# Optimization of Application Placement Towards a Greener Cloud Infrastructure

Tania Lorido-Botran(✉), Jose Antonio Pascual, Jose Miguel-Alonso, and Jose Antonio Lozano

Intelligent Systems Group, University of the Basque Country, UPV/EHU, Paseo Manuel Lardizábal, 1 20018 Donostia-San Sebastian, Spain
{tania.lorido,joseantonio.pascual,j.miguel,ja.lozano}@ehu.es

**Abstract.** Cloud infrastructures are designed to simultaneously service many, diverse applications that consist of collections of Virtual Machines (VMs). The policy used to map applications onto physical servers (placement policy) has important effects in terms of application performance and resource efficiency. This paper proposes enhancing placement policies with network-aware optimizations trying to simultaneously improve application performance, resource efficiency and, as a consequence, power efficiency. The per-application placement decision is formulated as a bi-objective optimization problem (minimizing communication cost and minimizing the number of physical servers assigned to the application) whose solution is searched using an evolutionary algorithm with problem-specific crossover and mutation operators. Experiments carried out with a simulator demonstrate how a low-cost optimization technique results in improved placements that achieve all the target objectives.

**Keywords:** Cloud computing · Tree-network topology · VM placement · Multi-objective optimization · Energy consumption

## 1 Introduction

In recent years, the utilization of cloud infrastructures to host applications has spread widely. The characteristic that makes these cloud systems so appealing is their elasticity, that is, resources can be acquired on-demand, depending on the time-varying application needs, but paying only for those actually booked (a scheme known as pay-as-yo-go). Virtualization technologies enable the cloud infrastructure to provide such elastic usage. The resources offered by physical servers, organized in several data centers, are provided in the form of abstract compute units that are implemented as Virtual Machines (VMs). Each VM is assigned a pre-configured set of resources: number of cores, amount of memory, disk and network-bandwidth.

Virtualized data centers support a large variety of applications, including batch jobs (scientific applications), and web applications (e.g. an online bookshop or a blog hosting site). Each application is deployed on a set of VMs, which can be allocated to any collection of physical servers in the data center.


A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2014, LNCS 8602, pp. 690–701, 2014.
DOI: 10.1007/978-3-662-45523-4_56

The problem of assigning a physical location to each VM is known as *VM placement* and it is performed by the manager of the cloud infrastructure. This manager is typically called the Infrastructure-as-a-Service (IaaS) provider.

The challenge for the provider is to host a large and diverse set of applications (VM sets from different clients) in the infrastructure trying to (1) maximize its revenue and (2) provide a good service to the clients. An adequate application placement would be able to maximize the resource usage of physical servers and reduce the energy consumption of the data center, for example by turning off (or setting to idle state) the inactive servers and switches. At the same time, the infrastructure management policies should balance the obtained revenue with the Quality of Service (QoS) agreed with the client, guaranteeing that each application receives the resources payed for.

The VM placement problem has been extensively explored in the literature [11] [12]. Most efforts have been directed towards optimizing the usage of CPU, memory and disk resources, and reducing the energy consumption of physical servers. However, not enough attention has been paid to the utilization of the network. An inappropriate placement of VMs with heavy communication requirements could lead to the saturation of certain network links, with the subsequent negative impact on applications (longer execution or response times). Besides, as stated in [9], the network power has been estimated at 10-20% of the overall power consumption. For this reason, the VM placement policy should try to reduce not only the use of physical servers, but also the use of network links and switches to reduce the total power footprint.

The most common topology of data center networks is a tree of switches arranged in several tiers. The communication latency of any pair of VMs depends on the distance between the physical servers in which they are allocated. This, in turn, depends on their position in the tree. Distance is measured as the number of hops from the sender VM to the recipient one. The collection of VMs forming an application communicate between them following a certain communication pattern. In batch jobs implementing, for example, a scientific computation, the pattern may be all-to-all. In web applications, the VMs are arranged into several layers and there may be intra- and inter-layer communication. Other patterns are possible, depending on the particular characteristics of the application.

Based on the communication pattern of an application, it is straightforward to compute the input/output network bandwidth needed by each VM. The most communicative VM subsets should be placed as *close* as possible (minimizing the distance between them in terms of network hops). This means using the minimum number of physical servers, because intra-server communication is the cheapest. The constraint is that the external aggregated bandwidth required by the all the VMs in a server, from the same or from different applications, cannot exceed the bandwidth of its network connection.

