# Interplay of Virtual Machine Selection and Virtual Machine Placement

Zoltán Ádám Mann

# Interplay of virtual machine selection and virtual machine placement

Zoltán Ádám Mann

paluno – The Ruhr Institute for Software Technology
University of Duisburg-Essen, Essen, Germany

**Abstract.** Previous work on optimizing resource provisioning in virtualized environments focused either on mapping virtual machines to physical machines (i.e., virtual machine placement) or mapping computational tasks to virtual machines (i.e., virtual machine selection). In this paper, we investigate how these two optimization problems influence each other. Our study shows that exploiting knowledge about the physical machines and about the virtual machine placement algorithm in the course of virtual machine selection leads to better overall results than considering the two problems in isolation.

## 1 Introduction

As cloud data centers are serving an ever growing demand for computation, storage, and networking, their efficient operation has become a high priority. On one hand, the operation of data centers incurs huge costs and environmental impact. According to a recent study, data center electricity consumption in the USA alone will increase to 140 billion kWh per year by 2020, costing US businesses 13 billion USD annually in electricity bills and emitting nearly 100 million tons of $CO_2$ per year [25]. On the other hand, servers often run with low utilization – in fact, a significant percentage of running servers do not do any useful work [1].

Virtualization has been widely adopted in data centers to consolidate workload on the necessary number of physical machines (PMs), with the aim of achieving high utilization and switching off unused PMs to save energy. For this purpose, virtual machines (VMs) are used as the virtual infrastructure for running the workload. Live migration technology makes it possible to migrate a running VM from one PM to another one without noticeable downtime. This way, data center operators can react to changes in the workload and always use the appropriate number of turned-on PMs to accommodate the active VMs, taking into account their current resource needs. However, too aggressive consolidation must be avoided because overloading physical resources leads to performance degradation. Furthermore, live migration of VMs incurs increased resource consumption, so that the number of migrations must be limited.

Optimization relating to the management of VMs has received considerable attention in the last couple of years because of its impact on costs, application

performance and carbon emission [29]. As shown in our recent survey [21], most previous research efforts fall into one of two categories: VM placement and VM selection. The goal of *VM placement* is to determine a mapping of VMs to PMs with the objective of minimizing energy consumption while obeying performance constraints and keeping the number of VM migrations low [23]. On the other hand, *VM selection* is concerned with assigning computational tasks to VMs.

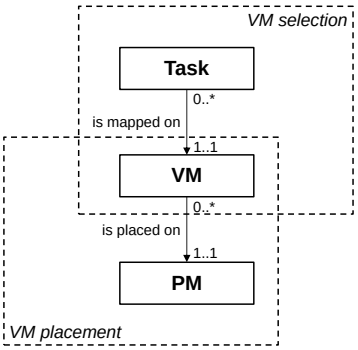
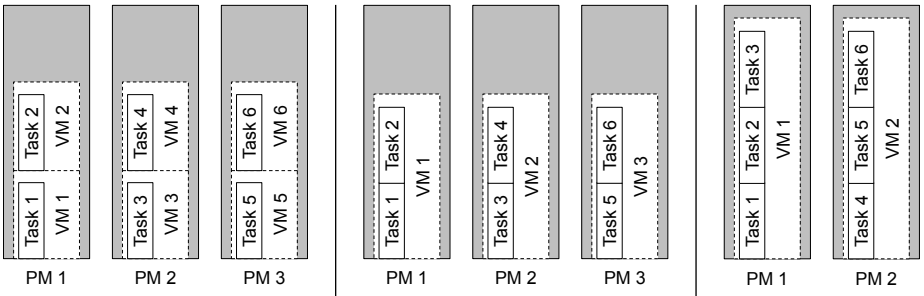



**Fig. 1.** Overview of VM selection and VM placement

The separation between the VM placement problem and the VM selection problem (see Fig. 1) is rooted in the fact that the two kinds of optimization are performed by different actors: VM placement is carried out by cloud providers, whereas VM selection is done by cloud users. Moreover, the two problems are quite different: VM placement is about physical resources, power consumption, and live migration, whereas VM selection is typically concerned with lease costs and application-level performance metrics. The central notion that connects the two perspectives is the VM.

Although VMs play an important role, we argue that VMs are just a tool for mapping users' tasks to PMs in a safe and manageable fashion. Users' main objective is to find hosts for their tasks, providers' objective is to utilize their infrastructure by accommodating workload that is valuable for users. VMs can be seen as wrappers around tasks that make all this possible, at the price of some overhead. In this respect, VM placement and VM selection are just two sides of the same coin. Most importantly, the two problems influence each other.

