J Grid Computing (2016) 14:75–90
DOI 10.1007/s10723-015-9343-x

Topology-Aware Virtual Machine Placement in Data Centers

Rodrigo A. C. da Silva · Nelson L. S. da Fonseca

Received: 11 November 2014 / Accepted: 30 July 2015 / Published online: 8 September 2015

Abstract This paper presents the Topology-aware Virtual Machine Placement algorithm, which aims at placing groups of virtual machines in data centers. It was designed to occupy small areas of the data center network in order to consolidate the network flows produced by the virtual machines. Extensive simulation is used to show that the proposed algorithm prevents the formation of network bottlenecks, therefore accepting more requests of allocation of virtual machines. Moreover, these advantages are obtained without compromising energy efficiency. The energy consumption of servers and switches are taken into account, and these are switched off whenever idle.

Keywords Virtual machine · Data center · Consolidation · Virtual machine placement · Energy consumption

1 Introduction

Recent advances in information technology have led to the wide adoption of the cloud computing paradigm. The National Institute of Standards and Technology (NIST) defines cloud computing as "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction" [1]. In a pay-per-use fashion, users can acquire computational resources dynamically and promptly request and release resources according to their needs.

In virtualized data centers, servers host full machines implemented in software, called virtual machines (VMs). Once a VM is instantiated, it can be used to process the target workload, and, after processing this, that virtual machine is released so that the server can accommodate another VM.

Cloud users usually request a set of virtual machines rather than an individual one, especially for processing computation-intensive applications. These VMs can work in a cooperative manner to process the workload. Moreover, applications create flows not only between virtual machines but also onto the Internet. For example, data-intensive scientific workflows running on a cloud produce both inter VM flows and flows to the Internet [2].

One strategy for decreasing the consumption of resources by these flows is to place communicating virtual machines close to each other, thus requiring shorter paths for the flows from one to the other. The shorter the paths, the lower is the number of switches and links visited by these flows, decreasing the energy consumption, which is a major issue in data center

R. A. C. da Silva · N. L. S. da Fonseca (✉)
Institute of Computing, State University of Campinas,
Av. Albert Einstein, 1251 Cidade Universitária, 13083-852,
Campinas SP, Brazil
e-mail: nfonseca@ic.unicamp.br

R. A. C. da Silva
e-mail: rodrigo@lrc.ic.unicamp.br

management. In 2010, the energy consumption of data centers was about 1.5 % of all the energy consumed in the world and this should increase as a consequence of the increase in the adoption of cloud computing [3]. One approach for the reduction of energy consumption is to use servers and switches in a low consumption mode when they are idle [4].

One of the main issues in data center management is the determination of the physical machine (server) on which a virtual machine will be placed. This decision is known as the virtual machine placement problem. The solution to this problem is crucial for the minimization of the resource consumption of a request involving a group of communicating VMs. Consolidating jobs and network flows on few network equipment makes it possible for idle servers and switches to be turned off and, therefore, to save energy. The placement of a group of communicating VMs is equivalent to the virtual network mapping (embedding) problem, which is NP-hard and, therefore, requires efficient heuristics to provide solutions in an acceptable processing time.

In this paper, we present a novel algorithm for the virtual machine placement problem. It aims at consolidating groups of communicating VMs in a small area of the data center. The algorithm uses a recursive approach which tries to find solutions using the smallest possible area of the data center. The algorithm is oriented towards hierarchical topologies such as Fat-tree and it tries to find solutions at the lowest possible level of hierarchical topologies. The energy consumption of both servers and switches is considered in the criterion for the placement of VMs. The algorithm has been extensively evaluated using simulation. We employ the Fat-tree topology and show that, in scenarios with heavy traffic between VMs, the proposed algorithm uses network resources efficiently, admitting a larger number of VMs than other algorithms. Moreover, these advantages are obtained without compromising energy efficiency. Results in this paper revise results of a preliminary investigation [5] and extend the study by considering diverse traffic scenarios.

The rest of this paper is organized as follows. In Section 2, we present related work on virtual machine placement. In Section 3, we describe the proposed algorithm. In Section 4, we present the evaluation of the algorithm. Finally, in Section 5, we draw some conclusions and suggest future work.

2 Related Work

Recently, the problem of virtual machine placement in data centers has been studied in the literature [6–16]. Algorithms for VM placement can have different goals, such as energy efficiency and reduction of response time. These strategies generally involve virtual machine consolidation, which aims at concentrating the workload on the smallest possible number of servers, and network flows consolidation, which aims at concentrating flows passing in a smaller number of switches. This section reviews different strategies for VM placement dealing with energy and networking issues in virtualized data centers.

Considering the common three-tier data center architecture [17], the authors in [6] propose an energy-aware placement strategy, including both servers and switches in the decision. The metric proposed favors high server utilization and penalizes the selection of underutilized servers and considers the load of queues at the switches. This approach evaluates the trade-off between load balancing of traffic and consolidation of workloads, being particularly relevant in data centers running data intensive jobs that impose low computational loads. Their approach does not, however, consider heavy internal traffic demands.

The algorithm in [6] was enhanced by selecting a set of servers with high connectivity and then choosing the server with the smallest available computing capacity. Such change improves the energy efficiency and prevents the formation of network congestion [7].

Multi-objective evolutionary algorithms were employed to solve the placement problem in [8]. The objectives were the consolidation of VMs on a small set of processors as well as the minimization of associated energy costs for servers and network equipment. A Fat Tree topology and tiered applications, such as a web server with an associated database, were considered in the performance evaluation. The algorithms are suitable for enhancement of the application performance and energy consumption.

In [9], the data center is modelled as a set of servers, with algorithms for virtual machine placement and migration designed to save energy without violating service-level agreements (SLA's). The problem is modeled as a bin-packing problem, and the proposed solution is a variation of the best fit decreasing algorithm. Results show potential energy savings without a significant number of SLA violations. Although the

authors manage to consolidate virtual machines on few servers, they do not account for the impact of network traffic, which may not be consolidated efficiently.

In [10], the virtual machine placement problem is addressed. This approach also considers the data center as a set of servers. In addition to traditional approaches designed to reduce energy consumption, it employs active cooling control and the authors conclude that the proposed algorithm performs well in medium to large data centers. They conclude that active cooling control results in relevant savings.

In [11], the authors use a multi-dimensional bin-packing model for virtual machine placement. Each computational resource (processor and disk) is a bin, and a heuristic is proposed to achieve high server utilization, based on the euclidean distance of the current VM location. The numerical evaluation considers only servers in the data center.

Algorithms for the placement of precedence-constrained parallel virtual machines are introduced in [12]. The authors propose reducing energy consumption by consolidating virtual machines on the available physical machines yet not degrading the makespan. They assessed the scheduling proposed using benchmarks of real-world distributed applications and achieved efficient results.

In [18], energy reduction is achieved by considering only consolidation of network flows in a Fat Tree topology. Correlated flows are identified and assigned to network paths in a greedy way. Link rate adaptation and the switching off of idle switches are the techniques employed to save energy. Simulations, based on real traces of Wikipedia, demonstrated relevant savings in the number of switches used and the algorithm outperforms two others presented in the literature [19, 20].

Although energy consumption has been widely used as a criterion for VM placement, some algorithms do not adopt it [13–16]. In [13], the authors proposed a virtual machine placement algorithm based on tree-like topologies. It considers inter-VM traffic and it views the data center as a set of servers and switches. The placement decisions are based on network graph cuts. However, the solution proposed was not intended to be executed online.

