A Virtual Machine Placement Algorithm for Resource Allocation in Cloud-Based Environments

David H. S. Lima[1,3], Andre L. L. Aquino[2], Marilia Curado[1]

[1]University of Coimbra,
Centre for Informatics and Systems of the University of Coimbra,
Department of Informatics Engineering, Portugal

[2]Computer Institute, Federal University of Alagoas, Maceió, Brazil

[3]Federal Institute of Alagoas, Rio Largo, Brazil

{dhlima, marilia}@dei.uc.pt, alla@laccan.ufal.br

Abstract. *This research paper presents a Virtual Machine Placement algorithm designed to optimize the service allocation process, which is one of the most significant challenges in resource allocation and management. Our solution employs a ranking strategy based on global resource scarcity, where we consider all the information contained in datacenter hosts during the allocation process. We compare our approach with two different literature-based methods regarding task allocation time and the number of allocated resources. To evaluate our proposal, we consider scenarios in which requisitions arrive at different times or simultaneously. The first scenario represents a realistic situation where no knowledge is available regarding task arrival. Meanwhile, the second scenario represents a dense situation where many tasks arrive together. Our evaluations show that our solution yields better results in both cases, especially when requisitions arrive simultaneously. Our solution reduced the allocation time by around 25% on average and achieved better load distribution among hosts.*

1. Introduction

Cloud computing [Taleb et al. 2017] is an on-demand Internet-based computing service. It shares computing resources among the users via the Internet based on the pay-for-use model. This paradigm offers computing resources (e.g., servers, storage, networks, and applications) as a service to the user. The cloud organizes the computing resources at one place called cloud data center and manages these data centers by an organization called cloud service providers (CSPs) [Rimal et al. 2009]. Each data center contains thousands of physical machines (PM) and resides in a different geographic point for performance enhancement and reliability while connected to other data centers with high-speed telecommunication links. Companies share their data centers' PMs with cloud technology as a virtualized resource pool, known as Virtual Machines (VMs). Cloud providers can create multiple VM instances on physical machines to improve resource utilization.

Even though virtualization technology offers a potential platform to improve resource utilization, some issues still obstruct the maximum hardware utilization. One of these issues is the deployment of VMs in a virtualized data center. Having a data center with an enormous number of PMs, the primary challenge is mapping each created VM to the most suitable PM, achieving the maximum resource utilization for each one.

This problem is known as Virtual Machine Placement (VMP) [Usmani and Singh 2016, Silva Filho et al. 2018, Talebian et al. 2020]. Proposals allow providers to control the usage of their resources to reduce the number of unused PMs while meeting specific objectives defined in the Service Level Agreement (SLA).

The placement of VMs over physical hosts plays a critical role in cloud environments because they determine how resources are allocated to VMs within the infrastructure. The proper placement of VMs can significantly impact the performance, availability, and cost-effectiveness of cloud services. When the cloud provider allocates resources to VMs, one possible objective is to maximize resource utilization by placing VMs over a minimal set of physical hosts [Masdari et al. 2016]. The importance of virtual machine placement algorithms in cloud environments lies in their ability to optimize resource utilization, improve performance, ensure high availability, and reduce costs. Given a data center composed of a set of PMs and a corresponding queue of client requests for VM instantiation, the VMP problem is determining a PM to instantiate each VM in the queue. A VMP solution may focus on traffic [Omer et al. 2021], energy [Feng et al. 2021], application [Alboaneen et al. 2021], network topology [Sridharan and Domnic 2021], resource [Gupta et al. 2018], or a combination of them.

Developing effective algorithms to optimize the placement of virtual machines is essential for maximizing the benefits of cloud computing. Effective placement can improve the efficiency, cost-effectiveness, scalability, and quality of service of a cloud system. Consequently, the importance of solving the Virtual Machine Placement problem cannot be overstated.

The aim of this paper is to assess how well the Virtual Machine Placement (VMP) algorithm performs in maximizing the utilization of resources in cloud systems. The research will examine the VMP algorithm's capacity to optimize the allocation of virtual resources in cloud-based environments, reducing the wastage of resources and enhancing the overall performance of the system. Additionally, the paper will investigate how the VMP algorithm affects resource utilization.

