I.J. Intelligent Systems and Applications, **2021, 6, 1-12**
Published Online December 2021 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijisa.2021.06.01

Heuristic-based Approach for Dynamic Consolidation of Software Licenses in Cloud Data Centers

Leila Helali
MARS Research Laboratory LR17ES05, University of Sousse, Tunisia
E-mail: leilahelali.ing@gmail.com

Mohamed Nazih Omri
MARS Research Laboratory LR17ES05, University of Sousse, Tunisia
E-mail: mohamednazih.omri@eniso.u-sousse.tn

Received: 18 September 2021; Revised: 21 October 2021; Accepted: 01 November 2021; Published: 08 December 2021

Abstract: Since its emergence, cloud computing has continued to evolve thanks to its ability to present computing as consumable services paid by use, and the possibilities of resource scaling that it offers according to client's needs. Models and appropriate schemes for resource scaling through consolidation service have been considerably investigated, mainly, at the infrastructure level to optimize costs and energy consumption. Consolidation efforts at the SaaS level remain very restrained mostly when proprietary software is in hand. In order to fill this gap and provide software licenses elastically regarding the economic and energy-aware considerations in the context of distributed cloud computing systems, this work deals with dynamic software consolidation in commercial cloud data centers $\boldsymbol{DS^3C}$. Our solution is based on heuristic algorithms and allows reallocating software licenses at runtime by determining the optimal number of resources required for their execution and freed unused machines. Simulation results showed the efficiency of our solution in terms of energy by 68.85% savings and costs by 80.01% savings. It allowed to free up to 75% physical machines and 76.5% virtual machines and proved its scalability in terms of average execution time while varying the number of software and the number of licenses alternately.

Index Terms: Software license Optimization, Resource Management, Energy Efficient, Cost, Virtualization.

1. Introduction

The computational needs today driven by the externalization and the new emerged paradigms such as the softwarization and the virtualization of networks, which resulted in a massive number of generated services where most of them use proprietary software and licenses [1], leads to rethinking the resource management in the virtualized cloud environment. Yet, with the software license cost explosion where 50% of costs (software licensing, maintenance, and deployment) are incurred in a data center [2], when these resources and the hardware resources are used inefficiently, operational expenditures and energy consumption will be pushed to the maximum. This can cause a maximum of 15% exploitation of resources by around 30% of cloud servers which participates to the large amount of energy consumed by the data centers actually that worth more than 2% globally [3]. The efficient use of software license resources is an essential basis for successful business. Managing these resources along with the virtualized hardware resources and the generated energy and costs in the cloud represents a recent challenge. This is even more challenging to reconsider the elasticity aspects in the presence of licensed applications where the classical consolidation solutions are not sufficient. Dynamic consolidation allows determining the number of resources needed at runtime to run the workload and satisfy the request of the users. Performing this technique when licensed software are in hand is not a trivial task. The consolidation problem in cloud data centers (DCs) has attracted much attention in the academic and industrial environment. The main objective is to maximize resource exploitation by keeping the minimum number of active physical machines (PMs), minimizing energy consumption and costs, among others. Unfortunately, the literature work in this area focusses on the IaaS level: the technical characteristics of equipment and standard resources (network, CPU, memory, etc.) by exploring virtual machine (VM) consolidation. Some service placement solutions are presented in the literature in fog computing like [4]. However, software licenses that are typically an even more expensive resource than compute-power [3] have been omitted. Thus, in this work, we focus on software license consolidation in cloud data centers by multiplexing several licenses in the same VM. Through this technique, the same VM is shared between

different licenses instead of dedicating a VM to each license. This makes it possible to fill the gap remaining in the VM that can run multiple licenses and minimizes not only the number of physical machines but also the number of virtual machines while minimizing the total cost including the energy costs. This became possible thanks to the containerization technique which ensures the isolation, fast, and dense deployment of licensed applications when virtual machines are considered. In our work, we consider containerized software license applications that run in VMs due to the advantages of this virtualization architecture in terms of resource management, security, and isolation, among others. Furthermore, here, we consider the VM-based licensing model with regard to economic considerations without treating the compliance aspect. Note that license costs and software costs are used interchangeably in the Software Asset Management (SAM) context. Thereby, in our work, we follow the same trends [5]. The software license charges may be included in the VM price (License Charges Included) when VM-based virtualization technology is used, or considered separately. In the latter case, the cloud user, when deploying the software in VM, considers his own software license (Bring Your Own License) [6]. Here, we consider the second scenario, where the cost of software is separated from the cost of the virtualization element that hosts it (i.e. VM in our present work).

To address the above-mentioned challenges, the main contributions of our work are as follows:

- To propose a new strategy for SL candidates' selection for migration that allows additional probable savings via efficient resource allocation. This policy is named Maximum License Cost (MLC) and allows us to save the cost of the most expensive software license in the migration list.
- To propose adapted version of the well-known Best Fit Decreasing (BFD) heuristic algorithm that is widely used to resolve similar problems at other service levels, to support the Software License Consolidation (SLC) case.
- To implement and evaluate our proposed solution and discuss its results and show its effectiveness based on some key metrics. This step allows us to visualize the effectiveness of our proposed strategy in terms of energy and cost savings by decreasing the user's monetary costs resulting from the decrease in the resources provisioned (number of virtual machines), and also the decrease in the license costs and infrastructure cost (mainly energy costs).

The present paper is organized as follows. In section 2, we present the related state of the art. In section 3, the problem is defined and the assumptions of our model are presented. While section 4 describes the system architecture of our solution, details its basic components, and presents our proposed methodology to consolidate the software licenses in the minimum number of machines while minimizing total energy and cost. Section 5 is devoted to presenting our solution's evaluation, and section 6 discusses the obtained results and findings. In section 7, we conclude the work and give an outlook of some future directions.

