Power-efficient Assignment of Virtual Machines to Physical Machines*

Jordi Arjona Aroca[1,2], Antonio Fernández Anta[1], Miguel A. Mosteiro[3,4], and Christopher Thraves[4]

[1] Institute IMDEA Networks, Madrid, Spain
({jorge.arjona,antonio.fernandez}@imdea.org)
[2] Universidad Carlos III de Madrid, Madrid, Spain
[3] Department of Computer Science, Kean University, Union, NJ, USA
(mmosteir@kean.edu)
[4] GSyC, Universidad Rey Juan Carlos, Spain (cbthraves@gsyc.es)

Abstract. Motivated by current trends in cloud computing, we study a version of the generalized assignment problem where a set of virtual processors has to be implemented by a set of identical processors. For literature consistency we say that a set of virtual machines (VMs) is assigned to a set of physical machines (PMs). The optimization criteria is to minimize the power consumed by all the PMs. We term the problem Virtual Machine Assignment (VMA). Crucial differences with previous work include a variable number of PMs, that the VMs cannot be implemented fractionally (i.e., each VM must be assigned to exactly one PM), and a parametric minimum power consumption for each active PM. We show that the VMA problem is NP-hard in the strong sense and we present a VMA offline approximation algorithm. For this VMA protocol, we show the trade-off between the running time and the approximation ratio achieved. Furthermore, restricting the VMA problem to realistic applications, we observe that such protocol is a PTAS[5] for the VMA problem, while there is no FPTAS[6]. Moving to online VMA algorithms, we show upper and lower bounds on the competitive ratio when only 2 PMs are available, lower bounds when some arbitrary number m of PMs are available, and an upper bound when the number of machines is unbounded. We also carry extensive simulations using real-world input such as Google cluster data. To the best of our knowledge, this is the first time the VMA problem is studied for this cost function.

1 Introduction

The current pace of technology developments, and the continuous change in business requirements, may rapidly yield a given proprietary computational

* This work is supported by the National Science Foundation (CCF-0937829, CCF-1114930), Comunidad de Madrid grant S2009TIC-1692, MINECO grant TEC2011-29688-C02-01, and National Natural Science Foundation of China grant 61020106002, MICINN grant Juan de la Cierva.

[5] Polynomial-Time Approximation Scheme

[6] Fully Polynomial-Time Approximation Scheme

platform obsolete, oversized, or insufficient. Thus, outsourcing has recently become a popular approach to obtain computational services. Furthermore, in order to attain flexibility, such service is usually virtualized, so that the user may tune the computational platform to its particular needs. Users of such service need not to be aware of the particular implementation, but only of the specification of the virtual machine they use. This conceptual approach to outsourced computing has been termed *cloud computing*, in reference to the cloud symbol used as an abstraction of a complex infrastructure in system diagrams. Current examples of cloud computing providers include Amazon Web Services [1], Rackspace [5], and Citrix [2].

Depending on what the specific service provided is, the cloud computing model comes in different flavors, such as *infrastructure as a service*, *platform as a service*, *storage as a service*, etc. In each of these models, the user may choose specific parameters of the computational resources provided. For instance, processing power, memory size, communication bandwidth, etc. Thus, in a cloud-computing service platform, various ***virtual machines (VM)*** with user-defined specifications must be implemented by, or ***assigned to***[7], various ***physical machines (PM)***[8]. Furthermore, such a platform must be scalable, allowing to add more PMs, should the business growth require such expansion. In this work, we call this problem the ***Virtual Machine Assignment (VMA)*** problem. (A precise definition is given in Section 1.1.)

From the previous discussion, it can be seen that, underlying VMA, there is some form of bin-packing problem. However, in VMA the number or capacity of PMs (i.e., bins for bin packing) may be increased if needed. The optimization criteria for VMA depends on what the particular objective function sought is. Previous work related to VMA has focused on minimizing the number of PMs used (cf. [12] and the references therein) to minimize energy consumption. However, power consumption is usually superlinear on the ***load*** of a given computational resource [11, 19]. Hence, the use of extra PMs may be more efficient energy-wise than a minimum number of heavily-loaded PMs. On the other hand, the addition of a new PM to the system usually implies some fixed power-consumption increase [9, 19], even if such PM is not loaded. In this work, we combine both power-consumption factors. That is, for some parameters $\alpha > 1$ and $b > 0$, we seek to minimize the sum of the α powers of the PMs loads *plus* the fixed cost b of using each PM.

[7] The cloud-computing literature use instead the term *placement*. We choose here the term assignment for consistency with the literature on general assignment problems.

[8] We choose the notation VM and PM for simplicity and consistency, but notice that our study applies to any computational resource assignment problem, as long as the minimization function is the one modeled here.

