Joint optimization of the cost of computation and virtual machine image storage in cloud infrastructure

José Luis Díaz, Javier García, Joaquín Entrialgo, Manuel García, Daniel F. García
Department of Computer Science
University of Oviedo
Campus de Viesques, 33204, Gijón, Spain
{jldiaz, javier, joaquin, mgarcia, dfgarcia}@uniovi.es

***Abstract*—In the field of cost optimization in cloud computing infrastructure, different strategies are used to calculate Virtual Machine (VM) allocations to support workloads while minimizing costs. Usually, VM allocations strategies are focused on determining the number and types of VMs required to support workloads at every moment, but generally they lack procedures to account for the storage costs of VMs. A significant part of these costs is generated by the storage of the Virtual Machine Images (VMIs) required to deploy VMs. In this paper, we present an improvement to a state-of-the-art VM allocation strategy, by integrating VMI costs. To achieve this, the allocation model of the strategy is extended. Then a wide set of experiments is carried out to study the improvements in the cost optimization. Several factors are analyzed and the most significant ones identified. The experimentation shows savings up to 20%, but are very dependent on the VMI sizes, the characteristics of the workload and the cloud infrastructure.**

***Index Terms*—cloud computing, virtual machine image, virtual machine allocation, cost optimization, infrastructure as a service.**

I. Introduction

Cost is an essential aspect of service deployments in cloud computing environments. In the case of Infrastructure as a Service (IaaS) platforms, Virtual Machine (VM) allocation strategies are usually employed to deal with cost issues. These strategies find allocations of VMs to support workloads, minimizing costs. A VM allocation is a mix of VMs, including their types, the number of each type, and their pricing categories. An example of a VM type in the Amazon IaaS Platform (EC2) is m5.2xlarge, which has 8 virtual cores and 16 GB of memory [1]. Pricing categories are usually on-demand or reserved.

Allocation strategies generate VM allocations periodically to constantly adjust the computational power of a VM deployment to the level of the workload. Generally, a time slot is defined to determine the periods at which the strategies are applied. Strategies can be focused in the long or the short term. Long-term strategies generate an allocation for each time slot in a long period (usually, one year). Thus, they can take advantage of reserved VMs, such as those offered by major IaaS platforms, like Amazon EC2 [2], Azure Virtual Machines [3] and Google Compute Engine [4]. Short-term strategies produce allocations for the next time slot. When long-term and short-term strategies are used in combination, the long-term strategies establish the number and types of reserved VMs required for the reservation period, and the short-term strategies determine the number and types of the on-demand VMs that provide the necessary extra computational power in each time slot to reach the required level of performance. Long-term and short-term strategies employ long-term and short-term workload predictions respectively to compute allocations.

Transactional services, such as web services, are frequently designed using a layered architecture. To determine the cost of a transactional service deployment in a cloud environment, the cost of the database layer and the cost of the VMs implementing the business logic layer must be taken into account. In turn, the latter cost is made up of three components: 1) the costs of the VMs in execution, 2) the cost of the storage of the VM root volumes, and 3) the cost of the storage of the Virtual Machine Images (VMIs) used to deploy the VM root volumes (VMIs are referred to as AMIs in the Amazon EC2 environment [5]). Finally, with regard to the costs of the VMs in execution, reserved and on-demand VMs may be taken into account. In the field of transactional services, very few research works implement long-term and short-term strategies to take advantage of reserved and on-demand VMs. One of these works is described in [6], which introduces the Malloovia VM allocation strategy. However, Malloovia lacks mechanisms to calculate the storage costs of VM root volumes and VMIs. In this paper, we present an extension of Malloovia, called Malloovia+I (Malloovia + Images), to take VMI costs into account. The storage costs of root volumes will be addressed in future research. The cost of the database layer is out of the scope of Malloovia and Malloovia+I.

Fig. 1 illustrates a VM allocation example carried out by both Malloovia and Malloovia+I. The elements with a solid red line represent the Malloovia allocation. The Malloovia+I allocation includes the elements with both the solid red line and the dashed blue line. The example represents the business logic layer of a service made up of three applications (App_x in

This work was supported by the Spanish National Plan for Research, Development and Innovation [RTI2018-094849-B-I00].