2. Related Work

Consolidation work in the cloud mainly concerned the IaaS level with virtual machine consolidation. With the popularity of container technologies and the container-based cloud, new consolidation opportunities at CaaS level emerged, from which we can cite for example [7, 8, 9]. When at the SaaS level, we can cite [10] which deal with software migration and dynamic consolidation in addition to VM placement at system start-up. The consolidation problem was considered as a constraint satisfaction problem (CSP) formalized in mixed integer nonlinear programming (MINLP). The goal was to optimize software in terms of cost and energy consumption using a constraint solver that uses an exhaustive search based on the in-depth algorithm. The major disadvantage of this work is that the choice of the destination VM in which the software is to be moved is not optimal and the consolidation engine corrects the placement. This requires redoing the work instead of placing the software in the VMs optimally from the beginning. Besides, this work does not consider the costs of software licenses.

Another work named CloudBridge [2] aims to find an integrated hardware-software consolidation strategy that minimizes the total cost of cloud data centers. Considering that a typical application consists of multiple layers of software (OS, middleware, application) and that software consolidation can be in one or more of these layers. Thus, consolidating 2 workloads into a specific software level requires the entire software stack to be migrated over to the other instance. A two-level algorithm is presented to a) find the best placement of workloads on a server and b) find the best level of software consolidation for the workloads placed on each server based on an algorithm called BP2 [2]. Hardware consolidation is performed using the PCP [2] algorithm. It uses workload analysis to group workloads based on their peak periods. The major issue of this approach is that it does not allow guaranteeing the isolation for software consolidation. One must have built-in support for migrating the layers of the relevant software stack. Otherwise, the migration cost becomes very important.

The work of Zoltan [11] considers the software license aspect in VM consolidation. This work was interested in a joint optimization problem of mapping VMs to PMs (VM placement) and application components to VMs (VM selection). The author has studied the effects of VM selection policies (for the component in question) on the placement of VMs in PMs and vice versa. He basically sought for the selection of suitable VMs to place software components and

suitable PMs to accommodate these VMs, without considering neither deployment nor consolidation aspects of these components. It is a classical VM consolidation work that considers the license fees among other parameters.

3. Problem Definition and Assumptions

Let us consider an incoming workload that consists of n amount of software with m licenses. The software license consolidation problem revolves around containerized licensed software that are deployed in virtual machines which are themselves hosted on physical machines (PMs). This is an NP-complete problem [12] like VM consolidation. It is assumed as a bin packing problem that seeks to pack the software licenses in virtual machines (and thus PMs) that minimize the cost and energy values under resource capacity constraints. In this problem, we are looking for a new placement of software licenses with the principal objective of minimizing the overall license, VMs, and energy costs and resolve the situations of over/under-utilization of physical machines. We are concerned with the question: how to determine a mapping of software licenses to VMs in order to achieve this goal. We consider the following assumptions to specify the boundaries of our model.

- Each container executes one software license and each license will be placed/ moved with its dependencies. In the current work, we use containers to encapsulate software instances without detailing aspects related to container-based virtualization-related issues.
- Each containerized SL executes inside a single virtual machine and each VM is hosted by a single PM.
- Only CPU and memory computing resources are considered and the required amount of these resources for a containerized SL is known in advance.
- All instances of software are co-locatable together.
- In the cloud, it seems that no border to run every software instance in every machine. Zoltan [11] considered that, for a given license, the fee is proportional to the number of machines (virtual or physical) running software components belonging to this license. For the sake of simplicity, we adopt this point of view in our work and consider a VM-based license where a license is paid once for all the software instances running in the same VM.
- In the current work, we do not consider the constraints related to the number of instances of the same software in a given VM as the compliance considerations are out of the scope of the present contribution.

4. Proposed Work and Methodology

This section is devoted to presenting the proposed work. We will first describe our proposed architecture of software license consolidation and its components. Then, we will present the adopted methodology allowing us to reallocate the software licenses dynamically.

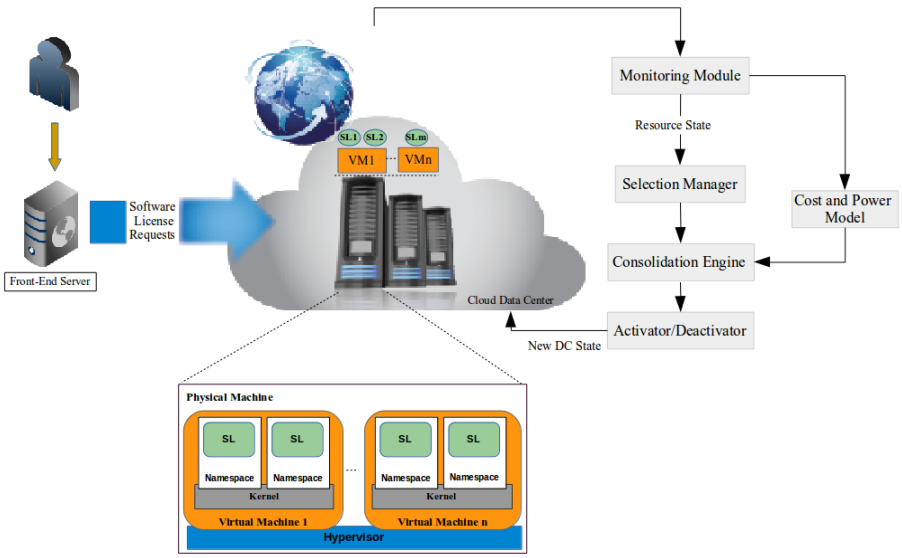


Fig.1. System architecture for our $\boldsymbol{DS^3C}$ solution

4.1. Proposed Architecture

The architecture of our proposed model is illustrated in Fig. 1. Concerning the virtualization architecture, we adopt a VM-container virtualization since it is considered the best virtualization technology until now [3]. In fact, the VM-based virtualization is based on what we call hypervisor, the management layer that allows sharing a single physical machine between many operating systems through virtual machines. In containerization, resources and processes are isolated using namespaces and containers share the OS kernel [13]. Indeed, in our previous survey [3], we deeply studied and compared these two virtualization technologies and deduced that containers surpass VMs in many parameters such as their lightweight, fast, dense deployment, and efficient resources utilization but they cannot assure the level of security and isolation that VMs provide. In our case, to guarantee a good level of security while managing software licenses, we consider the VM-container architecture. Thus, a software license is encapsulated in a container with its dependencies, which in turn run in a VM hosted in a PM as mentioned in the previous section.

