Consolidation of VMs to improve energy efficiency in cloud computing environments

Thiago Kenji Okada[1], **Albert De La Fuente Vigliotti**[1], **Daniel Macêdo Batista**[1], **Alfredo Goldman vel Lejbman**[1]

[1]Institute of Mathematics and Statistics (IME) – University of São Paulo (USP)
São Paulo, SP, Brazil

{thiagoko,albert,batista,gold}@ime.usp.br

Abstract. *Improvement of energy efficiency in IT is an important research topic nowadays. The reduction of operational costs, generated heat and environment impact are some of the reasons for this. Thanks to the advent of cloud computing, it is possible to improve energy efficiency in data centers by running various virtual machines in a single physical machine. However, the cloud providers generally invest in performance, not energy efficiency. This paper focuses on the problem of an energy efficient initial VM placement, and describes three new algorithms for this problem, one based on the First Fit Decreasing algorithm, and the other two based on the Best Fit Decreasing algorithm. They are compared with other algorithms in the literature, and a reduction of power consumption up to 3.24% was observed, as well a reduction of execution time in several orders of magnitude. Scripts used to analyze traces publicly provided by Google are another contribution of the paper, since they are useful for those working in mechanisms for cloud computing.*

1. Introduction

The advent of cloud computing is changing the way investments in computing resources are made. The characteristics of cloud computing services, including the illusion of infinite computing resources, elimination of upfront costs, and the ability to pay per use [Armbrust et al., 2010], made those kinds of services popular. Different service providers are rapidly deploying new data centers to keep up with the increasing demand for cloud services.

Performance was the sole concern in cloud environments, to keep up with the demand, while energy consumption did not receive much attention [Beloglazov et al., 2012]. However, worldwide energy usage in data centers accounted for between 1.1% and 1.5% of total electricity use in 2010 [Koomey, 2011]. As the energy costs keep increasing, focus on improving the energy efficiency in data centers is important, but it is necessary to maintain similar performance while doing so.

Different approaches to improve energy efficiency may be applied on data centers, including development of applications that use resources more efficiently, and use of modern hardware, including technologies like Dynamic Voltage and Frequency Scaling (DVFS) [Beloglazov and Buyya, 2010]. Nevertheless, an efficient approach is to reduce the amount of hardware over-provisioning by the implementation of more efficient VM (Virtual Machine) placement algorithms.

Virtualization technology allows several VMs in a single physical machine. However hardware is typically over-provisioned in order to sustain peaks of resource demand during short-periods of time, since cloud providers are bound to Service-Level Agreements (SLAs) [Feller et al., 2011]. The result of this strategy is an average low resource utilization of data centers: for example, in [Barroso and Hölzle, 2007] is shown that the mean value of CPU utilization in data centers is 36.44% with 95% confidence interval.

Therefore, improving VM placement can lead to interesting gains in energy efficiency. For example, if we can allocate more VMs using less resources, the tendency is to reduce the power consumption of the data center.

This paper focuses on an energy efficient initial VM placement, instead of optimizing an existing VM placement like other works in the literature. This work presents three different algorithms, one based on the First Fit Decreasing (FFD) algorithm [Yue, 1991] and two based on the Best Fit Decreasing (BFD) algorithm [Suoerv and Johnson, 1973] to solve the problem of VM placement, including modifications that allow each algorithm to suspend physical hosts and that evaluate the impact of a new VM allocation in the entire data center energy consumption. We modeled the VM placement problem as a bin packing problem [Beloglazov and Buyya, 2012; Vigliotti and Batista, 2014b].

We implemented and evaluated these algorithms in pyCloudSim[1], a simulation framework written in the Python language by the authors, and compared them with other algorithms in the literature [Beloglazov and Buyya, 2012; Vigliotti and Batista, 2014b] using traces available from Google's clusters [Reiss et al., 2011]. Before presenting the results of the experiments, an extensive analysis of the Google's traces is detailed. The scripts developed to make this analysis represent another contribution of this paper and they are publicly available. In the experiments, our algorithms performed better than other algorithms presented in the literature, while being multiple times faster.

The remainder of this paper is structured as follows: In Section 2, we review the literature about the area. In Section 3, we describe the problem and our proposed algorithms. In Section 4, we describe the simulation environment, including the analysis of Google's traces. In Section 5, we present the performance evaluation. Finally, in Section 6, we present the conclusion of this work and future directions.

2. Related work

[Barroso and Hölzle, 2007] describe techniques to reduce the power consumption in servers. These techniques are directly related with our simulation model, that uses dynamic-voltage-frequency scaling (DVFS) and sleeping states to reduce power usage. DVFS reduces voltage according to the current CPU frequency, reducing overall consumption, while sleeping states put the host in a state of low power consumption and disable host functionalities until "wake up" by changing its state.