We propose the Rank-Heuristic Virtual Machine Placement (RHVMP) algorithm that determines the physical machine to host the received task using a ranking-heuristic strategy. RHVMP prioritizes scarce resources by globally analyzing them during the allocation process. In comparison to two techniques presented by [Son et al. 2019], Least Full First (LFF) and Most Full First (MFF), we analyzed RHVMP's host's task allocation waiting time, the number of tasks, and resources allocated in each host. Our proposal outperformed both techniques, especially in high-stress scenarios, where there is a burst of tasks to be managed.

Regarding the evaluation process, we consider two different scenarios: i) Realistic scenario to evaluate the techniques considering that each task has its arrival time, duration, and resource requirements similar to the real world (known further as Random Arrival). ii) Highly dense scenario to reproduce worst-case situations. Thus, we consider that all tasks have the exact arrival time, i.e., all tasks already arrived when the system started running (known further as Equal Arrival). Using this configuration, we evaluate how the techniques deal with a burst of tasks at certain times. Considering these scenarios, we evaluate the techniques regarding their average task waiting time, percentage of allocated

tasks by hosts, and percentage of used resources by hosts. Through time, our solution could allocate tasks much faster than the other techniques (on average 25%).

We organize the remaining of the paper as follows. Section 2 reviews existing algorithms in the literature. Section 3 presents the details of the proposed algorithm. Section 4 shows performance evaluation with a detailed description of the performance metrics, experimental setup, and simulation results. Section 5 gives the concluding remarks of the research work.

2. Related Work

There are different approaches to the VMP problem in the literature, including solutions based on machine learning algorithms and multi-objective optimization.

[Fatima et al. 2018] present an enhanced levy-based particle swarm optimization algorithm with variable-sized bin packing known as Particle Swarm Optimization Algorithm with Variable Sized Bin Packing PSOLBP for solving the VM-placement problem. They compared their solution with a simple particle swarm optimization (PSO) and a hybrid levy flight and particle swarm optimization (LFPSO) one. The proposed algorithm efficiently minimized the number of running PMs.

[Duong-Ba et al. 2018] study the problem of optimal VMP and migration to minimize resource usage and power consumption in a data center. They also formulate the optimization problem as joint multiple objective functions and solve it by leveraging a convex optimization framework. Their proposal is known as Multi-level Join VMP and Migration (MJPM), and it is based on the relaxed convex optimization framework to approximate the optimal solution. The theoretical analysis demonstrates the effectiveness of their proposed algorithms that substantially increase data center efficiency in energy consumption.

[Jiang and Chen 2018] propose the Self-Adaptive Resource Allocation algorithm (SARA). It is an online resource allocation algorithm that uses VM consolidation to achieve energy efficiency and reduce service-level agreement violations of data centers. It considers the power usage of servers, the number of migrations, and the path length of migrations in data center networks. The authors used both synthetic and real data traces to evaluate their proposal. Compared to two state-of-the-art mechanisms, the proposed algorithm successfully reduces the power consumption, the number, and the path length of migrations for a dynamic cloud service.

[Qin et al. 2020] show a VMP algorithm based on multi-objective reinforcement learning known as VMP Algorithm Based on Multi-Objective Reinforcement Learning (VMPMORL). The main objective is to find a Pareto approximate set to simultaneously minimize energy consumption and resource wastage. For simplification, authors only consider CPU and memory resources. Compared with other multi-objective RL algorithms in VMP, VMPMORL uses the concept of the Pareto set and solves the weight selection problem. The experimental results show that VMPMORL outperforms these existing methods in most cases. The authors were able to show that VMPMORL is scalable for large-size VM requests.

[Chen et al. 2018] implement a correlation-aware VMP scheme. According to the historical resource utilization data, the authors used Neural Networks and the fac-

tor model concept to forecast the resource utilization trend. Then, they designed three correlation-aware placement algorithms to enhance resource utilization while meeting the user-defined service level agreement. To evaluate their proposal, they used synthetic data, and the results of simulations showed that the efficiency of their VMP scheme outperforms the previous work by about 15%-30%.