For the consolidation process, to consolidate the workload on the minimum number of machines running the software license applications while minimizing the energy and total DC cost, our model is designed in many steps.

After receiving the user's requests to run the licensed software, and depending on the actual configuration of the DC in terms of machines and used resources, the consolidation is triggered. The key components performing the dynamic reallocation steps of our consolidation strategy are discussed below.

- Monitoring Module: the role of this module is to observe the state of machines and resource utilization within a DC. Depending on the current load of each running machine, the monitor checks the state of this machine to detect the under-loaded or over-loaded ones, often, based on Under-load (UL)/Over-load (OL) threshold. The identified machines are added to a critical PM list.
- Selection Manager: this entity selects a list of software licenses from critical machines identified in the previous step. In other words, it selects the most suitable SLs for migration from over-loaded machines and places them along with all SLs from under-loaded machines in the migration SL list.
- Cost and Power Model: this module is the estimator that allows quantifying the cost and energy within a DC to select the best destination optimizing them in the next step.
- Consolidation Engine: this component is responsible for the selection of the most suitable destination machines where the software licenses in the migration list can be shifted. If no machine is available, then a new machine is activated.
- Activator/Deactivator: the role of this module is to activate new machines (VMs/PMs) if no suitable destination machine is found and deactivate the non utilized machines i.e. freed VMs and/or idle PMs.

4.2. Consolidation solution

Fig. 2 presents the proposed methodology of software license consolidation. The SLC process illustrates the key steps to optimize data center states in terms of energy consumption and costs and guarantees optimal resource utilization. These steps must be performed based on the level of utilization of all PMs running the workload (the input of our scheme), every 5 minute time-lapse to dynamically maintain an optimal state in the cloud data center.

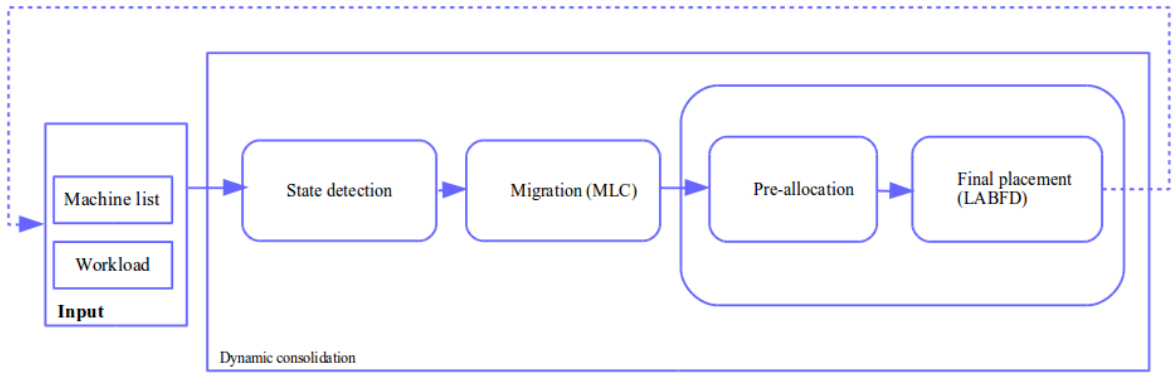


Fig.2. Methodology: A schematic diagram

The DC state is periodically evaluated and the consolidation decision is made based on resource state (state detection module). In this step, the critical machines are identified based on over-load (OL)/under-load(UL) threshold values to identify over-loaded/under-loaded PMs. A PM is considered over-loaded if its utilization reaches or exceeds the OL threshold. It is considered under-loaded if its utilization is less than the UL threshold. If a machine is over-loaded, some software licenses will be selected for migration to resolve the over-loaded state. For under-loaded hosts, all running software licenses will be migrated and they switch to the sleep mode after the migration ends. In fact, many competing technologies are used for optimizing configurations and allocation decisions [14]. Heuristic methods like

greedy approaches are very popular and promising strategies for solving this type of problem [15] due to their optimal computational time. Thus, in our work, we propose a new heuristic solution named Maximum License Cost (MLC) to select the software migration candidate (migration module). Also, we propose a License Adapted BFD (LABFD) version of the well-known BFD algorithm [16] to treat the destination selection of migrated software (final placement).

Algorithm 6: LABFD based Energy and Cost-Driven SL Relocation

```
Input: SLToMigrateList, PM_List
Output: migration_Plan
Begin
SLToMigrateList.sortDecreasingUtilization()
for SL in SLToMigrateList do
    minCost ← max
    destinationVM ← NULL
    L_VM ← preSelectedVMCandidates()
    for VM in L_VM do
        if VM has enough resources for SL then
            PM_src ← getSourcePM(VM)
            cost ← estimateCost(SL, VM, PM_src)
            energy ← estimateEnergy(SL, PM_src) * Ce
            CP ← w1 * cost + w2 * energy
            if CP <minCost then
                minCost ← CP
                destinationVM ← VM
    if destinationVM # NULL then
        add(SL,destinationVM) to migration_Plan
    else
        activate a VM from the list of inactive VMs or create a new VM
return migration_Plan
End
```

The main steps of containerized software license migration candidates' selection and allocation destination selection are detailed below. In the candidate's selection step, we propose a novel strategy (MLC) to select some SLs that have to be migrated if the migration condition is met. On a critical machine, the critical SL(s) that have the maximum license fees will be given the highest priority to be migrated and it is added to the list of migratable SLs. The general idea of our proposed strategy of destination machine selection is to select, in a first step, the VMs which contain the same license of a given SL (pre-allocation step) after sorting the list of migratable SLs in decreasing order, then select, in a second step, the right allocation which minimizes the overall cost and energy consumption and respects the constraints relating to resources. Thus, migrate the most expensive license from the critical PM, allows minimizing the license fees. For each selected SL, the requirements in terms of resources are analyzed to select the destination machine that can offer its required resources and at the same time give the best combination of cost and energy. The last step concerns the SL reallocation by selecting the suitable VM in terms of pre-established parameters which allows generating an optimal state within the DC cloud and updating it to restart the optimization process.