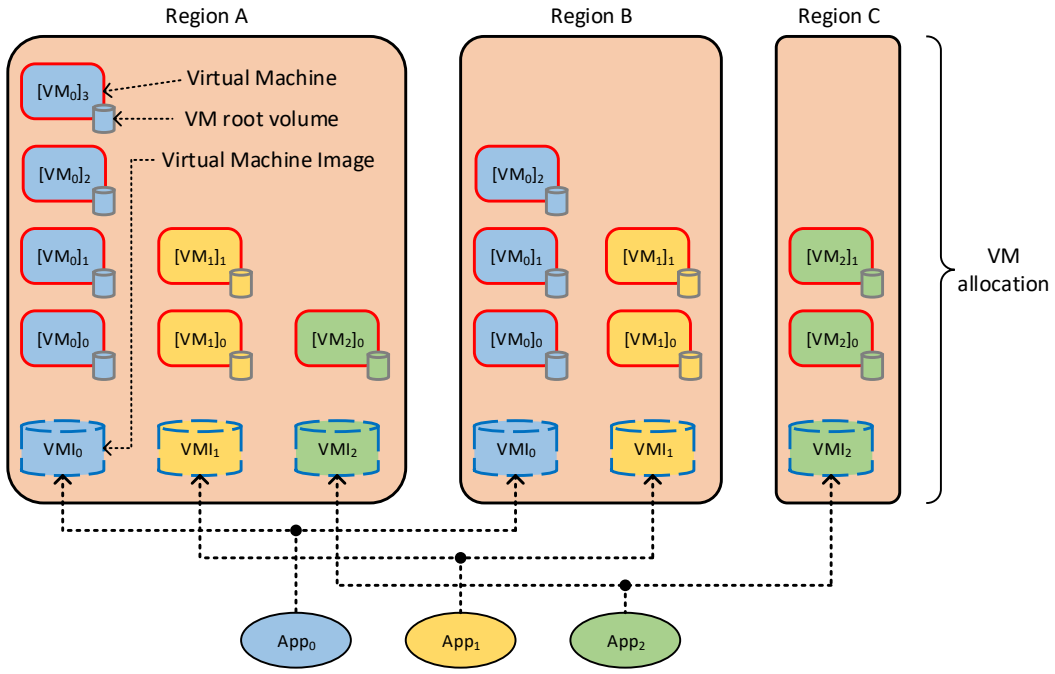

Fig. 1: Example of allocation with Malloovia (VMs) and Malloovia+I (VMs + VMIs)

the figure). The figure shows an example of an allocation in a particular time slot. The allocation of Malloovia is made up of the number, types and categories (reserved or on-demand) of VMs for each application in different regions of a provider. Types and categories of the VMs are not shown in the figure. Malloovia+I adds VMIs to the allocation produced by Malloovia. As the figure shows, a VMI must be stored in a region when the corresponding application is deployed in that region. The integration of VMIs in the allocation strategy improves its capacity for obtaining lower allocation costs.

II. Related work

This related work section is devoted to the research of the cloud cost optimization problem in the area of IaaS providers. In recent years, users have increased the use of IaaS services and as a result the budget to finance them. Furthermore, studies of cloud consultant companies have found that a substantial amount of the budget invested in IaaS is wasted [7]. So, cloud cost optimization studies are paramount from the users' point of view.

A. Optimization only with computation cost

Initially, cost optimization focused on the most demanded and expensive resource, computation. Over the last decade, a great number of research papers on cost cloud optimization were published. Here we review some of the most relevant. In [8] the authors present a comprehensive research paper on cloud cost optimization, where they develop a model solved using stochastic integer programming. In the model, the authors take into account both on-demand and reserved VMs. The main drawback of this model is that the workload is expressed as a number of required VMs instead of users' requests. A different approach is used in [9], where the authors optimize the resource provisioning for different types of workloads. In [10] the authors provide a multi-cloud optimization model solved by a greedy heuristic that includes both reserved and on-demand VMs. This model takes into account QoS, but it is limited to only one type of VM. Another cloud cost optimization strategy can be found in [11], where the users' workload is aggregated by a broker who hires the resources to the cloud providers. In [12] the cost optimization model analyzes the case of a hybrid cloud. The authors develop an online dynamic provisioning algorithm able to work under uncertainty about VMs prices and future required workload. A more recent work is Lloovia [13], which introduces a generic cost optimization model that deals with both reserved and on-demand VMs. The model also supports other real cloud aspects such as different types of VMs and the limits imposed by the cloud providers on the number of VMs that can be hired simultaneously. This work is extended in Malloovia [6], where the authors improve the optimization model to be able to include different applications, each of them with different resource requirements.

B. Optimization with computation and storage costs

All these works optimize the cloud cost due to computation only. There are a few research papers that develop more complete optimization models. In [14] the authors develop a cost optimization model for a workflow type application. This model incorporates the costs of computation, storage of the data and transmission, but only for on-demand VMs. In [15]

the authors present a case study of a migration to the cloud of an e-learning application, which fits in the type of BoT. The authors develop a model in order to optimize the cloud migration cost taking computational resources and data storage into account. This model uses reserved and on-demand VMs. Finally, in [16] the authors develop a cost optimization model for two-layered web applications. The model encompasses the computational cost and the storage cost for the database application.

There is another group of research papers focused mainly on the optimization of the cloud storage cost. In [17] the authors present a model for optimizing the cost of storage services in geographically distributed datacenters. The model minimizes cost by exploiting pricing discrepancies across providers while guaranteeing the latency of requests. A similar work is presented in [18], but in this case the QoS requirements include the level of availability of the service. This work also covers both reserved and on-demand resources. A case study of data storage optimization on the cloud is presented in [19] and later extended in [20]. These works develop a model to minimize the cost incorporating both the storage and data transfer costs. A different approach is used in [21], where the authors study VM and data placement over physical machines in order to reduce storage cost, network traffic and bandwidth usage. It is solved heuristically using the ant colony methodology. Finally, in [22] the authors present a model to minimize the storage cost based on two service layers, hot storage and cool storage, depending on the frequency of data access. The model is extended to a multi-cloud model in [23] and with two new on-line algorithms in [24].

One special case of the storage optimization cost is represented by Content Delivery Networks (CDN). Two examples of this type of research are [25] and [26]. The authors in [25] develop a model to reduce both storage cost and latency. The model determines whether some content should be replicated in a nearby server in order to reduce latency. The model is solved using genetic algorithms. The work presented in [26] studies the problem of cost optimization for dynamic OSN (Online Social Networks) on multiple geo-distributed clouds. The model incorporates the storage cost, transfer cost and maintenance cost. It obtains a solution using a heuristic algorithm.