[Son et al. 2019] propose a framework for simulating Network Function Virtualization (NFV) functionalities in both edge and cloud computing environments. Additionally, the authors show two different techniques for task allocation, namely, Least Full First (LFF) and Most Full First (MFF). LFF is a technique that searches the data center to find available physical machines with less allocated resources. The main idea of this technique is to disperse VMs across the data center since a PM hosting a VM has less priority than a machine without any hosted VM. MFF is a technique that selects the most allocated machine with enough capacity to host the VM. The selected physical machine still has enough resources for the VM, but other VMs, compared to the other candidate machines, have more resources allocated. The MFF technique makes it possible to have unused hosts in the data center since it tries to fulfill the hosts before activating the others.

In contrast to the literature, RHVMP uses the concept of resource scarcity to determine which host is more suitable for VM allocation. This ensures that we can obtain a better allocation solution for tasks by prioritizing resources that are less abundant in the datacenter. Table 1 summarizes the presented related works and compares them with our proposal.

Table 1. Related Work Comparison

Work	Objectives	Approach	Resources	Dataset
[Son et al. 2019]	VMP	Heuristic	Generic	Synthetic
[Fatima et al. 2018]	VMP	Particle swarm with Variable Sized Bin Packing	Generic	Not informed
[Duong-Ba et al. 2018]	Minimize energy consumption and resource wastage	Multi-objective optimization	Generic	Synthetic
[Jiang and Chen 2018]	Energy efficiency	Heuristic	Generic	Synthetic and real traces
[Qin et al. 2020]	Minimize energy consumption and resource wastage	Multi-objective Reinforcement Learning	CPU and RAM	Synthetic
[Chen et al. 2018]	VMP	Neural Network model and Heuristic	Generic	Synthetic
RHVMP	Resource usage maximization	Heuristic	CPU and RAM	Real traces

3. Rank-Heuristic Virtual Machine Placement

This section presents an overview of the problem and its formulation. It also shows the proposed techniques for the virtual machine placement problem.

3.1. Problem Definition and Formulation

Proper placement of virtual machines can significantly impact the overall performance of a cloud system. Inefficient VM placement can result in the underutilization of resources,

leading to wastage of energy, increased costs, and reduced system scalability. Therefore, the Virtual Machine Placement problem is a critical area of research in cloud computing. Some variables must be considered as the set of hosts, how the tasks arrive in the datacenter and the type of resources.

We consider a model with N hosts in the datacenter in this work. The tasks are distributed in a host set to maximize the number of tasks allocated. Tasks can arrive in the datacenter in two different ways: batches or streams. When they arrive in streams, the datacenter deals with them as soon as they arrive. In a case where they arrive in batches, it is necessary to wait for the formation of a batch accumulating the arrived tasks for a period of time. For a more realistic evaluation, we considered that tasks arrived in streams. In the real world, when a request arrives, it is expected that it be executed as soon as possible.

The host dimensions in the VMP problem are related to the type of resources considered in the process. Some solutions consider only one dimension (i.e., a generic resource). Meanwhile, others consider multiple dimensions (CPU, memory, disk, bandwidth). Considering just one dimension, on the one hand, creates more simple solutions, but on the other hand, the solutions are not precise since it misses many variables. This work considers a model with two dimensions (CPU and memory).

The VMP problem is an NP-hard problem [Bobroff et al. 2007]. We have adapted a version of the bin packing problem in our solution. For our case, we adapted the problem to a multiple multidimensional 0-1 knapsack problem [Song et al. 2008] to VMP.

Considering:

- $H = \{h_1, h_2, h_3, ..., h_m\}$ the set of available hosts in the datacenter;
- τ_k is the arrival time of the task k;
- $T = \{t_1, t_2, t_3, ..., t_n\}$ the sequence of tasks, such as if $p < q$, so $\tau_p \leq \tau_q$;
- $D = \{d_1, d_2, d_3, ..., d_g\}$ the set of the dimensions;
- $x_{i,j} \in \{0, 1\}, i = \{1, 2, 3, ..., n\}$ and $j = \{1, 2, 3, ..., m\}$
  - $x_{i,j} = 1 \Leftrightarrow$ host h_j allocates task t_i
  - $x_{i,k} = 0 \; \forall \; k \neq j$
- $w_{i,g}$ is the value of the dimension g for task t_i;
- $c_{m,g}$ is the capacity of host m for the dimension g;
- $F_{m,g}(\tau_n)$ is the free capacity of dimension g in host m in the instant of arrival of t_n

Maximize:

$$\sum_{i=1}^{n} \sum_{j=1}^{m} x_{i,j}$$

Subjected to:

$$\sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{g} x_{i,j} \times w_{i,k} \leq c_{j,k}$$

As presented, the main objective of our solution is to maximize the number of tasks allocated.

