%}$, varying from 0 to 1, is the CPU utilization during a time window. Conversely, $(1 - CPU_{\%})$ corresponds to the CPU idleness during the same time window. Thus, Eq. (2) represents the minimum energy consumed by the RAM in a time window as a function of the CPU utilization. The background power for every 4GB DDR3 of RAM, at 1333 MHz, in Active Standby is $E_{Act\_Sb} = 5.36$ W, and in Active Powerdown is $E_{Act\_Pd} = 3.28$ W [22].

Operational power is the product of memory bandwidth and the power required to run a particular command, read or write, per unit of bandwidth. There is only a small difference between the power consumed by a read command and a write one. In our equations, we thus do not differentiate read and write commands because we assume that they consume the same amount of energy. Moreover, we assume that there are read/write commands in RAM during a random amount of CPU time. The operational energy consumption is defined by:

$$E_{Oper} = RAM_B \times \frac{(E_{B,r} + E_{B,w})}{2} \times CPU_{\%} \times U(0, 1), \quad (3)$$

where $RAM_B = 10.7$ GB/s is the peak transfer rate of DDR3 RAM at 1333 MHz. The operational power of the read command is $E_{B,r} = 0.939$ W/(GB/s), and of the write command is $E_{B,w} = 1.023$ W/(GB/s) [22]. $U(0, 1)$ represents the choice of a random value from a uniform distribution. This value describes the amount of CPU time that is spent on read/write commands involving RAM. We have employed this approach because there is no publicly available cloud application profile describing that. The uniform distribution is suitable because it tries different values without bias. This is illustrated in Figs. 1a, b, and c, which show the Monte Carlo simulations of Eq. (3) using the following statistical distributions: uniform, standard normal, and exponential, respectively.

In summary, Eq. (3) represents the energy consumed by the RAM in a time window as function of the percentage of the CPU time used on memory operations. Finally, we obtain the total energy consumed by the RAM:

$$E_{RAM} = E_{Bg} + E_{Oper}. \quad (4)$$

3.2. Metrics

In this section, we present the metrics employed to measure VM migration cost and SLA violation. Because we want to compare our heuristics with other heuristics from the relevant literature, we adopted the same metrics from Beloglazov and Buyya [12]. That allows us to offer a fairer comparison.

3.2.1. Migration cost

Live migration of a VM is the process of transferring the memory pages of that VM from one physical server to another at run-time [18]. In this process, the hypervisor first makes a copy of the memory pages and then transfers it in several rounds. While the most used memory pages are being copied, the VM execution is interrupted, which causes a short downtime. As soon as all memory pages have been transferred to the destination server, the VM can run again. Live migration also encompasses the transfer of a server execution state. However, because the amount of energy consumed in state transferring can be ignored, we do not consider it in our study. Let j be a VM that employs the amount of memory M_j and let C be the effective network link capacity between the source and the destination servers. The time needed for migrating the VM j through this network link is defined by:

$$T_{M_j} = \frac{M_j}{C}. \quad (5)$$

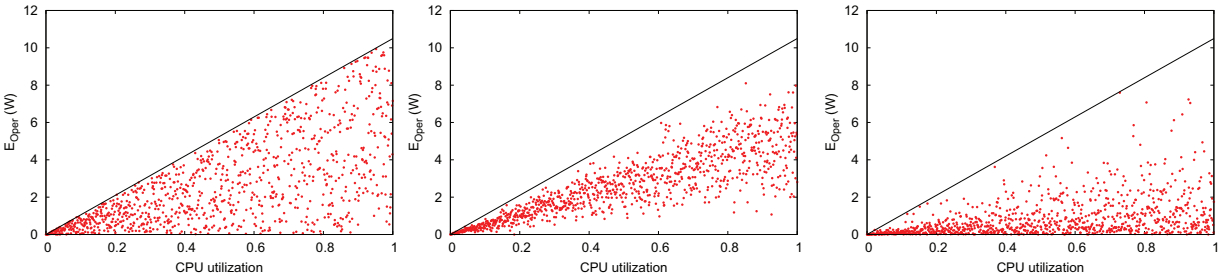
Because of the downtime, the migration has a negative effect on the performance of the applications running inside the VM. Beloglazov and Buyya [12] estimate the performance degradation of a VM j due to migrations, denoted by C_{d_j}, as being nearly 10% of the CPU utilization during all migrations of this VM. We will use the same approach in the next sections of this work.

3.2.2. SLA violation

We employ three metrics to quantify the service level provided for VMs and their applications [12]. The first metric is the SLA violation Time per Active Host (SLATAH), which describes the percentage of time in which servers stay at 100% of their CPU utilization. When a server reaches its maximum processing capacity, the performance of the applications running inside the VMs starts to be limited by the server state.

Table 1
Power consumption, in watts, as a function of the percentage of CPU utilization.

Server	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
HP G4	86	89.4	92.6	96	99.5	102	106	108	112	114	117
HP G5	93.7	97	101	105	110	116	121	125	129	133	135



(a) Uniform. (b) Standard normal. (c) Exponential.

Fig. 1. Monte Carlo simulations using different statistical distributions.

Thus, it is reasonable to assume that the VMs are not provisioned with the required service level. Formally, let N be the number of servers, let T_{s_i} be the total time in which server i has experienced CPU utilization of 100% and let T_{a_i} be the total time that server i has stayed active. The SLATAH metric is defined in Eq. (6).

The second metric is the Performance Degradation due to Migrations (PDM), which estimates the total performance degradation caused by the migrations of VMs. Let M be the number of VMs, let C_{d_j} be the estimate of the performance degradation experienced by VM j due to migrations (see Section 3.2.1) and let C_{r_j} be the CPU capacity required by VM j during its whole lifetime. The PDM metric is defined in Eq. (7).

$$SLATAH = \frac{1}{N}\sum_{i=1}^{N}\frac{T_{s_i}}{T_{a_i}} \tag{6}$$

$$PDM = \frac{1}{M}\sum_{j=1}^{M}\frac{C_{d_j}}{C_{r_j}} \tag{7}$$

The third metric, SLA Violation (SLAV), is the combination of (6) and (7):

$$SLAV = SLATAH \times PDM. \tag{8}$$

4. A joint CPU-RAM approach for dynamic VM consolidation

The energy-aware VM consolidation problem can be divided into four subproblems:

1. Detecting when a server is overloaded so that some VMs can be migrated from it to other servers;
2. Detecting when a server is underloaded so that its VMs can be migrated, and the server can be switched to a low-power mode;
3. Selecting VMs to be migrated from an overloaded server; and
4. Deciding the new placement of VMs that are selected for migration from overloaded and underloaded servers.