A simplified example is shown in Fig. 2. Here, we consider PMs of capacity 1 (for this conceptual example, the exact metric is unimportant) and six tasks with resource need 0.3 each. Further, we assume that a VM adds an overhead of 0.05 to the size of the contained task(s) in terms of resource consumption. The three subfigures show the effect of different VM selection algorithms on VM placement. In Fig. 2(a), the VM selection algorithm selects a dedicated VM for each task, resulting in 6 VMs of size 0.35 each, the placement of which requires at least 3 PMs. In Fig. 2(b), tasks are grouped pairwise into VMs, resulting




(a) Separate VM for each task

(b) 2 tasks per VM

(c) 3 tasks per VM

**Fig. 2.** Examples of the impact of VM selection decisions on the possibilities of VM placement

in 3 VMs of size 0.65 each, the placement of which again requires 3 PMs. In Fig. 2(c), groups of 3 tasks are mapped to VMs, resulting in 2 VMs of size 0.95 each, and these can be hosted by 2 PMs. Therefore, this third scenario leads to approximately 33% energy savings. However, if we continue this line of thought and map 4 tasks into a single VM, this would result in a VM of size 1.25, which cannot be accommodated by the available PMs (or, if we map such a VM to one of the available PMs, this will lead to severe resource overload).

As demonstrated by this example, VM selection influences VM placement in a non-trivial way. Therefore we argue that, at least in a private cloud setting, where VM selection and VM placement are in the hand of the same organization, the two kinds of optimization should be carried out in a closely coupled way. Even in a public cloud setting, it is important to understand the inter-dependence of the two optimization problems, so that the provider can motivate users (by means of appropriate pricing mechanisms) to use VM sizes that allow good placement. So, the main question that this paper addresses can be summarized as follows: how to perform VM selection in such a way that the resulting VMs allow an advantageous VM placement?

In particular, we show that incorporating knowledge about the capacity of the PMs into the VM selection algorithm already leads to substantial improvement compared to PM-oblivious VM selection. Further improvement is possible if the VM selection algorithm also exploits knowledge about the current placement of VMs on PMs as well as about the VM placement optimization algorithm.

## 2 Previous work

As shown in our recent survey [21], most previous research efforts on VM mapping problems fall into one of two categories: VM placement is concerned with mapping VMs to PMs in a data center, while VM selection considers the problem of mapping tasks to VMs.

### 2.1 VM placement

Several algorithms have been suggested for VM placement. Some focus only on the computational capacity of PMs and computational load of VMs [2, 4, 7, 14, 34, 22], whereas others also include other resources like memory, I/O, storage, or network bandwidth [5, 11, 24, 31].

One of the cost factors considered by most works is the number of active PMs because it largely determines the total energy consumption [3, 7, 13, 34]. Some also take into account the load-dependent dynamic power consumption of PMs [4, 14, 11, 30]. A further objective of some works is to minimize the number of overloaded PMs because of the performance degradation that results from overloads [4, 7, 31, 34]. Some works also considered the cost of VM migrations [4, 7, 13, 30].

The special case of the VM placement problem in which a single resource type is considered and the only objective is to minimize the number of used PMs is equivalent to the well-known bin-packing problem. On the one hand, this means that the VM placement problem is strongly NP-hard so that the existence of an efficient exact algorithm is very unlikely. On the other hand, simple packing heuristics like First-Fit (FF), Best-Fit (BF), Worst-Fit (WF) and First-Fit-Decreasing (FFD) are known to perform very well on bin-packing. Hence, several authors proposed to adopt such heuristics to the VM placement problem [4, 19, 31, 34, 3, 2, 14].

### 2.2 VM selection

Concerning VM selection, also many different problem models have been suggested. Similarly to the VM placement problem, most works focus on computational power [8, 20, 27, 35] but some consider also other resource types like memory [18, 26]. The main optimization objective is to find the best trade-off between performance and VM lease costs, which typically means that either the minimum required performance is given and costs must be minimized or the acceptable costs are constrained and performance must be maximized.

Several different models have been investigated also in terms of VM lease costs. Most works consider costs proportional to VM usage time [6, 20, 32, 33, 35, 15], but some also add fees depending on consumed resource usage [18, 26] or discounts for long-term VM rental [12, 18].

Existing VM selection algorithms assume that VMs have fixed rental fees and fixed capacities. However, in a private cloud, VM capacities can be arbitrarily chosen and also changed, and instead of rental fees, real operations costs have to be minimized, which are incurred at the level of PMs.

### 2.3 Inter-dependence of VM placement and VM selection

In contrast to the works cited above, we do not handle VM placement or VM selection in isolation, but are interested in their interplay. We are aware of two papers that have a somewhat similar aim. The recent work of Piraghaj et al. [28]

focuses on selecting optimal VM sizes based on the characteristics of the tasks to be allocated. The objective is to reduce energy consumption by minimizing resource wastage. Each VM is assumed to have a fixed size irrespective of its workload, and the difference between the VM's size and the total size of its workload is wasted.

In contrast, we assume that a VM's real size (what is taken into account by the provider in VM placement decisions) follows the resource requirements of its workload. The rationale is that resource usage is most of the time significantly below the peak, yielding a great opportunity for consolidating VMs based on their current load and continuously adapting the placement accordingly, always using just the necessary number of active PMs [34, 30]. Another important difference is that the work of Piraghaj et al. did not consider migrations, whereas we do. Through these differences we arrive at a more realistic model, in which the sought trade-offs and the objectives are also different (consolidation through migration versus minimization of wastage through sizing).