3.2. A Heuristic for the Virtual Machine Placement Problem

We propose the Rank-Heuristic Virtual Machine Placement (RHVMP) algorithm, which employs a ranking strategy based on global resource scarcity. During the allocation process, we consider all the information contained in the datacenter hosts. In our proposal, each host h has an associated score ($S(h)$), which is calculated based on the scarcity of a particular resource. For example, if there is a shortage of memory in the datacenter, memory will be prioritized over CPU. The score is known beforehand and recalculated whenever there is a change in the host, such as task allocation, deallocation, or resizing. Figure 1 summarizes the execution process.

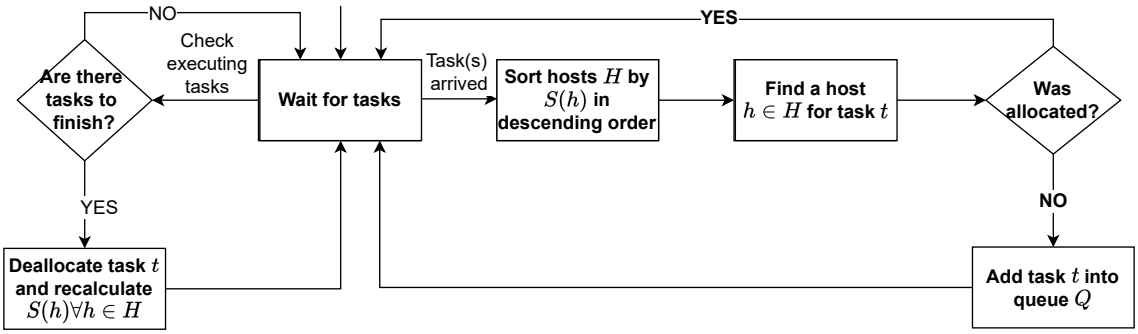

Figure 1. Execution flowchart of RHVMP

The score $S(h)$ is calculated following five steps:

1. Considering the knowledge regarding the arrived tasks and the set of hosts, we need to calculate the weights. For this, we sum each dimension of all tasks and then we do the same for all hosts;
2. For each dimension, we divide the obtained sum of the tasks by the value obtained by the hosts. This way, we found the *dimension weight*. In other words, the scarcer a resource, the greater its weight;
3. Now, it is necessary to calculate the *offer relation* of each host dimension. We divide the free capacity of each dimension of the host by the corresponding values of the task we want to allocate;
4. Lastly, for each host dimension, we need to calculate the *weighted offer relation*. This value is obtained by multiplying the respective offer relations (step 3) with the weights (step 2). Then, for each host, we sum all of its dimensions and sort them according to these results;
5. Tries to allocate task t in host $h \in H$ sequentially.

To evaluate our solution, we compared it with two different literature techniques [Son et al. 2019], classified as Immediate Dynamic Non-preemptive. We selected both Least Full First (LFF) and Most Full First (MFF) solutions since they are the two most commonly used for comparison in the literature. As mentioned, both LFF and MFF are immediate techniques. When LFF receives a task, it searches for the host with the least available resources that can accommodate the task. The algorithm performs a linear search to find the candidate host. Then, LFF submits the task to the selected host to execute it. In contrast, MFF searches for the host with the most available resources to perform the allocation process.

4. Results and Evaluation

Our evaluation uses a unique data center composed of H identical hosts serving as many VMs as needed (one-to-many relationships). Each task is only associated with only one VM. We create the VMs in execution time, and according to the requirements provided by the tasks, i.e., the VMs have the exact configuration of PE and RAM required by the allocated task.

We use the Google Cluster Dataset 2011.2 [Wilkes 2011]. It represents a collection of 30 days of data center monitoring. The dataset is composed of the description of both tasks and hosts. A task t and a host h have two arguments: the number of processing elements (PE) and RAM. The values of PE and RAM for both the Hosts and the Tasks are relative. So, we use the range $[0, 1]$ to mask the real values, where 1 represents the physical machine with the most amount of the respective resource. The best PM in the data center would have a configuration as $pe = 1$ and $ram = 1$. Meanwhile, a PM with the following configuration $pe = 0.5$ and $ram = 0.5$ has half of the resources of the best one.