In this section, we introduce novel solutions to address subproblems 2 and 4. We provide additional information on each of the subproblems, paying particular attention to those where we present our contributions.

4.1. Server overload detection

In general, an overloaded server is identified by setting fixed or adjustable utilization thresholds. If the CPU utilization of a server exceeds the threshold, the server is assumed to be overloaded.

Beloglazov and Buyya [12] have proposed and evaluated four algorithms to solve the problem of server overload detection: Median Absolute Deviation (MAD), Interquartile Range (IQR), Local Regression (LR), and Local Regression Robust (LRR). The Local Regression (LR) algorithm provides the best results in general. In our experiments, we observed that the Interquartile Range (IQR) algorithm also achieves promising results. Given this, in Section 5 we evaluate the LR and IQR algorithms.

LR and IQR use statistical analysis to adapt dynamically the current CPU utilization threshold. LR consists on creating simple models for data subsets to generate a curve that closely represents the real data. IQR is a statistical dispersion measurement that is acquired from the difference between the first and third quartile.

4.2. Server underload detection

Typically, underloaded servers are detected using a simple approach of choosing the server that currently has the least CPU utilization and attempting to migrate its VMs without overloading other servers. However, as we will show in Section 5, instantaneous values of CPU utilization do not describe a stable state of the server and thus leads to inappropriate choices and performance degradation. To

contribute to this scenario, we introduce a novel algorithm, denoted Underload Detection (*UD*), where the procedure for determining underloaded servers is performed in two steps. First, a list containing the servers that are neither turned off nor overloaded is created. The CPU utilization of each server in this list is used to compute an average CPU usage in the data center. This CPU usage is calculated as an Exponential Weighted Moving Average (EWMA), as follows:

$$\overline{Y}_t = \alpha \times \frac{1}{N} \sum_{i=1}^{N} cpu\_usage_i + (1-\alpha) \times \overline{Y}_{t-1}, \quad (9)$$

where α is a constant smoothing factor between 0 and 1, cpu_usage_i is the CPU utilization of the *ith* server in the list, N is the number of servers in the list, $\overline{Y}_t$ is the EWMA computed at time t, and $\overline{Y}_{t-1}$ is the EWMA at time $t-1$. The server with the least CPU utilization is selected as input for the next step.

The second step of our algorithm consists of checking whether the CPU utilization of the candidate server is below a percentage ϕ of the CPU usage in the data center ($\overline{Y}_t$). If that is the case, the algorithm checks if it is possible to migrate the VMs of the server to other hosts without overloading them. If this migration is possible, it is performed, and the server is turned off or switched to a low-power mode. Otherwise, the server stays turned on. UD is presented in Algorithm 1.

Algorithm 1 Underload Detection Algorithm.

Input: *hostList* - list of servers that are neither turned off nor overloaded; $\overline{Y}_{t-1}$ - current CPU usage in the data center.
Output: *underloadServer* - underloaded server; $\overline{Y}_t$ - new CPU usage in the data center.

```
1: minimumUtilization ← MAX;
2: underloadServer ← NULL;
3: avgUtilization ← 0;
4: for each server in hostList do
5:     utilization ← cpuUtilization(server);
6:     avgUtilization ← avgUtilization + utilization;
7:     if utilization < minimumUtilization then
8:         minimumUtilization ← utilization
9:         underloadServer ← server
10: avgUtilization ← avgUtilization/ hostList.size();
11: if it is the first run then
12:     Ȳ_t ← avgUtilization;
13: else
14:     Ȳ_t ← α × avgUtilization + (1 − α) × Ȳ_{t−1};
15: if minimumUtilization > Ȳ_t × φ then
16:     underloadServer ← NULL;
17: return underloadServer, Ȳ_t
```

We provide a performance evaluation for UD with different values of ϕ in Section 5.4.

4.3. VM selection

After identifying a server as overloaded, it is necessary to select one or more VMs for migration. The algorithm for VM selection runs iteratively and performs the following steps. First, a particular VM is selected according to some policy, and the server is checked again. If the server is still overloaded, another round takes place to select an additional VM to migrate from the server. These steps are repeated until the server is not overloaded anymore.

Beloglazov and Buyya [12] have evaluated three VM selection policies. The Minimum Migration Time (MMT) policy provides the best results. MMT selects the VM that requires the minimum time to complete a migration. Under normal conditions, the migration time is a function only of the size of the VM in terms of RAM, *i.e.*, the smaller the amount of RAM used by the VM, the lower the time required to migrate it. In this work, we also use the MMT policy.

4.4. VM placement

As described earlier, some related work [12,13,16] has formulated VM placement as a bin-packing problem with variable bin sizes and prices. We have used the same approach, but in our study bins are physical servers; items represent VMs that have to be allocated; bin sizes are the combination of CPU and RAM available on servers; and prices correspond to the power cost incurred by CPU and RAM when VMs are allocated to servers.

To determine the new placement of VMs, we modified the Best Fit Decreasing (*BFD*) algorithm proposed by Beloglazov and Buyya [12]. In its original version, the algorithm sorts all VMs in decreasing order of their current CPU utilization and allocates each VM to a server that provides the least increase of power consumption caused by this allocation.

In our modification of BFD, which is denoted as CPU and RAM Energy Aware (*CREW*), there is a change in the estimation of power consumption caused by the allocation, so that it also accounts for the power consumption incurred by using RAM. This consumption is quantified by Eq. (4), as outlined in Section 3.1. CREW is presented in Algorithm 2.

Algorithm 2 CPU and RAM Energy Aware VM Placement Algorithm.

Input: *hostList* - List of all the servers
vmList - List of VMs to be allocated
Output: *allocation* - Allocation of VMs

```
1: vmList.sortInDecreasingOrderOfCpuUtilization()
2: for each vm in vmList do
3:     minPower ← MAX
4:     allocatedHost ← NULL
5:     for each host in hostList do
6:         if host has both CPU and RAM enough for vm then
7:             power ← estimatePower(host,vm) {Eq. (1)}
8:             if power < minPower then
9:                 allocatedHost ← host
10:                minPower ← power
11:    if allocatedHost ≠ NULL then
12:        allocation.add(vm, allocatedHost)
13: return allocation
```

Table 2
Sizes of VMs.