C. Optimization taking into account VMI costs

All these works study the problem of cloud cost optimization taking into account computational resources and/or storage data resources. However, none of them considers the cost introduced by the storage of the Virtual Machine Image (VMI) in the regions where it must be deployed. The only research paper where a similar problem is studied is [27]. In this work the authors develop a model for dynamic provisioning cost optimization under VM price uncertainty, that is, they are oriented towards the spot instances price model of Amazon. A different related problem is presented in [28], where the authors study a cost-effective replication of the number of VMIs in order to obtain the required availability. However, this work does not consider any real cost, it is titled cost-effective because it minimizes the number of required VMIs.

This work extends our previous research by incorporating the cost due to the storage of the VMIs on the total infrastructure cost. The study is carried out under long-term working conditions and therefore both reserved and on-demand VMs are included. Moreover, other real cloud provider characteristics are incorporated such as different types of VMs and the limits imposed on the number of VMs that can be run simultaneously.

III. Model

A. Previous model

In [6], Malloovia is introduced as an algorithm to find the optimal allocation of virtual machines in a cloud provider, capable of serving multiple applications given a prediction of the workload for each one, with a minimum cost. However, Malloovia's model did not include the cost of VMIs. A summary of the parameters of Malloovia is presented here, to help the reader to understand the extensions proposed to take into account the storage cost.

The set of all possible VMs offered by the cloud provider is modeled with "instance classes" (IC_i), which represent a family of VMs of the same price (p_i) per time unit, price model (reserved or on-demand), and performance (perf_{ai}) for each application. Also, each IC belongs to one or more "limiting sets" (LS_i), which are used to implement some limits imposed by cloud providers on the maximum number of VMs (or VM types) which can be run simultaneously. This concept is related to the availability zone or geographic region in which the VM is deployed. An example of instance class in Amazon EC2 is an on-demand m4.xlarge instance in region us-east-1. Note that the instance class is not simply the VM type, but also the region in which it is deployed.

The set of all Instance Classes provides a model of the infrastructure, in which a set of applications $A = \{A_1, \ldots, A_{N_A}\}$ is deployed, each one subject to a different workload over time.

Malloovia divides time into *slots* of length t (for example, 1 hour). At any time slot t_k, the workload is characterized as a vector $\boldsymbol{l}_k$, whose components are the workload during that time slot for each application, i.e. $\boldsymbol{l}_k = \{l_{k,a}\}$.

To find the optimal solution, the prediction of the workload for each application in each time slot for the whole planning period is required. This way the optimizer can take advantage of the reserved instances, which are cheaper but are paid even when not used. This prediction is called LTWP (Long-Term Workload Prediction). Since having an accurate LTWP is unrealistic, Malloovia operates in two phases. Phase I uses that LTWP (or a good approximation of it) to find an optimal allocation for each time slot in the whole planning period, which uses both reserved and on-demand instances. This result is used to purchase the required number of reserved VMs and start the deployment. Then, Phase II runs online, using the reserved instances given by the optimal solution of Phase I, but

re-computing a new allocation for the on-demand VMs at each time slot from a Short-Term Workload Prediction (STWP), which uses only the next time slot. If the STWP is very similar to the LTWP, then Phase II will produce the same allocation as Phase I, but if STWP differs substantially from LTWP, Phase II can accommodate it by using the fixed number of reserved instances given by Phase I and adjusting the allocation of the on-demand ones.

To reduce the computational cost of solving Phase I, the LTWP can be compressed by using a histogram $H(\mathbb{L})$, which stores all the different load levels present in $\boldsymbol{l}_k$ as well as the number of times each one is repeated. So, for example, with three applications, one element of that histogram could be $H(\{100, 120, 500\}) = 150$, which means that there are 150 time slots in the whole planning period with the same workload, equal to 100 units for application 1, 120 for application 2, and 500 for application 3. Since the workload is the same in those 150 time slots, the optimal allocation will also be the same for all of them. This can reduce the size of the problem substantially when workloads are repetitive. If they are not, Malloovia proposes *quantizing* the workloads by rounding them up to given values, so that the number of possible workload values is reduced.

B. Extensions to previous model

The previous model did not take into account the cost of the storage. To include it, two types of storage must be considered. First, the storage attached to each VM, which is usually some kind of block storage whose price is dependent on its size and on the length of time the volume exists. This cost will not be modeled at this stage. Instead, since the volume exists only while the VM is running and it is deleted afterward, we assume that this cost can be included as part of the price p_i per time unit of each instance class IC_i.

Second, each application requires a Virtual Machine Image (VMI) from which a new VM can be created. The image contains all the required code and data for the app to boot and run.

Most cloud providers allow the use of custom images, but these must be stored in the same region as the VMs created from that image. This means that, if the allocator decides to run the application A_a in any VM of the region r, at any time slot, then the image I_a must be stored in a storage solution for that region (e.g.: AWS S3).