Ganesan et al. [10] consider a Software-as-a-Service provider that wants to allocate the components of its applications to VMs. The focus of the work is on VM sizing, namely, determining the dedicated and shared capacity for the VMs, based on past observations of the applications' workload. Their algorithm also outputs *recommendations* for VM placement, like which VMs can be placed statically and which ones need dynamic placement. However, the actual allocation of VMs to PMs is not carried out; they assume that it is done by some external algorithm. In contrast, we are interested in the impact of sizing on placement; it is unfortunately not possible to tell how good that approach is in this respect. Another limitation of that paper is the assumption that each application component is mapped to a separate VM, whereas we also allow to co-locate multiple tasks in the same VM.

## 3 Problem model

VM selection and VM placement are difficult problems on their own, so combining them results in a very complex problem. In this paper, we focus on the following aspects, related to performance and costs (and leave further aspects, such as security and fault tolerance, for future research):

- Energy consumption of the PMs, which depends on the number and load of turned-on PMs
- Overhead (extra resource consumption) of virtualization
- Overhead (extra time) associated with launching new VMs
- Overhead (extra resource consumption) of VM migrations
- Performance degradation resulting from PM overloads

It is important to note that the impact of these aspects are conflicting: e.g., because of the overheads of virtualization, it would be advantageous to combine into a single VM as many tasks as possible; on the other hand, too big VMs

limit the consolidation possibilities, thus potentially leading to higher energy consumption and/or more PM overloads.

We consider $d$ resource types; for example, if CPU and memory are considered, then $d = 2$. Each task $j$ has a size $s(j) \in \mathbb{R}_+^d$ describing its resource need according to the considered resource types. Similarly, the size of a VM $v$ is $s(v) \in \mathbb{R}_+^d$, the vector of its resource needs. Each task must be mapped to exactly one VM; a VM may accommodate multiple tasks. For a task $j$, $v(j)$ denotes its hosting VM; for a VM $v$, $T(v)$ denotes the set of tasks that it hosts. The size of a VM is determined by the size of the tasks it hosts:

$$s(v) = s_0 + \sum_{j \in T(v)} s(j), \tag{1}$$

where $s_0 \in \mathbb{R}_+^d$ is the size of an empty VM, representing the overhead of virtualization, in terms of extra resource consumption. This overhead stems from the (load-independent) resource needs of the guest operating system, hence a constant overhead is a good approximation, although for some resource types more sophisticated models of the virtualization overhead might be more realistic.

Each PM $p$ has a capacity $c(p) \in \mathbb{R}_+^d$. Each VM $v$ must be hosted by exactly one PM $p(v)$; a PM $p$ may host multiple VMs and their set is denoted by $V(p)$. To guarantee the required level of performance, the following capacity constraint must hold:

$$s(p) = \sum_{v \in V(p)} s(v) \leq c(p). \tag{2}$$

Note that here "$\leq$" means that in each dimension the left-hand side must be less than or equal to the right-hand side.

The power consumption of a PM is a function of its CPU load. As in several previous works [2, 17, 11, 30], we use a linear approximation, i.e., the power consumption of a PM with CPU capacity $c$ and CPU load $x$ is given by

$$W(x) = W_{min} + (W_{max} - W_{min}) \cdot x/c, \tag{3}$$

where $W_{min}$ and $W_{max}$ are the minimum and maximum power consumption of the PM, respectively.

To simplify the presentation, we assume that each PM has the same capacity and the same power consumption characteristics.

The following decision points – and hence optimization opportunities – exist:

- VM selection:
  - If a new task arrives, it must be mapped to a VM. For this purpose, either one of the existing VMs must be selected or a new VM must be created.
  - If a VM becomes empty, it can be destroyed or kept for later reuse.
- VM placement:
  - If a new VM is created, it must be mapped to a PM. For this purpose, either one of the turned-on PMs must be chosen or a new PM must be turned on.

- A VM can be migrated from its old PM to a new one.
- If a PM becomes empty, it can be switched off.

Note that the VM placement can be re-optimized again and again with live migrations; in contrast, a task is mapped to one VM for its entire life[1].

The aim is to make these decisions in such a way that the performance of the system is as high as possible (requiring the number of migrations, the number of PM overloads and the number of VM launches to be minimized) and its cost is as low as possible (requiring the number of turned-on PMs and their energy consumption to be minimized).

## 4 VM selection and VM placement algorithms to assess

Our aim is to investigate the interplay between VM selection and VM placement. For both subproblems, several algorithms are conceivable, leading to a huge number of possible combinations. To keep the number of experiments manageable, we chose to fix an algorithm for VM placement and consider a series of algorithms for VM selection that differ in how much knowledge they exploit about the underlying VM placement.

Specifically, we use the algorithm of Beloglazov et al. for VM placement as a representative example of previously proposed VM placement algorithms, which was shown to achieve a good trade-off between energy consumption, number of migrations, and number of PM overloads [2]. Whenever a new VM is requested, the first PM that has sufficient free capacity is chosen to host it or a new PM is turned on if no such PM could be found. Moreover, the VM placement is re-optimized at regular time intervals, consisting of the following steps:

1. From each overloaded PM, a minimal set of VMs is removed so that the PM is not overloaded anymore. (A PM is overloaded if its load exceeds its capacity in at least one dimension.)
2. From each underloaded PM, all its VMs are removed. (A PM $p$ is underloaded if $s(p) \leq \lambda \cdot c(p)$, i.e., its load is below $\lambda$ times its capacity in each dimension, where $0 < \lambda < 1$ is a given constant.)
3. The list of removed VMs is sorted in decreasing order of CPU load.
4. For each removed VM, the first PM with sufficient free capacity is chosen.
5. Emptied PMs are switched off.