[Beloglazov and Buyya, 2012; Beloglazov, 2013] present the Power Aware Best Fit Decreasing (PABFD) which is based on the Best Fit Decreasing algorithm. However, the focus of those studies is in the optimization of an existing VM placement, instead of focusing in the initial VM placement problem. Therefore it does not overlap with our

[1]`https://github.com/vonpupp/pyCloudSim/tree/v0.1` . Accessed on December 14, 2014

study. Other differences between the PABFD and our algorithms based on BFD is that our algorithms allows physical hosts to suspend and, also, because one of our algorithms evaluate the impact of a new VM allocation in terms of the entire data center energy consumption.

[dos Santos Júnior, 2014] shows an implementation of FFD for the VM placement problem. While it does a comprehensive study of the problem, it does not compare FFD with other algorithms, as we do in this work. The difference between the implementation proposed in [dos Santos Júnior, 2014] and our algorithm based on FFD is that our algorithm allows physical hosts to suspend.

[Vigliotti and Batista, 2014a] present two approaches for the problem of VM placement. The first one (called *Iterated-KSP*) is an iterated approach for a Knapsack based heuristic, solving the problem n-times (once per host). The second one (called *Iterated-EC*) is based on an Evolutionary Computation model, using an iterated approach. Experiments in Section 5 show that our algorithms are faster and perform better than the algorithms presented in [Vigliotti and Batista, 2014a].in [Vigliotti and Batista, 2014a] by including new algorithms in the comparison and adding support to Google's traces data.

In [Vigliotti and Batista, 2014b] the authors present two variations to the algorithms presented in [Vigliotti and Batista, 2014a]. The first one, called (*Iterated-KSP-Mem*), is a modification of Iterated-KSP that doubles the size of virtual memory to allocate VMs that consume too much virtual memory. The second one, called (*Iterated-EC-Net*), is a modification of Iterated-EC aiming to reduce both the power consumption and network utilization, instead of only the former. While we did not include Iterated-EC-Net in our comparison, since Google's traces data does not include network usage information, our experiments show that our algorithms are faster and perform better than the other algorithm presented in [Vigliotti and Batista, 2014b].

To the best of our knowledge, there is no work in the literature that compares different algorithms for the initial VM placement problem and there is no algorithm to this objective that runs faster and more efficiently than the algorithms we are proposing in this paper.

3. Energy efficient VM placement

The energy-efficient VM placement problem can be divided in two sub-problems. The first one is the placement of a requested VM on a host, while the second one is the optimization of the current VM placement. The focus on this work is the first one.

Similar to [Feller et al., 2011; Vigliotti and Batista, 2014a], we modeled our problem as a instance of the well-known multi-dimensional bin-packing (MDBP) problem. To find the optimal solution of a MDPB problem is NP-hard [Jung et al., 2010], therefore it is interesting to use heuristics instead of calculating the exact solution, for efficiency reasons, since a slow algorithm would not be interesting in real data centers receiving several requests to allocate new VMs at every second.

An energy efficient VM placement aims to reduce the number of active physical machines, by increasing the workload in the current active machines. The idea is simple: if the machine already has work to do, increasing the amount of work will not increase the power consumption of the data center significantly. However, waking up a new machine will impact the power consumption of the data center more significantly, since a machine in suspended state consumes a little fraction of the power of an active one. Even in the case when the data center is composed by several modern machines that do not have a consumption directly proportional to the usage, our work is valid, since we consider how much Watts are consumed by each one of the operation modes of the processor(s) of the machine.

The strategy used to place the VMs must be careful to not overload a host, since it is important to not violate any Service-Level Agreements (SLAs) between the cloud provider and the client.

The next subsections present the three proposed algorithms. The first algorithm (Subsection 3.1) and the second algorithm (Subsection 3.2) modify existing algorithms by including instructions related to the suspension of physical machines. The third algorithm (Subsection 3.3) extends existing algorithms by including instructions related to suspension of physical machines and also including instructions to evaluate if a new VM allocation affects the entire data center energy consumption.

3.1. The First Fit Decreasing algorithm

The first algorithm presented in this work is a modification of the greedy algorithm described in the literature [Yue, 1991; Esnault, 2012], called First Fit Decreasing. The pseudo-code for the algorithm is shown in Algorithm 1.

```
   Input: $hostList$, $vmList$
   Output: A VM placement
 1 Sort $vmList$ in the order of decreasing utilization;
 2 foreach $vm$ in $vmList$ do
 3     foreach $host$ in $hostList$ do
 4         if $host$ has enough resources for $vm$ then
 5             Allocate $vm$ in $host$;
 6             break;
 7 foreach $host$ in $hostList$ do
 8     if $host$ has no allocated $vm$ then
 9         Suspend $host$
```

Algorithm 1: The First Fit Decreasing (FFD) algorithm for VM placement.