Two examples of VM placement policies that can be used in data centers are first fit (FF) and round robin (RR). Each of them has a different characteristic that makes it appropriate for its use in the data center. The objective of FF is to reduce the number of physical servers in use, saving energy. RR tries to equalize

the utilization of all servers to avoid excessive wearing-out of server subsets and thermal peaks. We demonstrate how it is possible to take these policies as starting points and use optimization techniques to improve the benefits for both the infrastructure provider and the application.

The remaining of this paper is organized as follows. After a review of the literature (Section 2), we provide in Section 3 models for cloud applications, data center organizations, and the energy consumed by servers and switches. Then we formulate VM placement as a multi-objective optimization problem (Section 4). We assess the benefits of our approach using the experiments defined in Section 5, whose results are discussed in Section 6. We end in Section 7 with some conclusions and future lines of work.

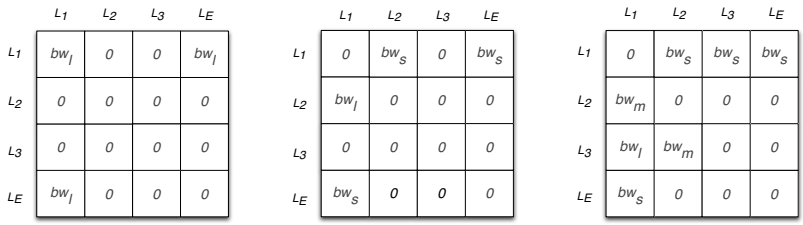
## 2 Related Work

Open-source tools for cloud management use rather simple placement policies. For example, Eucalyptus [1] implements FF and RR strategies that only consider the VM requirements and the availability of resources. It also implements a PowerSave policy that is similar to the ranking algorithm available in OpenNebula [4]: choosing first the most used servers (with room for the new demand) with the objective of minimizing the number of used servers and, therefore, the power consumption. Commercial tools for capacity planning, such as NetIQ PlateSpin Recon [3], VMware Capacity Planner [5] and IBM Workload Deployer [2] also focus on maximizing the resource usage and power consumption savings. None of these tools explain how VM placement is carried out.

Neither open-source nor commercial tools consider the impact of network topology and the communication patterns of applications, but it has been analyzed in several research works [7] [8] [9] [11] [12]. For example, Meng et al. [11] propose grouping VMs and servers into clusters, addressing VM placement for each (VM-cluster, server-cluster) pair as a Quadratic Assignment Problem (QAP). The VM clustering tries to maximize the intra-cluster communication and reduce the inter-cluster communication, but all VM-clusters have equal size. The server set assigned to a VM-cluster is fixed. This work does not consider the energy consumed by physical servers. Mann et al. [9] propose an approach similar to ours, but using a greedy heuristic. However, their work does not consider the large variety of applications that can run in the cloud. Georgiou et al. [8] also propose a greedy heuristic to improve the network utilization, but they do not try to consolidate the VMs in the minimum number of physical servers.

## 3 Application, Data Center and Power Models

This section presents several models that will be later used to define and solve the VM placement problem. First, we present an application model, which covers a wide range of application types that run on cloud environments. Next, we define a general model for describing the interconnection network topology of tree-based data centers. Finally, a power model is introduced to estimate the power consumption of physical servers and switches.




(a) Batch (b) L-WA (c) H-WA

**Fig. 1.** Simplification of the communication matrix for each application type (defined per layer instead of per VM)

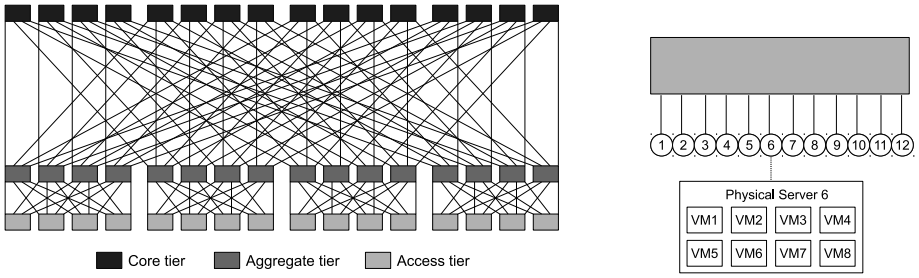
### 3.1 Modeling Applications

A cloud environment is suitable to run a diversity of applications, formed by a collection of VMs typically organized in one or more layers. The way VMs communicate among them determine the communication pattern of the application.