Next, the considered VM selection algorithms are presented. We start with the ones that are completely oblivious of the underlying PMs and the VM placement algorithm, and then gradually increase the exploited knowledge:

– **Simple.** This approach creates a new VM for each task, like in [10].

[1] Although some applications may support the migration of individual tasks, but this cannot be assumed in general.

- **Multiple($k$).** Tasks are assigned to VMs in groups of $k$, where $k \in \mathbb{Z}_+$ is a given constant. In the order as tasks arrive, a VM is created for task 1, which is then used for tasks $2, \ldots, k$ as well. For task $k+1$, a new VM is created, which is used for tasks $k+2, \ldots, 2k$ as well, and so on.
- **Maxsize($\mu$).** In contrast to the previous algorithms, this one exploits some knowledge about the PMs. The idea here is to ensure that the size of each VM is at most $\mu \cdot c$, where $0 < \mu \leq 1$ is a given constant and $c$ is the capacity vector of the PMs. When a new task arrives, it is checked which of the existing VMs could host it without exceeding the $\mu \cdot c$ threshold. If no such VM exists, a new one is created. If multiple appropriate VMs exist, one of them is selected according to a *selection policy*, which can be one of FF, BF, WF. Since these policies work in a single dimension whereas VM and task sizes are multi-dimensional, a *selection metric* is used to convert a $d$-dimensional vector to a number. Possible metrics are the sum, product, maximum, or minimum of the coordinates, the length of the vector, or the imbalance of the vector, defined as the difference between the maximum and minimum coordinate.
- **Consolidation-friendly.** This algorithm exploits not only knowledge about the PMs but also about the current VM placement and the VM placement algorithm. When a new task arrives, it is first checked whether there is a PM that is not underloaded and has enough free capacity to accommodate the new task. Such PMs are preferred because in this case, no overhead nor a PM overload is generated, and also no consolidation opportunity is obstructed. When there are multiple such PMs, one of them is selected using an appropriate PM selection heuristic and metric, similarly as in the Maxsize algorithm. When no such PM exists, then one of the underloaded PMs is selected with the same policy and metric. In this case, a consolidation opportunity is obstructed, but still no overhead is generated. In any case, after a PM has been chosen, one of its VMs has to be selected using a selection policy and metric, and the new task is mapped to this VM. Finally, if no appropriate PM could be found, then a new VM is created to accommodate the new task.

The Simple, Multiple, and Maxsize algorithms are used as representatives of the class of previously proposed VM selection algorithms. However, the Maxsize algorithm is already more advanced than the existing algorithms because existing algorithms just assume some given VM size limit without considering how this size limit should relate to the PMs' capacity. The Consolidation-friendly algorithm was designed by us specifically to show how detailed knowledge about the underlying VM placement algorithm can be exploited during VM selection.

## 5 Empirical results

All the proposed algorithms were implemented in a simulation framework in C++, which is freely available from `https://sourceforge.net/p/vm-alloc/task_vm_pm/`. To obtain practically relevant results, we used real-world test

data. For the tasks, we used a real workload trace from the Grid Workloads Archive, namely the AuverGrid trace, available from `http://gwa.ewi.tudelft.nl/datasets/gwa-t-4-auvergrid`. From the trace, we used the first 10,000 tasks that had valid CPU and memory usage data. The simulated time (i.e., the time between the start of the first task and the end of the last one) is a bit over 29 days, thus giving sufficient exposure to practical workload patterns.

As PMs, we simulated HP ProLiant DL380 G7 servers with Intel Xeon E5640 quad-core CPU and 16 GB RAM. Their power consumption varies from 280W (zero load) to 540W (full load) [16]. Throughout the experiments, we focus on two resource types: CPU and memory, i.e., $d = 2$. For memory sizes, absolute values are used in MB. For CPU capacities and loads, relative values are used, where 100% is the capacity of one physical CPU core. Concerning virtualization overhead, previous work reported 5-15% for the CPU [36] and 107-566 MB for memory [9]. In our experiments, we use 10% CPU overhead and 200 MB memory overhead. The parameter of the VM placement algorithm, $\lambda$, is set to 0.4 as in [2]. The VM placement is re-optimized every 5 minutes.