This algorithm sorts the VMs on the list of requests into non-increasing order by CPU usage (line 1) and then processes the list in that order by placing each VM into the first host which fits (lines 2-5). It continues to the next VM (line 6) until all VMs are allocated. Finally, lines 7-9 suspend idle physical hosts, to reduce power consumption.

[Yue, 1991] shows that the result given by this algorithm for a list L of items is no more than $\frac{11}{9}OPT(L) + 1$, where $OPT(L)$ is the optimal solution. Our modification of FFD is power aware: if no VMs are allocated to a host, it suspends that host, greatly reducing the energy consumption.

3.2. The Power Aware Best Fit Decreasing algorithm

The second algorithm is based on the work from Beloglazov, called Modified Best Fit Decreasing (MBFD) [Beloglazov and Buyya, 2012] or, more recently, Power Aware Best Fit Decreasing (PABFD) [Beloglazov, 2013]. According to [Beloglazov and Buyya, 2012], this algorithm is a variation of FFD algorithm, and the same demonstration from [Yue, 1991] is still valid.

The algorithm is shown in Algorithm 2. While the algorithm itself is similar to FFD, the main difference is that PABFD calculates the increase in power consumption after the allocation of the current VM (line 12) in each host in data center (lines 7-14), storing the lowest increase in power consumption (lines 15-17). The VM is then allocated to the host that had the lowest increase in power consumption (lines 18-19). Finally, lines 20-22 actually suspend idle physical hosts, similar to Algorithm 1.

We modified the PABFD algorithm from [Beloglazov, 2013] to consider that every host is suspended before the initial allocation (lines 1-2). This change made the allocation of a VM in a unused host very expensive in terms of energy consumption, since it would need to wake-up the host first (lines 9-11), and allows our algorithm to prioritize hosts that already had VMs allocated to them.

3.3. The Global Power Aware Best Fit Decreasing algorithm

Our third algorithm is a modification of the algorithm described in Section 3.2. In this modification, we changed the function that returns the estimate in power consumption after a VM allocation, i.e., we changed the function $estimateIncreaseInPower$ in line 13 from Algorithm 2 to the code showed in Algorithm 3.

Instead of looking to the power consumption increase after the allocation of a new VM, our algorithm calculates the power consumption of the whole data center (lines 3-4) after simulating an allocation of VM in the current host (line 2). We named it *Global Power Aware Best Fit Decreasing* (GPABFD) due to this.

4. The simulation framework and workloads

We evaluate the performance of our proposed algorithms through simulation. We noticed that the available cloud computing simulators like CloudSim [Buyya et al., 2013] make their decision related to VM placement only based on CPU usage. Because of that, we extended the pyCloudSim simulator [2]. The simulator is a free software over the Apache 2.0 license.

Although the simulator was previously used to evaluate the performance of VM placement mechanisms in [Vigliotti and Batista, 2014a,b], on those works it used traces from PlanetLab machines, which not necessarily represent the higher diversity in workloads encountered in cloud computing. This fact motivated us to extend the pyCloudSim by including support to the Google cluster traces, traces that captures different workloads (Subsection 4.1).

Presenting the details of pyCloudSim is out of the scope of this paper, nevertheless more details about the framework can be found in [Vigliotti and Batista, 2014a]. We will

[2] `https://github.com/m45t3r/pyCloudSim`. Accessed on December 14, 2014.

```
Input: $hostList$, $vmList$
Output: A VM placement
1  foreach $host$ in $hostList$ do
2      Assume $host$ is suspended;
3  Sort $vmList$ in the order of decreasing utilization;
4  foreach $vm$ in $vmList$ do
5      $minPower \leftarrow \infty$;
6      $allocatedHost \leftarrow Null$;
7      foreach $host$ in $hostList$ do
8          if $host$ has enough resources for $vm$ then
9              if $host$ is suspended then
10                 $hostWasSuspended \leftarrow True$;
11                 Assume Wake-on-Lan on $host$;
12             $power \leftarrow estimateIncreaseInPower(host, vm)$;
13             if $hostWasSuspended = True$ then
14                 Assume $host$ is suspended;
15         if $power < minPower$ then
16             $allocatedHost \leftarrow host$;
17             $minPower \leftarrow power$;
18     if $allocatedHost \neq Null$ then
19         Allocate $vm$ in $host$;
20 foreach $host$ in $hostList$ do
21     if $host$ has no allocated $vm$ then
22         Suspend $host$
```

Algorithm 2: The Power Aware Best Fit Decreasing (PABFD) algorithm for VM placement.

```
Input: $host, vm$
Output: Power consumption of data center
1  $totalPower \leftarrow 0$;
2  Simulate allocation of $vm$ in $host$;
3  foreach $host$ in $hostList$ do
4      $totalPower \leftarrow totalPower + estimatePower(host)$;
5  Remove $vm$ from $host$;
```

Algorithm 3: The Global Power Aware Best Fit Decreasing (GPABFD) modification.

focus, in Subsection 4.1, to provide information about the Google's traces used in our experiments, which is useful to researchers working with cloud computing mechanisms and needing more information about these traces to execute experiments.