In our simulations, we use 200k tasks and only 126 PMs representing 1% of the total tasks presented in the dataset. The idea of using one percent of all PMs in the dataset is to create more disputes among the PMs. If we use all PMs, the system will have more available resources than needed because the machines are much more potent than the required configuration by the tasks.

All techniques were implemented using the Java programming language. For the evaluation process, all PMs were homogeneous, meaning they had the same configuration: $pe = 0.5$ and $ram = 0.25$. We obtained these values by considering an average PM in the dataset. Additionally, all tasks were heterogeneous, comprising PE, RAM, and Duration, with Duration representing how long the task would run on a host. We defined an average task with $pe \approx 0.04$, $ram \approx 0.03$, and $duration \approx 5000$.

We evaluate three different characteristics: i) *Task Waiting Time* (Figures 2 and 3): to represent the allocation waiting time of a task in a PM. We evaluate the task waiting time in the queue before its allocation to an available host; ii) *Tasks by Time* (Figure 4): to represent the number of allocated tasks in the data center in a certain moment. We evaluate the number of tasks allocated in the system through time; iii) *Tasks distribution by Host* (Figure 5): to represent the distribution of tasks in each host in a time snapshot. We evaluate the load balancing of each task in the system. All results present an asymptotic confidence interval of 95%.

In Figure 2, the X-axis represents the evaluated techniques, and the Y-axis represents the execution time. We calculated the waiting time from when the task arrived in the data center until we allocated the task to a PM. We considered two scenarios: equal (EA) and random (RA) task arrival time. The Equal Arrival scenario assumed that all tasks arrived at the same time, while the Random Arrival scenario was more realistic, with tasks arriving at different times. In the equal task arrival time scenario, our solution had a better waiting time than the others because we processed all tasks as a single block and prioritized resources that were scarce. In the random scenario, all techniques had statistically similar results. This occurs due to the different arrival times of the tasks, which do not cause stress on the hosts. As a result, there is no competition between which tasks will be

allocated and which ones will be queued waiting for their turn.

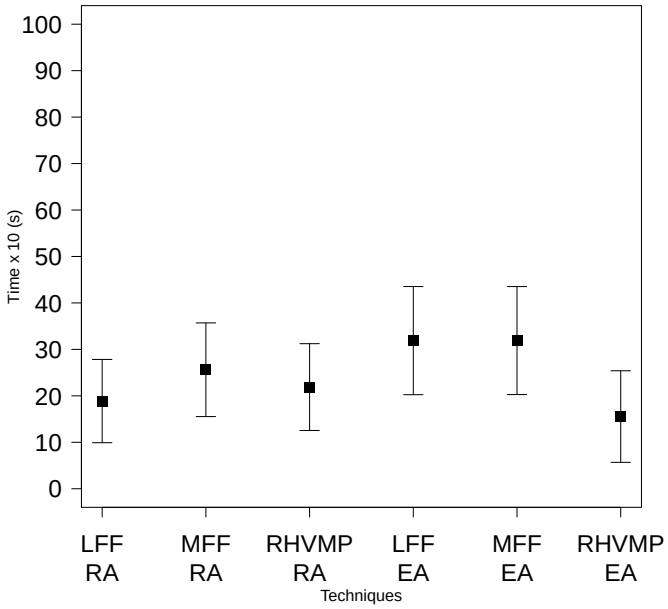

Figure 2. Tasks' average waiting time

Figure 3 displays a density plot illustrating the distribution of average waiting times for tasks. The X-axis represents the waiting time, while the Y-axis represents its frequency. In a similar task arrival scenario, our proposed technique, RHVMP, has the majority of waiting time values at zero, while the other methods have more spread-out values. This observation aligns with the results presented in Figure 2. In some cases, LFF and MFF performed poorly by allocating a machine that could have been used for different tasks, leading to a longer waiting time. Therefore, we will only analyze scenarios with the equal arrival time property, considering that the three techniques yielded statistically similar results in the random arrival scenario. In this scenario, RHVMP outperformed the other two techniques.