5. Performance Evaluation

To evaluate the performances of our proposed solution, we conducted a set of experiments. In this section, firstly, we elaborate our research questions followed by the experimental design of our evaluation methodology. After that, we present and briefly discuss the obtained results. Our experiments are designed to answer the following research questions:

- To what extent can our approaches improve cost, energy consumption, and resource utilization in the data center?
- To what extent can our approach ensure the scalability within a DC?

5.1. Experimental Design

In the cloud environment, it is hard to do experiments in the real world, mostly in our case treating the commercial software consolidation. For that, and for the sake of generality, we use extensive simulation to evaluate our proposed schema. This allows doing repeated simulations easily.

Setup: The execution of our proposed solutions is carried out on Intel® CoreTM i3-8100 CPU @ 3.60 GHz, 8 GB RAM, Windows 10. We implemented our solution in java language using Eclipse IDE and JDK 8. For the sake of simplicity, we consider the same number of licenses for each software. It's worth recalling that, all (containerized) SLs run in VMs, and VMs are allocated to PMs. Moreover, each VM can run many SLs and each PM can host many VMs. The performance of our proposed solutions has been evaluated under different load conditions. Based on the current utilization of all machines in the data center, the optimization process is performed every 5-minute time interval. Moreover, we consider homogeneous resources to evaluate the impact of the proposed scheme on the energy, cost, and resource utilization of DCs, and all the used PMs are of the type Haswell (Xeon 2695). To take the values of power consumption of this host type at different utilization levels, we used the SPECpower benchmark[1] as shown in Table 1. For resource leasing, many scenarios can take place [17]. For VMs pricing, we use the common pay as you go model. Most infrastructure providers offer hourly services, whose costs are mostly based on usage-dependent rates [18]. In our work, we used M1 medium instances VM types. For software pricing, inspired by App Service pricing[2] of Azure, we generated the software license fees randomly in [0,1). The parameter settings of our experiments are summarized in the Table 1.

Table 1. Parameter settings

	Parameter	Value
Simulation parameters	OL threshold	0.8
	UL threshold	0.4
	w1	0.5
	w2	0.5
Resource configuration	#PMs	100
	#VMs	400
	#Software	100
	#Licenses	10
	#SLs	1000
Power consumption values for Haswell (Xeon 2695) (Wh)	0% server utilization	70
	100% server utilization	120
Pricing	VM cost	$0.120 per VM per hour
	License fee	Random in [0,1]
SL generation parameters	Correlation coefficient (P)	1
	Normalized total PM CPU	1
	Normalized total PM Mem	1
	Normalized VM CPU	0.25
	Normalized VM Mem	0.25
	CPU *	0.025
	Mem *	0.025

[1] https://www.spec.org/

[2] https://azure.microsoft.com/en-us/pricing/details/app-service/window

Algorithm 2: Synthetic Dataset Generation

Input: N_S

Output: $\prec U^i_{CPU}, U^i_{mem} \succ Set$

Begin

for $i=1$ *to* N_S **do**

  $U^i_{CPU} \leftarrow 2 * U^*_{CPU}$

  $U^i_{mem} \leftarrow 2 * U^*_{mem}$

  r $\leftarrow$ rand(1.0)

  if $(r < P \wedge U^i_{CPU} \geq U^*_{CPU}) \vee (r \geq P \wedge U^i_{CPU} < U^*_{CPU})$ **then**

    $U^i_{mem} \leftarrow U^i_{mem} + U^*_{mem}$

return $\prec U^i_{CPU}, U^i_{mem} \succ Set$

End

Dataset: For workloads, inspired by [19] we generated synthetic instances to evaluate our solution. All the used values are normalized and detailed in Table 1. Te normalized values allow to run up to 4 VMs per host and up to 10 SLs in each VM. To generate our software synthetic instances, we used the random workload generator given by Algorithm 2. Then, for each software, a number of licenses are created (initially 10) and their costs are also randomly generated as mentioned above. To simplify concerns, we suppose that all software running in the Dc are licensed. We generated 500

software and created variable licensed instances.

5.2. Evaluation Metrics

Our objective is, principally, to optimize energy consumption and the total cost (i.e. fee of licenses and virtual machines renting). Thus, we consider these two metrics in the evaluation method whose equations are given below. The overall cost of a PM involves the sum of the license fees running in that PM and the cost of the VMs that host them. As we adopt the VM-based licensing model, license fees must be paid per VM for software of the same license running in that VM.

Energy consumption: The power consumption of a machine PM_i is modeled as the sum of the static part (P^{idle}) and dynamic part (see Eq 1), and its energy model is given by the Eq 2.

$$P_i\big(U_i(t)\big) = \mathrm{P_i^{idle}} + (\mathrm{P_i^{max}} - \mathrm{P_i^{idle}}) * U_i(t) \tag{1}$$

$$E_i = \int_{t1}^{t2} P_i\big(U_i(x)\big)dx \tag{2}$$

Where $\mathrm{P_i^{idle}}$ and $\mathrm{P_i^{max}}$ represent the energy consumption of a $\mathrm{MP_i}$ when its utilization is 0% and 100% respectively. U_i represents the CPU utilization of the PM in question and it is estimated by the sum of the utilization of all software instances running in all VMs hosted in that PM (see Eq 3).

$$U_i = \sum_j \sum_k \alpha_j^k \, \delta_i^j U_k \tag{3}$$

U_k represents the CPU utilization of the software license SL_k and α_j^k and δ_i^j are binary variables indicating if the software license SL_k is running in the VM_j and the VM_j is hosted by the PM_i respectively. It's evident from the Eq 1 that a PM, even when its CPU utilization worth 0, still consumes an important amount of power accounting for 70% of peak power [3]. This significantly affects the energy consumption savings if an idle PM can be deactivated in time. The total energy consumption in a DC cloud is given by the Eq. 4.

$$E = \sum_{i=1}^{N_{PM}} E_i \tag{4}$$

Where E_i represents the energy model given by the Eq. 2 and N_{PM} represents the total number of PMs.