The paper is organized as follows. First, in Section 1.1 we include a formal definition of the problem, we overview related work in Section 1.2 and we detail our results in Section 1.3. Some preliminaries are included in Section 2. For the model described, we show that the VMA problem is NP-hard in the strong sense, even if α is fixed, in Section 3. Then, we present an offline VMA approximation algorithm, which in fact is a PTAS for this problem for any *realistic* model of power consumption. We also study the competitiveness of online VMA algorithms (Section 4). For settings where a bounded number m of PMs are available, we show a lower bound on the competitiveness of online algorithms (with respect to an optimal offline assignment). We improve such bound for settings where only 2 PMs are available. We also show upper bounds on the competitiveness of online algorithms for the setting where 2 PMs are available. Finally, we show an upper bound on the competitiveness of online VMA algorithms when the number of VMs is not bounded. We also present and evaluate experimentally some heuristics in Section 5.

1.1 Problem Definition

We define the ***Virtual Machine Assignment (VMA)*** problem as follows:

Input: A set $S = \{s_1, \ldots, s_m\}$ of m identical physical machines. Rational numbers α and b, where $\alpha > 1$ and $b > 0$. A set $D = \{d_1, \ldots, d_n\}$ of n virtual machines. A function $\ell : D \to \mathbb{R}$ that gives the CPU load each virtual machine incurs.
Output: A partition $\pi = \{A_1, \ldots, A_m\}$ of D.
Objective function: Minimize the power consumption given by the function

$$P(\pi) = \sum_{i \in [1,m]: A_i \neq \emptyset} \left(\Big(\sum_{d_j \in A_i} \ell(d_j) \Big)^{\alpha} + b \right).$$

For convenience, we define the following notation. We overload the function $\ell(\cdot)$ to be applied over sets of virtual machines, so that $\ell(A_i) = \sum_{d_j \in A_i} \ell(d_j)$. Also, let us define the function $f(\cdot)$, such that $f(x) = 0$ if $x = 0$ and $f(x) = x^{\alpha} + b$ otherwise. Then, the objective function is to minimize

$$P(\pi) = \sum_{i=1}^{m} f(\ell(A_i)).$$

1.2 Related Work

To the best of our knowledge, previous work on VMA has been only experimental [17, 26, 31, 23] or has focused on different cost functions [16, 18, 7, 12].

First, we provide an overview of previous theoretical work for related assignment problems (storage allocation, scheduling, network design, etc.). The cost functions considered in that work resemble or generalize the power cost function under consideration here. Secondly, we overview related experimental work.

Chandra and Wong [16], and Cody and Coffman [18] study a problem for storage allocation that is a variant of VMA with $b = 0$ and $\alpha = 2$. Hence, this problem tries to minimize the sum of the squares of the machine-load vector. They study the offline version of the problem and provide algorithms with constant approximation ratio. A significant leap was taken by Alon et al. [7], since they present a PTAS for the problem of minimizing the L_p norm of the load vector, for any $p \geq 1$. This problem has the previous one as special case, and is also a variant of the VMA problem when $p = \alpha$ and $b = 0$.

Bansal, Chan, and Pruhs minimize arbitrary power functions for speed scaling in job scheduling [11]. The problem is to schedule the execution of n computational jobs on a *single* processor, whose speed may vary within a countable collection of intervals. Each job has a release time, a processing work to be done, a weight characterizing its importance, and its execution can be suspended and restarted later without penalty. A scheduler algorithm must specify, for each time, a job to execute and a speed for the processor. The goal is to minimize the weighted sum of the flow times over all jobs plus the energy consumption, where the flow time of a job is the time elapsed from release to completion and the energy consumption is given by s^α where s is the processor speed and $\alpha > 1$ is some constant. For the online algorithm *shortest remaining processing time first*, the authors prove a $(3 + \epsilon)$ competitive ratio for the objective of total weighted flow plus energy. Whereas for the online algorithm *highest density first (HDF)*, where the density of a job is its weight-to-work ratio, they prove a $(2 + \epsilon)$ competitive ratio for the objective of fractional weighted flow plus energy.

A generalization of the above problem is studied by Gupta, Krishnaswamy, and Pruhs in [19]. The question addressed is how to assign jobs, *possibly fractionally*, to unrelated parallel machines in an online fashion in order to minimize the sum of the α-powers of the machine loads plus the assignment costs. Upon arrival of a job, the algorithm learns the increase on the load and the cost of assigning a unit of such job to a machine. Jobs cannot be suspended and/or reassigned. The authors model a greedy algorithm that assigns a job so that the cost is minimized as solving a mathematical program with constraints arriving online. They show a competitive ratio of α^α with respect to the solution of the dual program which is a lower bound for the optimal. References to previous work on the particular case of minimizing energy with deadlines can be found in this paper.