The following evaluation concerns the distribution of tasks and resource utilization among hosts at different time snapshots. Our analysis is performed at two distinct moments: (i) when 25% of the tasks are allocated and (ii) when 100% of the tasks are allocated. Figure 4 represents the tasks allocated through time and helps to explain how those snapshots affect the obtained results. The X-axis represents the percentage of allocated tasks, and the Y-axis represents the time to allocate the task to a host. Both LFF and MFF have the same behavior since they sequentially execute the tasks. They take a similar time to allocate all tasks to the available hosts. In contrast, our solution can allocate tasks much faster than the other two solutions. In some cases, RHVMP spent three times less time than the other two solutions to allocate the same number of tasks. As pointed out before, these results are directly impacted by how we occupy the hosts. For example, if larger tasks arrive and the most powerful host is not accessible, they could have to wait to be allocated since we allocate fewer hosts.

Figure 5 shows the distribution of allocated tasks among hosts. The X-axis repre-

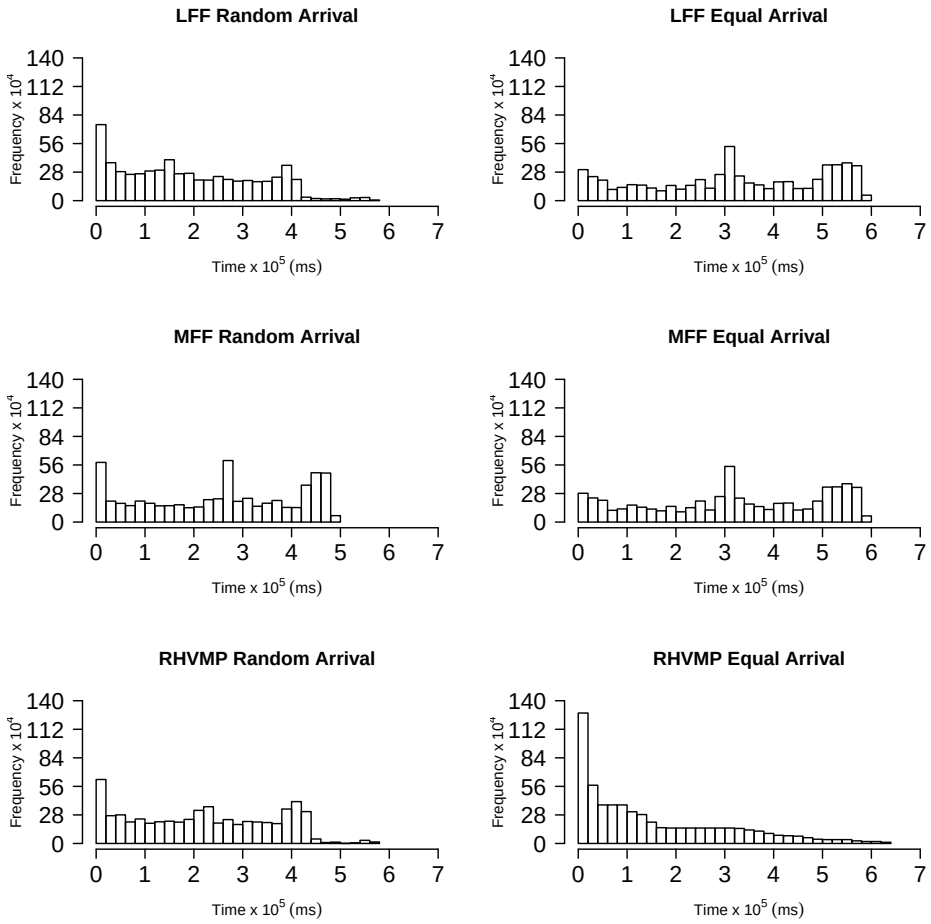

Figure 3. Density plot for average waiting time

sents the techniques, and the Y-axis represents the percentage of tasks allocated to each host. The left side presents the allocation distribution with 25% of the tasks, and the right side shows the distribution with 100% of the tasks. In both scenarios, it is evident that our technique presents a better way of distributing tasks throughout the data center compared to the other two techniques, which have a more heterogeneous form of task allocation. In other words, in MFF and LFF, some hosts are more heavily occupied than others, while RHVMP tries to minimize this variation. MFF obtained the worst results since it first tries to fill the hosts with fewer available resources. Moreover, in both LFF and MFF, there is a wider range in the values, indicating that the variation of resource usage is more significant compared to RHVMP. Our solution obtained similar results with minimum difference among hosts in both time snapshots.