Cost: The cost of our proposed model is formed by the fees paid by the user for renting a VM ($feeVM_j$) within an hour along with the license cost ($\mathrm{licenseFee}_{VM_j}$) which is estimated by the sum of the license fees of software running in a given VM. This latter is given by Eq. 5 where τ_k^l is a binary variable that indicates if a software license SL_k belongs to the license l.

$$\mathrm{licenseFee}_{VM_j} = \sum_k \tau_k^l \, \alpha_j^k \mathrm{licenseFee}_{SL_k} \tag{5}$$

Moreover, the total cost of a VM is estimated as its renting fee plus the sum of the license fees of each software instance type and is given by Eq. 6.

$$\mathrm{Cost}_j = \mathrm{FeeVM}_j + \mathrm{licenseFee}_{VM_j} \tag{6}$$

In the same way, the cost of a PM_i is estimated by the sum of the cost of all VMs running on it (Eq 7).

$$Cost_i = \sum_j \delta_i^j Cost_j \tag{7}$$

Finally, C represents the total cost within the DC.

$$C = \sum_{i=1}^{N_{PM}} Cost_i \tag{8}$$

Time: Further, to evaluate the scalability of our proposed strategies, we used the computational time of the proposed algorithms in the evaluation of the variation in software and license numbers alternately.

Number of PMs and VMs: Furthermore, the number of PMs in use during the simulation time and optimization process is used to show how our schemes optimize this number and then minimize the DC resource utilization. Finally, to show

the potential of the software license consideration in the consolidation approach and its effectiveness in terms of resources, we quantified the savings in terms of the number of VMs that can help to further decrease the number of active PMs implicitly and optimize the user monetary costs.

5.3. Results

In order to investigate the efficiency of our proposed migration candidate selection strategy (MLC) when combined with our proposed LABFD migration destination selection heuristic, we compared it with the baseline Maximum Usage (MU) selection strategy [20]. The MU was adapted to treat the software license migration and allows migrating the SLs having the maximum utilization of the CPU resource. Thus, for each evaluation metric, we note the obtained values along with the related savings. These combined strategies are compared to the initial data center state when no migration is performed (NO). The results of our proposed strategy are numerically given in Table 2. Equally, we quantified the percentage of savings for each approach.

The schematic representation of the energy results is given by Fig. 3 showing the superiority of our proposed migration candidates selection MLC approach over the baseline MU when combined with our proposed destination selection heuristic LABFD. We remark that the energy consumption decreased drastically reaching 68.75% savings by LABFD when combined with the MLC approach, while it reaches 65.47% savings when the baseline MU approach is used in combination with the LABFD heuristic.

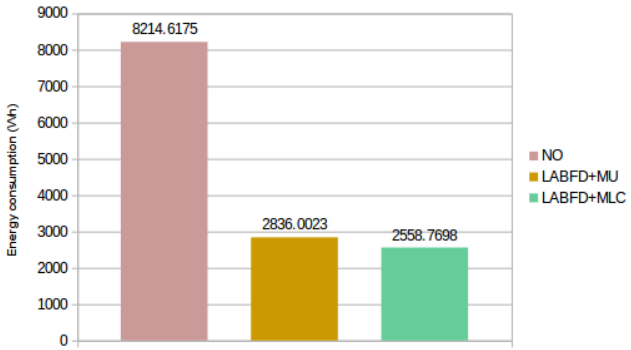


Fig.3. Energy consumption results (best) of our proposed heuristics

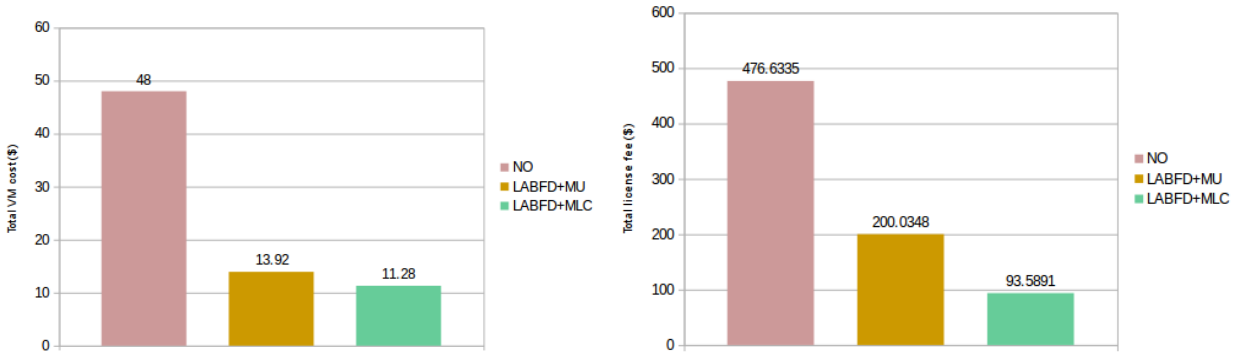
In terms of cost, the gains made compared to the initial state (NO consolidation performed) are also remarkable. In fact, the total cost is composed of total VM renting cost and the total license fees, as numerically illustrated in Table 2.