We propose a simple model that allows us to represent the structure of any kind of application providing a few parameters: number of layers of the application ($L$), the number of VMs in each layer ($N_i$ being $i$ the layer identifier) and a matrix of the communication needs (or bandwidth, measured in Mb/s) between each pair of VMs $i$ and $j$ ($BW = [bw_{i,j}]$) and with the external world.

For this work, we particularize this model to define two classes of applications: (1) batch jobs, typical of scientific workloads, and (2) web applications. Batch jobs represent the execution of parallel applications, or workflows comprising parallel tasks. The main characteristic of a batch job is the intense internal communication (between tasks-VMs of the same application). In this work, we model them as a single-layer application, with communications following patterns such as all-to-all, neighbour-to-neighbour in a 2D (virtual) arrangement, and neighbour-to-neighbour in 3D.

Web applications are usually implemented using a three-layer architecture: a load balancer that receives end-user requests, a business layer that processes those requests (and replies to them), a and database (DB) or persistence layer. The load balancer distributes the input requests evenly along the VMs of the business layer; it may be implemented on a hardware device, or as a DNS-based redirection—thus, we do not include it in the application model. The number of VMs at the persistence layer depends on the database requirements of the applications. A light workload can be managed by a single DB server that supports both read and write operations; we represent this class of applications as *L-WA*. For applications with heavy database demands (*H-WA*), a master-slave replication scheme may be applied: one of the VMs of the persistence layer is the *master* node that processes all the write operations, while the queries (read operations) are evenly distributed along the remaining VMs in the layer. Whenever a change (write) is done on the master node, it is propagated to the read VMs.




(a) 3-tier fattree build with 8-port switches (b) 24-port access switch

**Fig. 2.** Representation of the physical configuration of a data center (network and servers)

IaaS providers offer different types of VMs, with different resource sets. In this work, we will consider small, medium and large instances, with different characteristics only in terms of allotted network bandwidth (in Mb/s): $bw_s$=50, $bw_m$=150 and $bw_l$=300, respectively. Our batch applications use a single layer $L_1$ of large instances. For *L-WA* web applications, the business layer ($L_1$) uses small instances and the database is represented as a single, large VM in layer ($L_2$). For *H-WA* applications, the database is modelled as a $L_2$ layer for reading, with several medium-size VMs, and a single-VM (of large size) layer $L_3$ for writing. Figure 1 shows the communication pattern between layers for each application type, that is reflected on the $BW$ matrix. The additional layer $L_E$ represents the traffic to/from the Internet and the application itself.

### 3.2 Describing the Data Center Structure

As stated before, current data centers are usually built using tree-based topologies, such as fat tree and VL-2 [11]. This kind of networks are composed of several tiers of switches (we assume homogeneous switches) and several servers connected to the bottom tier of the tree (the edge or access tier). Each server is divided into several *slots*, where each slot can be a fraction of a core, an entire core or several cores. Application VMs are assigned to different slots of the data center servers. Throughout this work we assume that a VM consumes a slot, and that one slot is equivalent to one core of a multi-core server.

The physical configuration of a data center is defined as the number of servers ($P$), the number of cores per server ($C_p$) and the network topology. In particular, a tree-based topology is defined by the number of uplinks and downlinks of the switches ($S_{up}$ and $S_{down}$), the bandwidth (Mb/s) offered by each switch port ($S_{bw}$) and the number of tiers of the tree ($T$). The communication latency between two cores $i$ and $j$ depends on the distance between them, measured in terms of *hops*. Matrix $D = [d_{i,j}]$ defines the distance between any pair of cores (actually, the servers to which they belong).

We have focused on data centers built using fat trees as interconnection network, composed of three tiers (as depicted in Figure 2(a)) with the same

**Table 1.** Parameter values of energy utilization in physical servers and switches

| | Consumption at | Server value (W) | Switch value (W) |
|---|---|---|---|
| $E_{max}$ | 100% utilization | 200 | 100 |
| $E_{idle}$ | Idle state | 10 | 10 |
| $E_{active}$ | One active core/port | 160 | 31 |
| $E_{rem}$ | Remaining $U_{active} - 1$ cores/ports | 40 | 69 |

number of switches in each one of them (see Figure 2(b)). We consider that core switches are directly connected to the Internet. In this kind of tree the distance between two servers (matrix $D$) is computed as follows: cores in the same physical server are at distance 0; servers connected to the same access switch are at distance 2; if aggregation or core switches are required, distance grows to 4 and 6 respectively. The physical configuration of the data center used in this work is: ($P = 1728$ servers, $C_p = 8$ cores, $T = 3$ tiers, $S_{up} = 12$ ports, $S_{down} = 12$ ports, $S_{bw} = 1000$ Mb/s).

