Energy Savings on a Cloud-Based Opportunistic Infrastructure

Johnatan E. Pecero[1], Cesar O. Diaz[1], Harold Castro[2], Mario Villamizar[2], Germán Sotelo[2], and Pascal Bouvry[1]

[1] University of Luxembourg, L-1359 Luxembourg-Kirchberg, Luxembourg
{firstname.lastname}@uni.lu,
[2] Universidad de los Andes, Bogotá D.C., Colombia
{hcastro,mj.villamizar24,ga.sotelo69}@uniandes.edu.co

Abstract. In this paper, we address energy savings on a Cloud-based opportunistic infrastructure. The infrastructure implements opportunistic design concepts to provide basic services, such as virtual CPUs, RAM and Disk while profiting from unused capabilities of desktop computer laboratories in a non-intrusive way.

We consider the problem of virtual machines consolidation on the opportunistic cloud computing resources. We investigate four workload packing algorithms that place a set of virtual machines on the least number of physical machines to increase resource utilization and to transition parts of the unused resources into a lower power states or switching off. We empirically evaluate these heuristics on real workload traces collected from our experimental opportunistic cloud, called UnaCloud. The final aim is to implement the best strategy on UnaCoud. The results show that a consolidation algorithm implementing a policy taking into account features and constraints of the opportunistic cloud saves energy more than 40% than related consolidation heuristics, over the percentage earned by the opportunistic environment.

Keywords: cloud computing, green computing, performance of system.

1 Introduction

In this paper, we consider a Cloud-based opportunistic infrastructure called UnaCloud [1]. UnaCloud implements an Infrastructure as a Service cloud model oriented to the provision of computing resources for the development of scientific projects and to support related activities (i.e., to execute applications such as BLAST, Hmmer or Gromacs). It uses a commodity underlying infrastructure implementing opportunistic design concepts to provide computational resources such as CPU, RAM and Disk, while profiting from the unused capabilities of desktop computer laboratories in a non-intrusive manner offering some of the most important advantages of cloud computing, as for example up-front investment elimination and the appearance of infinite resources available on demand. UnaCloud is composed of (almost) homogeneous computing resources of computer laboratories. This may not be the case for other opportunistic platforms

A.R. Lomuscio et al. (Eds.): ICSOC 2013 Workshops, LNCS 8377, pp. 366–378, 2014.

or community-based Cloud, in which private computer owners donate a portion of their idle computing resources to be used by anyone inside a community for supporting a specific project.

We aim to investigate energy savings on UnaCloud that can serve as a basis for related opportunistic cloud models. Although UnaCloud offers the performance capability of deploying Virtual Machines (VMs) in a sustainable way by using idle resources opportunistically, it lacks of energy saving consideration during the placement of the VMs. UnaCloud implements a random placement of VMs to the idle resources of the Physical Machines (PMs).

Techniques such as Dynamic Voltage and Frequency Scaling and Dynamic Power Management have been extensively studied and deployed to make the Cloud infrastructure components power efficient [2,3]. The consolidation of VMs to reduce the number of underutilized computing resources, and shutting down the unused resources or to transition parts into a lower power state is another efficient energy saving strategy. Therefore, in this paper we investigate four consolidation algorithms to minimize the Energy Consumption Rate (ECR) of UnaCloud. Three of the heuristics rely on classical bin-packing algorithms, and one is an ad hoc (opportunistic) consolidation algorithm proposed in [4], that places the VMs first on PMs already in use by a physical user. We empirically evaluate these heuristics on real workload traces. We use cloud workloads based on real production traces collected from the archives of UnaCloud. The traces have been collected during one year. Real UnaCloud scenarios provide a realistic VMs stream for performance evaluation based on simulations of VMs consolidation algorithms. We present results obtained by simulations, while the ultimate goal is to implement the best strategy on UnaCloud. The main reason is that UnaCloud is a production cloud infrastructure already in use making it difficult the implementation for testing the algorithms. However, based on the generated results a work in progress is considering the implementation on UnaCloud.

