Dynamic Virtual Machine (VM) Optimization in a Cloud Environment

Aziz SAIBOU[1*], Onyonkiton Theophile ABALLO[2], Arsene *Narcisse* DAGBA[3] and Alamou Taohidi LAMIDI[4]

[1234] LETIA, University of Abomey-Calavi, Abomey-Calavi, Benin

Abstract.

Cloud computing appears to be an ecosystem with flexible and scalable IT resources. The finding is that cloud services are increasing considerably and resource management is a real problem. This is the case with the static allocation of virtual machines (VMs) that fail to effectively manage multiple workloads in real time. This leads to inefficiencies and high costs. This article proposes a system for dynamic optimization of virtual machines in a cloud to satisfy the multiple and varied requests of users. The main objective is to design a system that dynamically adjusts the number and configurations of VMs according to cloudlet requests, while optimizing performance and costs. This solution allows to learn, scale and remove virtual machines taking into account the variation in demand. It therefore ensures the optimal use of data centre resources.

Keywords: *cloud computing, cloudlets, resources, static, Virtual Machines.*

Introduction

Cloud is a new system that proves its usefulness daily. Users can now access the requested resources on demand, paying only for what they consume. A new service model has been born for this purpose and has revolutionized the way Information Technologies resources are managed. However, the efficient allocation of resources (Virtual Machines VMs, CPUs, etc.) remains a major challenge. Static machine management (VMs) does not respond effectively to changes in demand, which can lead to inefficiencies in processing, quality of service and increased costs. Indeed, in the doctoral thesis of the author Rachida AOUDJIT (wife LAGAB) [1], charging decision-making depends on load and energy thresholds and the responsiveness of the system depends on these thresholds. Setting thresholds requires quantifying the load generated by running mobile nodes. Thus, a question to be resolved is to consider whether the thresholds are fixed or whether they adapt as the load of the network changes. The implementation of variable thresholds is not easy to solve because it poses a crucial question: how to disseminate the new value or what value should the new threshold take?

* Corresponding author : aziz.saibou@imsp-uac.org

To address this challenge, the present paper makes the following original contributions:

i) a dynamic VM scaling mechanism that automatically adjusts the number and characteristics of virtual machines according to Cloudlet demand. ii)A game-theory-based mathematical model (Nash equilibrium) governing VM creation and deletion decisions. iii) an experimental validation using the CloudSim framework demonstrating measurable improvements in resource utilization and cost efficiency compared to static provisioning.

This solution enables the automated provisioning, adjustment, and termination of VMs according to demand, ensuring optimal use of data center resources.

1 Literature Review

1.1 Cloud Concepts

1.1.1 Definition of Cloud

According to authors J. Geelan and K. Sheynkman [2], deployment, scaling and consumption of resources on demand are key elements for cloud computing. Other authors such as R. Buyya and al. [3] emphasize the service level agreement and the quality of service. Authors L. Vaquero and al. [4] and NIST [5] consider automatic scaling, the "pay only for what you consume" on-demand model, ubiquitous use and virtualization as essential elements of cloud. The cloud is a remote access by internet for resources treatment such as storage or servers, available on demand and built on virtualization. The figure 1 illustrated the deployment models of cloud.