4.1. Google cluster traces

Cloud computing has higher diversity in workloads profiles than traditional data centers. One big, public cloud provider may have different kinds of applications running in its cloud, including big data analysis, Web services, scientific computing, software deployment and automated tests, and others.

However, several traces of cluster data report more homogeneous workload. For example, one trace may have a great number of high-performance, non-latency sensitive tasks (like big data analysis), while other is more focused in long-running servers like Web servers. Google made available traces from one of its computing clusters that captures different behavior, including the former examples and others [Reiss et al., 2012].

Google cluster traces data is composed by various *jobs*. Each job is created by a *user*, and it is comprised of one or more *tasks*, including the requirements used for scheduling the tasks on physical machines. There is no known distribution that fits these data, but the resources appear to form a heavy-tailed distribution [Reiss et al., 2011].

The raw information of the current version of Google cluster data is composed of 40GB of compressed data. Some tasks have 0 units of CPU, RAM, Disk I/O or Disk space [Reiss et al., 2011] and some tasks are really long, including tasks running for the whole period of the traces. So we needed to filter the data before using it. To do this, we created a script to import the raw data available in the Google website to a SQL database[3], allowing us to query for various pieces of data included in the traces.

We need to make some assumptions on the data to do a proper simulation. Google does not provide details about their computing environment and the data itself is obfuscated, so we do not know if the application is running on a VM, a container or even on the real OS. Each job has a multitude of tasks, that may or not run in the same machine. However, a task is a single Linux program, possibly consisting of multiple processes, to be run on a single machine [Reiss et al., 2011]. So it is reasonable to consider each task as a single VM instance or a container.

Another assumption is about machine resources. Each resource data (including CPU, memory, disk space and I/O time) is normalized, relative to the largest capacity of the resource on any machine in the traces. So we need to assume that our environment is homogeneous (i.e. all machines have the same specifications), even when Google's cluster environment is heterogeneous [Reiss et al., 2011].

According to the official website of the traces [Wilkes and Reiss, 2013], the current version of the traces data represent 29 days of data from May 2011 on a cluster of 11k machines. The trace files are divided in 500 parts, from 00000 to 00499. Disk time data is only included in the first 14 days, because of a change in the monitoring system. So we evaluated the first 50 parts, from 00000 to 00049, that still included disk I/O time data. This is equivalent to 2713386 unique tasks. About 1% of the jobs in the traces

[3]`https://gist.github.com/m45t3r/0242be48a82bafee23f1`. Accessed on December 14, 2014.

have missing resource records, so we excluded them. We did not considered the network resource on this work, due to the fact that the Google Cluster Data traces do not include this resource metric.

From the rest of the traces data, we filtered the entries that had an average CPU usage from 15% to 80%, to get only tasks that had relevant CPU usage. This is equivalent to 3725 unique tasks. We then selected the first 288 tasks to represent the VMs for our simulation, since initial experiments showed that this was sufficient to show the difference between the algorithms presented in this work. This sample is sufficient since [Reiss et al., 2012] showed that there is no known distribution that seems to good fit the Google traces data. The possible explanation is due to unpredictability of humans factors in resource usage.

The histogram of CPU usage for these selected tasks is presented in Figure 1. The majority of the tasks, roughly 75% of selected tasks or about 2800 tasks, are in the 15%-20% range of average CPU usage. The rest of tasks, or approximately 25% of selected tasks, falls in the range starting from 20% of average CPU usage up to 45%, characterizing a heavy-tail distribution.

The histogram of memory usage for these selected tasks is in Figure 2. The majority of the tasks, roughly 67% of selected tasks or about 2500 tasks, are in the 0%-10% range of average memory usage. Roughly 29% of selected tasks or about 1100 tasks, are in the 10%-20% range. About 4% of the tasks falls in the range starting from 20% of average memory usage up to 60%, again characterizing a heavy-tail distribution.

The histogram of disk I/O time for these selected tasks is in Figure 3. It is interesting to note that disk usage is almost irrelevant in Google's traces. The majority of tasks, roughly 67% of selected tasks or about 2500 tasks, are in the 0%-0.5% range of average disk I/O time. About 33% of the tasks falls in the range starting from 0.5% of average disk I/O time up to 3%, characterizing another heavy-tail distribution.