Although Unacloud may aggregate desktops from independent individuals it is meant to work with machines within a computer laboratory, each laboratory is managed by a single administrator. It gives UnaCloud several advantages in terms of homogeneity, control and risk of failures. In our tests, a computer laboratory with 35 machines presents a maximum of two failures a day. Besides, UnaCloud has been used to run Bag of Tasks applications, with many short-lived jobs, making unnecessary to deal with checkpoints and fault tolerance issues [5].

This paper is organized as follows: Section 2 presents related work. Section 3 describes the energy consumption behavior on an opportunistic cloud infrastructure, and details the energy model. The evaluated consolidation algorithms are described in Section 4. Section 5 presents the simulation results. Section 6 concludes the paper.

2 Related Work

The problem of VM allocation can be divided in two: the first part is admission of new requests for VM provisioning and placing the VMs on hosts, whereas the

second part is optimization of current allocation of VMs. In this work we focus on the first part of the VM allocation problem. This problem can be modeled as a bin packing problem with variable bin sizes.

The problem in its single-objective variant is an NP-hard problem, and thus is expensive to compute with increasing numbers of PMs and VMs. Different heuristics and linear programming based solutions are used for getting a near optimal solution. Several proposed power-aware packing algorithms use a variant of First Fit Decreasing (FFD) heuristic to reduce resource fragmentation. Static consolidation considers that resource utilization does not change during execution and the number of reconfigurations depends solely of creation and deletion of VMs, one example is Entropy [6]. Dynamic VM management assumes that resource needs change over time, and thus VMs can be moved during their execution in order to improve the optimality placement, one framework that considers dynamic management of VMs is Snooze [7].

Verma et al. [8] modeled the VM workload placement as an instance of the one dimensional bin-packing problem and extended FFD to perform the placement. Li et al. [9] proposed the EnaCloud framework and a modified version of the Best-Fit algorithm is implemented. Buyya et al. [10] presented simulation-driven results for a workload consolidation algorithm based on a modified version of the Best Fit Decreasing algorithm. The algorithm sorts all the VMs in decreasing order of current utilization and allocate each VM to a host that provides the least increase of power consumption due to this allocation. This allows leveraging heterogeneity of the nodes by choosing the most power-efficient ones. Feller et al. [11] dealt with the workload consolidation problem and model it as an instance of the multi-dimensional bin-packing problem. The authors proposed a nature-inspired workload consolidation algorithm based on the Ant Colony Optimization framework. The proposed algorithm outperforms FFD, however, at greater complexity. Beloglazow and Buyya [12] proposed a resource management policy for virtualized cloud data centers. The objective is to consolidate VMs leveraging live migration and switch off idle nodes to minimize power consumption, while providing required Quality of Service.

Most of the related consolidation algorithms consider a pool of dedicated computing resources, however the work in this paper differs from related state of the art since we deal with an opportunistic cloud-based infrastructure.

3 Energy in an Opportunistic Cloud Environment

In this section we present the energy model. First, we present the energy function of a desktop machine, then the energy model is described.

3.1 Parameter Tunning

In opportunistic cloud solutions there are some desktops computers which are donned by users of the same or different institutions to aggregate processing capabilities of the system. Those desktop computers can be modeled as a Set of

Physical Machines (SPMs) each one with some hardware specifications including CPU cores, RAM memory, hard disk and networking. Researchers require the execution of a Set of Virtual Machines (SVMs) each one requiring a minimum hardware specification. Due to opportunistic environment, we assume that there are two states (idle and busy) of a PM to be selected to deploy a VM, and according to its state the ECR and the estimated execution time of a VM executing a CPU-intensive task can change. To analyze the relation between the CPU usage and the ECR consumed by a PM, experimental tests of a previous and recent work [5] show that the CPU usage and ECR are not directly proportional. Figure 1 shows the function $f(x)$ (based on a regression calculated on experimental tests) that allows to estimate the ECR of a PM according to its CPU usage. Here we present CPU usage as the percentage provided by the operating system which is in aggregation of the utilization across all cores of the CPU.