In [14], the authors studied traffic patterns in a production data center and proposed a traffic-aware placement algorithm. Their strategy is based on the knowledge of the network topology as well as on an estimation of the traffic matrix. The algorithm separates clusters of VMs and servers, and then matches these clusters to minimize the communications between VMs.

The work in [15] proposes a network-aware virtual machine placement algorithm, which separates the data center into partitions of servers and virtual machines with these partitions mapped according to a bin-packing problem. The algorithm does not consider forwarding data center traffic to the Internet.

The authors of [16] proposed two VM placement algorithms for the Portland network architecture. These algorithms aim to allocate communicating virtual machines in physical proximity to avoid the creation of network bottlenecks. One algorithm was designed for rapid placement of closely located VMs, while the other tries to identify network regions that can best host the VMs and then, using the first algorithm, maps these VMs on the servers. Such an approach can reduce the intensity of traffic in the links of top-level switches.

Other approaches for the migration of VMs take advantage of the data center network topology [21, 22]. Migration algorithms can balance the load among servers by transfering VMs from overloaded to underloaded servers while taking into consideration the migration paths in the spanning tree of the network topology [21]. These algorithms are formulated as a packing integer program, with relaxation and rounding techniques for fractional solutions tailored to unimodular constraints. Such a solution relieves the load on overloaded servers and yet satisfies edge constraints of the spanning tree.

The S-CORE [22] VM live migration scheme adopts a distributed approach for collecting information about network flow for making decisions about VM migration. S-CORE alleviates congestion from the core layers of a data center in a three tier topology and can be implemented as a plugin in Xen hypervisor.

The algorithm proposed in this paper considers an arrival model for groups of virtual machines as well as their pattern of communications. Moreover, the selection of paths is performed jointly with the decision to mapping VMs onto servers, in a way to minimize the use of switches needed in certain areas of the data center.

Table 1 presents a comparison of the approaches addressed and the algorithm proposed, in relation to energy and network awareness, VM group-based arrival process and both internal (between VMs) and

Table 1 Comparison of related papers

Approach	Network awareness	Energy awareness	Arrival in groups	Internal traffic	External traffic	Flow path allocation
[6]	Yes	Yes	No	No	Yes	No
[7]	Yes	Yes	No	No	Yes	No
[8]	Yes	Yes	Yes	Yes	Yes	No
[9]	No	Yes	No	No	No	No
[10]	No	Yes	No	No	No	No
[11]	No	Yes	No	No	No	No
[12]	No	Yes	Yes	Yes	No	No
[18]	Yes	Yes	No	Yes	Yes	Yes
[13]	Yes	No	No	Yes	No	No
[14]	Yes	No	No	Yes	No	No
[15]	Yes	No	No	Yes	No	No
[16]	Yes	No	Yes	Yes	No	Yes
This paper	Yes	Yes	Yes	Yes	Yes	Yes

external (to the Internet) traffic. The proposed algorithm has several of the features of other algorithms in the literature and it is the only one to consider arrival of groups of VMs and both internal and external traffic to the data center, deciding on path allocation for these network flows.

3 Proposed Algorithm

In this paper, the data center network is modeled as a graph, with vertices representing servers and switches and edges representing links. Requests for allocation of groups of VMs are also modeled as a graph, with vertices representing virtual machines and edges representing the flow between pairs of VMs. The notation presented in Table 2 will be used to present the algorithm.

The topology of the data center network defines the connectivity of servers and, consequently, the patterns of communication. Communication between two virtual machines placed on different servers can take longer and consume more energy in less connected data center networks. Most of the current data centers topologies are hierarchical. They generally follow the model proposed in [17], although other hierarchical topologies (and, in some cases, recursive ones), such as Fat-tree [23], BCube [24] and DCell [25] have been proposed to enhance scalability.

The aim of the algorithm proposed here, entitled Topology-aware Virtual Machine Placement (TAVMP), is to position groups of communicating virtual machines in small areas of a hierarchical data center network so that few switches are needed to serve the network flows. In order to achieve this goal, the data center is divided in hierarchical areas containing switches and servers.

In TAVMP, a parameter $j \geq 0$ is used to describe the level of an area in the hierarchical topology. For example, the Fat-tree topology (Fig. 1) has three levels. The racks are at level 0, POD's (groups of racks) at Level 1, and the whole data center (a set of POD's), at the highest level, Level 2.

TAVMP has a recursive approach and works as follows. It receives as input the VM group and the data center topology and divides the topology into smaller areas, using an auxiliary algorithm to find subgraphs (SUB). Recursion is then performed for each area, and, when the lowest level area is reached, the placement decision is made, i.e., which servers will be used to host the virtual machines, as well as the network paths for the flows. This placement decision is made by another algorithm, entitled Placement in Current Area (PCA), which evaluates areas for placement of groups of VMs. When the recursion finishes, there may be various options for the placement of the VMs. The area chosen is the one which involves the minimum power consumption. Furthermore, if none of the areas are suitable for accommodating the group of VMs due to lack of resources, PCA is then applied to areas on the next higher level area and, if necessary, on the data center level. When TAVMP decides on the

Table 2 Notation used in this paper

Notation	Description
$\mathcal{D} = (\mathcal{E}, \mathcal{L})$	graph representing the data center
$\mathcal{E} = \mathcal{H} \cup \mathcal{S}$	$\mathcal{H}$ is the set of servers/hosts and $\mathcal{S}$ is the set of switches
$\mathcal{L}$	links representing the physical connections
$availableMIPS(h), h \in \mathcal{H}$	available processing resources (in MIPS) of server h
$availableRAM(h), h \in \mathcal{H}$	available memory of server h
$availableBW(l), l \in \mathcal{L}$	available bandwidth in link l
$paths(h1, h2, length)$	function that returns all the paths between servers $h1$
$h1, h2 \in \mathcal{H}, length \in \mathbb{N}$	and $h2$ with length $length$
$pathsPair(h1, h2, \mathcal{D})$	function that returns all the paths between servers $h1$
$h1, h2 \in \mathcal{H}$	and $h2$ in the area of the data center represented by $\mathcal{D}$
$pathsToInternet(h)$	function that returns all the shortest paths between
$h \in \mathcal{H}$	server h and the switch connected to the Internet
$\mathcal{G} = (\mathcal{V}, \mathcal{F})$	graph representing a VM group
$\mathcal{V}$	the set of virtual machines
$\mathcal{F}$	the set of network flows between the VMs
$requestedMIPS(v), v \in \mathcal{V}$	demanded processing resources (in MIPS) by virtual machine v
$requestedRAM(v), v \in \mathcal{V}$	demanded memory by virtual machine v
$requestedBW(f), f \in \mathcal{F}$	demanded bandwidth by flow f
$adj(v), v \in \mathcal{V}$	the adjacent vertices of v, i.e., the VMs that are connected to v
$flow(v1, v2), v1, v2 \in \mathcal{V}$	function that returns the flow $f \in \mathcal{F}$ between $v1$ to $v2$
$requestedBWInternet(v)$	requested bandwidth for the flow between v and the Internet.
$v \in \mathcal{V}$	The value is 0 if v does not communicate with the Internet
$suitable(h, v)$	function that is true if the host h is suitable for the virtual
$h \in \mathcal{H}, v \in \mathcal{V}$	machine v, i.e., if $availableMIPS(h)$ and $availableRAM(h)$ are, respectively, greater or equals to $requestedMIPS(v)$ and $requestedRAM(v)$
$host(v), v \in \mathcal{V}$	function that returns the chosen host $h \in \mathcal{H}$ to place v
$place(v, h), v \in \mathcal{V}$, $h \in \mathcal{H}$	function that chooses server h to host the VM v
$energy(v, h), v \in \mathcal{V}$, $h \in \mathcal{H}$	function that estimates the power increase for server h to host the VM v according to the adopted energy model

final placement for the virtual machines of a group, these VMs are instantiated and the flows are allocated, updating the network status.