Table 2. Experimental results (best) for different consolidation schemes

Evaluation results		NO	LABFD+MU	LABFD+ MLC
Energy consumption	Results [Wh]	8214.6175	2836.0023	**2558.7698**
	Savings	-	65.47%	**68.85%**
Cost	License fees[$]	476.6335	200.0348	**93.5891**
	Savings	-	58.03%	**80.36%**
	VM cost[$]	47.9999	13.9199	**11.2799**
	Savings	-	71%	**76.5%**
	Total cost[$]	524.6335	213.9548	**104.8691**
	Savings	-	59.21%	**80.01%**
Number of used machines	#PMs	100	28	**25**
	Savings	-	72%	**75%**
	#VMs	400	116	**94**
	Savings	-	71%	**76.5%**

The results of total VM cost and total license fees are given by Fig. 4a and 4b successively. From these figures, we notice that our MLC policy surpasses the baseline MU selection strategy and gave the best results when combined with the adapted LABFD heuristic. From the detailed results of simulation during the execution time, we remark that the LABFD+MU approach decreased slightly after a fluctuating behaviour at the beginning. Contrariwise, the LABFD+MLC approach declined sharply especially at the beginning before stabilizing when approaching the optimal state. This is evident from Fig.4 that illustrates the VM and license cost results of the data center when reaching the optimal state. We can notice the important savings when the MLC policy is used beside the MU baseline mostly for the license costs. Basically, our destination selection strategy LABFD realized important savings when combined with the two SL candidates' selection policies with a significant superiority of MLC policy. With LABFD+MLC, the savings

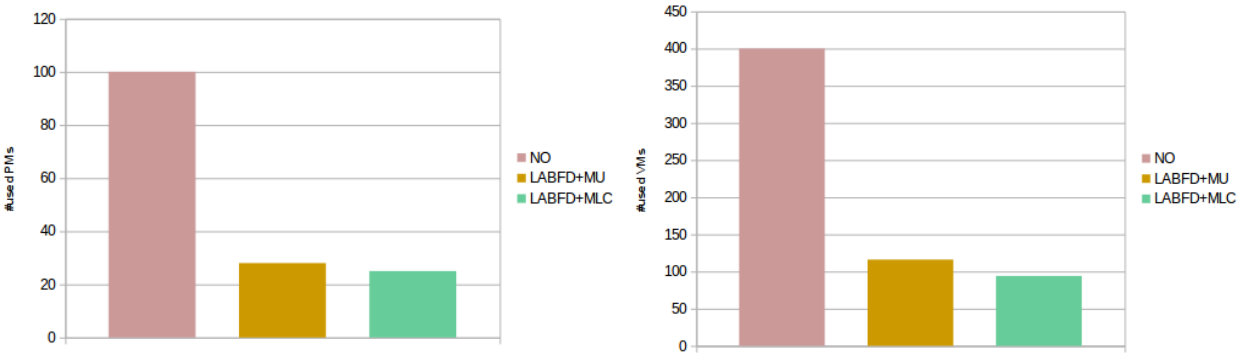
reach 80.36% in total license costs against 58.03% for LABFD+MU. For VM costs and total costs, the first one realized 76.5% and 80.01% against 71% and 59.21% for LABFD+MU, successively.



(a) Total VM cost (b) Total license fees

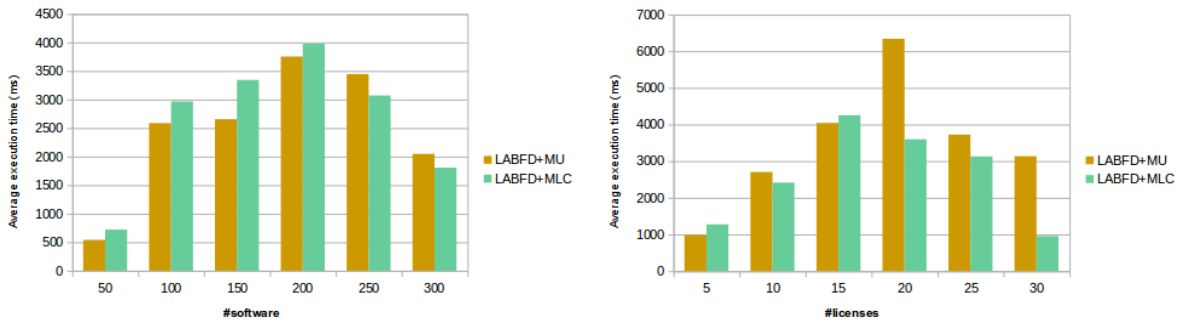
Fig.4. The total cost (best) of VMs (a) and license (b)

Consolidating workload in cloud data centers aims to use the DC resources efficiently while optimizing costs and energy, among others. To show how our proposed approach behaves in terms of the number of used servers (see Fig.5a) and VMs (see Fig.5b), we quantified these numbers along with the related savings. The results show that our strategy allows deactivating the maximum number of PMs, while utilizing the other machines more efficiently. Indeed, combined with the MLC policy, the LABFD heuristic achieved the best results in terms of active PMs and comes with 75% savings compared to the initial DC state. Generally, the savings of LABFD are important even when combined with the baseline MU policy, reaching 72% in the number of used servers. This shows the capacity of our proposed destination selection strategy to be combined either with the state of the art migration candidates' selection policies or to be adapted and combined with new policies. The same findings are obtained in the evaluation of the number of used VMs. Even if the LABFD heuristic gives good results in combination with the two candidates' selection policies, the MLC one always gives the best results with 76.5% savings against 71% savings by the baseline MU.



(a) Number of used PMs (b) Number of used VMs

Fig.5. The number of used machines after the consolidation process



(a) Varying the number of software (b) Varying the number of licenses

Fig.6. Average execution time function of the number of software (a) and licenses (b)

The final set of experiments is done with the regard to the scalability consideration. This later is quantified by the number of software licenses that can be treated in an acceptable time. In this regard, the scalability of our strategies is traduced by the execution time of the proposed approaches to perform consolidation regarding the number of software

variations and the number of license variations. These two dimensions we consider on scalability test vary between 50 and 300 software with the first one with an increment by 50, and the second dimension varies between 5 and 30 licenses with an increment by 5 given the same number of VMs and PMs. The results are shown in Fig. 6.

It is evident that the proposed LABFD approach converges to a minimized execution time for a number of software equal to 250 and a number of licenses equal to 20 when combined with the MLC policy compared to the baseline MU policy. The peak was reached with 2000 software licenses i.e. 200 software at around 4 seconds (see Fig. 6a) when we vary the number of software, and 20 licenses at around 6.336 seconds when we vary the number of licenses (see Fig. 6b). We remark that, in many cases, the LABFD+MLC strategy takes more time to perform the consolidation and reach the optimal state, especially, at the beginning. In general, it takes more time before reaching the peak then, from 2000 SLs, it gave a better time than the MU approach. This is explained by the fact that the MLC policy needs more time to find the best configuration to check the existence of software with the same license of the migrated SL while the MU policy is based on the maximum utilization of that SL. Hence, by increasing the number of software licenses, the MLC policy takes less time to reach the optimal state, which reflects the scalability of the MLC approach in a DC with large amount of software. It's notable that the MLC policy is more effective while varying the number of licenses because when this number increases, it tends to find more licenses of the same software. Finally, for the LABFD heuristic, we notice that the execution time may be important in some cases because it does the sorting of the selected list of migratable SLs, but, overall, this heuristic performed the consolidation in a reasonable execution time.