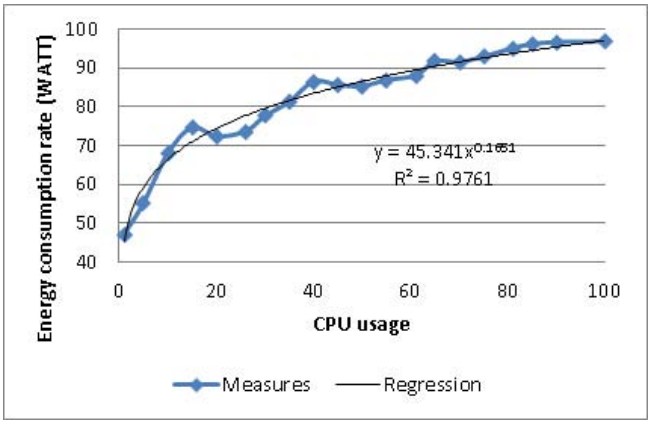

Fig. 1. Desktop Computer energy consumption. $f(x) = y(x) = 45.341x^{0.1651}$

Table 1 provides the ECRs required to execute a VM according to the state of the PM. We assume that a VM will execute a CPU-intensive task during a τ time. While the VM is in execution the ECR of the PM will increase to $f(x)$, where $f(x)$ is a real function that returns the ECR of a PM given its CPU usage percentage. The idle state represents a turned on PM without a user (a student, administrative, etc.) using it while the VM is in execution. The busy state represents a turned on PM with a user using it and a VM in execution.

When the physical machine is in idle state, the ECR consumed by the VMs is equal to the difference between $f(x)$ and $f(0)$, where $f(0)$ is the ECR consumed by the PM while is in idle state and $f(x)$ is the ECR when a virtual machine is in execution (by number of cores required from VMs determining the CPU usage). In the busy state, the ECR consumed by the VMs is equal to the difference among $f(x) + ECR_{MON}$, and ECR_{user}, where ECR_{MON} is the ECR of the monitor when there is a user using the PM, and ECR_{user} is the mean ECR of a

physical machine when there is a user using it and there is not a VM in execution. In busy state the execution time of the VM takes more time because there is an user consuming computational resources (and competing by the resources required by the VM), therefore the execution time of the VM can be calculated as $100/L_{free} \times \tau$, where L_{free} is the percentage of CPU dedicated to the virtual machine and τ is the estimated running time (provided on demand when a VM is requested) of the VM. We assume that the execution time of an opportunistic CPU intensive task is linearly proportional to the amount of processor used in the PM that is running it. To estimate the percentage of CPU used by users of the PMs, during daylight working hours we executed different tests that show that the CPU utilization does not exceed 10% on average [13]. That is $L_{free} = 90\%$.

To estimate the ECR on different states, the results of Table 1 can be completed with the function $f(x)$ depicted in Figure 1. Additional parameters such as ECR_{MON} and ECR_{USER} were calculated using specific tests. In tests using a commodity desktop computer, the ECR_{MON} was equal to 20W and the ECR_{USER} was equal to 87W [13].

Table 1. ECR used by a VM executing a CPU-intensive task

Computer state	**Execution with VM**	**ECR for an intensive CPU Task (ECR with VM - without VM)**
1 (idle)	τ	$f(x) - f(0)$
2 (busy)	$\frac{100}{L_{free}} \times \tau$	$f(x) + ECR_{MON} - ECR_{user}$

τ is the sum of τ_{user} and τ_{free} if τ_{user} is less than τ. τ_{user} is given by the time of the PM with an user when it is executing VMs and τ_{free} is the rest of the time to finish the VM, Eq 1 shows the relation of the execution time of the VM when there is an user in the PM.

$$\begin{aligned} \tau &= \tfrac{100}{L_{free}}\tau_{user} + \tau_{free} = \tfrac{10}{9}\tau_{user} + \tau_{free} \\ 1.1 \times \tau_{user} + \tau - \tau_{user} &= 0.1 \times \tau_{user} + \tau \end{aligned} \tag{1}$$

Table 2 shows that from the energy consumption point of view, the best desktop PMs to deploy a VM are those in a busy state.

Table 2. Experimental results of energy consumption

Computer state	**Execution time**	**Mean ECR (W)**
1 (idle)	τ	$f(x) - 47$
2 (busy)	$0.1 \times \tau_{user} + \tau$	$f(x) + 20 - 87$

3.2 Energy Model

In this section we show a mathematical description to calculate the ECR required for executing a SVM on an opportunistic cloud infrastructure.