Recently, Im, Moseley, and Pruhs studied online scheduling for general cost functions of the flow time, with the only restriction that such function is non-decreasing [21]. In their model, a collection of jobs, each characterized by a release time, a processing work, and a weight, must be processed by a *single* server whose speed is variable. A job can be suspended and restarted later without penalty. The authors show that HDF is $(2+\epsilon)$-speed $O(1)$-competitive against the optimal algorithm on a unit speed-processor, for all non-decreasing cost functions of the flow time. Furthermore, they also show that this ratio cannot be improved significantly proving impossibility results if the cost function is not uniform among jobs or the speed cannot be significantly increased.

Similar cost functions have been considered for the minimum cost network-design problem. In this problem packets have to be routed through a (possibly multihop) network of speed scalable routers. There is a cost associated to assigning a packet to a link and to the speed or load of the router. The goal is to route all packets minimizing the aggregated cost. In [8] and [9] the authors show offline algorithms for this problem that achieve polynomial and poly-logarithmic approximation, respectively, where the cost function is the α-th power of the link load plus a link assignment cost, for any *constant* $\alpha > 1$. The same problem and cost function is studied in [19] (the assignment cost is omitted for clarity). As for the scheduling problem, the authors model a greedy algorithm as solving a mathematical program with constraints arriving online. They show a competitive ratio of α^α with respect to the solution of the dual program which is a lower bound for the optimal.

The experimental work related to VMA is vast and its detailed overview is out of the scope of this paper. Some of this work does not minimize energy [15, 24, 27] or it applies to a model different than ours (VM migration [29, 30], knowledge of future load [25, 30], feasibility of allocation [12], multilevel architecture [26, 29, 22], interconnected VMs [13], etc.). On the other hand, some of the experimental work where minimization of energy is evaluated focus on a more restrictive cost function [33, 22, 34].

In [22], for an energy cost model that is linear, the authors evaluate experimentally the allocation of VMs to clusters following 7 placement policies, some of them included in popular cloud platforms [4, 3]. Namely, Round Robin, Striping, Packing, Load Balancing (free CPU count), Load Balancing (free CPU ratio), Watts per Core, Cost per Core. We adapt 5 of these policies (defined later) to our model and cost function for the purpose of simulations.

In [30], the authors focus on an energy-efficient VM placement problem with two requirements: CPU and disk. These requirements are assumed to change dynamically and the goal is to consolidate loads among servers, possibly using migration at no cost. In our model VMs assignment is based on a CPU require-

ment that does not change and migration is not allowed. Should any other resource be the dominating energy cost, the same results apply for that requirement. Also, if loads change and migration is free, an offline algorithm can be used each time that a load changes or a new VM arrives. In [30] it is shown experimentally that energy-efficient VMA does not merely reduce to a packing problem. That is, to minimize the number of PMs used even if their load is close to their maximum capacity. For our model, we show here that the optimal load of a given server is a function only of the fixed cost of being active (b) and the exponential rate of power increase on the load (α). That is, the optimal load is not related to the maximum capacity of a PM.

1.3 Our Results

In this work, we study offline and online versions of the VMA problem. First, using a reduction from 3-PARTITION, we show that VMA is NP-hard in the strong sense, even if α is constant. This result implies that the VMA problem does not have a fully polynomial-time approximation scheme (FPTAS), even if α is constant. Then, we present a VMA algorithm that achieves an approximation ratio of $(1+\epsilon)^\alpha$ with respect to the power consumption of an optimal assignment, for any constant $\epsilon > 0$. We also show a $O(\min(n,m)(n + g(1/\epsilon) \cdot \log^{O(1)} n))$ upper bound on the running time of such algorithm, where $g(\cdot)$ is some function that grows at least exponentially. We observe that, when α is constant as in any realistic power consumption model, this algorithm is a polynomial-time algorithmic scheme (PTAS) for the VMA problem. Hence, for constant α, we fully characterize the offline version of the VMA problem, since a PTAS is presented and no FPTAS may exist.

Then we move on to online VMA algorithms. That is, we assume that VMs are revealed to the algorithm one by one, and the assignments made by the algorithm are final. First, we show that, when the number of PMs is bounded, no online VMA algorithm is $3^\alpha/(2^{\alpha+2}+\epsilon)$-competitive for any $\epsilon > 0$, and we show a stronger lower bound of $3^\alpha/2^{\alpha+1}$ for a system where only 2 PMs are available. Moving to upper bounds, for a system with only 2 PMs, we present an online algorithm that is optimal when $\ell(D) \leq (b/(2^\alpha - 2))^{1/\alpha}$ and achieves a competitive ratio of at most $\max\{2, (3/2)^{\alpha-1}\}$ otherwise. Finally, for a system where PMs may be added on demand, we present a VMA online algorithm that, if no VM d_i has $\ell(d_i) < x^*$ (where $x^* = (b/(\alpha-1))^{1/\alpha}$) achieves optimal competitive ratio 1. Otherwise, it achieves a $2^{\alpha-1} + x^*/(\sum_{d_i:\ell(d_i)<x^*} \ell(d_i))$ competitive ratio.

The proofs omitted from this document can be found in [10].