This causes two new terms in the cost of the allocation:

- **Deployment cost**. This is the cost of the initial transfer from the customer to the region storage (via Internet), plus the cost of maintaining that storage for the time of the deployment. This cost can be computed as part of Phase I, but the images needed by the on-demand machines must also be taken into account, because we assume that the required images are uploaded once, at the beginning of the planning period.
  Most cloud providers do not charge for the bandwidth required for this first upload. However, they may charge for the number of POST operations required to complete the upload, as it is the case for AWS S3. Since there is a limit on the size that can be transferred in a single POST, the number of POST will be proportional to the size of the VMI. This cost may be significant.
  After Phase I is run, the output contains the number and types of reserved instances to hire in each region, and the VMIs that should be uploaded to each region. Our model assumes that no new VMIs will be upload during the planning period. This does not preclude the possibility of uploading patches and bug fixes to some images, but it is assumed that the VMI sizes remain practically constant, so the storage cost is not affected. The upload of patched VMIs will have a minimal cost, because of the POST. Moreover, it is a cost that cannot be optimized anyway, so it doesn't need to be taken into account in Phase I.
- **Working cost**. In Phase II, each time a new on-demand VM is created the image containing the boot disk must be transferred from the region storage to the block storage of that VM. Most providers do not charge for data transfers inside a region, so instantiating a new VM from the image is free. In this case, the cost-optimal strategy is to destroy the block storage of the VM once the on-demand VM is no longer required, and create a new block storage each time a VM is instantiated again. Using this strategy, the cost of the block storage can be included as part of the cost per time unit of having a running VM instance. Therefore, no changes are required in Malloovia's model to take this cost into account. However, new restrictions are needed in Phase II to ensure that no on-demand instances are created in a region for which the needed image was not deployed in Phase I.

C. New variables

In addition to the idea of "Limiting Set" already present in Malloovia, the model requires the concept of a "Deployment Region". Each instance class (IC) belongs to one (and only one) Deployment Region, despite being part of one or more Limiting Sets. Each VMI can be deployed in one (or more) Deployment Region, which will make that VMI available for all VMs in that region.

Deployment Regions and Limiting Sets are independent concepts in theory, although they can be related in practice. Limiting sets are required to model the limits imposed by some cloud providers on the number of machines of each type which can be run at once (and translates to concepts such as "availability zones" and "regions" in AWS parlance). Deployment Regions, on the other hand, are required to model the link between the VMI storage and the instance class that can use it (and translates to what AWS calls simply "regions").

Each application a has an associated image, I_a, which has a certain size S_{I_a} in GB.

Each deployment region r has new pricing parameters:

- pt_r which is the price of transferring 1 GB from the Internet to that region. This cost includes both the price per transferred GB, which is usually zero, and the price

for the number of POST required, which is proportional to the VMI size.
- ps_r which is the price of the storage, per GB and time slot.

The unknowns solved by Malloovia are:

- $X_{ai\boldsymbol{L}}$ which is the optimal number of on-demand VMs of Instance Class IC_i to be deployed in a time slot to run application A_a, when the workload vector in that time slot is $\boldsymbol{L}$
- Y_{ai} which is the optimal number of reserved VMs of Instance Class IC_i to be purchased at the beginning of the planning period, to run application A_a.

The extension requires new "artificial" unknowns:

- Z_{ar} which is a boolean indicating whether application a is ever run in deployment region r (and thus the corresponding image I_a has to be stored in that region).

D. Cost

The cost computed by the previous version of Malloovia has to be increased by the following amount, to take into account the cost of deploying the VMIs:

$$\text{VMI deployment cost} = \sum_a \sum_r Z_{ar} S_{I_a} (pt_r + T \cdot ps_r) \quad (1)$$

where T is the number of time slots in the deployment period.

Hence the new objective function to minimize is:

$$\begin{aligned} C = & \sum_a \sum_i Y_{ai} p_i^{\text{res}} T/t \\ & + \sum_a \sum_i \sum_{\boldsymbol{L} \in \mathbb{L}} X_{ai\boldsymbol{L}} p_i^{\text{dem}} H(\boldsymbol{L}) \\ & + \sum_a \sum_r Z_{ar} S_{I_a} (pt_r + T \cdot ps_r) \end{aligned} \quad (2)$$

E. Restrictions

To enforce that image I_a is stored in the same deployment region where app a is run, new auxiliary variables are needed, namely V_{ar} which are the number of levels in the LTWP in which app a is used in region r. These can be computed as follows:

$$V_{ar} = \sum_{\boldsymbol{L} \in \mathbb{L}} \sum_{i \in R_r^{\text{dem}}} X_{ai\boldsymbol{L}} + \sum_{i \in R_r^{\text{res}}} Y_{ai} \quad (3)$$

where R_r^{dem} (respectively R_r^{res}) is the set of all indexes i corresponding to all on-demand (respectively reserved) Instance Classes in the same deployment region r, and $\mathbb{L}$ the set of all different workloads.

Using these new variables, and the auxiliary variables Z_{ar}, the following restrictions should be enforced (in which M is a number large enough, larger than the maximum possible V_{ar}, which can be obtained from the limiting sets):

$$\begin{aligned} M Z_{ar} - V_{ar} &\geq 0 \\ V_{ar} - M(Z_{ar} - 1) &\geq 1 \end{aligned} \quad (4)$$

The first one forces Z_{ar} to be 1 whenever V_{ar} is not zero. The second one forces Z_{ar} to be 0 whenever V_{ar} is zero (no instances run application a in deployment region r at any time slot). These values of Z_{ar} are used in the cost function as shown above in eq. (1) and (2).