Once the consolidation process has finished, the program calculates the power consumption. $P_i^{core}(t)$ is the power of a core i at time t that belongs to a PM. Eq 2 shows how is defined this power.

$$P_i^{core}(t) = s_i(t) \cdot \left((1 - y_i(t))\, P_i^{idleC} + y_i(t) P_i^{workC} \right) \tag{2}$$

where P_i^{idleC} and P_i^{workC} are power consumed in idle and work state of the core. $s_i(t)$ denotes if the core is on or not at time t as $s_i(t) = 1$ and $s_i(t) = 0$ respectively. $y_i(t)$ denotes if the core is working or not at time t. When a core is on, without working, consumes P_i^{idleC} but if the core is on and working, it consume P_i^{workC}. The model assumes that power consumption of all system components is essentially constant regardless of the machine activity.

The power consumption of a PM is denoted by $P_j^{mach}(t)$ as Eq. 3 shows.

$$P_j^{mach}(t) = z_j(t) \cdot \left((1 - w_j(t))\, P_j^{idleM} + w_j(t) P_j^{workM}(t) \right) \tag{3}$$

where P_j^{idleM} and P_j^{workM} are power consumed in idle and busy states of the PM. $z_j(t)$ denotes different states of the PM as Eq. 4 shows. The value of $z_j(t) = 1.1$ refers to the extra power consumption because of competing by the resources required as aforementioned. $w_i(t)$ denotes if the core is working or not at time t. Hence, power consumed by machine have direct relation to power consumed by core.

$$z_j(t) = \begin{cases} 1 & if\ mach\ is\ on \\ 1.1 & if\ mach\ is\ on\ and\ user \\ 0 & otherwise \end{cases} \tag{4}$$

We assume that P_j^{idleM} is the sum of P_i^{idleC} of all cores belong to machine j and is the same when machine is in idle state. P_j^{workM} is calculated from the equation resulted from Figure 1 as Eq. 5 shows.

$$P_j^{workM}(t) = V_{j_{ini}} x_j(t)^{\left(\log_{10} \frac{V_{j_{max}}}{V_{j_{ini}}} \right)/2} \tag{5}$$

where $x_j(t)$ is taken from Figure 1 as Eq. 6 shows. We have specific homogeneous machines with a behavior as it shows in Figure 1 and related with Eq.5, we can deduce $V_{j_{ini}} = 45.341$ and if $V_{j_{max}} = 97$ we have $P_j^{workM}(t) = 45.341 x_j(t)^{0.1651}$.

$$x_j(t) = \frac{T\_u\_c}{T\_c} \times 100 \tag{6}$$

Where T_u_c refers to Total used cores and T_c refers to Total cores. As we assumed for P_j^{idleM}, we assume the power when a PM j is at full charge, it is when all the cores are working, therefore we can deduce:

$$\begin{gathered} P_j^{idleM} = \textstyle\sum_i^{T\_c} P_i^{idleC} \\ P_j^{mach}(t_{max}) = P_j^{workM}(t_{max}) = P_i^{workC} \times T\_c \\ P_j^{workM}(t) = P_i^{core} \cdot T\_u\_c \end{gathered} \tag{7}$$

Now, from equations 5 and 7 we can conclude:

$$\begin{aligned} P_j^{idleM} &= V_{j_{ini}} \\ P_{j_{max}}^{mach} &= V_{j_{max}} \end{aligned} \tag{8}$$

To calculate the energy consumed by a PM, once the placement process has finished, we sort all the VMs assigned to each machine by execution time (et) in descending order:

$$\tau_1 > \tau_2 \tag{9}$$

where τ_1 denotes the biggest execution time of VM in the machine assigned and τ_2 the next execution time in descending order. Using equation 10 we can calculate the energy consumed by a PM

$$E_j = \sum_{k=1}^{T\_v-1} P_j^{mach}(\tau_k) \cdot (\tau_k - \tau_{k+1}) + P_j^{mach}(\tau_{T\_v}) \cdot (\tau_{T\_v}) \tag{10}$$

where T_v refers to total of VMs assigned to a PM j. The total energy consumed by the system is denoted as E_{total} (see Eq. 11)

$$E_{total} = \sum_{j}^{T\_m} E_j \tag{11}$$

where T_m refers to the total PMs in the system.