2 Preliminaries

The following observations will be used in the analysis. We call ***power rate*** the power consumed per unit of load in a PM. Let x be the load of a PM. Then, its power rate is computed as $f(x)/x$. The load at which the power rate is minimized, denoted x^*, is the ***optimal load***, and the corresponding rate is the ***optimal power rate*** $\rho^* = f(x^*)/x^*$. Using calculus we get:

Observation 1 *The optimal load is:*

$$x^* = (b/(\alpha - 1))^{1/\alpha}.$$

Equivalently, for any $x \neq x^*$, $f(x)/x > \rho^*$.

Lemma 1. *Given an instance of the VMA problem with a set of VMs $D = \{d_1, \ldots, d_n\}$, any solution $\pi = \{A_1, \ldots, A_m\}$ where $\sum_{d \in A_i} d \neq x^*$ for some $i \in [1, m]$, satisfies*

$$P(\pi) > \rho^* \ell(D) = \rho^* \sum_{d \in D} \ell(d).$$

Proof. The total cost of π is $\sum_{i \in [1,m]} f(\ell(A_i))$ which, from Observation 1, is at least

$$\sum_{i \in [1,m]: A_i \neq \emptyset} \ell(A_i)\rho^* = \rho^* \sum_{i \in [1,m]: A_i \neq \emptyset} \sum_{d \in A_i} \ell(d) = \rho^* \sum_{d \in D} \ell(d).$$

Corollary 1. *Given an instance of the VMA problem with VMs $D = \{d_1, \ldots, d_n\}$, any solution π satisfies $P(\pi) \geq \rho^* \sum_{i=1}^{n} \ell(d_i)$.*

3 Off-line Algorithms

In this section we show that the VMA problem is NP-hard in the strong sense. Then, we show that there are algorithms to approximate the optimal off-line solution of VMA. Furthermore, if α is a constant there is a PTAS for the problem.

Theorem 1. *VMA is strongly NP-hard, even if α is constant.*

It is known that strongly NP-hard problems cannot have a fully polynomial-time approximation scheme (FPTAS) [32], we have the following corollary.

Corollary 2. *VMA does not have a fully polynomial-time approximation scheme (FPTAS), even if α is constant.*

Observe that the problem remains NP-hard when m is fixed to 2. The proof uses a simple reduction from the partition problem, in which it is decided whether a multiset of integers can be partitioned into two subsets of equal sum.

Theorem 2. *For every constant $\epsilon > 0$, there is an algorithm for the VMA problem with approximation ratio $(1+\epsilon)^\alpha$ and time complexity $O(\min(n,m) \cdot (n + g(1/\epsilon) \cdot \log^{O(1)} n))$, for some function $g(\cdot)$ that grows at least exponentially.*

When α is a constant, this result provides a PTAS for the VMA problem, by simply choosing for each constant $\delta > 0$ an appropriate constant ϵ such that $1 + \delta = (1+\epsilon)^\alpha$, and using this new value in the above theorem.

Corollary 3. *When α is constant, there is a polynomial-time approximation scheme (PTAS) for the VMA problem.*

4 Competitiveness of Online Algorithms

In this section we study the online version of VMA. First, we prove a lower bound on the competitiveness of any online algorithm in a system with a bounded number m of PMs, and a stronger lower bound for $m = 2$. Later, we present algorithms with bounded competitiveness for systems with 2 PMs and with unbounded number of PMs.

4.1 Lower Bounds

We show now that for m PMs there is a general lower bound on the competitive ratio of $3^\alpha/(2^{\alpha+2} + \epsilon)$, for any $\epsilon > 0$. Let us first give the following lemmas.

Lemma 2. *Let $x^* < \ell_1 \leq \ell_2$. Then $f(\ell_1 + \ell_2) > f(\ell_1) + f(\ell_2)$.*

From this lemma, it follows that if a PM has at least 2 VMs, each with load larger than x^*, and there are unused PMs, the power consumption can be reduced by moving one VM to an unused PM. When this is done in a given partition we say that we are *using* Lemma 2.

Lemma 3. *Let $L > 0$ and $\ell_2 < \ell_1 \leq L/2$. Then $f(\ell_1) + f(L - \ell_1) < f(\ell_2) + f(L - \ell_2)$.*

This lemma carries the intuition that balancing the load among the used PMs as much as possible reduces the power consumption. The following theorem gives a lower bound on the competitiveness of any algorithm.

Theorem 3. *When the number of PMs is m (bounded), no online VMA algorithm can achieve a competitive ratio of $3^\alpha/(2^{\alpha+2} + \epsilon)$, for any $\epsilon > 0$.*

The next result shows that the above lower bound can be made stronger for $m = 2$.