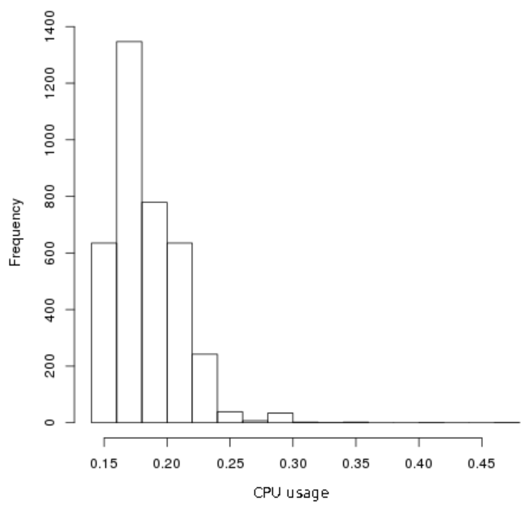

Figure 1. Histogram of Number of tasks X Average CPU Rate

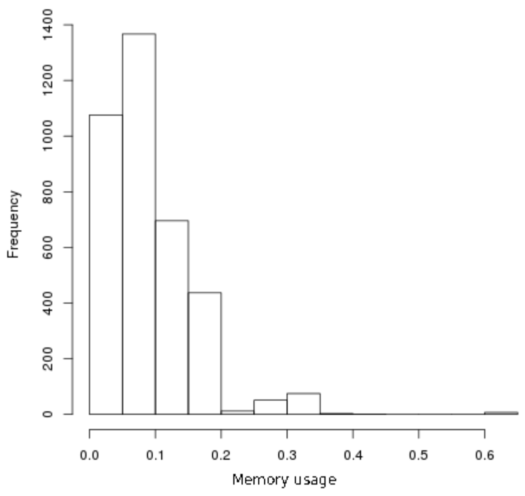

Figure 2. Histogram of Number of tasks X Average Memory Usage

5. Performance evaluation

The hardware used on the experiments was a virtual machine hosted in USP Cloud[4] scientific infrastructure. The physical machine has an Intel® Xeon® CPU E7- 2870 @ 2.40GHz. The virtual machine has 8 virtual CPU cores, 32GB of allocated RAM and two virtual disks, one of 20GB and another one of 500GB. The code and data of all experiments is available on GitHub[5].

We compared the proposed algorithms (FFD, PABFD and GPABFD) with an algorithm called Energy-Unaware, which simulates a First Come First Served (FCFS) algorithm without energy awareness. In this algorithm, we firstly randomize the VM request list to simulate requests coming from users. Another comparison is done with all algorithms available in [Vigliotti and Batista, 2014b], excluding Iterated-EC-Net, due to the fact that the Google traces do not include network usage information.

We have two different simulation scenarios, one with 35 physical machines and another one with 100 physical machines, each one starting with 16 VMs up to 288, in 16 VMs increments. The first scenario simulates a heavily used cloud, using more resources than are available. The second scenario simulates a more common situation, where the cloud provider overprovision their resources, but generally have more than sufficient resources for all the current tasks running in the cloud.

Each physical machine used the power model described in [Vigliotti and Batista, 2014a], that uses a linear power-to-frequency relationship for a server. For the real power usage we use data provided from SPECpower benchmark[6]. Each experiment was repeated 30 times to calculate the average and confidence intervals with a level of confidence equal

[4]`http://www.sti.usp.br/?q=node/5370`. Accessed on December 14, 2014.

[5]`https://github.com/m45t3r/sbrc-2015-simulation`. Accessed on December 14, 2014.

[6]`http://www.spec.org/power_ssj2008/results/res2014q3/power_ssj2008-20140615-00658.html`. Accessed on December 14, 2014.

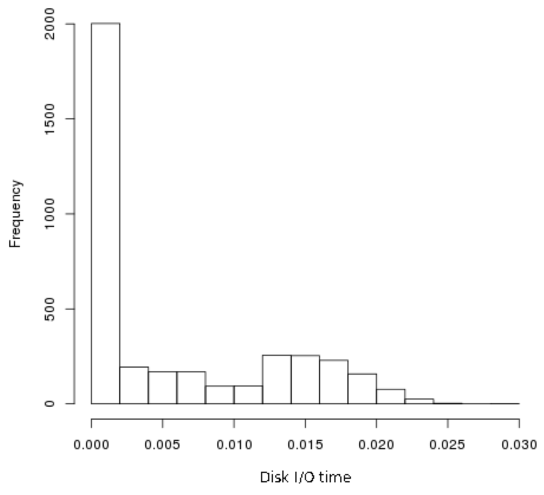

Figure 3. Histogram of Number of tasks X Average Disk I/O Time

to 95%.