4 Evaluated Consolidation Strategies for UnaCloud

UnaCloud usually executes VMs to run CPU-intensive applications. Therefore, we consider that the VMs consolidation problem is constrained by a single resource, in this case all the VMs are CPU bound, then the problem corresponds to the one dimensional bin packing problem with different bin capacity (i.e., different number of cores per PM).

The most popular and used heuristics to deal with the one dimensional bin packing problem are First-Fit, First-Fit Decreasing and Best-Fit algorithms [14]. These heuristics use a greedy weight function applied to the items such that every item is assigned a single value. In case of First-Fit Decreasing and Best-Fit algorithms the items are sorted and then placed sequentially into a decreasing order. The heuristics, specially First-Fit Decreasing, are best known to be very effective both in theory with performance guarantees and in practice. Several systems have implemented variants of the three heuristics. Therefore, we consider them to be evaluated in the context of UnaCloud.

To minimize the ECR rate used by the opportunistic cloud, the consolidation strategies should try to place the VMs first on a PM in a busy state (i.e., with a VM already assigned on it or with a physical user) that satisfies the VM requirements, instead of placing the VM on a PM in a different state. In [4]

we proposed a packing algorithm that prioritizes the deployment of VMs on PMs already in use. The algorithm, called Sorting in [4], is a variant of First-Fit Decreasing. The algorithm starts by sorting the set of VMs and PMs. VMs are sorted in decreasing order of required cores and estimated running time. PMs are sorted by three attributes: (1) PMs that already have virtual machines running on them, (2) PMs in busy state (a user is using them), and (3) PMs with more available CPU cores. Then, VMs are placed on the PMs in a First-Fit policy. After a VM is placed on a PM the ordered list of the PMs is sorted again. The attributes used to sort PMs allow that PMs with a VM already assigned or in a busy state have priority over the others, hence reducing the ECR rate as described in Section 3.

Next section presents the evaluation comparison of First-Fit, First-Fit Decreasing, Best-Fit, and Sorting using UnaCloud scenarios.

5 Experimental Results

This section presents the empirical evaluation of the investigated consolidation algorithms. The aim is to gain a first insight into the performance of the algorithms on different real scenarios before implementing the best one on UnaCloud.

UnaCloud currently has access to three computer labs with 109 desktop computers, whose aggregated capabilities may deliver up to 592 processing cores (70 PMs have four cores each and 39 PMs with eight cores), 572 GB of RAM, 8 TB of storage and 1TB of shared storage in a Network Attached Storage (NAS).

5.1 Workload

In order to provide performance comparison, we use workloads based on real VMs production traces. Real UnaCloud scenarios provide a realistic VMs stream for performance evaluation based on simulations of VMs allocation algorithms. In this paper, traces from the archives of UnaCloud are used. The traces have been collected during one year. The total number of VMs in the workload requested during the year is up to 9800 each one requiring either 1, 2, 3, 4, 6, or 8 cores, 1, 2, 3, 4, 6, or 8 GB of RAM, and 20 GB of storage.

In order to estimate the energy consumed by a given placement, we use the information provided in Section 3. The VMs are also characterized by different time periods from 45 minutes up to 43200 minutes ($\approx$ 720 h), the time requested by users on demand to execute VMs. The ECR rate values represent the power drawn by the PMs at the utilization given by the placement over the execution time. We assume that when a PM does not have a VM assigned to it and a user is not working on it the PM can be turned off. Hence, no energy is consumed by the PM and is not included in the computation of the total ECR rate.