Theorem 4. *There is no online VMA algorithm that achieves a competitive ratio of* $3^\alpha/2^{\alpha+1}$*, if* $m = 2$*.*

4.2 Upper Bounds

In this section we present first a VMA algorithm (detailed in Algorithm 1) and show an upper bound on its approximation ratio. The algorithm is online, that is, the VMs are revealed to the algorithm one by one. A_1 and A_2 are the sets of VMs assigned to PMs s_1 and s_2, respectively, at any given time.

Algorithm 1: Online VMA algorithm for $m = 2$.

```
for each VM d_i do
    if ℓ(d_i) + ℓ(A_1) ≤ (b/(2^α − 2))^{1/α} or ℓ(A_1) ≤ ℓ(A_2) then
        d_i is assigned to s_1
    else
        d_i is assigned to s_2
```

We prove the approximation ratio of Algorithm 1 in the following theorem.

Theorem 5. *For a system where* $m = 2$*, there exists an online VMA algorithm that achieves the following competitive ratios.*

$$\rho = 1, \text{ for } \ell(D) \leq \left(\frac{b}{2^\alpha - 2}\right)^{1/\alpha},$$

$$\rho \leq \max\left\{2, \left(\frac{3}{2}\right)^{\alpha-1}\right\}, \text{ for } \ell(D) > \left(\frac{b}{2^\alpha - 2}\right)^{1/\alpha}.$$

We introduce now an online VMA algorithm for the case when the number of PMs is unbounded. The algorithm uses the load of the new revealed VM in order to decide the PM where it is assigned. If the load of the revealed VM is larger than x^*, the algorithm assigns this VM to a new PM without any other VM already assigned to it. Otherwise, the algorithm schedules the revealed VM to the most loaded PM whose current load is smaller than x^*. Note that, since the case under consideration assumes the existence of an unbounded number of PMs, there always exists at least one PM whose current load is smaller than x^*. A detailed description of this algorithm is shown in Algorithm 2. As before, A_j is the set of VMs assigned to PM s_j at a given time.

We prove the approximation ratio of Algorithm 2 in the following theorem.

Algorithm 2: Online VMA algorithm for unbounded number of PMs.

for *each VM* d_i **do**
  if $\ell(d_i) \geq x^*$ **then**
    d_i is assigned to a new PM
  else
    d_i is assigned to the PM s_j such that $\ell(A_k) \leq \ell(A_j) < x^*$ for all k

Theorem 6. *For a system with unbounded number of PMs, Algorithm 2 achieves a competitive ratio of 1 if no VM* d_i *has* $\ell(d_i) < x^*$*, and of* $2^{\alpha-1}+x^* \big/ \sum_{d_i:\ell(d_i)<x^*} \ell(d_i)$*, otherwise.*

5 Experimental Evaluation

In this section we experimentally evaluate the performance of the online VMA algorithm we proposed in Section 4, extended to be able to handle a bounded number of PMs. Additionally, we compare it with other online placement algorithms.

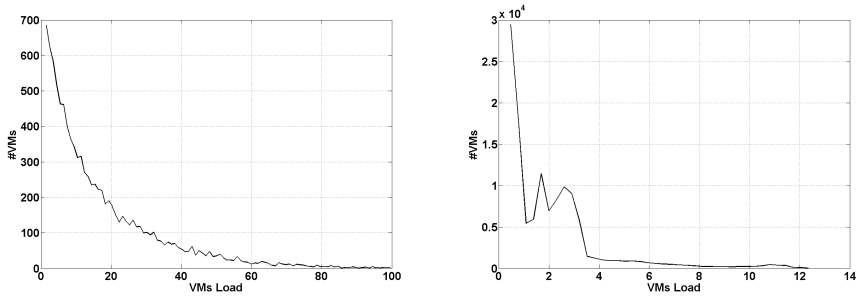
5.1 Experimental Setup

The online algorithm whose performance is evaluated here, which we call Algorithm VMA, behaves exactly as Algorithm 2 if possible. Otherwise, a new VM is assigned to the least loaded PM.

The performance of Algorithm VMA is first compared with a lower bound, denoted $LBVMA$, that is obtained as follows. The input VMs are sorted in non-increasing order of their loads. Then, using this order, as many VMs as possible with load at least x^* are assigned to different PMs. Let L be total load of the VMs still unassigned. If there are at least $\lfloor L/x^* \rfloor$ PMs still unused, $LBVMA$ uses exactly $\lfloor L/x^* \rfloor$ PMs. Otherwise all PMs are used by $LBVMA$. Finally, the load L is assigned among all used PMs as if it could be infinitely divided (i.e., as a fluid), using a water-filling algorithm [14].

Algorithm VMA is also compared with the following algorithms proposed in the literature.

- Random Placement (RP) [26]: It chooses a PM for each VM uniformly at random.
- Next Fit (NF) [26]: Starting initially at the first PM, each new VM is assigned to the next PM after the latest PM to which a VM was assigned (in a cyclic fashion).
- Least Full First (LFF) [26]: Each new VM will be assigned to (one of) the least loaded PM in the system.