Figure 4 shows the average number of VMs placed in our first simulation scenario, with 35 hosts. The FFD, PABFD and GPABFD algorithms performs worse than the rest of the algorithms in this situation where there is not sufficient resources for all VMs. With 288 VMs, for example, FFD, PABFD and GPABFD allocates 25.5% less VMs than Iterated-KSP, the best algorithm in this experiment. This performance of FFD, PABFD and GPABFD can be explained since these algorithms have a greedy nature, and once a decision is made they do not optimize a solution further.

Figure 5 shows the average number of VMs placed with 100 hosts, and confirms that there is sufficient resources for all VMs, at least for the initial allocation. In this case, all the algorithms performed similarly.

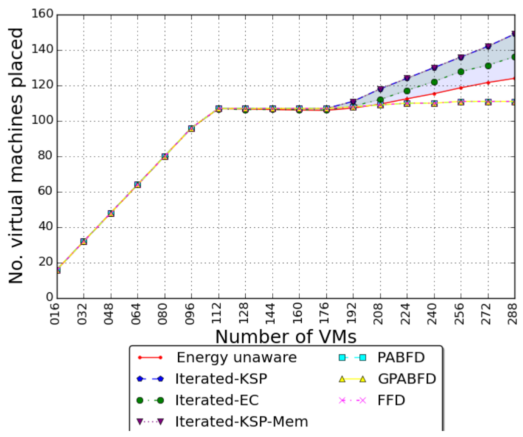

Figure 4. Average placed VMs comparison (35 hosts)

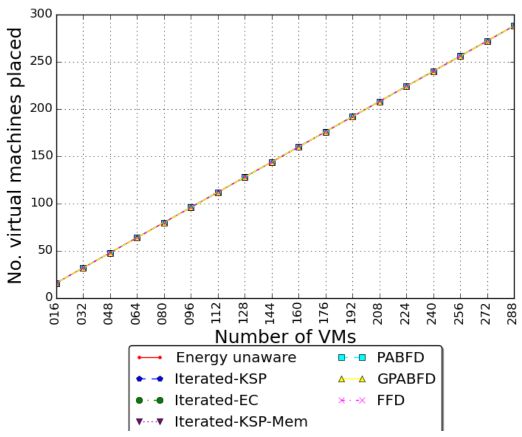

Figure 5. Average placed VMs comparison (100 hosts)

Figure 6 shows the average power consumption with 35 hosts. All algorithms excluding Energy-Unaware have similar consumption. This result is confirmed by Figure 8, where all algorithms excluding Energy-Unaware suspended a similar number of VMs.

Figure 7 shows the average power consumption with 100 hosts. FFD, PABFD and GPABFD performs similarly, but performs better than Iterated-KSP, Iterated-KSP-Mem, Iterated-EC and Energy-Unaware after 192 VMs. FFD for example, consumed 3.24% less power than Iterated-KSP in the 288 VMs case. Even including the confidence intervals (at most $\pm 0.53\%$ with 95% confidence), FFD, PABFD and GPABFD consumed less power than the other algorithms. This result is confirmed by Figure 9, where our algorithms suspended more VMs than the Iterated-KSP, Iterated-KSP-Mem, Iterated-EC and Energy-Unaware, starting from 192 VMs.

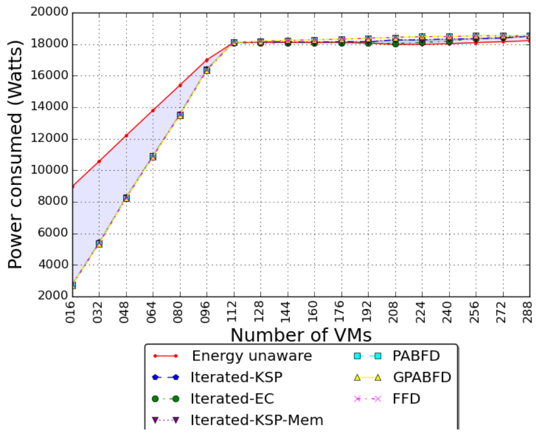

Figure 6. Average energy consumption comparison (35 hosts)

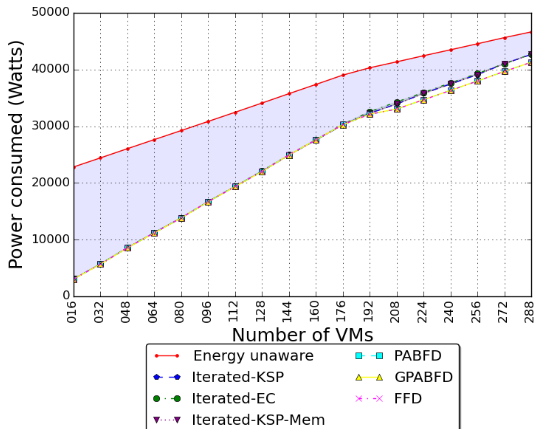

Figure 7. Average energy consumption comparison (100 hosts)