TAVMP is presented in Algorithm 1. The input is the data center network, the virtual machine group (request) and the highest level of the hierarchy. In Line 1, subgraphs of the network representing areas at a lower level are generated. All these subgraphs are visited (Line 3) and recursion for each area is conducted (Line 4). The area with the least power consumption increase is selected (Line 5). If all attempts to allocate VMs at the lower level fail, a new attempt is made for the next higher level, as shown in Line 7.

Algorithm 1 TAVMP

Input: $\mathcal{D} = (\mathcal{E}, \mathcal{L})$, $\mathcal{G} = (\mathcal{V}, \mathcal{F})$, j
Output: The placement
1 $subgraphs \leftarrow SUB(\mathcal{D}, j)$
2 if $subgraphs \neq \emptyset$ **then**
3 $\quad \forall l \in subgraphs$
4 $\quad\quad placement \leftarrow TAVMP(l, j-1)$
5 $\quad$ Choose *placement* with smallest power increase
6 if *No placement was successful* **then**
7 $\quad placement \leftarrow PCA(\mathcal{D}, \mathcal{G})$
8 return *placement*

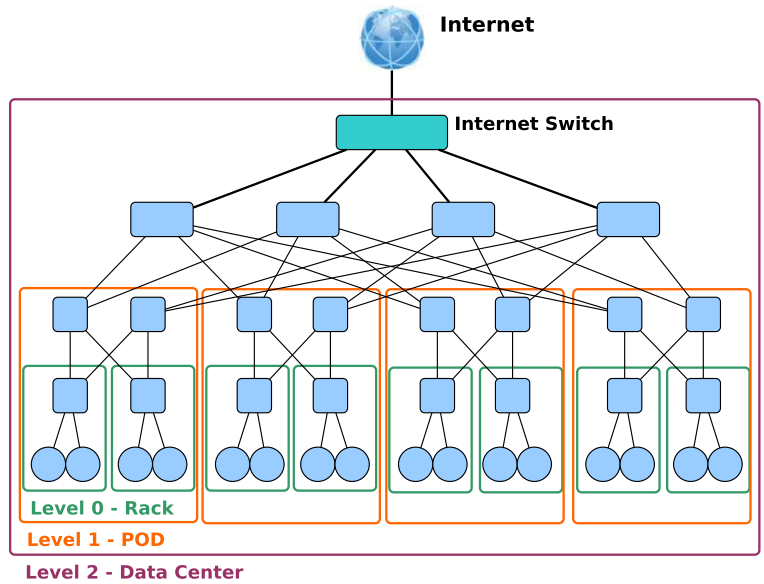


Fig. 1 Fat-tree architecture divided into levels. Each circle is a physical server, and each rectangle a switch

Algorithm 2 presents the PCA algorithm. The input to this algorithm is the graph representing an area in the topology and the group of VMs. The idea is to choose the servers with resources available to support the computational demands of the VMs (Line 4), but which will minimize the increase in total power consumption. Moreover, these servers must be connected by links with enough bandwidth to support the communication demands for the group. In the beginning, the algorithm marks all the virtual machines as unassigned. When a server is chosen to host a VM, it is marked as assigned. One VM at a time is placed, and all the servers are analyzed in order to find the one that minimizes energy consumption.

The PCA algorithm not only places the VMs on servers, but also decides the paths to accommodate the flows. A path is chosen to minimize the number of switches visited by a flow. This step considers both the flows between VMs, Lines 5 to 8, and the flows to the Internet, Lines 9 to 11. If the energy increase caused by the choice of server and paths is the lowest found for that area, that server is selected to host the VM, which is then marked as assigned. Although in Lines 14 and 15 a server is chosen to host a VM, this designated hosting server can be changed in future interactions if a more energy efficient placement is found. The PCA algorithm succeeds if all the VMs are placed (Line 16), otherwise, it fails. When successfully, the overall power increase is calculated (Line 17).

Algorithm 2 PCA

Input: $\mathcal{D} = (\mathcal{E}, \mathcal{L}), \mathcal{G} = (\mathcal{V}, \mathcal{F})$
Output: The placement
1 Mark all $v \in \mathcal{V}$ as unassigned
2 $\forall v \in \mathcal{V}$
3 $\forall h \in \mathcal{H}$
4 **if** $suitable(h, v)$ **then**
5 $\forall cv \in adj(v)$
6 **if** cv was assigned **then**
7 $paths \leftarrow pathsPair(h,host(cv),\mathcal{D})$
8 Choose path for $flow(v, cv)$
9 **if** *requestedBWInternet(v)* $\neq 0$ **then**
10 $paths \leftarrow pathsToInternet(h)$
11 Choose path to Internet from *paths*
12 **if** *Needed flows were allocated* **then**
13 **if** *energy(v,h) is minimum for v* **then**
14 $place(v, h)$
15 Mark v as assigned
16 **if** All $v \in \mathcal{V}$ *were assigned* **then**
17 **return** *Placement and estimated power increase*
18 **else**
19 **return** *Placement was not possible*

Algorithm 3 presents the SUB algorithm. SUB finds subgraphs in the Fat-tree topology. However, TAVMP may accept other topologies as long as SUB is changed to divide the subgraphs representing areas in the given topology. The input to SUB is the area to be divided and its level, and its output is a set of

subgraphs. If the algorithm is at the lowest level, the SUB algorithm finishes (Line 3). Otherwise, the algorithm builds new subgraphs using a server not represented in any generated subgraph (Line 6). In Line 8, the paths between this server and the others in the same subgraph are defined and then included in the current subgraph (Line 10). For the Fat-tree topology, the path length is $2j$. For example, in Fig. 1, for $j = 2$, the length is 4. If we choose the leftmost server, the resulting paths would lead to the other three servers of the first POD, including all the four switches. The algorithm finishes when all the servers are represented in some subgraph.

Algorithm 3 SUB

```
   Input: D = (E, L), j
   Output: The set of subgraphs
 1 resultSet ← ∅
 2 if j = 0 then
 3     return resultSet
 4 Mark all h ∈ H as unvisited
 5 while There exists an unvisited h1 ∈ H do
 6     newSubgraph ← ({h1}, ∅)
 7     ∀ unvisited h2 ∈ H
 8         paths ← paths(h1, h2, 2 · j)
 9         ∀path ∈ paths
10             Add all d ∈ D of path to newSubgraph
11     Mark all h ∈ H of newSubgraph as visited
12     Add newSet to resultSet
13 return resultSet
```

The computational complexity of the SUB algorithm, *paths*, *pathsPair* and *pathsToInternet* depends on the network topology. Nevertheless, for TAVMP, they can be computed beforehand, since the data center topology does not change over time. Thus, the computational complexity for such computation is $O(1)$. The complexity of TAVMP in conjunction with SUB will be derived here. The recursion in TAVMP basically iterates over all the data center servers. In the worst case scenario, all the levels are analyzed. If J is the number of topology levels, TAVMP iterates over all servers J times. Each host is evaluated for each virtual machine in the group. Finally, for each host, all the flows from that VM are analyzed. For each VM, there are at most $|\mathcal{V}|$ flows, which corresponds to the existence of flows to all the other VMs in the group as well as to the Internet. Therefore, the complexity depends on $J \cdot |\mathcal{H}| \cdot |\mathcal{V}| \cdot |\mathcal{V}|$. Since the network topology is fixed and J is a constant, the computational complexity of TAVMP algorithm is $O(|\mathcal{H}||\mathcal{V}|^2)$.