(a) Trace A (synthetic traces) (b) Trace B (Google traces)

Fig. 1. VMs load distributions used in the evaluations.

- Striping (S) [22]: Each new VM is assigned to (one of the) PM with the smallest number of VMs assigned.
- Watts per Core (WC) [22]: Assign each new VM to the PM whose power would suffer the smallest increase.

The behavior of the aforementioned algorithms is evaluated by inputting two sets of traces, synthetic and real, shown in Figure 1. We call them *Trace A* and *Trace B*, respectively. Trace A is generated by randomly choosing the load of each VM following a power-law distribution with exponential cutoff (power-law distributions are similar to Zipf and Pareto distribution, cf. [6, 28]), which has been chosen so 100 is the maximum load of a VM. We select 10000 integer loads randomly using this distribution. This leads us to the VM load distribution shown in Figure 1(a).

Trace B is obtained from public Google traces [20]. We extract all the tasks from these traces, assuming that each task is an independent VM. The VMs (tasks) are sorted by the time at which they join the system. The load of a VM is the maximum CPU load of the task. The trace then contains 124885 VMs with loads varying between 0.31 and 12.5. The resulting VM load distribution can be seen in Figure 1(b).

Each execution of the algorithms is run with a fixed number of PMs. This number of PMs increases from 1 to the number of VMs in the trace being used. This allows us to see how the power consumption and the algorithms behavior evolve when the available PMs in the system vary.

We run simulations for all values of $\alpha \in \{1.5, 2, 3\}$, and $x^* \in \{1, 3, 10, 30, 100, 300, 1000, 10000\}$, for both traces and for all algorithms (including the lower bound). (Due to space restrictions we only present results for $x^* \in \{30, 100, 300\}$, which are the most interesting.) With these two parameters, b is also

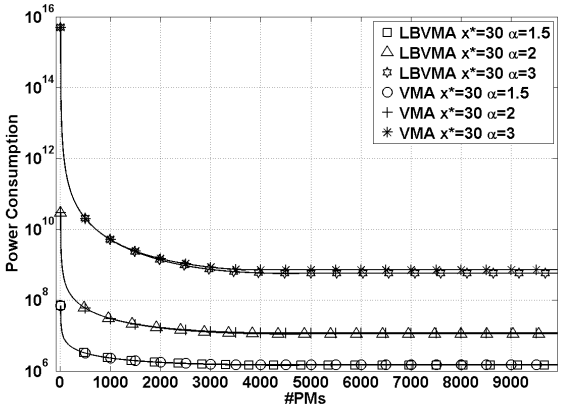


Fig. 2. Comparing the power consumed by VMA with the lower bound $LBVMA$ for $x^* = 30$ and $\alpha = \{1.5, 2, 3\}$.

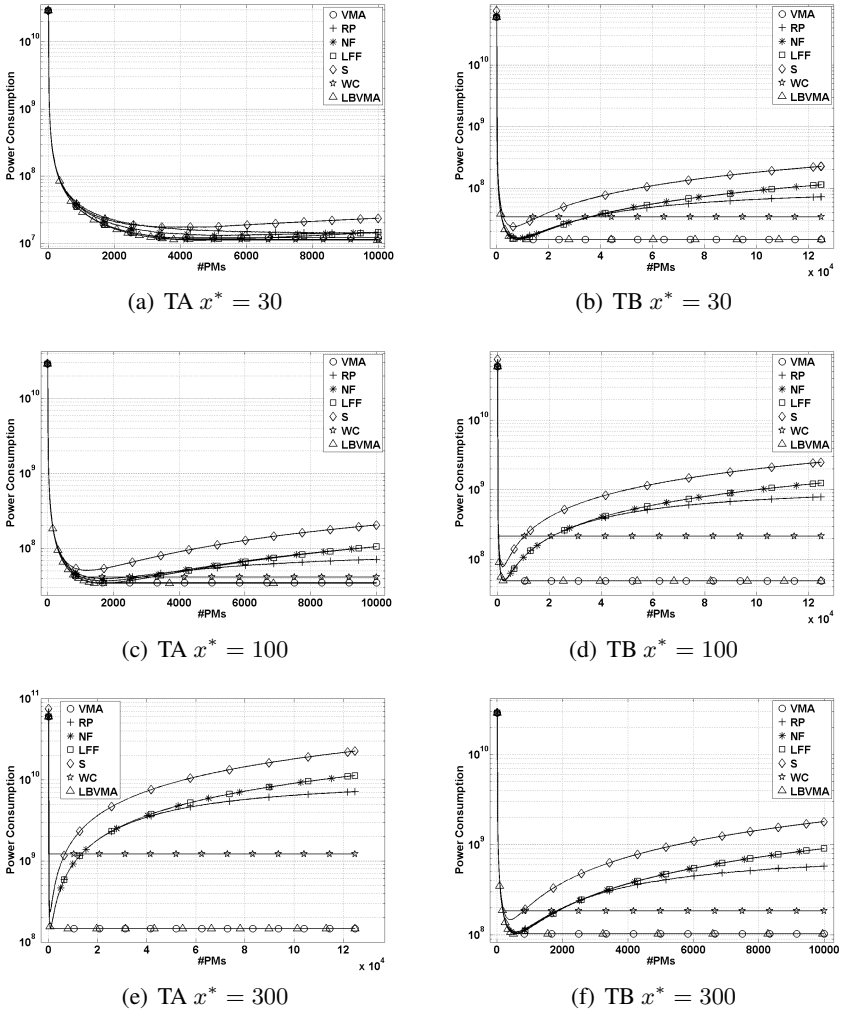
characterized and we can also study different situations, the cases in which the VM loads are larger, similar or smaller than x^*.