For each evaluated algorithm, the following *quality metrics* were measured:

- Total energy consumption
- Average number of turned-on PMs
- Maximum number of turned-on PMs
- Maximum number of concurrently used VMs (as indication of the necessary number of VM launches)
- Number of migrations
- Number of PM overloads

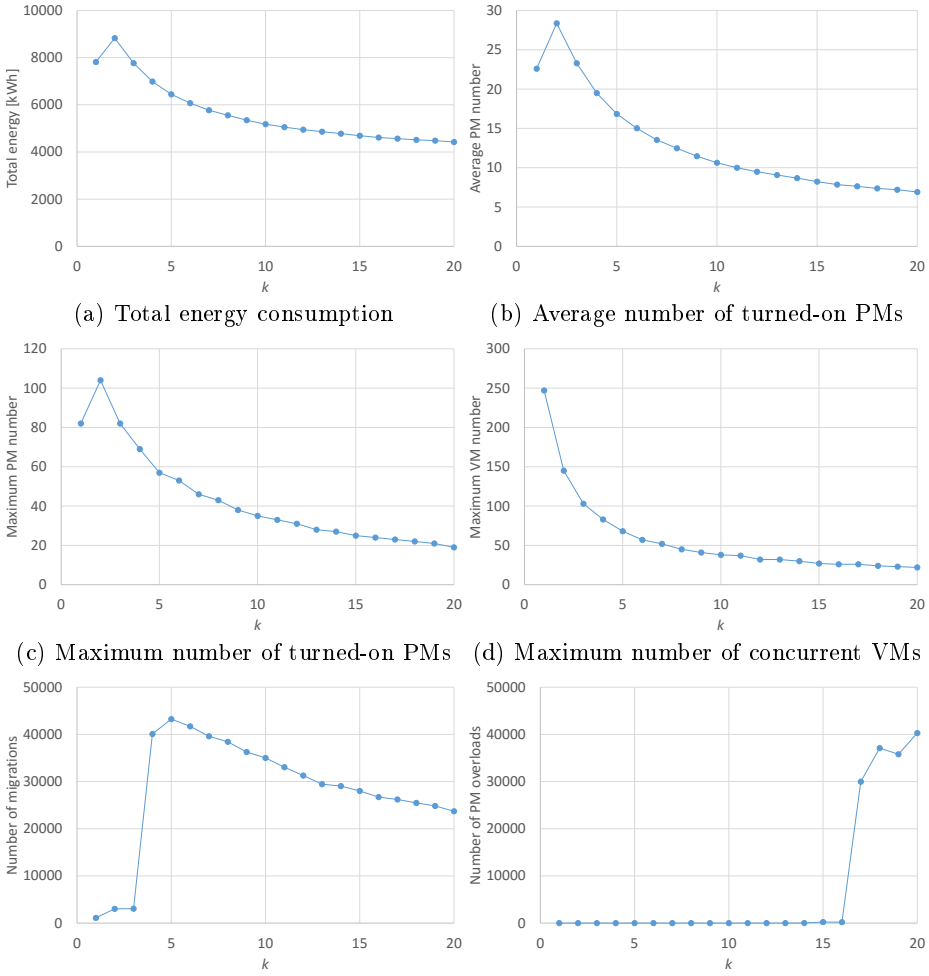
For each quality metric, smaller numbers are better.

First, the Simple and Multiple($k$) algorithms are evaluated. Note that Multiple(1) is exactly the Simple algorithm.

The results are shown in Fig. 3. As can be seen, the total energy consumption and the average and maximum number of turned-on PMs all show a similar pattern with an increase at the beginning, maximum at $k = 2$, and decrease afterwards. This can be attributed to two conflicting effects. With increasing $k$, the average VM size grows and the number of VMs decreases, which leads on the one hand to less consolidation opportunities, on the other hand to a decrease of the resource consumption overhead.

Based on these metrics, higher values of $k$ seem preferable. However, from Fig. 3(e) and 3(f) it can be seen that the number of PM overloads skyrockets at $k = 17$ and the number of migrations is exorbitantly large already for $k \geq 4$. Although there is a slow decrease of the number of migrations afterwards (thanks to the decreasing number of consolidation opportunities), but an acceptable level is reached only for very high values of $k$, where the number of overloads is already prohibitively large. Thus, the best compromise seems to be the case $k = 3$.

The biggest problem with the Multiple($k$) algorithm is that the value of $k$ at which the sudden explosion of the number of migrations and number of overloads takes place cannot be predicted nor controlled. This depends on several factors, like the capacity of the PMs and the workload's characteristics. Therefore, this




(a) Total energy consumption (b) Average number of turned-on PMs

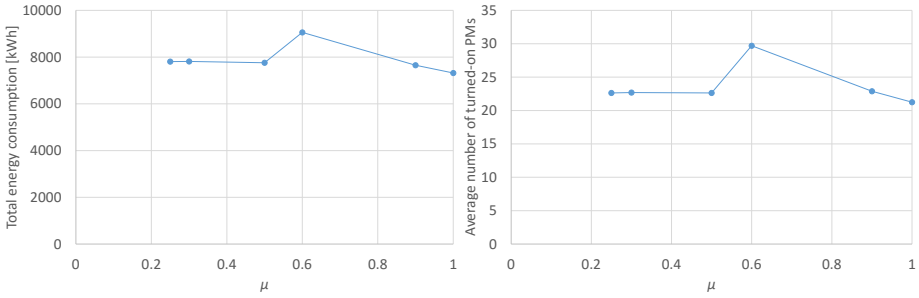
(c) Maximum number of turned-on PMs (d) Maximum number of concurrent VMs

(e) Number of migrations (f) Number of PM overloads

**Fig. 3.** Results of the Multiple($k$) algorithm for different values of $k$

algorithm is dangerous. Small values of $k$ are safer but lead to higher costs (higher energy consumption) and lower performance (more VMs need to be launched).