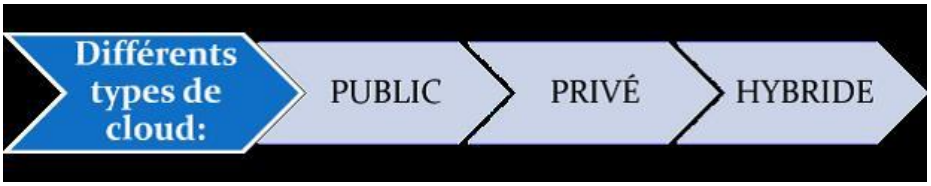


Fig. 1. The different types of cloud

1.1.2 The different components of the cloud

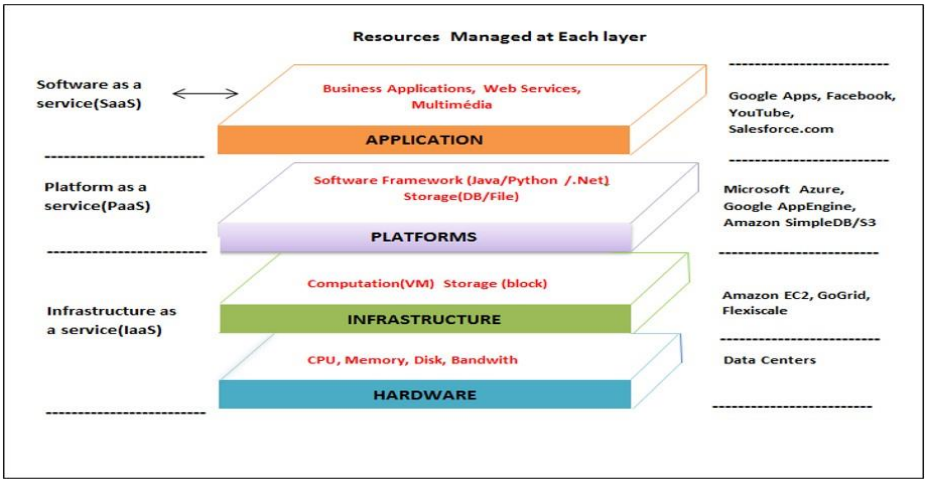


Fig. 2. Figure showing the different components of the cloud [6].

1.2 Virtualized Environments and Virtual Machines (VMs)

Virtualization is a technique that can logically represent physical resources.

- Operation: Several operating systems can be started on a physical server
- Resource optimization: Virtualization helps maximize the treatment of physical resources, reducing waste and infrastructure costs.

A virtual machine (VM) is a software processing unit that operates within another physical processing unit. These virtual machines can be installed on hypervisors such as: VMware ESXi, Microsoft Hyper-V, KVM (Kernel-based Virtual Machine).

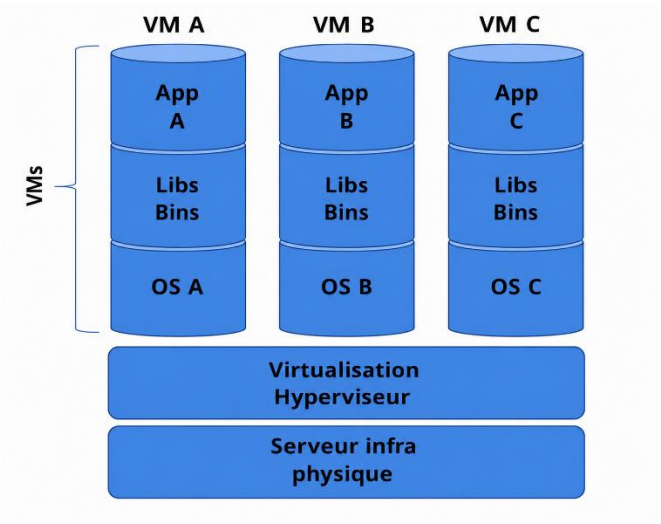


Fig. 3. Structure of a virtual machine and virtualization. [7]

1.3 Methods for Dynamic Optimization of Virtual Machines

Dynamic optimization of VMs in a cloud environment relies on several methods to enhance resource allocation and workload management:

1.3.1 Consolidation of VMs

Consolidate multiple VMs onto a small number of physical servers.

- Objective: Reduce energy consumption while maintaining performance.
- Associated algorithms: Bin Packing, Best-Fit Decreasing.

1.3.2 Live VM Migration

Migration refers to the seamless transfer of a running VM from one physical machine to another without disrupting its operation or affecting the availability.

- Techniques: Pre-copy (Move one VM from one physical server to another while making services accessible)
  Post-copy:(Move one VM from one physical server to another when it is on).
- Advantages: Better resource allocation, improved reliability.
- Disadvantages: Potential impact on performance during migration.

1.3.3 Dynamic Resource Provisioning

Automatically adjust resource size while taking into account implementation needs

- Examples: AWS Auto Scaling, Google Compute Engine Autoscaler.
- Advantages: proactive management of the increase, minimization of over-provision costs

1.3.4 AI and Machine Learning-Based Optimization

It is about making resources available using machine learning models. This allows to assign resources proactively. For example, we have SVM models (Support Vector Machines), neural networks.

- Advantages: Reduced allocation costs and improved performance.
- Limitations: Reliance on historical data and complex algorithms.