5. Final Considerations

We presented the Rank-Heuristic Virtual Machine Placement (RHVMP) algorithm used for the VMP problem. The Virtual Machine Placement consequence will directly lead the efficient resource utilization. Likewise, as VMs run on physical servers, energy consumption decreases. However, the cloud will consume more energy by maintaining and managing the nodes. In other words, a good VMP algorithm decreases the financial cost for both the infrastructure provider and the user since we will optimally use the resources.

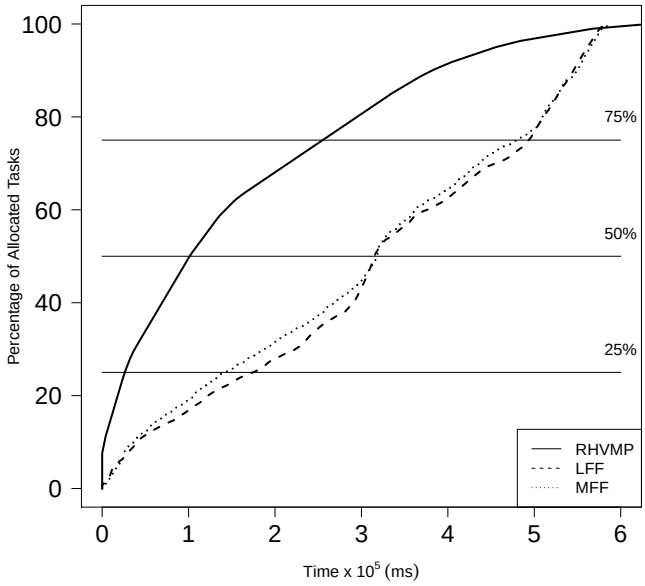

Figure 4. Allocated tasks through time

We compared our proposal with two different procedures (LFF and MFF) regarding average tasks' waiting time, percentage of tasks allocated through, and percentage of tasks assigned in hosts. In all three evaluations, our solution overperformed the other two procedures in highly dense scenarios. Considering the distribution of the tasks, RHVMP demonstrated to have a better load distribution. We also showed that RHVMP could allocate more tasks in less time when compared with the other solutions. Corroborated by the results, we can apply the RHVMP technique in scenarios where a more significant burst of tasks arrive simultaneously. As future work, we aim to include auto-scaling and migration procedures into our solution.

Acknowledgment

The work presented in this paper was co-financed by the Portuguese funding institution FCT - Foundation for Science and Technology under the Ph.D. grants SFRH/BD/144400/2019. In addition, we acknowledge the support from the Centre for Informatics and Systems of the University of Coimbra (CISUC) and the Research Foundation of the State of Alagoas (FAPEAL) under grants E:60030.0000000352/2021 and E:60030.0000000354/2021.

References

Alboaneen, D., Tianfield, H., Zhang, Y., and Pranggono, B. (2021). A metaheuristic method for joint task scheduling and virtual machine placement in cloud data centers. *Future Generation Computer Systems*, 115:201–212.

Bobroff, N., Kochut, A., and Beaty, K. (2007). Dynamic placement of virtual machines for managing sla violations. In *2007 10th IFIP/IEEE International Symposium on Integrated Network Management*, pages 119–128.