5.2 Experimental Scenarios

Different scenarios have been generated as follows. We consider that a given percentage of PMs has a user working on it. To simulate the scenarios and generate the instances used by the consolidation algorithms, we vary the percentage

from 0% up to 50% with the increment 10%. We randomly assign a user to a PM and we generate 30 instances for each scenario. The number of VMs to be placed on the PMs varies from 40 up to 130 with the increment 10. We generate 30 instances at random for each size from the real traces and we used them as workloads for all the simulations discussed below. The maximum number of cores on each size is up to the total available physical cores in order to support the worst packing scenario, in which all the PMs run at least one VM. We assume that a physical user utilizes a PM during 60 minutes up to 240 minutes. We randomly assign the time of the physical user working on the PM (uniformly in the interval [60, 240]).

5.3 Algorithms Comparison

We measured the amount of provisioned PMs, and the ECR rate of the placement for every algorithm. We report average results. We only report results for scenarios assuming 0% and 50% of PMs with a user. The objective is to explore the behavior of the investigated heuristics and the gain of the opportunistic environment.

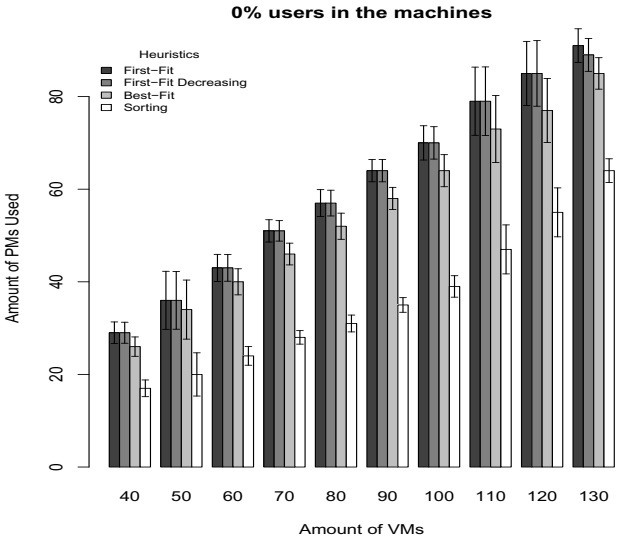

Fig. 2. Number of PMs to place the VMs when there are no users in PMs

Figures 2 and 3 show the number of PMs used by each of the heuristics to place all the VMs when there are no physical users using the PMs and when 50% of the machines already have a physical user, respectively. Figures 4 and 5 present the ECR rate for the considered scenarios without users and with 50% of machines with a user, respectively.

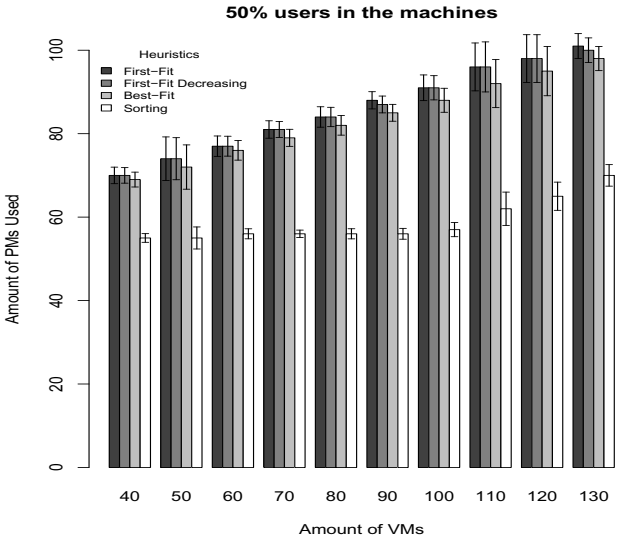

Fig. 3. Number of PMs to place the VMs when 50% of PMs have a physical user

As we can see the Sorting heuristic utilizes important lower amount of PMs yielding to superior average PM optimization and significant ECR lower consumption. The heuristic optimizes up to 41% more than First-Fit and First-Fit decreasing, and 36% more than Best-Fit when there are no physical users in the PMs. When 50% of the PMs are in an busy state Sorting requires 30% on average less PMs than the related heuristics. The main reason that Sorting needs more PMs in the second scenario is that it places the VMs to a maximum number of PMs in a busy state.