Instance	Description
Micro	613 MB of memory, 0.5 EC2 Compute Unit[a]
Small	1.74 GB of memory, 1 EC2 Compute Units
Extra large	3.48 GB of memory, 2 EC2 Compute Units
High-CPU medium	870 MB of memory, 2.5 EC2 Compute Units

[a] 1 EC2 = 1000 MIPS

4.5. Considerations about RAM in the VM consolidation problem

As we previously described, our proposal addresses two subproblems of the energy-aware VM consolidation problem: the detection of underloaded servers (subproblem 2) and the VM placement (subproblem 4). Actually, we also investigated the use of the information about RAM in the other two subproblems, but they did not present benefits as we will explain in the following.

The detection of an overloaded server (subproblem 1) involves only CPU because the concept of overload does not apply properly for RAM in data centers. Typical data center servers do not perform swapping of the VMs. Thus, after loading a VM in the RAM, a server keeps this VM until the VM be migrated or terminated. The solutions for subproblems 2 and 4 ensure that when a VM is migrated, the server hosting the VM has enough RAM to accommodate it. Once the VM is loaded in the RAM, it does not change its size. Additionally, eventual swapping inside the VMs are managed by a Network Attached Storage (NAS), which keeps the virtual disks.

The selection of VMs for migration (subproblem 3) follows a similar reasoning of the subproblem 1. While different policies can be applied for migration of the VMs in an overloaded server, the Minimum Migration Time (MMT) presents the best results because this policy provides the fastest reaction to the problem and presents the lower degradation due to migration. Additionally, the smallest VMs are the easiest one to fit.

5. Performance evaluation

In this section, we evaluate the impact of taking into account the energy consumption incurred by RAM in the heuristics designed for VM consolidation. We used the CloudSim [26] simulator, release 3.0.2, to conduct our tests. Among other features, CloudSim allows the modeling and simulation of large-scale data centers, virtualized servers, energy consumption by computing resources, and policies for VM allocation. Although the basic resources for our work were already available in CloudSim, we had to extend the simulator to add the RAM energy model.

We simulated a data center with 800 servers, half of which are HP ProLiant ML110 G4 servers, and half consists of HP ProLiant ML110 G5 servers. We also employed 32 GB of RAM in all servers, except in Section 5.2, where the amount of RAM varies. Table 2 lists the VM sizes used in this work. These sizes were based on instance types provided by Amazon EC2 [27]. This experimental setup has been adopted in many relevant works [12,21,28,29]. Algorithms for VM placement, including our proposal (CREW), typically ensure that VMs with high CPU utilization are allocated to the most energy efficient servers. Thus, from the point of view of energy efficiency, the server heterogeneity is not an issue for these algorithms. From the point of view of SLA violation, workload heterogeneity is more impacting than server heterogeneity, since workload composition changes over time, while processing capabilities, hardware features, processor architecture, and processor speed are rather static characteristics. Thus, we focused on assessing our proposals under two different representative workloads, as described in the following section.

Table 3
Workloads VMs per day and CPU utilization.

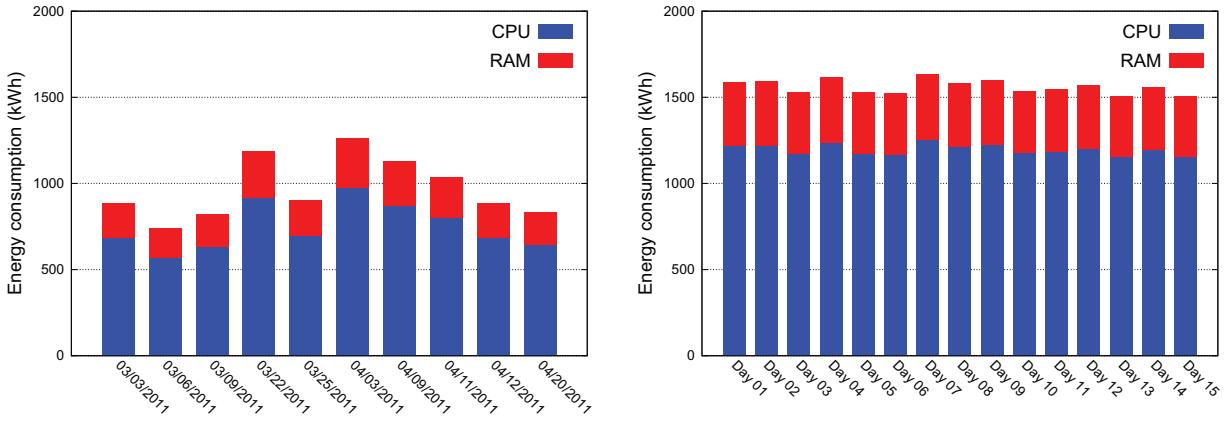
Workload	VMs per day	CPU utilization (%)
PlanetLab	1,174	11.13
Google trace – first 15 days	1,141	36.75

5.1. Workload data

We adopted a real workload provided by CloudSim. Such workload is part of the CoMon project [30] and was collected from the PlanetLab testbed [31]. The workload provides the CPU utilization of over 10,000 VMs executed in servers located in more than 500 places around the world. CPU utilization is measured every 5 min over a period of 10 days. Although this workload has been used in other works, we identified two shortcomings. First, it is from a testbed mostly used for scientific experimentation and, thus, it is reasonable to suppose that many real-world in-production applications are not represented in the dataset. Second, the workload provides no information on the amount of RAM required by each VM. To deal with this problem, we adopted a methodology where the types of VMs are randomly and proportionally generated according to Amazon EC2 instances (Table 2). As a result, all VM types have the same amount of instances.

To overcome these shortcomings and to broaden the evaluation of our proposal, we also adopted a workload based on the traces of a Google cluster composed of approximately 12,000 machines [20]. This trace was collected and made publicly available in 2011 and, since then, it has been used in several works [17,32,33]. The trace provides information about hundreds of thousands of jobs that are submitted by consumers. Each job is composed of one to tens of thousands of tasks, which are scheduled to run in the machines. Tasks are programs executed on the machines, and they are from different types: web servers, MapReduce-like systems, and high-performance computing. The whole trace describes the cluster usage over a period of 29 days, including the demand for RAM and the CPU utilization. A detailed description and analysis of the Google trace are provided in [32].