### 3.3 Modeling Power Requirements

Energy is consumed by servers and switches, and also by cooling and energy distribution systems. Reducing power use has direct benefits for the infrastructure provider (lowering the energy bill), while reducing the data center carbon footprint.

PowerNap [10] aims to reduce the consumption of unused servers by switching off memory, disk and other elements. In this work we assume that a strategy like this is used in the data center: unloaded servers and switches operate in an idle state that minimizes energy waste. We define a general model of power utilization of a device (server or switch), inspired in the one provided in [10].

$$E = \begin{cases} E_{idle} & U_{active} = 0 \\ E_{active} + \frac{E_{rem} \cdot (U_{active} - 1)}{U_{total} - 1} & U_{active} > 0 \end{cases}$$

The energy consumption $E$ of a server/switch (in Watts) depends on the number of active cores or ports $U_{active}$. At idle state, the consumption is equal to $E_{idle}$. The transition from the idle state to the activation of the first core/port implies an important increase in the energy utilization, because it requires turning on other resources (memory, disk) or internal fans. The consumption of each additional, active core/port is directly proportional to the active number of cores/ports. Table 1 shows the energy consumption values used in this work, for both servers and switches. Values for servers are based on those in [10].

## 4 Topology-Aware Optimization

The aim of this work is to find a suitable placement for the VMs forming an application onto a set of available cores (slots) in the data center servers. We

perform an initial selection of free cores using FF or RR (see Section 5); then, a bi-objective optimization algorithm fine-tunes the VM placement taking into account the communication needs of the application—and the corresponding cost considering the assigned cores and the topology of the data center network.

### 4.1 Problem Definition

Given an application $A$ with a VM set $V$ of size $N$, and a subset of available cores $C' \subset C$, where $C$ is the whole set of cores in the data center (note that usually $|C'| >> N$), the VM placement problem involves finding a mapping function $\varphi$ that assigns each VM, $v \in V$ to a core $c \in C'$:

$$\begin{aligned} \varphi : & V \rightarrow C' \\ & v \mapsto \varphi(v) = c \end{aligned}$$

A solution of the VM placement problem has the form $s = (c_1, c_2, \ldots, c_N)$ representing that the VM $i$ has been assigned to core $c_i$.

Two major selection criteria will be considered to choose a VM placement. First, we favor solutions that minimize communication latency. For this reason, the VM placement will try to allocate the most communicative VMs onto physically close cores, in terms of network distance. The second criterion focuses on reducing the number of servers allocated to the application. An allocation solution that fulfills the first criterion may not satisfy the second one. For example, given an application $A = \{v_1, v_2, v_3, v_4\}$ in which communication occurs between $v_1$-$v_2$ and $v_3$-$v_4$, the first criterion may place each pair of VMs on a different physical server. However, according to the second criterion, it would be better to place all the VMs in the same server. Both criteria try to improve the use of data center resources, by means of reducing the number of active servers and switches, but the first one specifically tries to benefit the application, optimizing its performance. Placement solutions must obey a restriction: external communication demands by all the VMs assigned to a server cannot exceed the bandwidth of its network link $S_{bw}$. This constraint does not take into account communication between VMs in the same server.

More formally, we describe VM placement as a bi-objective optimization problem subject to one constraint. The first objective function to minimize is defined as follows:

$$f_1 : \sum_{i,j \in V}^{N} bw_{i,j} \cdot d_{s(i),s(j)} \tag{1}$$

where $d_{s(i),s(j)}$ is the distance between the cores assigned to VMs $i$ and $j$ and $bw_{i,j}$ is the bandwidth required by VMs $i$ and $j$.

Given the function $\sigma(c) = p$ that returns the server $p$ to which core $c$ belongs to, and a solution $s$, we define the set of active servers for this solution as $P^s = \{p | \exists i \in \{1, \ldots, N\} \text{ s.t. } \sigma(s(i)) = p\}$. The second objective function to minimize is defined as:

$$f_2 : |P^s| \tag{2}$$

The solutions are subject to the following constraint:

$$\forall p \in P^s : S_{bw} - S^p_{bw} \geq 0 \tag{3}$$

where $S_{bw}$ is the bandwidth available for each physical server and $S^p_{bw}$ is the reserved bandwidth of server $i$, considering the previously allocated applications and also the new one.