6. Results Analysis and Discussions

Commercial software consolidation in cloud data centers is a new field that is pushed by the techno-economic transformations and the emergence of new paradigms like softwarization and network function virtualization. Our experimental evaluation demonstrates that consolidating the workload at the SaaS level considering commercial software comes with important results. We have noticed that the energy consumption declined sharply after performing the consolidation. This is due to the savings in the resource utilization which translates by up to 75% deactivated servers as they are considered the most power consumers in a data center [3]. This allows saving the operational expenditure but also helps to combat global warming by decreasing the CO_2 emissions. On the other hand, the MLC policy minimizes costs to the maximum and comes with considerable savings compared to the baseline MU migration candidate selection policy. Even if the MLC selection policy gave better results than the baseline MU, this latter, when combined with our proposed LABFD heuristic, made important gains, although it is based on the utilization of software licenses to select the most used SLs as migration candidates. This is due to the destination selection policy, principally, the pre-selection step. In this phase, we select the VM candidates, which already have SLs of the same type as the SL to be migrated and the cost model used to calculate the cost of licenses in a VM according to the license model considered, i.e. counts only once the cost of software belonging to the same license type. Thereby, the destination selection and migration candidate selection approaches are complementary and our strategies allow us to achieve significant gains regardless of the candidate selection policy. We notice that, for VM costs, the gains made by the two selection policies are very close. For license costs, it is evident that the gains made by the two selection policies are similar to those of the total cost as quantified above. This shows the weight of license costs and the importance of this component in total cost savings. As a matter of fact, the VM cost savings are proportional to the number of deactivated VMs as shown in Table 2. The savings in terms of the number of virtual machines confirm the effectiveness of our proposed strategies. To ensure the scalability of our proposed approach, we visualized its temporal behaviour. In this respect, the goal of this experiment was to quantify the execution time for performing the consolidation. The experiments focused on DC dimensionality in terms of the number of software licenses only by, firstly, varying the number of software and keeping the number of licenses constant for each software. Secondly, varying the number of licenses for each software considering the same amount of software. We can conclude from these experiments that the proposed approach proves its scalability with a reasonable execution time to perform the consolidation. The MLC approach is adapted to DCs with a large number of software even if it makes much time in a DC with a small number of software to search for the best configuration because if the number of software is large with only 10 licenses for each software, the chance to find an SL with the same license of a migrated software is not important. On the other hand, increasing the number of licenses for the same software tends to increase the chance to optimize costs because many licenses may be collocated in the same VM and thus paid once. This is more important when using the MLC policy which tends to migrate the most expensive SLs and thus many instances from these costly SLs, while co-located together, are paid only once, which explains the superiority of this policy compared to the baseline MU policy. For example, with 50 software (500 software licenses), the MLC policy assured 8.96% more gains in license costs than the MU policy. On the other hand, with 5 licenses (500 SLs), the difference in license cost savings reached 21.64% when the MLC policy reached 72.75% gains and the MU 51.11% in license cost savings. For all experiments either in the number of software or number of licenses, the MLC approach proved its superiority against the MU. That reflects the importance of license consideration in the consolidation work when commercial software are in hand. Further, when the number of licensed software instances increases, the execution time decrease to some extent from a given value as long as the number of SLs

processed can be accommodated in the current configuration (in terms of resource capacity and utilization). Indeed, when the number of SLs (either by increasing the number of software or that of licenses) increase, with the same number of VMs/PMs, we tend to find the best configuration earlier and the minimum number of migrations is done so the execution time decreases as shown in the previous section. In conclusion, our proposed heuristics, which are computationally lightweight, have shown good results in the average computational time that decrease when the problem size increases. This proves the scalability of our proposed strategies.

7. Concluding Remarks and Future Work

Consolidation techniques allow optimizing resource management in cloud data centers. They aim to maximize resource exploitation while minimizing costs and energy consumption, among others. Nevertheless, the overwhelming majority of the consolidation work focuses on resource optimization in the IaaS level [3]. However, the emergence of the softwarization paradigm and the network function virtualization along with the related webized services represent a critical actual need in terms of software asset management and resource provisioning and scalability at the SaaS level.

The purpose is to optimize data centers state in terms of related costs and resources optimization with the inherent energy consumption. Therefore, we proposed $\boldsymbol{DS^3C}$, a dynamic approach for software license consolidation and resource optimization in virtualized cloud data centers. The originality of our work is to treat the license-related aspect along with the software consolidation in resource management in the cloud which was not being explored in literature. Our approach allowed us to save up to 68.85% of energy and 80.01% in total costs while 75% of servers and 76.5% of virtual machines are being deactivated successively. Concluding this work, we can claim that license management in the cloud is still in its early days. New and more sophisticated methods that deal with license management-related challenges need to be designed. In our present work, we shed light on the licensed software in virtualized cloud systems from the resource management point of view and compliance considerations are not within the scope of the current work. Moreover, compliance is the most critical parameter in software license management processes. So, to be more realistic, in continuing work we would address later some known licensing models considering the compliance aspect. Furthermore, in the current work, we only considered homogeneous resources in a relatively medium-scale scenario. Yet, to meet the actual need of big data centers and validate the performances of our approach regardless of the resource homogeneity, we plan to extend it to treat heterogeneous resources while considering data center dimensionality. Also, we plan to extend some other heuristic solutions which are popular in similar consolidation work at infrastructure and container service levels and visualize their behaviour while combined with our new migration candidates' selection policy MLC. Finally, in the current work, we addressed the dynamicity aspect of consolidation that beats the resource wastage caused by static allocation [3]. Still, from the architectural point of view, we believe that designing distributed consolidation solutions can further optimize costs and enables the natural scaling [3]. Thus, we plan to adopt a distributed control and decision by implementing and evaluating the effects of multi-agent systems in consolidation work that have proven their advantages in the resource management area in the cloud [21].