Because the Google trace provides no information about VMs, we assumed that the aggregate of all tasks running in the same machine at the same time constitutes a VM. CPU utilization of each VM is measured every 5 min over a period of one day. The real demand for RAM, in each VM, is computed based on information about tasks. Once a VM is created, such demand is used to choose one of Amazon EC2 instances from Table 2. The Google trace contains



(a) PlanetLab trace. (b) Google trace – first 15 days.

Fig. 2. Energy consumption by CPU and RAM.

these proportions of Amazon EC2 instances: Micro (0.47%), Small (31.77%), Extra large (53.46%), and High-CPU medium (14.30%). Since the number of VMs available every day is much higher than that provided in the PlanetLab trace, processing all of them would require a lot of memory and time in the simulation. Thus, only part of the trace is used, randomly choosing 10% of VMs available each day. Table 3 shows the average number of VMs per day and the average CPU utilization in the PlanetLab workload and the Google trace sample. Notice that the average CPU utilization is higher in the Google trace sample. Thus, the opportunities to reduce energy consumption or SLA violation in this trace are significantly smaller than in the PlanetLab's.

5.2. Energy consumption by RAM

In Section 4, we described how we modeled the RAM energy consumption to be used in the CloudSim simulator. Based on this model, we show now how RAM utilization impacts on energy consumption. Results show that such energy consumption is too high to be neglected.

Figs. 2a and b illustrate the total amount of energy consumption by CPU and RAM, when CPU DVFS, a benchmark policy provided by CloudSim, is employed. In these figures, energy consumption varies as a function of the day in which the measurements were made. All servers have the same amount of RAM, *i.e.*, 32GB. In this configuration, the energy consumed by RAM is nearly 23% of the total in all datasets of PlanetLab and Google. According to our RAM energy model, presented in Section 3.1, background power is much higher than operational power. Thus, the energy consumption by RAM is widely determined by such a power.

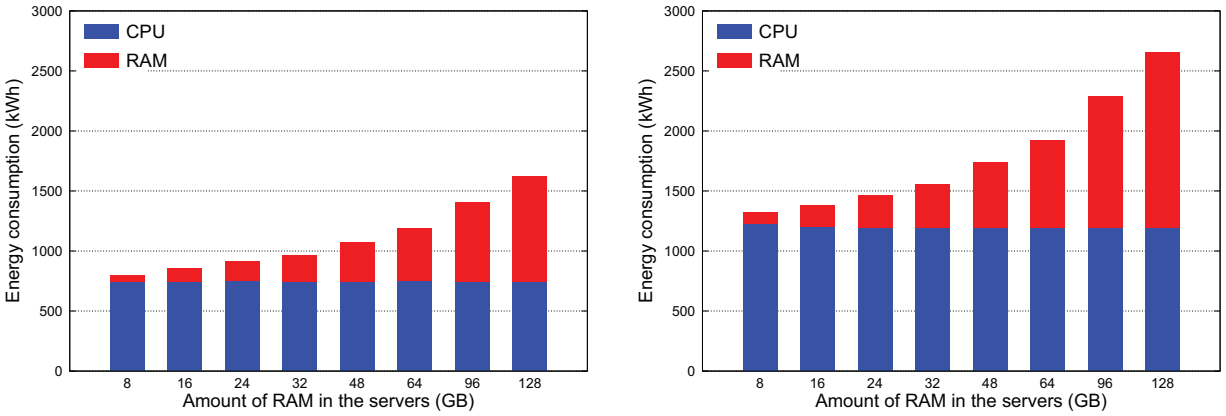
Figs. 3a and b show the average energy consumption of all datasets when the amount of RAM varies in the servers and CPU DVFS policy is applied. These results confirm our previous observation that the amount of RAM determines its consumption. Thus, as the amount of RAM increases, the energy consumption also increases. This is so because of the energy model, which reports background energy consumption even in the absence of any read or write operation. The background consumption can be lowered by techniques such as DVFS for RAM [22]. However, this technique has few benefits when the RAM is being used effectively most of the time. For example, the strategies employed in this work aim to keep the minimum number of servers turned on, while keeping the maximum number of VMs per server that do not violate the SLA. In this scenario, there is frequent RAM utilization by VMs, as well as by the migration process.

5.3. Heuristics aware of RAM power

In this section, we show the energy saving achieved by VM consolidation heuristics that are aware of RAM power. In this part of the evaluation, we used energy consumption and SLA violation as metrics for comparison, and we employed an unlimited amount of RAM in servers while performing VM consolidation. In Section 5.5, we complete the evaluation by showing the results when there is a limited amount of RAM.

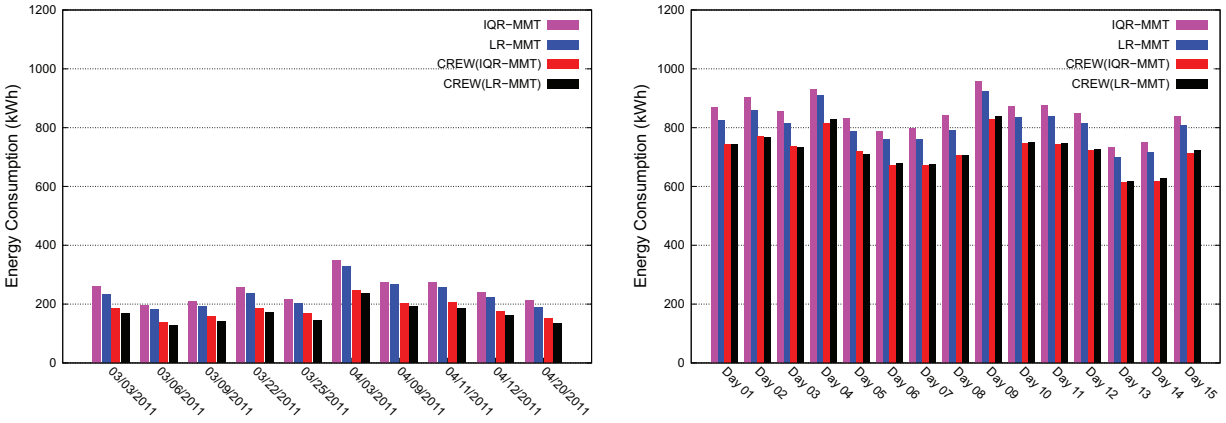
The following heuristics are evaluated. IQR-MMT and LR-MMT are heuristics for VM consolidation that are not aware of RAM power. IQR-MMT uses the IQR algorithm for overloaded server detection combined with the MMT VM selection policy. Similarly, LR-MMT uses the LR algorithm for overloaded server detection combined with the MMT VM selection policy. CREW(IQR-MMT) and CREW(LR-MMT) are improved versions of these heuristics. They take RAM power into consideration while performing VM consolidation. All heuristics evaluated in this section use instantaneous values of CPU utilization for detecting underloaded servers.