4 Performance Evaluation

The performance of the TAVMP was compared to that of the algorithm presented in [9], hereinafter called the Power Aware Best Fit Decreasing algorithm (PABFD), and to the performance of a Round Robin algorithm (ROUND). PABFD is a well known energy-aware algorithm found in the literature, and Round Robin has been used to represent a typical load balancing strategy. The algorithm in [6] was not considered since it considers queues at the switches and represents a different topology.

4.1 Simulation Settings

For the evaluation of the proposed algorithm, the Cloudsim Simulator [26] was used to simulate the placement of virtual machines. The simulator was extended by including the data center network topology and an energy consumption model. Results were obtained performing 100 different executions for each point in the graph and using a 95 % confidence interval derived by the independent replication method.

Fat-tree topology was used in the simulations as in [23], with the only difference being that a switch, entitled Internet switch, is connected to the core switches, as illustrated in Fig. 1. All rack, aggregation and core switches are commodity switches, each containing forty-eight 1 Gigabit ethernet (GE) ports and four 10 GE ports. The Internet switch contains 128 10 GE ports, and each core switch is connected to it.

Servers can be of two different types and four different configurations of virtual machines were considered in order to simulate a realistic environment (Table 3). The type of server and VM is uniformly distributed among those described in Table 3. The CPU usage of the VMs is taken from the data set in [27].

In the simulations, different sizes of the data center were evaluated by varying the Fat-tree parameter K between 10 and 16. The numbers of machines and switches (the Internet switch is ignored) for each value of K are: for $K = 10$, there are 250 servers and 125 switches; for $K = 12$, 432 servers and 180 switches; for $K = 14$, 686 servers and 245 switches; for

Table 3 Configuration of VM and data center physical servers

Configuration	MIPS	RAM (Mb)	CPU cores
Server Hp ProLiant Ml110 G4 Xeon 3040	1860	4096	2
Server Hp ProLiant Ml110 G5 Xeon 3075	2660	4096	2
VM Instance 1	2500	870	1
VM Instance 2	2000	1740	1
VM Instance 3	1000	1740	1
VM Instance 4	500	613	1

$K = 16$, 1024 servers and 320 switches. In each simulation, virtual machines are placed at the beginning of the simulation, creating a data center occupancy of 50 %.

4.2 Energy Consumption Model

A wide range of factors influences the energy consumption of data centers. The consumption of servers and switches mainly depends on the CPU processing and on the network load, respectively.

We adopt the server consumption model used in [27]. Whenever a server is idle, its consumption is about 70 % of the consumption under full load, a consumption which can be described as linear in relation to the CPU load. When no workload is being processed, a server can be switched to a low consumption mode, thus saving energy. Although a linear function can be employed, we base our model on a real data set of the SPEC power benchmark,[1] interpolating the measured power for each CPU usage level according to the current processing load, which is associated with the VM load hosted.

The energy model for switches [28] is calculated by considering three components: the switch chassis, line cards and ports. The following formula expresses this model:

$$P_{switch} = P_{chassis} + n_{lc} P_{lc} + \sum_{i=0}^{r} n_{r_i} \cdot P_{r_i}$$

P_{switch} is the total power consumed by a switch. $P_{chassis}$ is the fixed power for maintaining it powered on; P_{lc} is the power consumed by each line card in use and n_{lc} is the number of line cards. Each r_i is a potential transmission rate; n_{r_i} and P_{r_i} are the number of ports transmitting at rate r_i and the power used by a port transmitting at rate r_i, respectively.

Table 4 shows the power consumption values used in the paper. The network status is periodically analyzed, and idle servers and switches are powered off.

4.3 Traffic Model

Considering the single arrival of a virtual machine is not realistic because user perform computation by requesting groups of virtual machines. For example, in applications employing MapReduce, a group of virtual machines is in charge of processing the same workload, and the VMs communicate with each other, creating flows in the data center network. Each group has a variable number of VMs and each VM can produce network flows, either connecting a pair of virtual machines or a virtual machine to the Internet.

According to [29], the VM arrival and departure processes on different time scales exhibit self similarity. We model them on a 1-minute time scale, as suggested in [29], and use the generator described in [30] to create the self similar series, using the parameters described in Table 5.

Since a group of VMs departs at the same time, we match the generated number of VMs leaving the data center (departure process) with a group of the same size in execution to identify those leaving the data center. For the inter VM traffic, we generate the values of transmission rates according to the suggested in [13]. In each group, one VM has a flow directed to the Internet and there is a pre-defined probability that two VMs communicate. If there is a flow between a pair of

[1] http://www.spec.org/power_ssj2008/

Table 4 Power consumption of data center equipment

Switches Power Consumption (W)											
Type of switch	$P_{chassis}$		P_{lc}		P_r						
Rack, aggregation or core	146		Included in Chassis		0.42						
Internet	1558		1212		27						
Servers Power Consumption (W)											
	CPU load (%)										
Type of server	0	10	20	30	40	50	60	70	80	90	100
Hp ProLiant Ml110 G4 Xeon 3040	86	89.4	92.6	96	99.5	102	106	108	112	114	117
Hp ProLiant Ml110 G5 Xeon 3075	93.7	97	101	105	110	116	121	125	129	133	135

VMs, a Gaussian distribution is used to generate the transmission rate. Table 5 summarizes the values used for generating sizes of groups in the arriving/departing series, as well as traffic demands. Six scenarios were generated, identified by S and T: S is the size of the groups (MG or LG) and T is the traffic demand (LT, MT or HT).

4.4 Numerical Evaluation

In the simulation, the blocking ratio and energy efficiency were assessed to evaluate the performance of TAVMP. The blocking ratio is the percentage of VMs that were not placed in relation to the total number of requests for virtual machine allocations and occurs whenever the algorithm can not find servers on which to place group of VMs.

The blocking ratios produced by TAVMP, PABFD and ROUND are displayed in Fig. 2. To facilitate the visualization, Fig. 2a shows a range of blocking ratio up to 10 %, while Fig. 2b and c use a range of 20 % and 100 %, respectively. The Round Robin algorithm does not lead to optimized solutions since it tends to use as many different servers as possible to host

Table 5 Parameters used for the evaluated scenarios. **M** stands for mean, **SD** for standard deviation and **H** for the Hurst parameter

Parameter	Model	
Medium groups (MG)	Self-similar series M: 10; SD: 5; H: 0.7	
Large groups (LG)	Self-similar series M: 20; SD: 10; H: 0.7	
Low-intensive traffic (LT)	Internet flow: Gaussian M: 2 Mbps; SD: 0.2 Mbps	Pair flow: Gaussian M: 5 Mbps; SD: 0.5 Mbps Probability: 0.75
Medium-intensive traffic (MT)	Internet flow: Gaussian M: 4 Mbps; SD: 0.4 Mbps	Pair flow: Gaussian M: 10 Mbps; SD: 1 Mbps Probability: 0.75
High-intensive traffic (HT)	Internet flow: Gaussian M: 10 Mbps; SD: 1 Mbps	Pair flow: Gaussian M: 25 Mbps; SD: 5 Mbps Probability: 0.75