### 4.2 Multi-objective Optimization with NSGA-II

We have chosen the evolutionary algorithm NSGA-II [6] to solve the multi-objective VM placement problem. A solution or individual is represented as a vector that assigns each VM of the application to one available core. After generating an initial population of $N_{pop}$ individuals, an offspring is created from it applying a crossover and a mutation operator with probability $p_{cross}$ and $p_{mut}$ respectively. The resulting population $2N_{pop}$ is sorted, in order to select the best $N_{pop}$ individuals for the next generation. These steps are iterated along $N_{gen}$ generations. For further information about NSGA-II, please refer to [6].

**Guided Crossover.** The crossover operator is applied with probability $p_{cross}$. It combines two individuals to generate a new one, considering the specific characteristics of the problem. Given two parents $s_1$ and $s_2$ the crossover operator generates a new child $ch$ as follows. We define $\phi(i, s)$ as the communication cost of VM $i$ in a candidate solution $s$, considering all the destinations with which it communicates, the corresponding input/output bandwidths, and the distances:

$$\forall i \in \{1, \ldots, N\} : \phi(i, s) = \sum_{j=1, j \neq i}^{N} (bw_{i,j} + bw_{j,i}) \cdot d_{s(i),s(j)} \tag{4}$$

Child $ch$ will be constructed taking from the parents those cores that cause the lowest communication cost. That is, for each VM $i$, if $\phi(i, s_1) < \phi(i, s_2)$, then core $s_1(i)$ is assigned to VM $i$ of child $ch$. A correction step to remove any possible repeated position (cores) of each child may be required.

**Guided Mutation.** The mutation is applied with a probability $p_{mut}$. There are two types of mutation, that are selected based on another probability $p_{mtype}$. The first type performs a simple swap between any two elements of the chosen solution, without considering cores in the same server because this change would not affect the values of the objective functions. With probability 1-$p_{mtype}$, the second type of mutation is applied: one of the cores assigned to the solution is replaced with any free core $c$ from the whole network $C$, selected randomly using a distance-based distribution that favors physically close cores.

**Selection Criterion for Solutions in the Pareto Front.** The bi-objective optimization algorithm generates a collection of solutions for a given application (Pareto set), with different trade-offs between locality and number of allocated servers. As all Pareto optimal solutions are considered equally good, a selection criterion is required to choose one. We select the solution that is most beneficial for the provider: one that minimizes the *global* number of active servers in the data center $P_{active}$.

## 5 Experimental Framework

This section presents the experimental framework used to evaluate the VM placement strategies. The experiments have been performed using an in-house developed scheduling simulator. The initial mapping is generated with a topology-agnostic approach: FF that searches free cores sequentially, always starting at the first one, or RR that also performs a sequential search but starting from the last core used in the previous placement. We then apply the multi-objective optimization over this set of cores. Using this set, the initial population for NSGA-II is generated performing random reorderings of the cores. In all, four VM placement algorithms are considered: FF and RR, without and with optimization, in all cases obeying the bandwidth constraint.

Three initial workload scenarios have been considered, designed to generate low (25%), medium (50%) and high (75%) use of data center resources (servers). Each scenario consists of a sequence of arrival/departure operations (new applications, applications that end). Experiments carried out in the simulator are divided into two phases: first, a warming up until the target load of the scenario is reached and the system arrives to an steady state; then, 10 batches with sets of 1000 operations (equally distributed between arrivals and departures). The simulator gathers different per-batch metrics.

NSGA-II has been used with these parameters: $N_{pop}$=100, $N_{gen}$=100, $p_{cross}$= 0.8, $p_{mut}$=0.8 and $p_{mtype}$=0.5. Parameter tuning for the optimization process falls outside the scope of this work. For this parameter configuration, a run of NGSA-II in a desktop PC takes on average just 3". Given the Stochastic nature of the NSGA-II algorithm, we perform five repetitions for each scenario, using the same list of operations as input. Results gathered in the tables are obtained by calculating the mean of those repetitions.