Figs. 4a and b show the median values of energy consumption for the evaluated heuristics. In general, there is noticeable energy saving when CREW is applied to the two heuristics, IQR-MMT and LR-MMT, in all days of both workloads. Energy saving varies from near 8.8% to around 30.8%, depending on the heuristic and day. CREW(IQR-MMT) and CREW(LR-MMT) show a similar average pattern of energy saving, with a small advantage for CREW(LR-MMT). This advantage is an inheritance of the best performance of LR-MMT. The energy consumed by the Google workload is higher than PlanetLab's because its average CPU utilization is much higher.



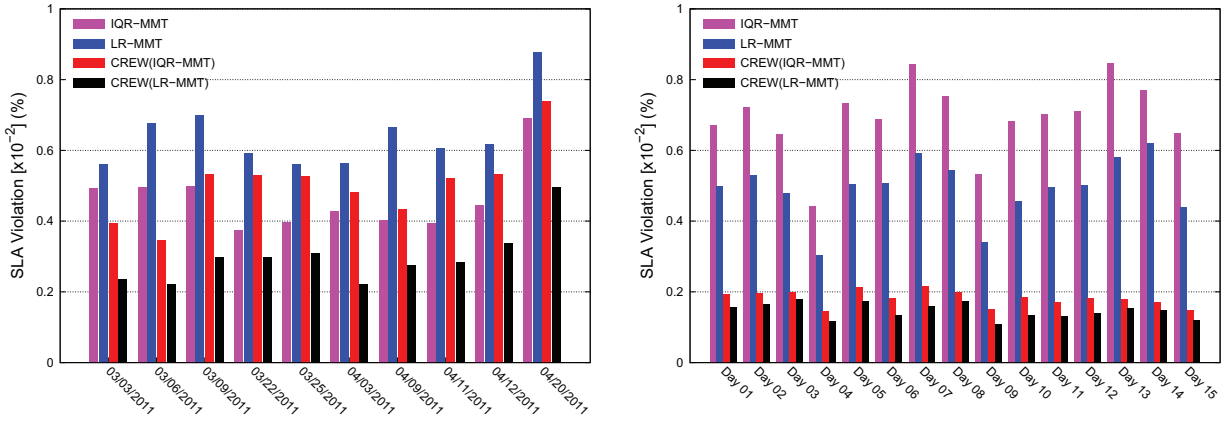
(a) PlanetLab trace. (b) Google trace – first 15 days.

Fig. 3. Average energy consumption by CPU and RAM.



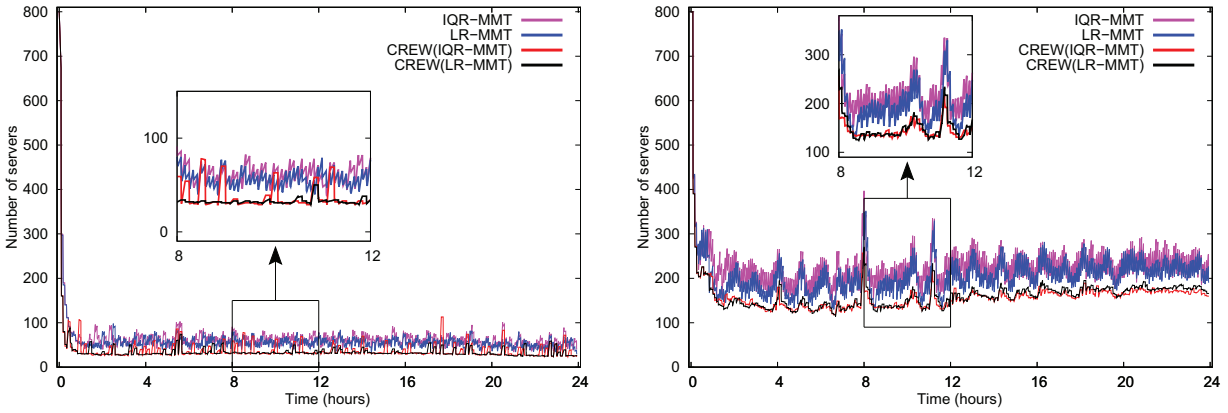
(a) PlanetLab trace. (b) Google trace.

Fig. 4. Median values of the energy consumption.



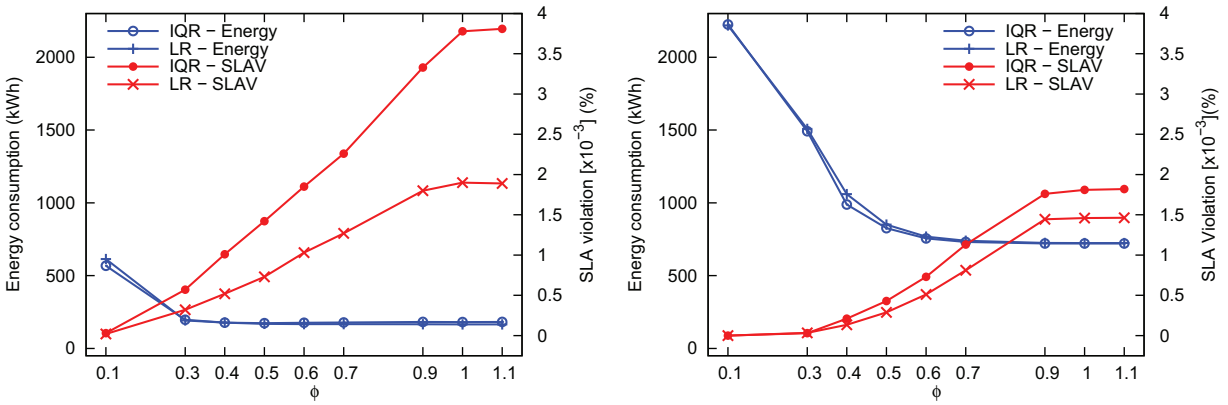
(a) PlanetLab trace. (b) Google trace.