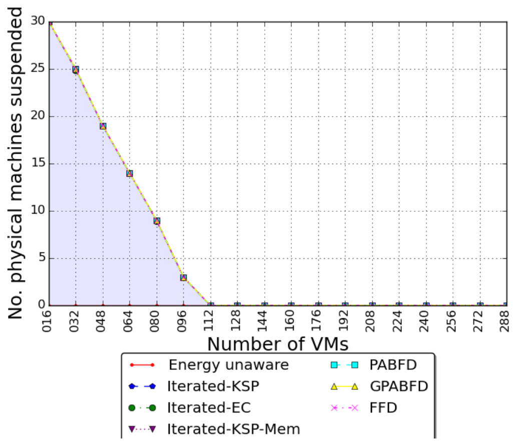

Figure 8. Average number of physical machines suspended comparison (35 hosts)

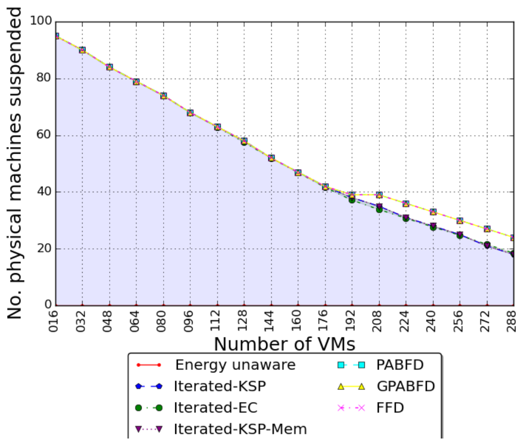

Figure 9. Average number of physical machines suspended comparison (100 hosts)

Figures 10 and 11 show the execution time of each algorithm. Since the FFD, PABFD and GPABFD algorithms are simpler than other algorithms, they are faster even with a great number of VMs. For example, FFD uses less than 2% of the time used by Iterated-KSP or Iterated-EC in the case with 288 VMs and 100 hosts. Even including the confidence intervals (at most $\pm 52.98\%$ with 95% confidence), FFD, PABFD and

GPABFD are at least one order of magnitude faster than the other algorithms. The FFD algorithm in particular shows execution times below one second in general. The high range of confidence intervals for execution time may be explained by performance variation between measurements, since the simulation was running in a cloud environment.

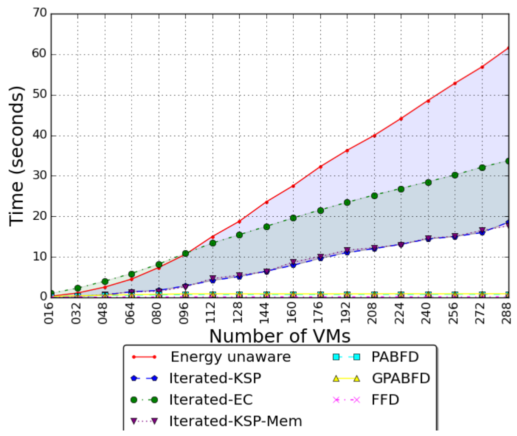

Figure 10. Average execution time comparison (35 hosts)

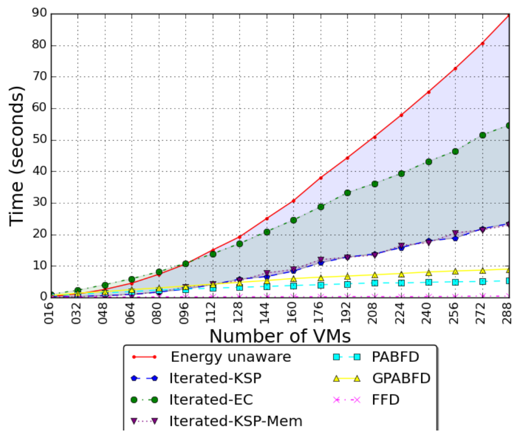

Figure 11. Average execution time comparison (100 hosts)

Table 1 summarizes the average execution time for all algorithms in the most demanding scenario simulated, with 100 hosts and 288 VMs. As explained, the FFD, PABFD and GPABFD algorithms are faster than all other algorithms by at least one order of magnitude.

Table 1. Average execution time with 100 hosts and 288 VMs

Strategy	Average time
Energy-Unaware	89.48
Iterated-KSP	23.38
Iterated-KSP-Mem	22.90
Iterated-EC	54.64
FFD	0.38
PABFD	5.30
GPABFD	8.99

6. Conclusion and future directions

The Fit-Decreasing family of algorithms (FFD, GABFD and PABFD) have an interesting performance according to the experiments. In a cloud with heavy usage, where there is not sufficient resources for every virtual machine currently in the system, they allocate less VMs than all other algorithms, including Energy-Unaware. However, it can be argued that this situation is not realistic, since it would mean that the cloud provider did not have sufficient resources to serve all its clients.