## 6 Analysis of Results

In this section we discuss the results provided by the simulator, with special focus on the effects that the different placement policies have on applications (which policy is most beneficial in terms of improving communications locality?) and the data center (which policy uses less resources and, therefore, requires less power?). Two approaches are compared, topology-agnostic RR/FF (Without Optimization, $WO$) and topology-aware RR/FF (with Optimization, $O$).

**Table 2.** Values of objective functions, *WO* - Without Optimization, *O* - with Optimization

| | | First fit | | | | Round Robin | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | $\mu_{f_1}$ | $\sigma_{f_1}$ | $\mu_{f_2}$ | $\sigma_{f_2}$ | $\mu_{f_1}$ | $\sigma_{f_1}$ | $\mu_{f_2}$ | $\sigma_{f_2}$ |
| High | WO | 264316.60 | 32567.10 | 14.40 | 2.42 | 174689.20 | 19749.78 | 10.40 | 0.49 |
| | O | 225017.80 | 32522.66 | 14.20 | 2.14 | 145600.60 | 18661.38 | 9.40 | 0.49 |
| Medium | WO | 296971.20 | 18588.08 | 16.40 | 3.26 | 172445.80 | 11480.79 | 8.60 | 0.49 |
| | O | 257856.60 | 20687.72 | 15.80 | 3.12 | 148800.00 | 11933.78 | 8.00 | 0.00 |
| Low | WO | 278839.00 | 10491.40 | 19.60 | 4.08 | 159689.60 | 6721.03 | 7.80 | 0.40 |
| | O | 240159.00 | 12813.12 | 17.80 | 3.37 | 138890.00 | 7062.93 | 7.00 | 0.00 |

### 6.1 Application-Related Metrics

Table 2 gathers the mean $\mu$ and standard deviation $\sigma$ of both objective functions $f_1$ (communications locality) and $f_2$ (number of servers assigned to the application). If we focus on topology-agnostic policies, clearly RR is better for applications, as it provides lower communication costs than FF in all scenarios (see $f_1$ values), while simultaneously providing better (smaller) $f_2$ values (number of servers per application). The most relevant result, though, is that applying optimization improves values of both objective functions for FF and also for RR.

### 6.2 Data Center-Related Metrics

The objective functions were designed to have a positive impact on the whole data center as well as on applications. This section evaluates the impact in terms of the number of active physical servers and the power consumption.

Table 3 contains the number of active servers $P_{active}$. $P_{min}$ is the minimum number of servers that would be necessary to allocate all applications. So, the cost in terms of servers of each VM placement policy can be evaluated as the extra number of servers used relative to $P_{min}$. FF policy obtains a lower number of extra servers than RR, thus being more appealing for the IaaS provider. The use of optimization makes this number even lower ($P_{dif}$), while simultaneously improving application-related characteristics.

These benefits in terms of number of active servers translate immediately into lower power requirements for servers. But optimization is focused on communications, and benefits are also expected in terms of reduction of the power required for switching. We have used the energy models described previously to measure power requirements, separately for servers/switches, and total. Results are summarized in Table 4. We see that RR requires more power than FF for servers (globally it uses more servers) but less for switches (makes better use of the network, because individual applications are allocated in fewer servers, and the upper-tier switches are used less). Using optimization we are able to improve both figures and, therefore, the total power required by the data center.

**Table 3.** Number of active servers in the data center used by the different VM placement strategies

| | | | First fit | | | Round Robin | | |
|---|---|---|---|---|---|---|---|---|
| | | $P_{min}$ | $P_{active}$ | $P_{active}$-$P_{min}$ | $P_{dif}$ | $P_{active}$ | $P_{active}$-$P_{min}$ | $P_{dif}$ |
| High | WO | 1311 | 1461.20 | 150.20 | 33.40 | 1601.80 | 290.80 | 17.60 |
| | O | | 1427.80 | **116.80** | | 1584.20 | **273.20** | |
| Medium | WO | 871 | 988.20 | 117.20 | 22.00 | 1193.80 | 322.80 | 36.40 |
| | O | | 966.20 | **95.20** | | 1157.40 | **286.40** | |
| Low | WO | 429 | 492.00 | 63.00 | 9.40 | 605.20 | 176.20 | 21.20 |
| | O | | 482.40 | **53.40** | | 584.00 | **155.00** | |

**Table 4.** Energy consumption (in Watts) of physical servers and switches. O* values represent the energy savings with respect to the *WO* approach.