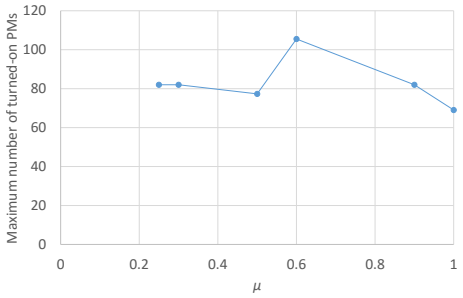
Next, we evaluated the Maxsize($\mu$) algorithm. More precisely, this is a family of algorithms, characterized by the value of $\mu$, the used selection policy, and selection metric. We tested 6 different values for $\mu$ (0.25, 0.3, 0.5, 0.6, 0.9, 1.0), 3 selection policies (FF, BF, WF), and 6 selection metrics (sum, product, maximum, minimum, imbalance, length). The effect of $\mu$ on the different quality metrics is shown in Fig. 4 (each data point corresponds to one value of $\mu$ and the average according to the two other parameters).



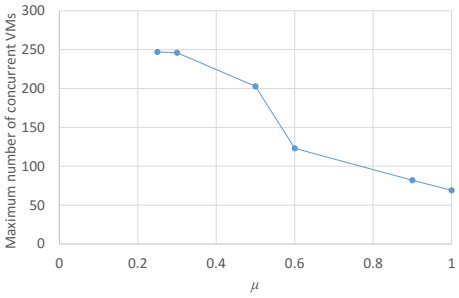

(a) Total energy consumption



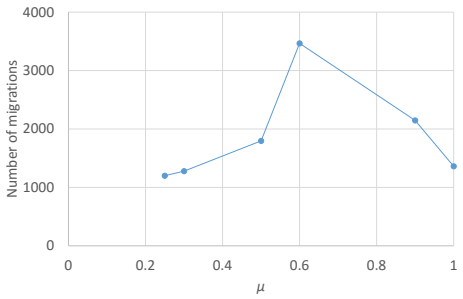

(b) Average number of turned-on PMs



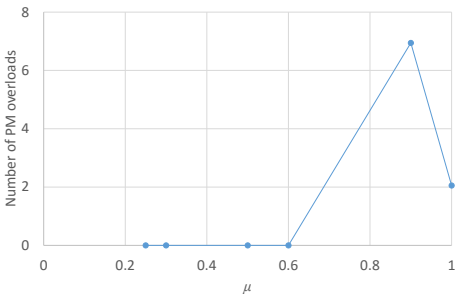

(c) Maximum number of turned-on PMs




(d) Maximum number of concurrent VMs




(e) Number of migrations




(f) Number of PM overloads

**Fig. 4.** Results of the Maxsize($\mu$) algorithm for different values of $\mu$

In contrast to the previous experience with the Multiple($k$) algorithm, these figures show no sudden large increases in any quality metric. This is a big advantage: apparently, the knowledge of the PMs' capacity allows the algorithm much better control of the PMs' utilization, leading to safer operation. Only the number of migrations (Fig. 4(e)) and the number of PM overloads (Fig. 4(f)) show some seemingly significant oscillations; however, when compared with the corresponding results of the Multiple($k$) algorithm, we can see that these oscillations span actually a quite small range.

Looking at the details, it can be observed that $\mu = 0.6$ leads to significantly worse performance than $\mu = 0.5$ according to most quality metrics. This is

logical, since VMs of size at most $0.5 \cdot c$ can be pairwise consolidated to a PM, but if their size can go somewhat beyond this limit, then the opportunities for consolidation decrease. This shows once again the importance of PM-level knowledge in VM selection. According to most quality metrics, $\mu = 1$ is the best choice.

The effect of the selection policy and selection metric on the considered quality metrics is much smaller than the effect of $\mu$. Therefore, to save space, these results are not shown (but they can also be found in our online repository mentioned above). The FF policy and the minimum metric were chosen as best, although their advantage over the others is small.

For evaluating the Consolidation-friendly algorithm, several parameters need to be tuned: 3 possibilities for the PM selection policy (FF, BF, WF), 6 possibilities for the PM selection metric (sum, product, maximum, minimum, imbalance, length), 2 VM selection policies (maximize, minimize), and 6 VM selection metrics (the same as for PMs), resulting in 216 possible configurations. Similarly as in the case of the Maxsize($\mu$) algorithm, none of the selection policies and selection metrics had a profound impact on the investigated quality metrics. This means that the algorithm is robust in the sense that changes in the parameters do not lead to abrupt changes in its behavior (in contrast to the Multiple($k$) algorithm). The details of fine-tuning the algorithm are skipped because of space constraints. The chosen best configuration uses FF and the minimum metric for PM selection (similarly to the Maxsize($\mu$) algorithm) and the maximize policy and product metric for VM selection.