5.2 Experimental Results

The results obtained are presented as graphs in which the power consumed is represented as a function of the number of PMs used. We start by using Trace A, comparing the resulting power consumed using our algorithm with the lower bound on the optimal power consumption given by $LBVMA$. This comparison is run for a fixed value of x^* ($x^* = 30$) and the different values of α considered, 1.5, 2, and 3. The results are shown in Figure 2. As it can be observed, there is no qualitative difference in the solutions when α varies. (Similar results are obtained with other values of x^* and with Trace B.) As can be seen, the power consumed by the partitions found with VMA is very close to the lower bound obtained with $LBVMA$. This shows that the performance of VMA is close to optimal.

We then compare both VMA and $LBVMA$ with algorithms RP, LFF, NF, WC and S. Figure 3 shows the result of running these algorithms with Trace A and Trace B, for $\alpha = 2$ and different values of x^* ($x^* = \{30, 100, 1000\}$).

Observe that with Trace A and small values of x^*, like $x^* = 30$, all the algorithms present similar results, and VMA is not showing a significantly better performance than the rest. This is so because, either explicitly or implicitly, when x^* is small in relation with the loads in the system, using a new PM is cheap, and all the algorithms (including VMA) essentially distribute the load evenly among the available PMs.



(a) TA $x^* = 30$ (b) TB $x^* = 30$

(c) TA $x^* = 100$ (d) TB $x^* = 100$

(e) TA $x^* = 300$ (f) TB $x^* = 300$

Fig. 3. Performance of the algorithms for Trace A and Trace B, $\alpha = 2$ and different values of x^*.

However, Figures 3(b) and 3(c) already shows a change in the behavior due to the smaller size of the loads in Trace B, on the one hand, and the higher value of x^* on the other. In these cases, WC and VMA, that have been designed to be energy-aware, take an advantage with respect to the other algorithms. In fact, both algorithms reach a stable number of PMs that is enough to place all the available load, not requiring new PMs and, hence, not paying for the switching on cost of additional PMs (characterized by b).

6 Conclusions and Open Problems

In this paper we have studied a particular case of the generalized assignment problem with applications to Cloud Computing. We have considered the problem of assigning virtual machines (VMs) to physical machines (PMs) so that the power consumption is minimized, a problem that we call virtual machine assignment (VMA). In our theoretical analysis we have shown that the VMA problem is NP-hard, we have shown a PTAS that solves VMA offline, and we have proved upper and lower bounds on the competitive ratio of VMA online algorithms. We have also carried out extensive simulations using synthetic data as well as real world inputs such as Google cluster data. The simulations show that in practice the performance of our online algorithms is very close to a lower bound on the offline optimal, and better than known techniques currently used.

In our model we have assumed that customers specify a CPU requirement that must be guaranteed and will not change, that migration of VMs among PMs is not feasible, and that the service provider may increase the number of PMs at a cost. To the best of our knowledge, this model has not been studied previously from a theoretical standpoint. Other models in the experimental literature include more client requirements (such as speed, disk, memory, etc.), migration of VMs for free, dynamic change of loads, and/or different cost functions. Being parametric, our energy cost function generalizes other functions considered in the literature. For settings where the dominating cost is other than the CPU load, our results still apply tuning appropriately the cost function. If the migration of VMs is for free, our results also still apply running the offline approximation algorithm each time a load changes or a new VM arrives. We leave for future work the consideration of migration of VMs at a cost, and the combination of various resource requirements, none of which is a bottleneck.

References

1. Amazon web services. `http://aws.amazon.com`. Accessed August 27, 2012.
2. Citrix. `http://www.citrix.com`. Accessed August 27, 2012.
3. Eucalyptus. `http://www.eucalyptus.com/`. Accessed January 20th, 2013.
4. Opennebula. `http://opennebula.org/`. Accessed January 20th, 2013.
5. Rackspace. `http://www.rackspace.com`. Accessed August 27, 2012.
6. L. A. Adamic and B. A. Huberman. Zipf?s law and the internet. *Glottometrics*, 3(1):143–150, 2002.
7. N. Alon, Y. Azar, G. J. Woeginger, and T. Yadid. Approximation schemes for scheduling. In *SODA*, pages 493–500, 1997.
8. M. Andrews, A. F. Anta, L. Zhang, and W. Zhao. Routing for power minimization in the speed scaling model. *IEEE/ACM Trans. Netw.*, 20(1):285–294, 2012.
9. M. Andrews, S. Antonakopoulos, and L. Zhang. Minimum-cost network design with (dis)economies of scale. In *FOCS*, pages 585–592, 2010.