References

[1] C. N. Höfer and G. Karagiannis. Cloud computing services: taxonomy and comparison. Journal of Internet Services and Applications, 2:81–94, 2011.

[2] A. Sen, A. Garg, A. Verma, and T. Nayak. Cloudbridge: On integrated hardware-software consolidation. ACM SIGMETRICS Performance Evaluation Review, 39, 2011.

[3] L. Helali and M. N. Omri. A survey of data center consolidation in cloud computing systems. Computer Science Review, 39, 2021.

[4] C. Guerrero, I. Lera, and C. Juiz. A lightweight decentralized service placement policy for performance optimization in fog computing. J Ambient Intell Human Comput, 10:2435—-2452, 2019.

[5] G. Bista, E. Caron, and A. -L. Vion. A study on optimizing vnf software cost. In 2020 Global Information Infrastructure and Networking Symposium (GIIS), pages 1–4. IEEE, October 2020.

[6] M. K M Murthy, M. N. Ameen, H. A. Sanjay, and P. M. Yasser. Software licensing models and benefits in cloud environment: A survey. In Kumar M. A., R. S., Kumar T. (eds) Proceedings of International Conference on Advances in Computing. Advances in Intelligent Systems and Computing, volume 174, pages 645–650. Springer, New Delhi, 2013.

[7] M. Imdoukh, I. Ahmad, Mohammad, and G. h. Alfailakawi. Machine learning-based auto-scaling for containerized applications. Neural Comput Applic, 32:9745—-9760, 2020.

[8] S. E l. Kafhali, I. -E -l. Mir, K -h.. Salah, and M. Hanini. Dynamic scalability model for containerized cloud services. Arab J Sci Eng, 45:10693—-10708, 2020.

[9] D. Patel, M. Patra, and B. Sahoo. Energy efficient genetic algorithm for container consolidation in cloud system. In 2020 7th International Conference on Signal Processing and Integrated Networks (SPIN), pages 1066–1071, 2020.

[10] A. Tchana, N. De Palma, I. Safieddine, and D. Hagimont. Software consolidation as an efficient energy and cost saving solution. Future Generation Computer Systems, 58:1–12, 2016.

[11] Z. Mann. Resource optimization across the cloud stack.
IEEE Transactions on Parallel and Distributed Systems, 29:169–182, 2018.

[12] A. Karve, T. Kimbrel, G. Pacifici, M. Spreitzer, M. Steinder, M. Sviridenko, and A. Tantawi. Dynamic placement for clustered

web applications. In in Proc. Of the 7th International Conference on World Wide Web, 2006.
[13] I. Mavridis and H. Karatza. Performance and overhead study of containers running on top of virtual machines. In 2017 IEEE19th Conference on Business Informatics (CBI), volume 2, pages 32–38. IEEE, 2017.
[14] M. G. Kambalimath and M. S. Kakkasageri. Cost Optimization based Resource Allocation Scheme for Vehicular Cloud Networks. International Journal of Computer Network and Information Security, 11, 2020.
[15] M. R. Chowdhury, M. R. Mahmud, and R. M. Rahman. Implementation and performance analysis of various vm placement strategies in cloudsim. Journal of Cloud Computing, 4, 2015.
[16] A. Beloglazov and R. Buyya. 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, 24:1397 –1420, 2012.
[17] C. Wu, R. Buyya, and K. Ramamohanarao. Cloud pricing models: Taxonomy, survey, and interdisciplinary challenges. ACM Computing Surveys, 52:108:1–108:36, 2020.
[18] S. H. Chun. Cloud services and pricing strategies for sustainable business models: Analytical and numerical approaches. Sustainability, 12:1–15, 2020.
[19] Y. Gao, H. Guan, Z. Qi, Y. Hou, and L. Liu. A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. Journal of Computer and System Sciences, 79:1230–1242, 2013.
[20] S. F. Piraghaj, A. V. Dastjerdi, R. N. Calheiros, and R. Buyya. A framework and algorithm for energy efficient container consolidation in cloud data centers. In 2015 IEEE International Conference on Data Science and Data Intensive Systems, pages 368–375, 2015.
[21] L. Helali and Z. Brahmi. Self-organizing agents for dynamic network- and qos-aware service composition in cloud computing. In Proceedings of 37th International Conference on Information Systems Architecture and Technology – ISAT 2016 – Part II. Advances in Intelligent Systems and Computing, volume 522, pages 111–124. Springer, Cham, 2017.

Authors' Profiles

Leila Helali is a Lecturer in Computer Science at the University of Sousse, Tunisia. Her research interests include reinforcement learning; cloud computing, compliance and resource optimization. Moreover, she has enough knowledge of applied machine learning, distributed systems and computer programming. She is a reviewer of international journals such as the journal of Supercomputing.

Mohamed Nazih Omri received his Ph.D. in Computer Science from University of Jussieu, Paris, France, in 1994. He is a professor in computer science at the University Of Sousse, Tunisia. From January 2011, he is a member of MARS (Modeling of Automated Reasoning Systems) Research Laboratory. His group conducts research on Information Retrieval, Data Base, Knowledge Base, and Web Services. He supervised more than 20 Ph.D. and Msc students in different fields of computer science. He is a reviewer of many international journals such as Information Fusion journal, Psihologija Journal, and many International Conferences such as AMIA, ICNC-FSKD, AMAI, etc.

How to cite this paper: Leila Helali, Mohamed Nazih Omri, "Heuristic-based Approach for Dynamic Consolidation of Software Licenses in Cloud Data Centers", International Journal of Intelligent Systems and Applications(IJISA), Vol.13, No.6, pp.1-12, 2021. DOI: 10.5815/ijisa.2021.06.01