**Table 1.** Comparison of the algorithms' results

| Algorithm | Energy | Avg #PM | Max #PM | Max #VM | Migrations | Overloads |
|---|---|---|---|---|---|---|
| Multiple(3) | 7766.14 | 23.28 | 82 | 103 | 3062 | 0 |
| Maxsize(1) | 7282.90 | 21.07 | 68 | 68 | 1209 | 0 |
| Consolidation-friendly | 7257.91 | 20.91 | 68 | 70 | 826 | 0 |

Finally, the results of the chosen best configuration of each algorithm are compared to each other in Table 1. As can be seen, the Multiple($k$) algorithm is significantly outperformed by the two others according to each quality metric. Moreover, the Consolidation-friendly algorithm offers considerable advantage over the Maxsize($\mu$) algorithm in terms of the number of migrations, and also some improvement in energy consumption and the average number of turned-on PMs, at the price of a marginal increase of the maximum number of concurrently active VMs.

## 6 Conclusions

In this paper, we analyzed the interplay of VM selection and VM placement algorithms. By fixing the VM placement algorithm and considering a series of

VM selection algorithms that exploit an increasing amount of knowledge about the underlying PMs and the VM placement, we showed the importance of such information. Specifically, already the knowledge of the PMs' capacity makes VM selection more efficient in terms of cost and also much more resilient to the negative impact of inappropriate parameter choices. Adding more knowledge about the details of the VM placement algorithm leads to a further improvement, especially in terms of the number of migrations.

This insight can be used especially in a private cloud setting, where all details of the PMs and the VM placement are available. In this case, exploiting this knowledge in the sizing of VMs and the mapping of tasks to VMs leads to considerable improvements. In a public cloud setting, the provider who has the knowledge about the PMs and the VM placement should shape usage-based pricing schemes in such a way that it corresponds to the real costs, so that users are incentivized to use actual VM sizes that lead to good consolidation.

Directions for future research include the investigation of further aspects that make the interplay of VM selection and VM placement even more complex, such as data transfer among the tasks or security and reliability considerations. Moreover, a software engineering challenge is how to design the interface between VM selection and VM placement tools so that they can exchange the necessary pieces of information.

## Acknowledgments

A part of this work was carried out when Z. Á. Mann was with Budapest University of Technology and Economics. This work was partially supported by the Hungarian Scientific Research Fund (Grant Nr. OTKA 108947) and the European Union's 7th Framework Programme (FP7/2007-2013) under grant agreement 610802 (CloudWave).

## References

1. Anthesis Group: 30% of servers are sitting "comatose". `http://anthesisgroup.com/30-of-servers-are-sitting-comatose/` (2015)
2. Beloglazov, A., Abawajy, J., Buyya, R.: Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Generation Computer Systems 28, 755–768 (2012)
3. Beloglazov, A., Buyya, R.: Energy efficient allocation of virtual machines in cloud data centers. In: 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing. pp. 577–578 (2010)
4. 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)
5. Biran, O., Corradi, A., Fanelli, M., Foschini, L., Nus, A., Raz, D., Silvera, E.: A stable network-aware VM placement for cloud systems. In: Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (Ccgrid 2012). pp. 498–506. IEEE Computer Society (2012)

6. Bittencourt, L.F., Madeira, E.R., da Fonseca, N.L.: Scheduling in hybrid clouds. IEEE Communications Magazine 50(9), 42–47 (2012)
7. Breitgand, D., Epstein, A.: SLA-aware placement of multi-virtual machine elastic services in compute clouds. In: 12th IFIP/IEEE International Symposium on Integrated Network Management. pp. 161–168 (2011)
8. Candeia, D., Araújo, R., Lopes, R., Brasileiro, F.: Investigating business-driven cloudburst schedulers for e-science bag-of-tasks applications. In: 2nd IEEE International Conference on Cloud Computing Technology and Science. pp. 343–350 (2010)
9. Chang, C.R., Wu, J.J., Liu, P.: An empirical study on memory sharing of virtual machines for server consolidation. In: IEEE 9th International Symposium on Parallel and Distributed Processing with Applications. pp. 244–249 (2011)
10. Ganesan, R., Sarkar, S., Narayan, A.: Analysis of SaaS business platform workloads for sizing and collocation. In: IEEE 5th International Conference on Cloud Computing (CLOUD). pp. 868–875 (2012)
11. Gao, Y., Guan, H., Qi, Z., Hou, Y., Liu, L.: A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. Journal of Computer and System Sciences 79, 1230–1242 (2013)
12. Genez, T.A.L., Bittencourt, L.F., Madeira, E.R.M.: Workflow scheduling for SaaS/PaaS cloud providers considering two SLA levels. In: Network Operations and Management Symposium (NOMS). pp. 906–912. IEEE (2012)
13. Gmach, D., Rolia, J., Cherkasova, L., Kemper, A.: Resource pool management: Reactive versus proactive or let's be friends. Computer Networks 53(17), 2905–2922 (2009)
14. Guazzone, M., Anglano, C., Canonico, M.: Exploiting VM migration for the automated power and performance management of green cloud computing systems. In: 1st International Workshop on Energy Efficient Data Centers. pp. 81–92. Springer (2012)
15. Hoenisch, P., Hochreiner, C., Schuller, D., Schulte, S., Mendling, J., Dustdar, S.: Cost-efficient scheduling of elastic processes in hybrid clouds. In: IEEE 8th International Conference on Cloud Computing. pp. 17–24 (2015)
16. HP: Power efficiency and power management in HP ProLiant servers. `http://h10032.www1.hp.com/ctg/Manual/c03161908.pdf` (2012)
17. Jung, G., Hiltunen, M.A., Joshi, K.R., Schlichting, R.D., Pu, C.: Mistral: Dynamically managing power, performance, and adaptation cost in cloud infrastructures. In: IEEE 30th International Conference on Distributed Computing Systems. pp. 62–73 (2010)
18. Lampe, U., Siebenhaar, M., Hans, R., Schuller, D., Steinmetz, R.: Let the clouds compute: cost-efficient workload distribution in infrastructure clouds. In: Proceedings of the 9th International Conference on Economics of Grids, Clouds, Systems, and Services (GECON 2012). pp. 91–101. Springer (2012)
19. Li, W., Tordsson, J., Elmroth, E.: Virtual machine placement for predictable and time-constrained peak loads. In: Proceedings of the 8th International Conference on Economics of Grids, Clouds, Systems, and Services (GECON 2011). pp. 120–134. Springer (2011)
20. Li, W., Tordsson, J., Elmroth, E.: Modeling for dynamic cloud scheduling via migration of virtual machines. In: Proceedings of the 3rd IEEE International Conference on Cloud Computing Technology and Science. pp. 163–171 (2011)
21. Mann, Z.A.: Allocation of virtual machines in cloud data centers – a survey of problem models and optimization algorithms. ACM Computing Surveys 48(1) (2015)