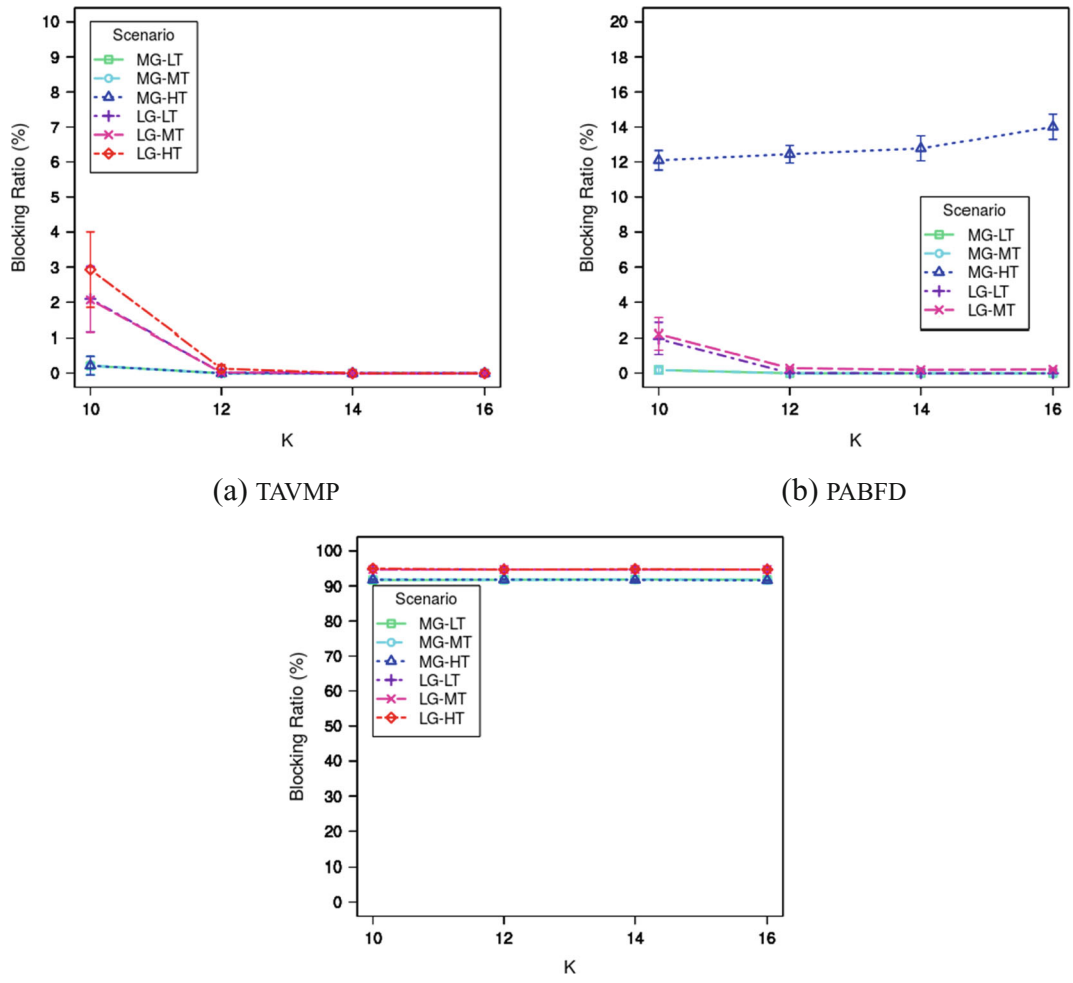


Fig. 2 Blocking ratio produced by the evaluated algorithms as a function of the data center size for the Hurst (H) parameter equals to 0.7

groups of virtual machines, placing them on different areas of the data center and using many network links.

The algorithm PABFD provides a similar performance when compared to TAVMP, except for the LG-HT and MG-HT scenarios. A huge difference in blocking occurs for the LG-HT scenario, since almost all the virtual machines are blocked by PABFD, while TAVMP manages to maintain the blocking ratio close to 3 % for $K = 10$ and below 1 % for K between 12 and 16. These scenarios represent heavy network traffic. Since TAVMP aims at restricting the area of the data center, flows between pairs of virtual machines tend to use fewer switches. Their links are thus seldom fully occupied, allowing the network to serve more requests.

Energy efficiency is defined in this paper as the total energy consumed in the data center, including that by both physical servers and switches, divided by the number of VMs accepted. The results for different data center sizes and configurations are shown in Fig. 3. The results given by ROUND are not shown since its blocking ratio was unacceptable. The use of LG-HT PABFD led to wide error bars due to the variation in the number of VMs accepted, as a consequence of the arrival of large groups.

The MG-HT and LG-HT blocking ratios in Fig. 2b and the energy efficiency in Fig. 3 are correlated. The PABFD algorithm consolidates the virtual machines on fewer servers; however, in the MG-HT and LG-HT

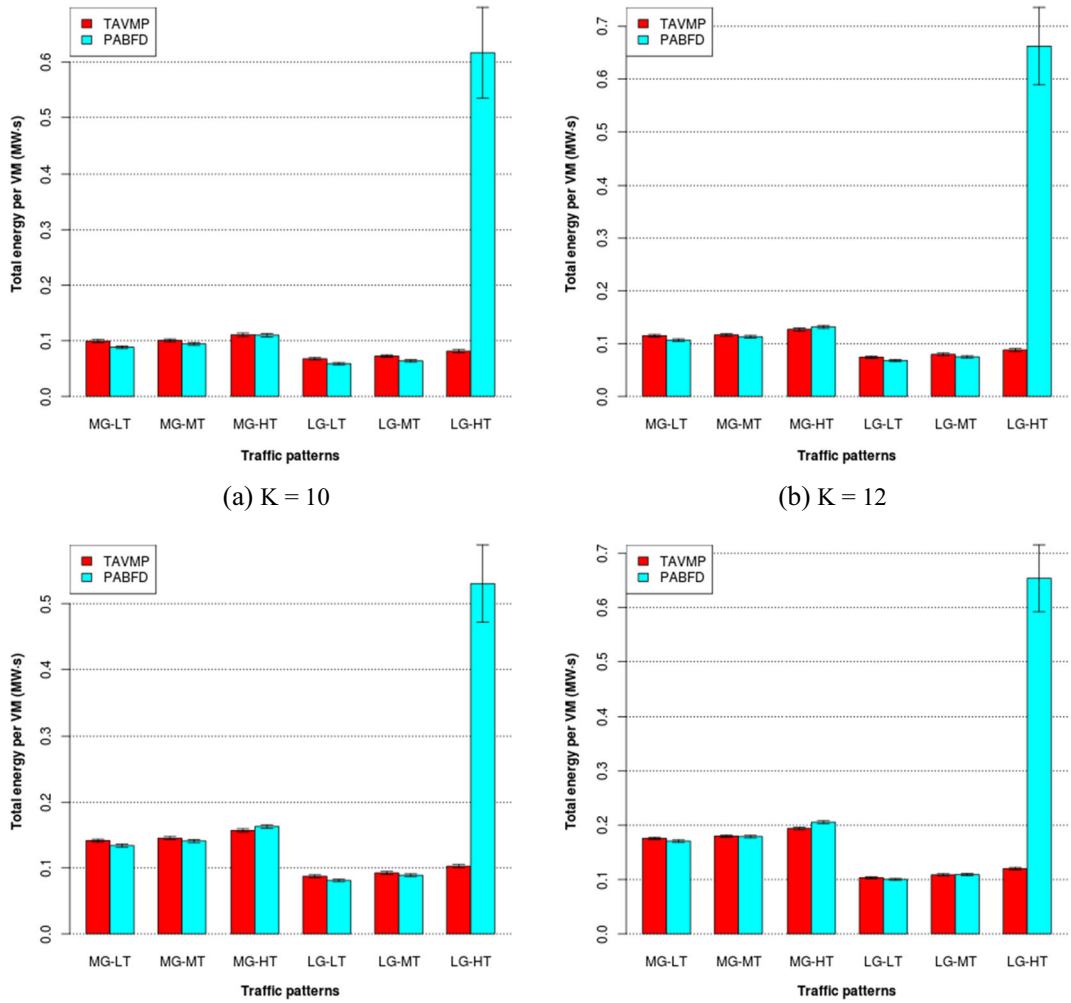
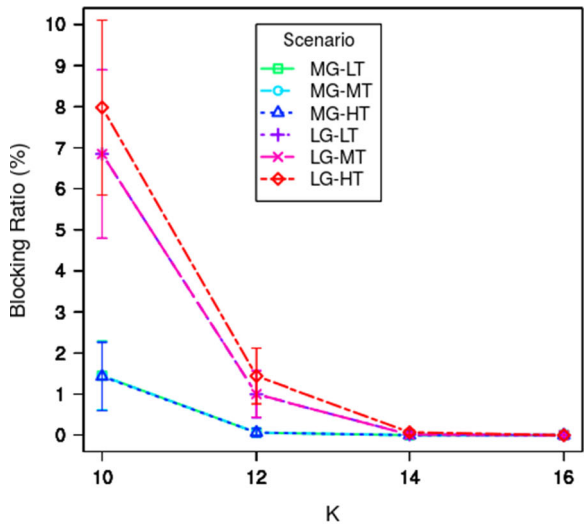