IV. Experiments

This section shows the results from a set of experiments aimed at comparing the new approach, Malloovia+I, with the previous state-of-the-art approach, Malloovia, and analyzing the savings that can be achieved with the new technique. In addition, the experiments also explore which factors influence these savings.

Regarding the latter, allocation problems are very complex and have many parameters, including parameters about the system (cloud providers used, pricing models, types of VMs, number of regions...) and parameters about the workload (number of applications, workload for each application and size of the VMIs).

In order to carry out a representative set of experiments, the case study presented in [6] was used as a base and many of its parameters were changed in different experiments. This case study included three applications with different workload patterns: application 0 had periodic spikes each hour and no load the rest of the time, application 1 had a workload more evenly distributed throughout the hour and application 2 had a combination of both. In addition, the average workload for each application was different.

Experiments with only one of these applications were carried out to determine if the shape of the workload influences the results. In addition, experiments using the three applications at the same time and six and nine applications (replicating two and three times the three base applications) have been carried out to see the influence of the number of applications.

For the parameters of the system, Amazon EC2 was selected as an IaaS provider because it is the leader in the sector and offers a variety of pricing models, types of VMs and regions. Specifically, the prices on April 1 2020 of 19 regions were used. Table I shows the base performance for each application on the instance types used. The performance was defined in

TABLE I: Performance of different VM types

VM type	rph app 0	rph app 1	rph app 2
c5.large	100	50	200
c5.xlarge	200	100	400
c5.2xlarge	390	195	780
c5.4xlarge	730	365	1460

TABLE II: VMI size and computational demand multipliers used to modify the base values in the experiments