| | | First fit | | | Round Robin | | |
|---|---|---|---|---|---|---|---|
| | | $E_{server}$ | $E_{switch}$ | $E_{total}$ | $E_{server}$ | $E_{switch}$ | $E_{total}$ |
| High | WO | 287532.5 | 30359.4 | 317891.9 | 307835.8 | 24397.9 | 332233.7 |
| | O* | 4858.7 | 1322.8 | 6181.5 | 2547.7 | 1412.9 | 3960.6 |
| Medium | WO | 199533.5 | 22097.8 | 221631.2 | 229187.8 | 17927.0 | 247114.8 |
| | O* | 3193.2 | 651.4 | 3844.6 | 5256.8 | 1587.5 | 6844.4 |
| Low | WO | 108183.5 | 13317.1 | 121500.6 | 124508.5 | 10942.6 | 135451.1 |
| | O* | 1385.8 | 325.6 | 1711.5 | 3010.6 | 913.9 | 3924.5 |

## 7 Conclusions and Future Work

Throughout this paper we have demonstrated that a IaaS provider can improve the VM placement policy in use by applying an optimization strategy, with benefits not only for the provider but also for the user. And this optimization can be done at a negligible cost: it is applied when allocating a new application, and it takes a few seconds. Benefits for the provider are measured in terms of used servers and switches, and immediately translate into power demands (resulting in a greener use of the data center). Benefits for the applications are achieved by reducing communication latencies.

This work can be improved in several aspects. One of them is taking into account that providers usually over-subscribe resources: users rarely exploit the 100% of the assigned resources (including cores, memory, network bandwidth, etc.) Therefore it is common practice to assign to a server "extra" slots. This practice rarely affects the QoS perceived by users, although it has to be carefully monitored in the rare event aggregated actual demands exceed server capacity. VM migration is the common solution to this problem, but it does not come for free: it affects QoS and network utilization. We plan to introduce over-subscription and VM migration in our models and experiments.

The elastic capacity of cloud environments allows the applications to dynamically scale the acquired resources (the number of VMs in horizontal scaling) depending on the input workload. Thus, the number of VMs will vary with time and the infrastructure provider should be able to optimize not only the initial placement, but also the addition of new VMs. We plan to adapt our proposal to deal with auto-scalable applications.

**Acknowledgments.** This work has been partially supported by the Saiotek and Research Groups 2013-2018 (IT-609-13) programs (Basque Government), TIN2013-41272P and COMBIOMED network in computational biomedicine (Carlos III Health Institute). Dr. Pascual is supported by a postdoctoral grant of the UPV/EHU. Mrs Lorido-Botran is supported by a doctoral grant from the Basque Government. Prof. Miguel-Alonso is a member of the HiPEAC European Network of Excellence.

## References

1. Eucalyptus, http://www.eucalyptus.com/
2. IBM Workload Deployer, http://www.ibm.com/software/products/us/en/workload-deployer
3. NetIQ PlateSpin Recon, https://www.netiq.com/products/recon/
4. OpenNebula, http://opennebula.org/
5. VMware Capacity Planner, http://www.vmware.com/products/capacity-planner/
6. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6**(2), 182–197 (2002)
7. Fan, P., Chen, Z., Wang, J., Zheng, Z.: Online Optimization of VM Deployment in IaaS Cloud, In: ICPADS. pp. 760–765 (2012)
8. Georgiou, S., Tsakalozos, K., Delis, A.: Exploiting Network-Topology Awareness for VM Placement in IaaS Clouds, In: CGC. pp. 151–158 (2013)
9. Mann, V., Kumar, A., Dutta, P., Kalyanaraman, S.: VMFlow: leveraging vm mobility to reduce network power costs in data centers. In: Domingo-Pascual, J., Manzoni, P., Palazzo, S., Pont, A., Scoglio, C. (eds.) NETWORKING 2011, Part I. LNCS, vol. 6640, pp. 198–211. Springer, Heidelberg (2011)
10. Meisner, D., Gold, B., Wenisch, T.: PowerNap: eliminating server idle power. ACM SIGPLAN Notices **44**(3), 205–216 (2009)
11. Meng, X., Pappas, V., Zhang, L.: Improving the Scalability of Data Center Networks with Traffic-aware Virtual Machine Placement. In: IEEE INFOCOM. pp. 1154–1162 (March, 2010)
12. Wo, T., Sun, Q., Li, B., Hu, C.: Overbooking-Based Resource Allocation in Virtualized Data Center. In: ISORCW, pp. 142–149 (2012)