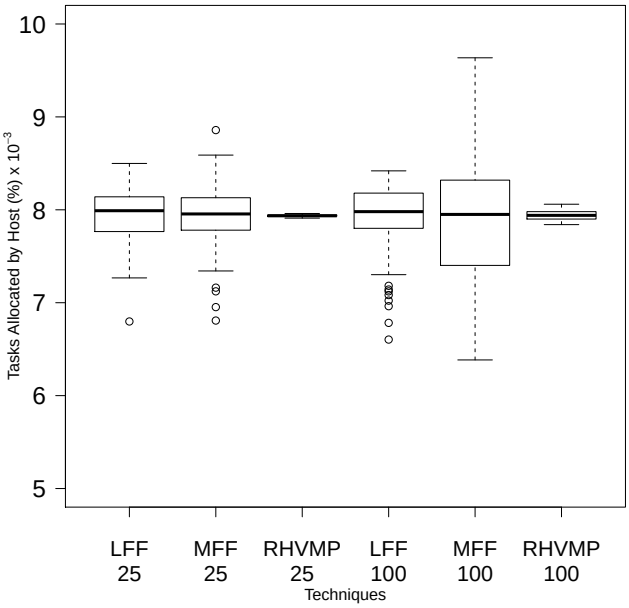

Figure 5. Percentage of allocated tasks by Hosts

Chen, T., Zhu, Y., Gao, X., Kong, L., Chen, G., and Wang, Y. (2018). Improving resource utilization via virtual machine placement in data center networks. *Mobile Networks and Applications*, 23(2):227–238.

Duong-Ba, T., Tran, T., Nguyen, T., and Bose, B. (2018). A dynamic virtual machine placement and migration scheme for data centers. *IEEE Transactions on Services Computing*, 14(2):329–341.

Fatima, A., Javaid, N., Sultana, T., Hussain, W., Bilal, M., Shabbir, S., Asim, Y., Akbar, M., and Ilahi, M. (2018). Virtual machine placement via bin packing in cloud data centers. *Electronics*, 7(12):389.

Feng, H., Deng, Y., and Li, J. (2021). A global-energy-aware virtual machine placement strategy for cloud data centers. *Journal of Systems Architecture*, 116:102048.

Gupta, M. K., Jain, A., and Amgoth, T. (2018). Power and resource-aware virtual machine placement for iaas cloud. *Sustainable Computing: Informatics and Systems*, 19:52–60.

Jiang, H.-P. and Chen, W.-M. (2018). Self-adaptive resource allocation for energy-aware virtual machine placement in dynamic computing cloud. *Journal of Network and Computer Applications*, 120:119–129.

Masdari, M., Nabavi, S. S., and Ahmadi, V. (2016). An overview of virtual machine placement schemes in cloud computing. *Journal of Network and Computer Applications*, 66:106–127.

Omer, S., Azizi, S., Shojafar, M., and Tafazolli, R. (2021). A priority, power and traffic-aware virtual machine placement of iot applications in cloud data centers. *Journal of Systems Architecture*, 115:101996.

Qin, Y., Wang, H., Yi, S., Li, X., and Zhai, L. (2020). Virtual machine placement based on multi-objective reinforcement learning. *Applied Intelligence*, 50(8):2370–2383.

Rimal, B. P., Choi, E., and Lumb, I. (2009). A taxonomy and survey of cloud computing systems. In *2009 Fifth International Joint Conference on INC, IMS and IDC*, pages 44–51. Ieee.

Silva Filho, M. C., Monteiro, C. C., Inácio, P. R., and Freire, M. M. (2018). Approaches for optimizing virtual machine placement and migration in cloud environments: A survey. *Journal of Parallel and Distributed Computing*, 111:222–250.

Son, J., He, T., and Buyya, R. (2019). CloudSimSDN-NFV: Modeling and Simulation of Network Function Virtualization and Service Function Chaining in Edge Computing environments. *Software: Practice and Experience*, 49(12):1748–1764.

Song, Y., Zhang, C., and Fang, Y. (2008). Multiple multidimensional knapsack problem and its applications in cognitive radio networks. In *MILCOM 2008 - 2008 IEEE Military Communications Conference*, pages 1–7.

Sridharan, R. and Domnic, S. (2021). Network policy aware placement of tasks for elastic applications in iaas-cloud environment. *Cluster Computing*, 24(2):1381–1396.

Taleb, T., Samdanis, K., Mada, B., Flinck, H., Dutta, S., and Sabella, D. (2017). On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration. *IEEE Communications Surveys Tutorials*, 19(3):1657–1681.

Talebian, H., Gani, A., Sookhak, M., Abdelatif, A. A., Yousafzai, A., Vasilakos, A. V., and Yu, F. R. (2020). Optimizing virtual machine placement in iaas data centers: taxonomy, review and open issues. *Cluster Computing*, 23(2):837–878.

Usmani, Z. and Singh, S. (2016). A survey of virtual machine placement techniques in a cloud data center. *Procedia Computer Science*, 78:491–498.

Wilkes, J. (2011). More Google cluster data. Google research blog. Posted at `http://googleresearch.blogspot.com/2011/11/more-google-cluster-data.html`.