Fig. 5. Median values of SLA violation as a function of the evaluated day.



(a) PlanetLab trace. (b) Google trace.

Fig. 6. Number of active servers over time.



(a) PlanetLab trace. (b) Google trace – first 15 days.

Fig. 7. Energy consumption, left Y, and SLA violation, right Y, as a function of ϕ.

Figs. 5a and b show the median values of SLA violation for the evaluated heuristics. In the PlanetLab workload, the highest average energy saving of LR-MMT implies on the highest average SLA violation. However, in the Google workload, IQR-MMT shows the worst average performance in both metrics. These differences in the performance of the heuristics can be attributed to their design and workload profile. IQR-MMT is more conservative than LR-MMT, as it saves less energy but keeps more resources to absorb the CPU usage surges, which are more frequent in the PlanetLab workload. CREW can improve the performance of LR-MMT on both workloads, but the CPU usage surges of the PlanetLab workload degraded the SLA violation of CREW(IQR-MMT). In general, CREW(LR-MMT) shows the best trade-off between energy consumption and SLA violation.

Beloglazov and Buyya [12] also proposed the metric for overall SLA violation, which is used to calculate the percentage of time in which the data center shows any level of SLA violation during the whole evaluation. Table 4 shows the average overall SLA violation of the heuristics in both workloads, PlanetLab and Google. The performance of IQR-MMT in the PlanetLab workload suggests that there is an opportunity to improve CREW. This result was our main reason for improving our proposal and developing UD.

Table 4
Average values of overall SLA violation.

Heuristic	PlanetLab	Google
IQR-MMT	0.09%	0.14%
LR-MMT	0.13%	0.12%
CREW(IQR-MMT)	0.14%	0.07%
CREW(LR-MMT)	0.11%	0.06%

The performance penalty for CREW in the metrics related to SLA violation is due to its design. CREW keeps the number of active servers as low as possible to save energy. This approach must be balanced with the one for detecting underloaded servers. However, the original strategy for that employs the instantaneous value of CPU utilization. Thus, critical servers for CREW can be wrongly turned off as a result of short-term fluctuations in CPU utilization. Figs. 6a and b show the number of active servers during a sample day. Since the number of active servers kept by CREW tends

Table 5
Summary of metrics in PlanetLab and Google workloads.

		ESV [x10^{-3}]	Energy (kWh)	SLAV [x10^{-3}] (%)	Overall SLA (%)	Migrations
PlanetLab	NPA	0.000	2410.800	0.000	0.00	0
	CPU DVFS	0.000	892.030	0.000	0.00	0
	IQR-MMT	25.970	216.865	11.975	0.19	38359
	LR-MMT	27.522	249.975	11.010	0.15	37199
	CREW(IQR-MMT)	6.780	181.515	3.735	0.11	27862
	CREW(LR-MMT)	3.013	164.660	1.830	0.08	15110
	CREW-UD[0.4](IQR-MMT)	1.719	177.210	0.970	0.05	15099
	CREW-UD[0.4](LR-MMT)	0.806	179.150	0.450	0.04	8558
	CREW-UD[0.3](IQR-MMT)	1.124	198.975	0.565	0.04	12444
	CREW-UD[0.3](LR-MMT)	0.536	191.600	0.280	0.03	7665
Google	NPA	0.000	2410.800	0.000	0.00	0
	CPU DVFS	0.000	1556.860	0.000	0.00	0
	IQR-MMT	59.886	849.440	7.050	0.14	61237
	LR-MMT	40.729	814.580	5.000	0.12	47635
	CREW(IQR-MMT)	13.191	720.810	1.830	0.07	28224
	CREW(LR-MMT)	10.408	727.830	1.430	0.06	24224
	CREW-UD[0.6](IQR-MMT)	5.493	762.940	0.720	0.05	18210
	CREW-UD[0.6](LR-MMT)	3.885	777.030	0.500	0.04	15106
	CREW-UD[0.5](IQR-MMT)	3.322	830.450	0.400	0.04	14754
	CREW-UD[0.5](LR-MMT)	2.259	836.710	0.270	0.03	12179

to be a critical minimum, any decision about turning off must be taken carefully.

5.4. Parameterization of UD

The UD strategy consists of checking if CPU utilization of the candidate server is below a percentage ϕ of the average CPU usage in the data center. To assert which would be a satisfactory choice for this percentage, we have conducted tests employing a range of values from 0.1 to 1.1. Figs. 7a and b show the energy consumption and SLA violation as a function of ϕ. Values of ϕ below 0.3 imply on very low SLA violation, but also lead to high energy consumption; values of ϕ above 0.7 achieve the smallest levels of energy consumption, but also generate a large increase in SLA violation. Since CPU usage surges are more frequent in the PlanetLab workload, a more conservative approach to ϕ, near to 0.3, is the proper choice for this workload. The Google workload has a better trade-off when ϕ is close to 0.6.

5.5. Analysis of the proposed solutions

Finally, we carried out a comparison between several solutions: Non Power Aware (NPA), CPU DVFS, IQR-MMT, LR-MMT, CREW, and CREW-UD. NPA works as a baseline to quantify the effective energy saving of the other solutions. CPU DVFS is a traditional solution that is widely available in the present machines, but it does not involve VM consolidation. QR-MMT, LR-MMT, and CREW were defined in Section 5.3. CREW-UD takes RAM power into account while performing VM consolidation and uses the average CPU utilization of the data center to detect underloaded servers. In this part of the evaluation, all solutions work with a limited amount of RAM of 32 GB. This approach differs from the one in Section 5.3.

Table 5 shows the median of the metrics employed to evaluate the solutions. We chose the median because it was the same statistic employed in [12]. However, we also carried out the evaluation with the average and obtained very similar results. The new metric ESV is the product of Energy and SLA violation (SLAV). This metric represents the combination of conflicting objectives: energy saving and SLA compliance.