Factor	Multipliers
Demand	1, 1/5, 1/10, 1/20
VMI Size	1, 10, 20, 30, 40, 50

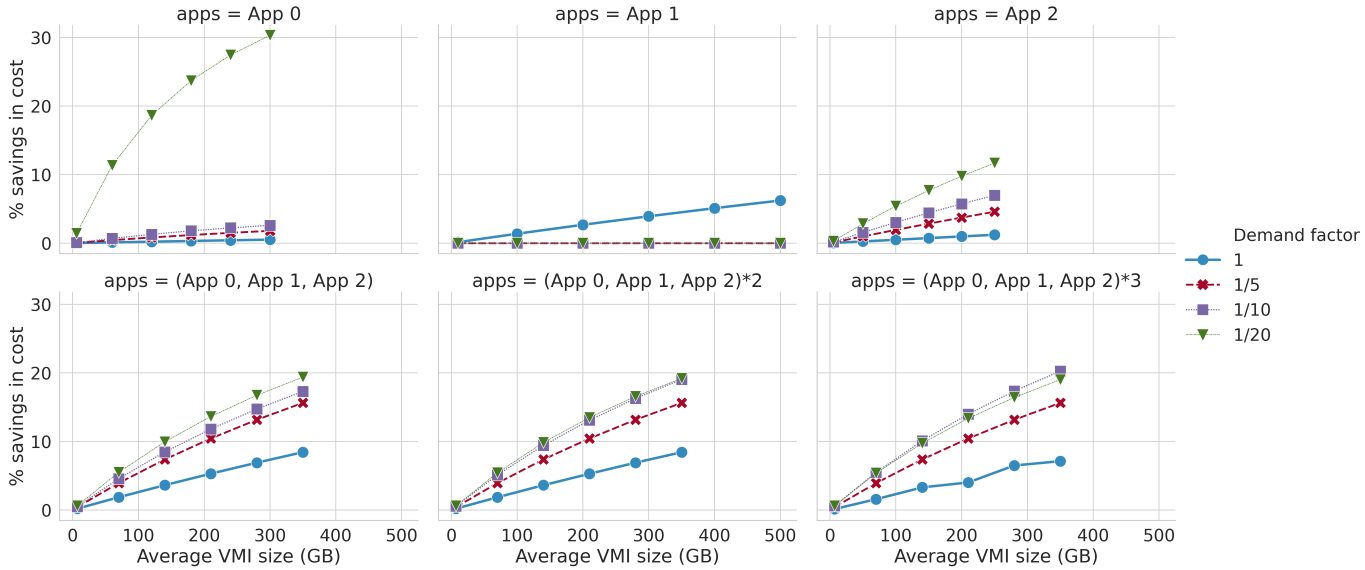

Fig. 2: Percentage of the cost saved by Malloovia+I related to Malloovia

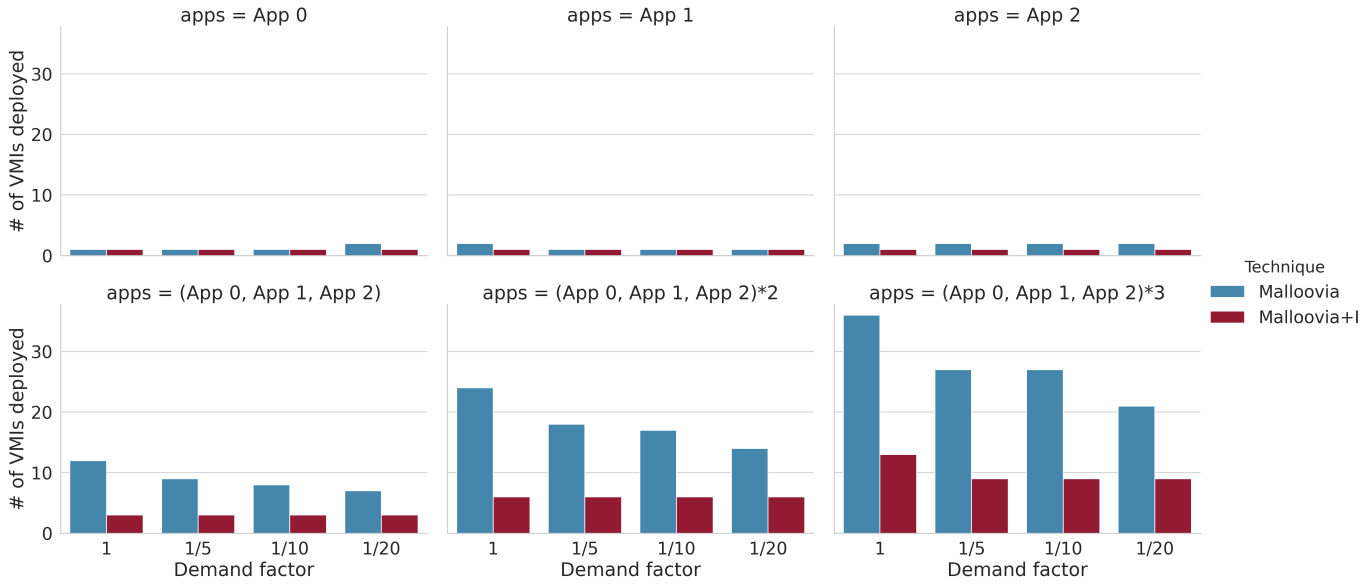

Fig. 3: VMIs deployed

requests per hour (rph) and extrapolated to each type of VM using its ECU (EC2 Compute Unit). In order to take into account the influence of the computational demand of the applications, a demand factor that reduces the base demand was introduced (see Table II).

Another very important parameter is the size of the VMIs for each application. To explore its influence, a base value of 6, 10 and 5 GBs for application 0, 1, and 2 respectively were selected, and then modified multiplying the values by a factor in different experiments (see Table II).

This experimental design gave 144 variations of the parameters. Each variation was solved both with Malloovia and Malloovia+I, resulting in 288 experiments. Malloovia optimizes the system without taking into account the cost of the VMIs. Thus, the VMI deployment cost had to be added to the cost obtained by Malloovia, in order to obtain the real total cost. Malloovia+I, on the other hand, takes this cost into account for the optimization problem.

Fig. 2 shows the percentage of the cost saved by using Malloovia+I instead of Malloovia. For instance, when only App2 is run, with a demand factor of 1/20 and a VMI size factor of 50, Malloovia+I obtains an allocation that costs $642.42, while the one obtained by Malloovia costs $567.25 and, thus, the percentage of the cost saved by using Malloovia+I is 11.67%, as shown in the left-most point of the top green line of the plot corresponding to App 2 in Fig. 2.

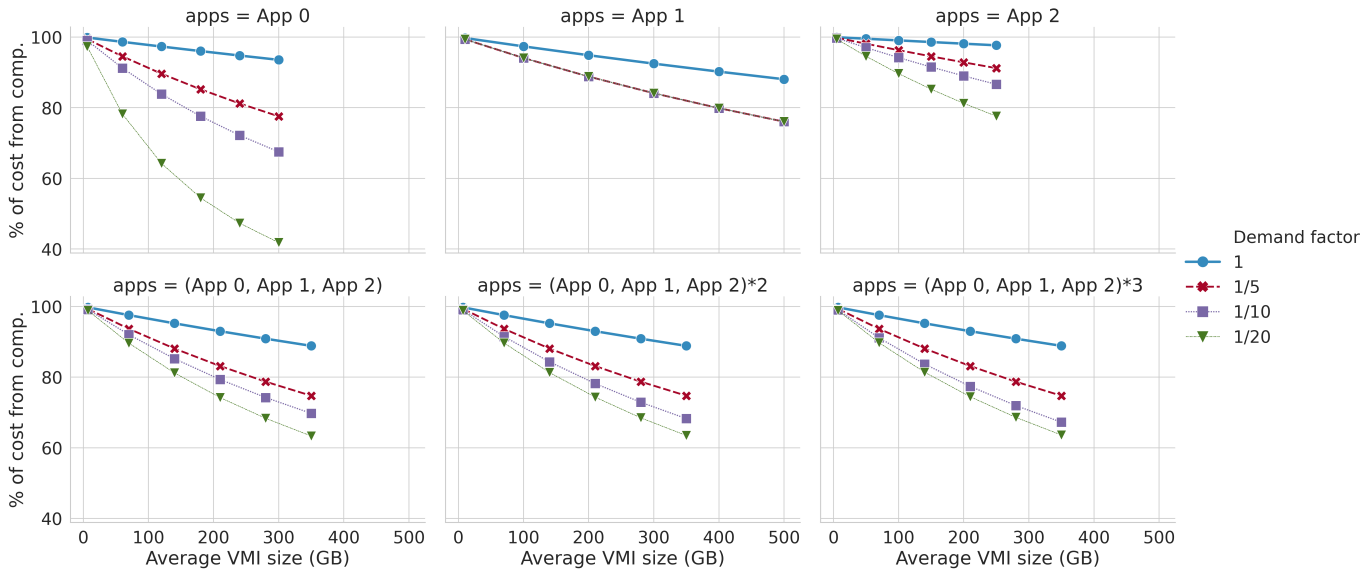

Fig. 4: Percentage of the cost that is computation cost

A first, somewhat obvious conclusion is that the larger the average size of the VMIs, the greater the savings. A second conclusion is that the computational demand is also a great influence: in many occasions its influence is greater than the influence of the size of the VMIs. For instance, executing only the first application with the smallest VMI size (top-left graph in Fig. 2), the savings are greater for computational demand factor 1/20 (top green line) than for any tested VMI size for demand factor 1 (bottom blue line).