Fig. 3 Energy per virtual machine for different sizes of data center

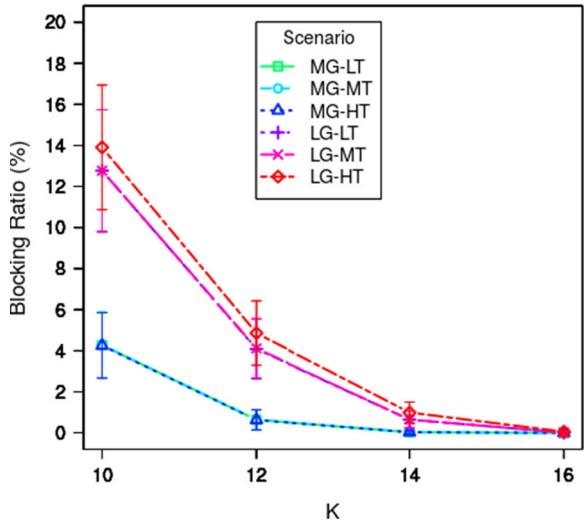
scenarios, it is not always possible due to the formation of network bottlenecks by the heavy traffic. Even with processing resources available, virtual machines may not be placed, thus causing blocking. This situation is less likely to occur when using TAVMP, since communicating VMs are closer together and use fewer network switches and less bandwidth, leaving the remaining links available to host new requests. As a result, for heavy traffic demands, TAVMP outperforms PABFD in almost all scenarios, specially when large groups create several network flows.

The Hurst parameter of the arrival process was varied to evaluate the impact of the arrival process self similarity and the performance of the algorithms. An increase in the Hurst parameter value implies longer periods of consecutive arrivals of groups of VMs (in a similar way, the increase of the H value of a packet flow implies on longer bursts of packets) which yield to higher blocking, especially for large groups of VMs and small data centers.

Figures 2a, and 4a, b display the blocking ratio for H equals to 0.7, 0.8, and 0.9, respectively, for the TAVMP algorithm. For $K = 10$ and medium size groups, the blocking ratio varied from 0.25 % to 4 % while for LG-LT and LG-MT from 2 % to 13 % and for LG-HT it varied from 3 % to 14 %. Such high

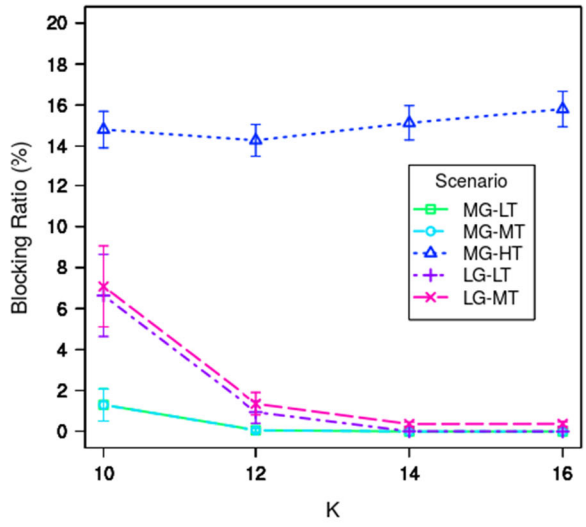


(a) H = 0.8

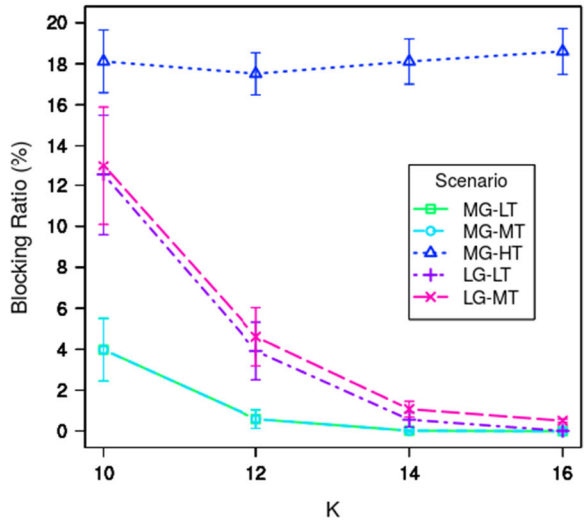


(b) H = 0.9

Fig. 4 Blocking ratio produced by the algorithm TAVMP as a function of the data center size for different values of the Hurst (H) parameter



(a) H = 0.8

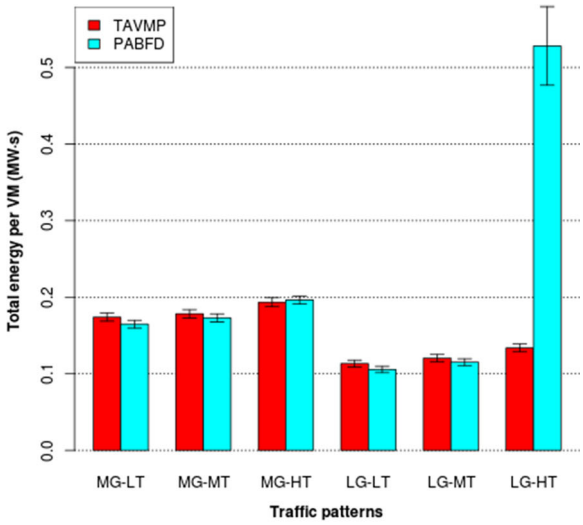


(b) H = 0.9

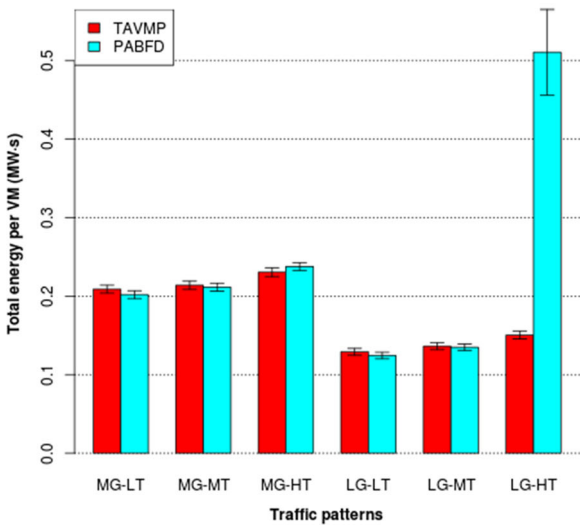
Fig. 5 Blocking ratio produced by the algorithm PABFD as a function of the data center size for different values of the Hurst (H) parameter

blocking when $H = 0.9$ can imply violation of service level agreements, incurring in financial losses to service providers. For data center size of $K = 12$ the blocking ratio doubles for large groups of VMs and quadruplicates for medium size groups.