NPA and CPU DVFS keep SLAV and overall SLA violation at zero, but also have the highest values for energy consumption. NPA and CPU DVFS do not perform VM migrations and; therefore, miss the opportunity to save energy by turning off underloaded servers. IQR-MMT and LR-MMT heuristics, which perform VM migrations, achieve energy saving that is notably higher than NPA and CPU DVFS. As a result, these heuristics have to deal with SLA violations that arise from the reduction of the number of available servers. It is worth noting that the SLAV and overall SLA metrics do not represent specific applications or their performance parameters. Thus, these metrics may underestimate the effective impact on the performance perceived by consumers. Therefore, it is important to keep the values of these metrics as low as possible, while respecting the energy saving target.

CREW improves IQR-MMT and LR-MMT in both metrics, energy and SLA violation. This improvement results from two primary activities: (1) reducing the number of active servers along the time, as shown in Section 5.3; and (2) reducing the number of migrations (see Table 5). CREW-UD offers a tuning knob that allows SLA violation to be reduced even further, but this reduction naturally increases energy consumption. This resource is useful because there is room to decrease SLA violation while keeping energy consumption below that of the solutions that are unaware of RAM power. For instance, a data center administrator can use this resource to establish his/her intended trade-off in association with a pricing policy.

6. Conclusions and future work

In this work, we have highlighted the importance of taking into account the energy consumed by RAM. Our tests showed that the energy consumption increases as a

function of the amount of RAM. According to our proposed model, 32GB already has a non-negligible impact, while 128GB can represent more than 50% of the whole energy consumption. Based on these observations, we developed strategies to improve the state-of-the-art heuristics for VM consolidation. Our proposals, CREW and UD, can reduce energy consumption and also decrease SLA violation. While CREW offers the best trade-off between energy saving and SLA violation, CREW-UD allows the trade-off to be adjusted to the data center needs.

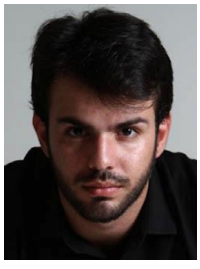
As future work, we will investigate different approaches to solving the problem of detecting overloaded servers. The results of this work suggest that alternative solutions can be implemented, which can enable us to save even more energy or be more computationally efficient. Additionally, we are also interested in investigating the impact of VM consolidation on the rest of data center hardware, particularly on the network and storage devices.

Acknowledgments

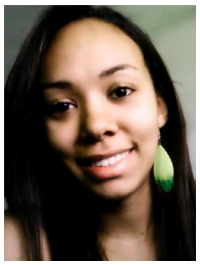
This work has been partially supported by the Goiás Research Foundation (FAPEG) under the grant 005/2012, and the Brazilian Research Agency (CNPq) under the grant 204993/2014-8.

References

[1] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, I. Brandic, Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility, Future Gener. Comput. Syst. 25 (6) (2009) 599–616.
[2] T. Mukherjee, A. Banerjee, G. Varsamopoulos, S.K. Gupta, S. Rungta, Spatio-temporal thermal-aware job scheduling to minimize energy consumption in virtualized heterogeneous data centers, Comput. Netw. 53 (17) (2009) 2888–2904.
[3] Greenpeace, Clicking clean: how companies are creating the green Internet, http://www.greenpeace.org/usa/global/usa/planet3/pdfs/clickingclean.pdf, 2014 (last access: feb-10-2015).
[4] X. Fan, W.D. Weber, L.A. Barroso, Power provisioning for a warehouse-sized computer, in: Proceedings of the Thirty-fourth Annual International Symposium on Computer Architecture, 2007, pp. 13–23.
[5] L. Barroso, U. Holzle, Energy-proportional computing, Computer 40 (12) (2007) 33–37.
[6] L. Benini, A. Bogliolo, G. De Micheli, A survey of design techniques for system-level dynamic power management, IEEE Trans. Very Larg. Scale Integr. (VLSI) Syst. 8 (3) (2000) 299–316.
[7] C.H. Hwang, A.C.H. Wu, A predictive system shutdown method for energy saving of event-driven computation, ACM Trans. Des. Autom. Electron. Syst 5 (2) (2000) 226–241.
[8] L.L. Andrew, M. Lin, A. Wierman, Optimality, fairness, and robustness in speed scaling designs, in: Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, 2010, pp. 37–48.
[9] M. Weiser, B. Welch, A. Demers, S. Shenker, Scheduling for reduced CPU energy, Mob. Comput. 353 (1996) 449–471.
[10] A. Wierman, L.L.H. Andrew, A. Tang, Power-aware speed scaling in processor sharing systems: Optimality and robustness, Perform. Eval. 69 (12) (2012) 601–622.
[11] H. Krallmann, C. Schroepfer, V. Stantchev, P. Offermann, Enabling Autonomous Self-Optimisation in Service-Oriented Systems, Springer, Netherlands, 2008, pp. 127–134.
[12] A. Beloglazov, R. Buyya, Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers, Concurr. Comput. Pract. Exp. 24 (13) (2012) 1397–1420.
[13] E. Feller, C. Rohr, D. Margery, C. Morin, Energy management in IaaS clouds: A holistic approach, in: Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing, 2012, pp. 204–212.
[14] D. Kusic, J.O. Kephart, J.E. Hanson, N. Kandasamy, G. Jiang, Power and performance management of virtualized computing environments via lookahead control, Clust. Comput. 12 (1) (2009) 1–15.
[15] R. Nathuji, K. Schwan, VirtualPower: Coordinated power management in virtualized enterprise systems, in: Proceedings of Twenty-first ACM SIGOPS Symposium on Operating Systems Principles, 2007, pp. 265–278.
[16] A. Verma, P. Ahuja, A. Neogi, pMapper: Power and migration cost aware application placement in virtualized systems, in: Proceedings of the Ninth ACM/IFIP/USENIX International Conference on Middleware, 2008, pp. 243–264.
[17] Q. Zhang, M.F. Zhani, R. Boutaba, J.L. Hellerstein, HARMONY: Dynamic heterogeneity-aware resource provisioning in the cloud, in: Proceedings of the Thirty-third International Conference on Distributed Computing Systems (ICDCS), 2013, pp. 510–519.
[18] C. Clark, K. Fraser, S. Hand, J.G. Hansen, E. Jul, C. Limpach, I. Pratt, A. Warfield, Live migration of virtual machines, in: Proceedings of the Second Conference on Symposium on Networked Systems Design & Implementation, 2, 2005, pp. 273–286.
[19] K. Lim, J. Chang, T. Mudge, P. Ranganathan, S.K. Reinhardt, T.F. Wenisch, Disaggregated memory for expansion and sharing in blade servers, SIGARCH Comput. Archit. News 37 (3) (2009) 267–278.
[20] Google, Google cluster-usage traces, https://code.google.com/p/googleclusterdata/wiki/TraceVersion2, 2015 (last access: feb-12-2015).
[21] Z. Cao, S. Dong, An energy-aware heuristic framework for virtual machine consolidation in cloud computing, J. Supercomput. 69 (1) (2014) 429–451.
[22] H. David, C. Fallin, E. Gorbatov, U.R. Hanebutte, O. Mutlu, Memory power management via dynamic voltage/frequency scaling, in: Proceedings of the Eighth ACM International Conference on Autonomic Computing, 2011, pp. 31–40.
[23] V. Stantchev, C. Schropfer, Service-level enforcement in web-services-based systems, Int. J. Web Grid Serv. 5 (2) (2009) 130–154.
[24] R. Sohan, A. Rice, A.W. Moore, K. Mansley, Characterizing 10 Gbps network interface energy consumption, in: Proceedings of the Thirty-fifth IEEE Conference on Local Computer Networks (LCN), 2010, pp. 268–271.
[25] SPEC, SPECpower_ssj2008 benchmark, https://www.spec.org/power_ssj2008/, 2015 (last access: feb-13-2015).
[26] CLOUDSLab, CloudSim: A framework for modeling and simulation of cloud computing infrastructures and services, http://www.cloudbus.org/cloudsim/, 2015 (last access: feb-13-2015).
[27] Amazon, Amazon EC2 instance types, http://aws.amazon.com/ec2/instance-types/, 2015 (last access: feb-13-2015).
[28] D. Alboaneen, B. Pranggono, H. Tianfield, Energy-aware virtual machine consolidation for cloud data centers, in: Proceedings of the Seventh IEEE/ACM International Conference on Utility and Cloud Computing (UCC), 2014, pp. 1010–1015.
[29] M. Al-Ayyoub, Y. Jararweh, M. Daraghmeh, Q. Althebyan, Multi-agent based dynamic resource provisioning and monitoring for cloud computing systems infrastructure, Clust. Comput. 18 (2) (2015) 919–932.
[30] K. Park, V.S. Pai, CoMon: A mostly-scalable monitoring system for PlanetLab, SIGOPS Oper. Syst. Rev. 40 (1) (2006) 65–74.
[31] PlanetLab, An open platform for developing, deploying, and accessing planetary-scale services , http://www.planet-lab.org, 2015 (last access: feb-13-2015).
[32] C. Reiss, A. Tumanov, G.R. Ganger, R.H. Katz, M.A. Kozuch, Heterogeneity and dynamicity of clouds at scale: Google trace analysis, in: Proceedings of the Third ACM Symposium on Cloud Computing, 2012, pp. 7:1–7:13.
[33] Q. Zhang, M.F. Zhani, S. Zhang, Q. Zhu, R. Boutaba, J.L. Hellerstein, Dynamic energy-aware capacity provisioning for cloud computing environments, in: Proceedings of the Ninth International Conference on Autonomic Computing, 2012, pp. 145–154.