Comparing the curves for different applications (top three graphs of Fig. 2), it can be seen that the percentage saved varies greatly. For instance, when only application 1 is executed (center graph in the first row of Fig. 2), the savings are always below 7%, but with application 0, it can reach 30% for demand factor 1/20.

It could be expected that with more applications the savings obtained with Malloovia+I would be greater, as there are more VMIs (one per application) to deploy in each region used. However, the experiments show that this is not the case, as can be seen by studying the bottom row of Fig. 2, which compares using 3, 6 or 9 applications with the same workload replicated in each group of 3 applications, so that the average workload is maintained. The graph shows that the savings are similar independently of the number of applications. The cause for this is that the percentage of the cost generated by the storing of the images does not change with the number of applications, as demonstrated below.

Two elements explain the savings obtained with Malloovia+I: the difference in the number of VMIs deployed (Fig. 3) and the relation of the computation cost to the storage cost (Fig. 4).

Fig. 3 can be used to explain how the number of VMIs deployed influences the savings. The size of the VMIs is not represented because we have found that it does not make a difference in the number of VMIs used, except for 9 applications and demand factor 1; however, with the same number of VMIs and bigger size, the absolute savings are greater.

The first result that can be explained with Fig. 3 is how the computational demand influences the savings obtained with Malloovia+I: in most cases, the new approach proposed in this paper uses the same or fewer VMIs than Malloovia, but when the computational demand factor is smaller (for instance, with computational demand factor 1/20), the difference in the number of VMIs is smaller, because fewer VMs are required and, therefore, Malloovia does not use so many more regions than Malloovia+I. This is not the case when only application 0 is executed and the demand factor is 1/20: this explains the large percentage saved in this situation compared to other demand factors, as already shown in the green line in the top-left graph in Fig. 2.

The computational demand also influences the other element that can be used to explain the savings of Malloovia+I: the ratio of the computation cost to the storage cost. When the computational demand is lower, the importance of the storage cost, where Malloovia+I can obtain savings compared to Malloovia, is more significant. This can be seen in Fig. 4, which plots the percentage of the total cost that comes from the computation in Malloovia. Where the computation cost is almost 100% of the total cost, the savings that can be obtained by Malloovia+I are very limited. Fig. 4 shows that the percentage of the cost attributed to computation is reduced when the size of the VMIs increases (in each graph, the computation cost percentage decreases when moving to the right) and that it is also reduced when the computational demand decreases (the lines corresponding to smaller computational demands are below the lines corresponding to greater computational demands).

V. Conclusions

This paper introduces a cost optimization technique for cloud computing allocation of VMs in IaaS providers that extends the previous state of the art by including the cost associated with storing the VMIs. The new technique is called Malloovia+I and extends a previous one, called Malloovia, by including new variables in the model used.

A significant number of experiments have been carried out to analyze how the new technique compares to the old one and which factors are more significant. The experiments found savings up to 20% (except in one particular case where they reach 30%), but they are very dependent on the characteristics of the workload and the system. The size of the VMIs, the shape of the workload and the computational demand have been found to be the most significant factors in the savings that can be obtained by Malloovia+I.

Future work will study the cost of storing the root volume of instances and other data used by the application, as well as the cost of data transmissions between VMs.

References

[1] Amazon, "Amazon EC2 instance types," https://aws.amazon.com/ec2/instance-types/, 2020, accessed Apr. 03, 2020.

[2] ——, "Amazon EC2," https://aws.amazon.com/ec2/, 2020, accessed Apr. 03, 2020.

[3] Microsoft, "Microsoft azure virtual machines," https://azure.microsoft.com/en-us/services/virtual-machines/, 2020, accessed Apr. 03, 2020.

[4] Google, "Google compute engine," https://cloud.google.com/compute/, 2020, accessed Apr. 03, 2020.

[5] Amazon, "Amazon machine images (ami)," https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIs.html, 2020, accessed Apr. 03, 2020.

[6] J. Entrialgo, J. L. Díaz, J. García, M. García, and D. F. García, "Cost Minimization of Virtual Machine Allocation in Public Clouds Considering Multiple Applications," in *Economics of Grids, Clouds, Systems, and Services*, C. Pham, J. Altmann, and J. A. Bañares, Eds. Cham: Springer International Publishing, 2017, vol. 10537, pp. 147–161.

[7] Flexera, "Flexera 2020 state of the cloud report," Flexera, Tech. Rep., 04 2020.

[8] S. Chaisiri, B. S. Lee, and D. Niyato, "Optimization of resource provisioning cost in cloud computing," *IEEE Transactions on Services Computing*, vol. 5, no. 2, pp. 164–177, Apr. 2012.