Figure 5 shows the blocking ratio for the PABFD algorithm and different values of the Hurst parameter. The blocking ratio for LG-HT was close to 100 % and it was not displayed in Fig. 5 for better visualization of the other curves. The PABFD algorithm does not minimize the path length between two communicating VMs, leading to a higher consumption of link and switch resources as well as a faster exhaustion of available resources. As a consequence, the blocking ratio of groups of VMs with high intensive traffic increases considerably. While for large groups with intensive traffic (LG-HT) the data center becomes inaccessible, for medium groups and intensive traffic (MG-HT) the blocking ratio reaches unacceptable values, such as 12 %, 15 %, and 18 %, for H equals to 0.7, 0.8, and 0.9,



(a) K = 14



(b) K = 16

Fig. 6 Energy per virtual machine for data center sizes $K = 14$ and $K = 16$ and $H = 0.9$

respectively. Such blocking ratio is much higher than those produced by TAVMP algorithm, which were at most 0.25 %, 1.5 % and 4 %, for H equals to 0.7, 0.8, and 0.9, respectively.

The error bars for large groups in Figures 4 and 5 are wide. As mentioned earlier, this happens due to the variation of virtual machine acceptance ratios of large groups. The increase of the Hurst parameter does not decrease the blocking ratio given by ROUND which was close to 100 %.

Figure 6 shows the energy consumption per VM for data center sizes $K = 14$ and $K = 16$, and $H = 0.9$. A slight increase in energy consumption occurred and, consequently, a slight decrease in energy efficiency. This happens since long bursts of groups of VMs tend to consume larger amounts of resources in short period.

4.5 Time per request

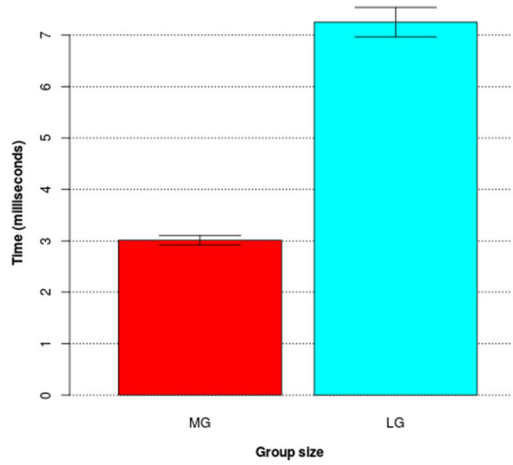
In order to assess the scalability of the proposed algorithm, we present an evaluation of execution time of TAVMP for different sizes of data centers and requests. In addition to the scenarios evaluated in the previous subsection, we have included the scenario of larger groups and data center. These scenarios are shown in Table 6. The evaluation was performed using a Hurst parameter value equal to 0.7 and HT traffic, since other variation of traffic and Hurst parameter does not significantly affect the execution time.

XG groups were assessed only for Fat-tree topology with $K = 40$ given their large size. Results for the time evaluation are presented in Fig. 7. The execution time for data centers size up to $K = 16$ is negligible. The execution time for the largest data center and medium and large groups is less than 1 second, which is quite acceptable. On the other hand, scenarios with extra large groups tend to demand a few seconds more to be processed. Such overhead is still acceptable, since these large groups of tasks demand a longer processing time, and the few additional seconds do not impact significantly on the response time.

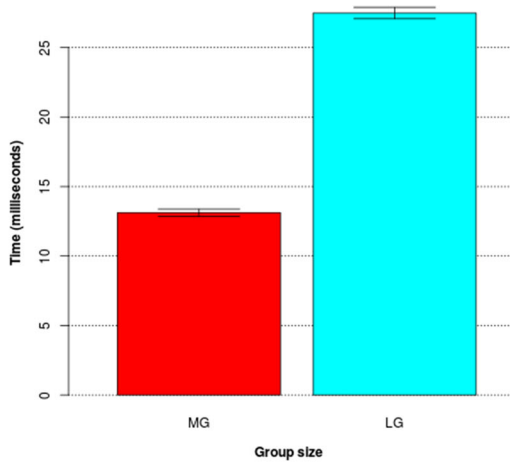
Groups with over 100 VMs require an execution time of around 3 minutes, which is generally not accepted, although, if such requests are not very frequent, the algorithm can still be used. Therefore, TAVMP is scalable and feasible for employment in real data centers.

Table 6 Additional parameters used for the evaluation of TAVMP time per request. **M** stands for mean, **SD** for standard deviation and **H** for the Hurst parameter

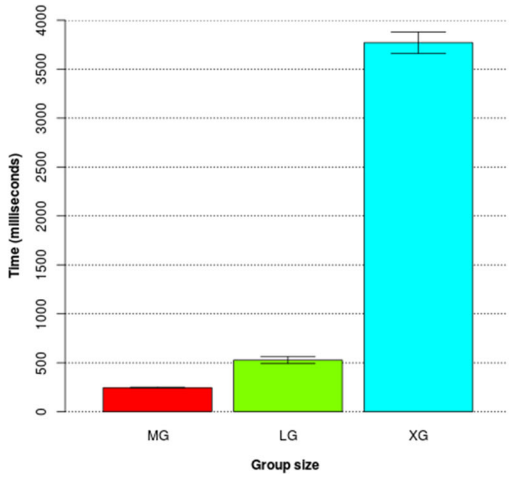
Parameter	Description
Extra large groups (XG)	Self-similar series M: 50; SD: 25; H: 0.7
Fat-tree $K = 40$	16000 servers 2000 switches



(a) K = 10



(b) K = 16



(c) K = 40

Fig. 7 TAVMP execution time for data center sizes $K = 10$, $K = 16$ and $K = 40$

5 Conclusion

This paper has introduced the Topology-aware Virtual Machine Placement algorithm designed to consolidate groups of communicating virtual machines in small areas of the data center. Its performance was assessed using simulations, and it was compared with two other algorithms. Results show that the proposed algorithm accepts more virtual machines without impacting energy efficiency. This is due to the consolidation of the flows in small areas, thus minimizing the use of network resources and avoiding network bottlenecks. The employment of the TAVMP algorithm helps to reduce the blocking ratio of requests, which is essential for cloud providers to provide high accessibility in service-level agreements. The algorithm presented does not consider different classes of service. As a consequence, it accepts a higher number of less demanding groups of virtual machines than accepts more demanding groups. Moreover, the proposed algorithm is scalable and can be used for large data centers with thousands of servers and switches.