10. J. Arjona Aroca, A. Fernández Anta, M. A. Mosteiro, and C. Thraves. Power-efficient Assignment of Virtual Machines to Physical Machines. *ArXiv e-prints*, Apr. 2013.
11. N. Bansal, H.-L. Chan, and K. Pruhs. Speed scaling with an arbitrary power function. In *SODA*, pages 693–701, 2009.
12. U. Bellur, C. S. Rao, and M. K. SD. Optimal placement algorithms for virtual machines. arXiv:1011.5064 (`http://arxiv.org/abs/1011.5064`), 2010.
13. J. F. Botero, X. Hesselbach, M. Duelli, D. Schlosser, A. Fischer, and H. de Meer. Energy efficient virtual network embedding. *IEEE Communications Letters*, 16(5):756–759, 2012.
14. S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.
15. M. Cardosa, A. Singh, H. Pucha, and A. Chandra. Exploiting spatio-temporal tradeoffs for energy-aware mapreduce in the cloud. In *CLOUD*, pages 251 –258, 2011.
16. A. K. Chandra and C. K. Wong. Worst-case analysis of a placement algorithm related to storage allocation. *SIAM J. Comput.*, 4(3):249–263, 1975.
17. S.-C. Chen, C.-C. Lee, H.-Y. Chang, K.-C. Lai, K.-C. Li, and C. Rong. Energy-aware task consolidation technique for cloud computing. In *CLOUD*, pages 115–121, 2011.
18. R. A. Cody and E. G. C. Jr. Record allocation for minimizing expected retrieval costs on drum-like storage devices. *J. ACM*, 23(1):103–115, 1976.
19. A. Gupta, R. Krishnaswamy, and K. Pruhs. Online primal-dual for non-linear optimization with applications to speed scaling. arXiv:1109.5931v1 [cs.DS] (`http://arxiv.org/abs/1109.5931`), 2011.
20. J. L. Hellerstein. Google cluster data. Google research blog, Jan. 2010. Posted at `http://googleresearch.blogspot.com/2010/01/google-cluster-data.html`.
21. S. Im, B. Moseley, and K. Pruhs. Online scheduling with general cost functions. In *SODA*, pages 1254–1265, 2012.
22. R. Jansen and P. Brenner. Energy efficient virtual machine allocation in the cloud. In *IGCC*, pages 1–8, 2011.
23. N. Liu, Z. Dong, and R. Rojas-Cessa. Task and server assignment for reduction of energy consumption in datacenters. In *NCA*, pages 171–174, 2012.
24. F. Machida, M. Kawato, and Y. Maeno. Redundant virtual machine placement for fault-tolerant consolidated server clusters. In *NOMS*, pages 32 –39, 2010.
25. C. Mark, D. Niyato, and T. Chen-Khong. Evolutionary optimal virtual machine placement and demand forecaster for cloud computing. In *AINA*, pages 348 –355, 2011.
26. K. Mills, J. Filliben, and C. Dabrowski. Comparing vm-placement algorithms for on-demand clouds. In *CLOUD*, pages 91–98, 2011.
27. M. Mishra and A. Sahoo. On theory of vm placement: Anomalies in existing methodologies and their mitigation using a novel vector based approach. In *CLOUD*, pages 275 –282, 2011.
28. M. E. Newman. Power laws, Pareto distributions and Zipf's law. *Contemporary Physics*, 46(5):323–351, 2005.
29. H. Nguyen Van, F. Dang Tran, and J.-M. Menaud. Autonomic virtual resource management for service hosting platforms. In *CLOUD*, pages 1–8, 2009.
30. S. Srikantaiah, A. Kansal, and F. Zhao. Energy aware consolidation for cloud computing. In *HotPower*, pages 10–10, 2008.
31. R. Van den Bossche, K. Vanmechelen, and J. Broeckhove. Cost-efficient scheduling heuristics for deadline constrained workloads on hybrid clouds. In *CLOUD*, pages 320–327, 2011.
32. V. V. Vazirani. *Approximation Algorithms*. Springer, Mar. 2004.
33. H. Viswanathan, E. Lee, I. Rodero, D. Pompili, M. Parashar, and M. Gamell. Energy-aware application-centric vm allocation for hpc workloads. In *IPDPSW*, pages 890 –897, 2011.
34. J. Xu and J. Fortes. A multi-objective approach to virtual machine management in datacenters. In *ICAC*, pages 225–234, 2011.