[9] J. Zhan, L. Wang, X. Li, W. Shi, C. Weng, W. Zhang, and X. Zang, "Cost-Aware Cooperative Resource Provisioning for Heterogeneous Workloads in Data Centers," *IEEE Transactions on Computers*, vol. 62, no. 11, pp. 2155–2168, Nov. 2013. [Online]. Available: http://ieeexplore.ieee.org/document/6205737/

[10] U. Bellur, A. Malani, and N. C. Narendra, "Cost optimization in multi-site multi-cloud environments with multiple pricing schemes," in *2014 IEEE 7th International Conference on Cloud Computing*, Institute of Electrical and Electronics Engineers, Inc. IEEE, Jun. 2014, pp. 689–696.

[11] W. Wang, D. Niu, B. Liang, and B. Li, "Dynamic cloud instance acquisition via IaaS cloud brokerage," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 6, pp. 1580–1593, Jun. 2015.

[12] S. Li, Y. Zhou, L. Jiao, X. Yan, X. Wang, and M. R.-T. Lyu, "Towards Operational Cost Minimization in Hybrid Clouds for Dynamic Resource Provisioning with Delay-Aware Optimization," *IEEE Transactions on Services Computing*, vol. 8, no. 3, pp. 398–409, May 2015.

[13] J. L. Díaz, J. Entrialgo, M. García, J. García, and D. F. García, "Optimal allocation of virtual machines in multi-cloud environments with reserved and on-demand pricing," *Future Generation Computer Systems*, vol. 71, pp. 129 – 144, 2017.

[14] P. Yi, H. Ding, and B. Ramamurthy, "Budget-Optimized Network-Aware Joint Resource Allocation in Grids/Clouds Over Optical Networks," *Journal of Lightwave Technology*, vol. 34, no. 16, pp. 3890–3900, Aug. 2016.

[15] P. lvarez, S. Hernndez, J. Fabra, and J. Ezpeleta, "Cost-driven provisioning and execution of a computing-intensive service on the Amazon EC2," *The Computer Journal*, vol. 61, no. 9, pp. 1407–1421, Sep. 2018.

[16] S. Mireslami, L. Rakai, M. Wang, and B. H. Far, "Dynamic Cloud Resource Allocation Considering Demand Uncertainty," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2019.

[17] Z. Wu, M. Butkiewicz, D. Perkins, E. Katz-Bassett, and H. V. Madhyastha, "*SPANStore*: cost-effective geo-replicated storage spanning multiple cloud services," in *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles - SOSP '13*. Farminton, Pennsylvania: ACM Press, 2013, pp. 292–308.

[18] G. Liu and H. Shen, "Minimum-Cost Cloud Storage Service Across Multiple Cloud Providers," in *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*. Nara, Japan: IEEE, Jun. 2016, pp. 129–138.

[19] C. Negru, F. Pop, O. C. Marcu, M. Mocanu, and V. Cristea, "Budget constrained selection of cloud storage services for advanced processing in datacenters," in *2015 14th RoEduNet International Conference - Networking in Education and Research (RoEduNet NER)*. Craiova, Romania: IEEE, Sep. 2015, pp. 158–162.

[20] C. Negru, F. Pop, M. Mocanu, V. Cristea, A. Hangan, and L. Vacariu, "Cost-aware cloud storage service allocation for distributed data gathering," in *2016 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*. Cluj-Napoca, Romania: IEEE, May 2016, pp. 1–5.

[21] T. Shabeera, S. Madhu Kumar, S. M. Salam, and K. Murali Krishnan, "Optimizing VM allocation and data placement for data-intensive applications in cloud using ACO metaheuristic algorithm," *Engineering Science and Technology, an International Journal*, vol. 20, no. 2, pp. 616–628, Apr. 2017.

[22] Y. Mansouri and A. Erradi, "Cost Optimization Algorithms for Hot and Cool Tiers Cloud Storage Services," in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. San Francisco, CA, USA: IEEE, Jul. 2018, pp. 622–629.

[23] Y. Mansouri, A. N. Toosi, and R. Buyya, "Cost Optimization for Dynamic Replication and Migration of Data in Cloud Data Centers," *IEEE Transactions on Cloud Computing*, vol. 7, no. 3, pp. 705–718, Jul. 2019.

[24] Y. Mansouri and R. Buyya, "Dynamic replication and migration of data objects with hot-spot and cold-spot statuses across storage data centers," *Journal of Parallel and Distributed Computing*, vol. 126, pp. 121–133, Apr. 2019.

[25] S. Sajithabanu and S. R. Balasundaram, "Cloud based Content Delivery Network using Genetic Optimization Algorithm for storage cost," in *2016 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*. Bangalore, India: IEEE, Nov. 2016, pp. 1–6.

[26] L. Jiao, J. Li, T. Xu, W. Du, and X. Fu, "Optimizing Cost for Online Social Networks on Geo-Distributed Clouds," *IEEE/ACM Transactions on Networking*, vol. 24, no. 1, pp. 99–112, Feb. 2016.

[27] J. L. L. Simarro, R. M. Vozmediano, F. Desprez, and J. R. Cornabas, "Image Transfer and Storage Cost Aware Brokering Strategies for Multiple Clouds," in *2014 IEEE 7th International Conference on Cloud Computing*. Anchorage, AK, USA: IEEE, Jun. 2014, pp. 737–744.

[28] D. Shen, F. Dong, J. Zhang, and J. Luo, "Cost-Effective Virtual Machine Image Replication Management for Cloud Data Centers," in *2014 IEEE Intl Conf on High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC,CSS,ICESS)*. Paris, France: IEEE, Aug. 2014, pp. 229–236.