As future work, the TAVMP algorithm can be evaluated for other network topologies and traffic patterns. We are currently working on extending TAVMP for the problem of load balancing in federated data center scenarios. In such a scenario, data centers are typically heterogeneous and their cooling infrastructure plays

an important role in energy efficient operations. The employment of the Power Usage Effectiveness metric (PUE), defined by the Green Grid consortium [31], is quite useful in the evaluation of the possible impact of virtual machine migration between data centers can have on energy efficient operations. Furthermore, TAVMP can be modified to adopt different policies for the selection of network paths to help avoiding the formation of bottlenecks. The design of novel algorithm for the provisioning of differentiated services is also a promising direction.

Acknowledgments The authors would like to acknowledge the financial support provided by the grant 2014/01154-4, São Paulo Research Foundation (FAPESP), and 131821/2013-0, National Council for Scientific and Technological Development (CNPq).

References

1. Mell, P., Grance, T.: The NIST Definition of Cloud Computing. Tech. Rep. (2009)
2. Szabo, C., Sheng, Q., Kroeger, T., Zhang, Y., Yu, J.: Science in the cloud: Allocation and execution of data-intensive scientific workflows. J Grid Comput **12**(2), 245–264 (2014)
3. Koomey, J.: Growth in data center electricity use 2005 to 2010. Analytics Press, Oakland (2011)
4. McKinsey & Company: Energy efficiency: a compelling global resource (2010)
5. da Silva, R.A.C., da Fonseca, N.L.S.: Algorithm for the placement of groups of virtual machines in data centers. In: 2015 IEEE International Conference on Communications (ICC), pp. 6080–6085 (2015)
6. Kliazovich, D., Bouvry, P., Khan, S.: Dens: data center energy-efficient network-aware scheduling. In: Green computing and communications (GreenCom), 2010 IEEE/ACM Int'l Conference on Int'l Conference on Cyber, Physical and Social Computing (CPSCom), pp. 69–75 (2010)
7. Kliazovich, D., Arzo, S., Granelli, F., Bouvry, P., Khan, S.: e-stab: energy-efficient scheduling for cloud computing applications with traffic load balancing. In: Green computing and communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing, pp. 7–13 (2013)
8. Pascual, J., Lorido-Botrán, T., Miguel-Alonso, J., Lozano, J.: Towards a greener cloud infrastructure management using optimized placement policies. J. Grid Comput., 1–15 (2014)
9. Beloglazov, A., Abawajy, J., Buyya, R.: Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Futur. Gener. Comput. Syst. **28**(5), 755–768 (2012). special Section: Energy efficiency in large-scale distributed systems
10. Lago, D.G.d., Madeira, E.R.M., Bittencourt, L.F.: Power-aware virtual machine scheduling on clouds using active cooling control and dvfs. In: Proceedings of the 9th International Workshop on Middleware for Grids, Clouds and e-Science, ser. MGC '11, pp. 2:1–2:6. ACM, New York (2011)
11. Srikantaiah, S., Kansal, A., Zhao, F.: Energy aware consolidation for cloud computing. In: Proceedings of the 2008 conference on Power aware computing and systems, ser. HotPower'08, 10–10 (2008)
12. Ebrahimirad, V., Goudarzi, M., Rajabi, A.: Energy-aware scheduling for precedence-constrained parallel virtual machines in virtualized data centers. J. Grid Comput., 1–21 (2015)
13. Biran, O., Corradi, A., Fanelli, M., Foschini, L., Nus, A., Raz, D., Silvera, E.: A stable network-aware vm placement for cloud systems. In: 2012 12th IEEE/ACM International Symposium on Cluster, cloud and grid computing (CCGrid), pp. 498–506 (2012)
14. Meng, X., Pappas, V., Zhang, L.: Improving the scalability of data center networks with traffic-aware virtual machine placement. In: INFOCOM, 2010 Proceedings IEEE, 1–9 (2010)
15. Dias, D., Costa, L.: Online traffic-aware virtual machine placement in data center networks. In: Global information infrastructure and networking symposium (GIIS), 2012, pp. 1–8 (2012)
16. Georgiou, S., Tsakalozos, K., Delis, A.: Exploiting network-topology awareness for vm placement in iaas clouds. In: 2013 Third International Conference on Cloud and Green Computing (CGC), 151–158 (2013)
17. Cisco Data Center Infrastructure 2.5 Design Guide, (2007)
18. Wang, X., Yao, Y., Wang, X., Lu, K., Cao, Q.: Carpo: Correlation-aware power optimization in data center networks. In: INFOCOM, 2012 Proceedings IEEE, 1125–1133 (2012)
19. Abts, D., Marty, M.R., Wells, P.M., Klausler, P., Liu, H.: Energy proportional datacenter networks. In: Proceedings of the 37th Annual International Symposium on Computer Architecture, ser. ISCA '10, pp. 338–347. ACM, New York (2010)
20. Heller, B., Seetharaman, S., Mahadevan, P., Yiakoumis, Y., Sharma, P., Banerjee, S., McKeown, N.: Elastictree: saving energy in data center networks. In: Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation, ser. NSDI'10, p. 17. USENIX Association, Berkeley (2010)
21. Jain, N., Menache, I., Naor, J.S., Shepherd, F.B.: Topology-aware vm migration in bandwidth oversubscribed datacenter networks. In: Proceedings of the 39th International Colloquium Conference on Automata, Languages, and Programming - Volume Part II, ser. ICALP'12, pp. 586–597. Springer, Berlin (2012)
22. Tso, F.P., Hamilton, G., Oikonomou, K., Pezaros, D.: Implementing scalable, network-aware virtual machine migration for cloud data centers. In: 2013 IEEE Sixth International Conference on Cloud Computing (CLOUD), 557–564 (2013)
23. Al-Fares, M., Loukissas, A., Vahdat, A.: A scalable, commodity data center network architecture. SIGCOMM Comput. Commun. Rev. **38**(4), 63–74 (2008)
24. Guo, C., Lu, G., Li, D., Wu, H., Zhang, X., Shi, Y., Tian, C., Zhang, Y., Lu, S.: Bcube: a high performance,

server-centric network architecture for modular data centers. SIGCOMM Comput. Commun. Rev. **39**(4), 63–74 (2009)
25. Guo, C., Wu, H., Tan, K., Shi, L., Zhang, Y., Lu, S.: Dcell: a scalable and fault-tolerant network structure for data centers. In: Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication, ser. SIGCOMM '08, pp. 75–86. ACM, New York (2008)
26. Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A.F., Buyya, R.: Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw. Pract. Exper. **41**(1), 23–50 (2011)
27. Beloglazov, A., Buyya, R.: Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. Concurrency and Computation: Practice and Experience **24**(13), 1397–1420 (2012)
28. Mahadevan, P., Sharma, P., Banerjee, S., Ranganathan, P.: Energy aware network operations. In: INFOCOM Workshops 2009, pp. 1–6. IEEE (2009)
29. Han, Y., Chan, J., Leckie, C.: Analysing virtual machine usage in cloud computing. In: 2013 IEEE Ninth World Congress on Services (SERVICES), pp. 370–377 (2013)
30. Paxson, V.: Fast, approximate synthesis of fractional gaussian noise for generating self-similar network traffic. SIGCOMM Comput. Commun. Rev. **27**(5), 5–18 (1997)
31. Belady, C., Rawson, A., Pfleuger, J., Cader, T.: Green grid data center power efficiency metrics: PUE and DCIE. The Green Grid, Tech. Rep. White Paper 6 (2008)