22. Mann, Z.A.: Rigorous results on the effectiveness of some heuristics for the consolidation of virtual machines in a cloud data center. Future Generation Computer Systems 51, 1–6 (2015)
23. Mann, Z.A.: A taxonomy for the virtual machine allocation problem. International Journal of Mathematical Models and Methods in Applied Sciences 9, 269–276 (2015)
24. Mishra, M., Sahoo, A.: On theory of vm placement: Anomalies in existing methodologies and their mitigation using a novel vector based approach. In: IEEE International Conference on Cloud Computing. pp. 275–282 (2011)
25. Natural Resources Defense Council: Scaling up energy efficiency across the data center industry: Evaluating key drivers and barriers. `http://www.nrdc.org/energy/files/data-center-efficiency-assessment-IP.pdf` (2014)
26. Oliveira, D., Ocana, K.A.C.S., Baiao, F., Mattoso, M.: A provenance-based adaptive scheduling heuristic for parallel scientific workflows in clouds. Journal of Grid Computing 10, 521–552 (2012)
27. Pandey, S., Wu, L., Guru, S.M., Buyya, R.: A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In: 24th IEEE International Conference on Advanced Information Networking and Applications (AINA). pp. 400–407. IEEE (2010)
28. Piraghaj, S.F., Dastjerdi, A.V., Calheiros, R.N., Buyya, R.: Efficient virtual machine sizing for hosting containers as a service. In: IEEE World Congress on Services. pp. 31–38 (2015)
29. Sáez, S.G., Andrikopoulos, V., Hahn, M., Karastoyanova, D., Leymann, F., Skouradaki, M., Vukojevic-Haupt, K.: Performance and Cost Trade-Off in IaaS Environments: A Scientific Workflow Simulation Environment Case Study, pp. 153–170. Springer (2015)
30. Svärd, P., Li, W., Wadbro, E., Tordsson, J., Elmroth, E.: Continuous datacenter consolidation. In: IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom). pp. 387–396 (2015)
31. Tomás, L., Tordsson, J.: An autonomic approach to risk-aware data center overbooking. IEEE Transactions on Cloud Computing 2(3), 292–305 (2014)
32. Tordsson, J., Montero, R.S., Moreno-Vozmediano, R., Llorente, I.M.: Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers. Future Generation Computer Systems 28(2), 358–367 (2012)
33. Tsamoura, E., Gounaris, A., Tsichlas, K.: Multi-objective optimization of data flows in a multi-cloud environment. In: Proceedings of the Second Workshop on Data Analytics in the Cloud. pp. 6–10 (2013)
34. Verma, A., Dasgupta, G., Nayak, T.K., De, P., Kothari, R.: Server workload analysis for power minimization using consolidation. In: Proceedings of the 2009 USENIX Annual Technical Conference. pp. 355–368 (2009)
35. Villegas, D., Antoniou, A., Sadjadi, S.M., Iosup, A.: An analysis of provisioning and allocation policies for infrastructure-as-a-service clouds. In: 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid). pp. 612–619 (2012)
36. Zhou, Y., Zhang, Y., Liu, H., Xiong, N., Vasilakos, A.V.: A bare-metal and asymmetric partitioning approach to client virtualization. IEEE Transactions on Services Computing 7(1), 40–53 (2014)