We can observe that Sorting gains significant energy saving (i.e. low ECR) regarding the related heuristics. The average gain of ECR by Sorting when no users are assigned to PMs is up to 38% regarding First-Fit, up to 34% concerning First-Fit Decreasing, and up to 35% with respect to Best-Fit. For the second scenario whit 50% of busy PMs all the investigated heuristics gain more energy than in the first scenario with all the PMs in an idle state. These results highlight the benefits and advantages of an opportunistic cloud, in this case UnaCloud, as a sustainable infrastructure.

The average ECR gain of Sorting is more important regarding related heuristics than in the first scenario. Sorting optimizes 48% more energy than First-Fit, it can gain up to 45% regarding First-Fit Decreasing, and up to 46% more than Best-Fit. The results highlight that prioritizing busy PMs to place the VMS is a good saving energy option to consolidate VMs.

It can be observed that First-Fit utilizes more PMs (approx. 4%) in the opportunistic environment to consolidate the VMs than Best-Fit in almost all the scenarios, however the ECR rate is lower than Best-Fit. We consider that it is

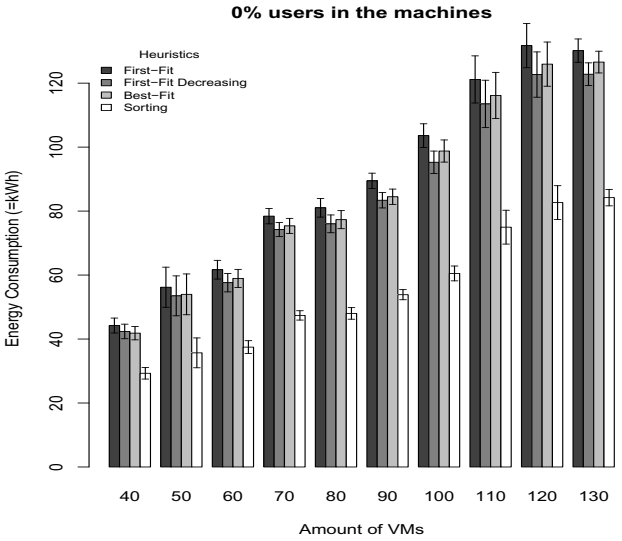

Fig. 4. Total ECR consumed when there are no users in PMs

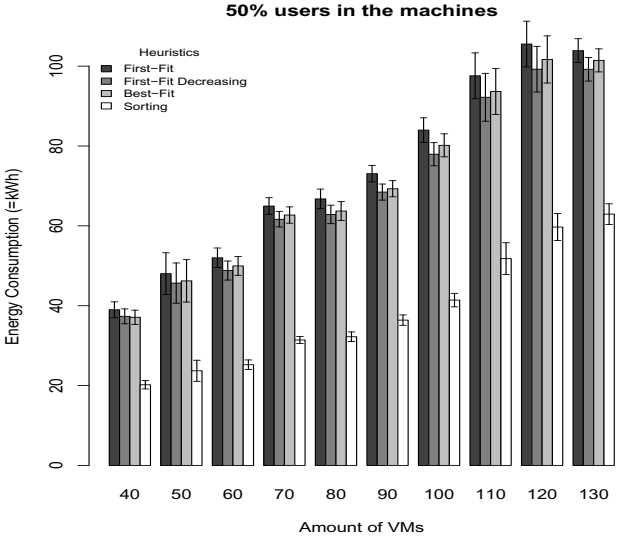

Fig. 5. Total ECR consumed when 50% of PMs have a physical user

due to the fact that First-Fit uses more PMs in an idle state than Best-Fit, nevertheless for a shorter time of period than Best-Fit, hence the ECR rate is less than Best-Fit.

6 Conclusions and Future Work

In this paper, we addressed energy savings on an opportunistic infrastructure, specially for UnaCloud. We investigated and empirically evaluated four state of the art consolidation algorithms. We focused on the optimization of the ECR rate. We have simulated different scenarios using real workload traces. The results showed that the opportunistic infrastructure seems to be a good option as a sustainable computing on demand infrastructure. The results also highlight that a consolidation algorithm implementing a policy that prioritizes the placement of VMs onto busy PMs can reduce the energy-consumption more than 40% against related heuristics, over the percentage earned by the opportunistic environment.