In a more realistic situation, where there is sufficient resources for every virtual machine, the Fit-Decreasing family of algorithms performs even better than more sophisticated algorithms like Iterated-KSP and Iterated-EC from [Vigliotti and Batista, 2014a].

They are very fast too, with execution times some orders of magnitude lower than the other existing algorithms.

Comparing the three proposed algorithms (FFD, PABFD and GPABFD), there is not much difference in energy consumption, but FFD is still faster than PABFD and GPABFD. This works concludes that FFD algorithm is an interesting candidate for cloud providers being both efficient in real workloads and fast.

For future work, there are other candidates to solve the problem of VM placement, including ant colonies optimization, dynamic programming and sharing aware algorithms. These algorithms could be compared with FFD, PABFD and GPABFD to show their advantages and disadvantages compared to our approach.

Another proposal for future work is to implement support in pyCloudSim to optimize an existing VM allocation, instead of just the initial allocation. This will probably need support for multithreading capabilities, so each physical machine run in a separate thread instead of the current solution of only one thread.

References

Armbrust, M., Stoica, I., Zaharia, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., and Rabkin, A. (2010). A view of cloud computing. *Communications of the ACM*, 53(4):50.

Barroso, L. A. and Hölzle, U. (2007). The Case for Energy-Proportional Computing. *IEEE Computer Society*, (December):33–37.

Beloglazov, A. (2013). *Energy-efficient management of virtual machines in data centers for cloud computing*. PhD thesis, The University of Melbourne.

Beloglazov, A., Abawajy, J., and Buyya, R. (2012). Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing. *Future Generation Computer Systems*, 28(5):755–768.

Beloglazov, A. and Buyya, R. (2010). Energy Efficient Resource Management in Virtualized Cloud Data Centers. *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 826–831.

Beloglazov, A. and Buyya, R. (2012). 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.

Buyya, R., Calheiros, R. N., and Nikolay, G. (2013). CloudSim website. `www.cloudbus.org/cloudsim/`. Accessed: 2013-12-04.

dos Santos Júnior, M. R. (2014). Mecanismos para consolidação de servidores. Master's thesis, Universidade de São Paulo.

Esnault, A. (2012). Energy-Aware Distributed Ant Colony Based Virtual Machine Consolidation in IaaS Clouds.

Feller, E., Rilling, L., and Morin, C. (2011). Energy-aware ant colony based workload placement in clouds. In *Proceedings of the 2011 IEEE/ACM 12th International Conference on Grid Computing*, GRID '11, pages 26–33, Washington, DC, USA. IEEE Computer Society.

Jung, G., Hiltunen, M., Joshi, K., Schlichting, R., and Pu, C. (2010). Mistral: Dynamically managing power, performance, and adaptation cost in cloud infrastructures. In *Proceedings of the 30th IEEE ICDCS*, pages 62–73.

Koomey, J. G. (2011). Growth in data center electricity use 2005 to 2010. *A report by Analytical Press, completed at the request of The New York Times*.

Reiss, C., Tumanov, A., Ganger, G. R., Katz, R. H., and Kozuch, M. A. (2012). Heterogeneity and dynamicity of clouds at scale: Google trace analysis. *Proceedings of the Third ACM Symposium on Cloud Computing*, page 7.

Reiss, C., Wilkes, J., and Hellerstein, J. L. (2011). Google cluster-usage traces: format+ schema. *Google Inc., White Paper*, pages 1–14.

Suoerv, T. and Johnson, D. S. (1973). *Near-optimal bin packing algorithms*. PhD thesis, Massachusetts Institute of Technology.

Vigliotti, A. P. M. d. l. F. and Batista, D. M. (2014a). Energy-efficient virtual machines placement. In *Proceedings of the SBRC 2014*, pages 1–8.

Vigliotti, A. P. M. d. l. F. and Batista, D. M. (2014b). A green Network-Aware VMs placement mechanism. In *Proceedings of the IEEE Globecom 2014*, page To appear.

Wilkes, J. and Reiss, C. (2013). Google Cluster Data Website. `https://code.google.com/p/googleclusterdata/wiki/ClusterData2011_2`. Accessed: 2014-12-12.

Yue, M. (1991). A simple proof of the inequality $FFD\ (L) \leq\ 11/9OPT\ (L) + 1, \forall\ L$ for the FFD bin-packing algorithm. *Acta mathematicae applicatae sinica*, 7(4):321–331.