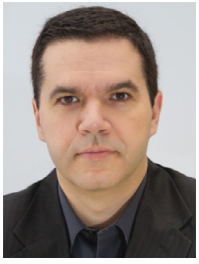
Pedro H. P. Castro is a public servant at Prosecution Service of the Union (MPU) in Minas Gerais, Brazil, where he is member of the Infrastructure and Information Technology Team (NUITI). He has completed his Bachelor's and Master's degrees, in 2011 and 2014, respectively, in Computer Science at Federal University of Goiás (UFG), Goiânia, Brazil. His research interests include Cloud Computing, Network Management and Information Security.



Vívian L. Barreto is an undergraduate student in Computer Science at Federal University of Goiás (UFG), Goiânia, Brazil, where she is member of the Computer Networks and Distributed Systems LABORAtory (LABORA). Her research interests include Distributed Systems, Energy Consumption and Cloud Computing.

Sand Luz Corrêa is an associate professor at Federal University of Goiás (UFG), Goiânia, Brazil, where she is member of the Computer Networks and Distributed Systems LABORAtory (LABORA). Prof. Sand received the Ph.D. degree in Computer Science from Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro, 2011, the M.Sc. degree in Computer Science from University of Campinas, São Paulo, 1997, and the B.Sc. in Computer Science from Federal University of Goiás, Brazil, 1994. Her research interests include Distributed Systems, High-end Computing Performance Modeling and High-performance, Power-aware Computing.

Lisandro Zambenedetti Granville is an associate professor at the Institute of Informatics of the Federal University of Rio Grande do Sul (UFRGS), Brazil. He received his M.Sc. and Ph.D. degrees, both in Computer Science, from UFRGS in 1998 and 2001, respectively. Lisandro has served as a TPC Co-Chair of IFIP/IEEE DSOM 2007, IFIP/IEEE NOMS 2010, TPC Vice-Chair of CNSM 2010, and general co-chair of CNSM 2014. He is also Chair of the IEEE Communications Society Committee on Network Operations and Management (CNOM) and co-chair of the Network Management Research Group (NMRG) of the Internet Research Task Force (IRTF). His areas of interest include management of virtualization for the Future Internet, Software-Defined Networking (SDN) and Network Functions Virtualization (NFV), and P2P-based services and applications.



Kleber Vieira Cardoso is an associate professor at Universidade Federal de Goiás (UFG), Goiânia, Brazil, where he is the Head of the Computer Networks and Distributed Systems LABORAtory (LABORA). Prof. Kleber obtained the B.Sc. in Computer Science from Universidade Federal de Goiás (UFG), Goiânia, 1997, the M.Sc. degree in Electrical Engineering from Federal University of Rio de Janeiro, Rio de Janeiro, 2002, and a Ph.D. in Electrical Engineering from Federal University of Rio de Janeiro, Rio de Janeiro, 2009. He is a member of TPC and has been on the Committee for several conferences and workshops in his fields of interest. He chaired the Graduate Program in Computer Science of UFG (2011–2013). His research interests include the Internet, High-Speed Networks, Performance Evaluation, Wireless Networks, Mobility, and Quality of Service. He is a member of the Brazilian Computer Society.