A work in progress is the implementation of Sorting on UnaCloud. However, we are adapting the heuristic by considering free-knowledge information (i.e., without assuming the estimated running time of VMs), and the dynamic feature of the infrastructure. We also plan to consider more than one resource during the placement of VMs, the problem can be modeled as a multi-dimensional packing problem.

Acknowledgment. This work was completed with the support of the FNR INTER/CNRS/11/03 Green@Cloud Project.

References

1. Rosales, E., Castro, H., Villamizar, M.: Unacloud: Opportunistic cloud computing infrastructure as a service. In: Cloud Computing 2011, pp. 187–194. IARIA (2011)
2. Wang, L., Khan, S.U., Chen, D., Koodziej, J., Ranjan, R., Zhong Xu, C., Zomaya, A.: Energy-aware parallel task scheduling in a cluster. Future Generation Computer Systems 29(7), 1661–1670 (2013)
3. Bilal, K., Khan, S., Madani, S., Hayat, K., Khan, M., Min-Allah, N., Kolodziej, J., Wang, L., Zeadally, S., Chen, D.: A survey on green communications using adaptive link rate. Cluster Computing 16(3), 575–589 (2013)
4. Diaz, C., Castro, H., Villamizar, M., Pecero, J., Bouvry, P.: Energy-aware vm allocation on an opportunistic cloud infrastructure. In: Proceedings of the 2013 13th IEEE/ACM Int. Symposium CCGRID, pp. 663–670. IEEE Computer Society (2013)
5. Castro, H., Villamizar, M., Sotelo, G., Diaz, C., Pecero, J.E., Bouvry, P.: Green flexible opportunistic computing with task consolidation and virtualization. Cluster Computing, 1–13 (2012)
6. Hermenier, F., Lorca, X., Menaud, J.M., Muller, G., Lawall, J.: Entropy: a consolidation manager for clusters. In: Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, VEE 2009, pp. 41–50. ACM, New York (2009)

7. Feller, E., Rilling, L., Morin, C.: Snooze: A scalable and autonomic virtual machine management framework for private clouds. In: Proceedings of the 2012 12th IEEE/ACM Int. Symposium CCGRID, pp. 482–489. IEEE Computer Society, Washington, DC (2012)
8. Verma, A., Ahuja, P., Neogi, A.: pMapper: Power and migration cost aware application placement in virtualized systems. In: Issarny, V., Schantz, R. (eds.) Middleware 2008. LNCS, vol. 5346, pp. 243–264. Springer, Heidelberg (2008)
9. Li, B., Li, J., Huai, J., Wo, T., Li, Q., Zhong, L.: Enacloud: An energy-saving application live placement approach for cloud computing environments. In: IEEE CLOUD, pp. 17–24. IEEE (2009)
10. Buyya, R., Beloglazov, A., Abawajy, J.H.: Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges. In: Arabnia, H.R., Chiu, S.C., Gravvanis, G.A., Ito, M., Joe, K., Nishikawa, H., Solo, A.M.G. (eds.) PDPTA, pp. 6–20. CSREA Press (2010)
11. Feller, E., Rilling, L., Morin, C.: Energy-aware ant colony based workload placement in clouds. In: Proceedings of the 2011 IEEE/ACM 12th Int. GRID, pp. 26–33. IEEE Computer Society, Washington, DC (2011)
12. Beloglazov, A., Buyya, R.: Energy efficient allocation of virtual machines in cloud data centers. In: Proceedings of the 2010 10th IEEE/ACM Int. Conference CCGRID, pp. 577–578 (2010)
13. Castro, H., Villamizar, M., Sotelo, G., Diaz, C.O., Pecero, J.E., Bouvry, P., Khan, S.U.: Gfog: Green and flexible opportunistic grids. In: Khan, S.U., Wang, L., Zomaya, A.Y. (eds.) Scalable Computing and Communications, Theory and Practice. Wiley&Sons (forthcomming)
14. Panigrahy, R., Talwar, K., Uyeda, L., Wieder, U.: Heuristics for vector bin packing (2011), `http://research.microsoft.com/pubs/147927/VBPackingESA11.pdf` (accessed July 20, 2013)