1.4 Related Works

The author Rajkumar Buyya and pioneer of the CloudSim Simulator has worked on dynamic resource allocation algorithms to efficiently use energy and optimize costs. His article entitled "Energy-Effective Management of Data Center Resources for Cloud Computing: A Vision, Architectural Elements, and Open Challenges" has made a significant contribution to his dynamic allocation algorithms[8].

It should be noted that Ching-Hsien Hsu, a specialist in distributed systems and cloud optimization, has contributed to the optimization of resource management by focusing on performance and reducing energy costs. This author also worked on multi-criteria planning and virtual machine migration strategies. He focused on indicators such as energy

consumption as well as service levels (SLA). Its reference's article "Dynamic Virtual Machine Resource Provisioning for Energy-Efficacy Cloud Computing"[9] is a key reference in this area" has a lot of impact in this area.

Albert Zomaya has made major contributions to data center resource scheduling and workload balancing algorithms for cloud infrastructures. His research focuses on energy-aware optimization techniques that dynamically adjust VM workload distribution to minimize power consumption. He is co-author of "A Taxonomy and Survey of Energy-Efficient Data Centers and Cloud Computing" [10].

Usman M. J. and al. proposed a resource allocation algorithm using a flower pollination algorithm to optimize energy efficiency compared to a genetic algorithm in the cloud. This approach has also reduced the number of migrations in resource use (CPU and RAM) [11]. Ali Aghasi and al. employed the Q Learning technique to address virtual machine placement [12].

Qiang Li is known for his work on performance optimization and cloud resource scheduling. He has investigated multi-objective scheduling and cost-aware resource provisioning in distributed computing environments. His research also explores SLA-driven optimization and adaptive VM scaling to address fluctuating workloads [13].

2 Proposed Solution

Following Albert Y. Zomaya's work on energy-aware optimization algorithms that dynamically adjust VM workloads based on demand to minimize power consumption, several researchers have integrated energy efficiency considerations and Service Level Agreements (SLAs) into their approaches. Several studies have addressed virtual machine migration techniques and are based on resource forecasting for placement optimisation and virtual machine allocation.

We design a system that automatically creates virtual machines by taking into account workloads (cloudlet). This makes it possible to optimize the performance and cost-effectiveness of the system. Our solution makes the use of resources more efficient than traditional methods of supply. It ensures the reduction of operational costs by using dynamic allocation of virtual machines and appropriate resource management.

2.1 Proposed Framework

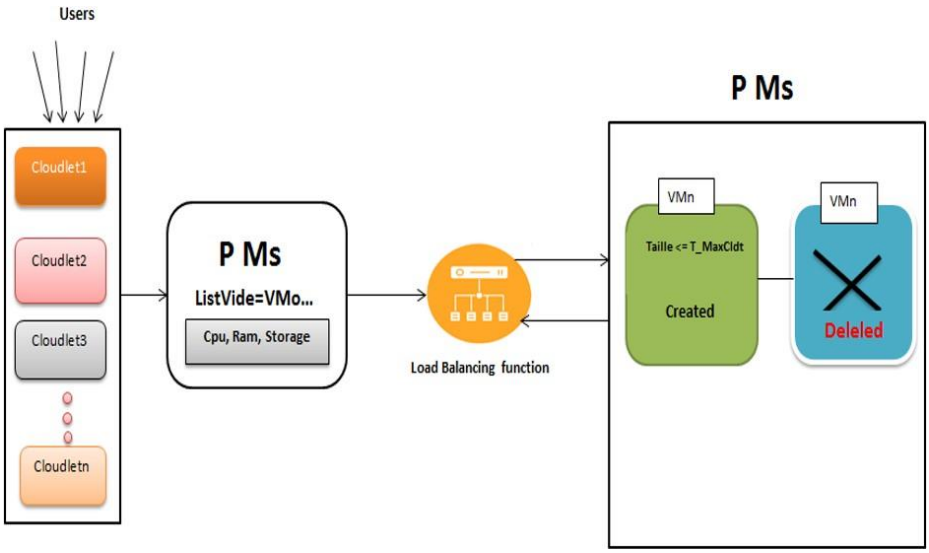


Fig. 4. Proposed framework

The framework operates as follows:

- Initialization of various cloudSim simulation parameters such as datacenter, virtual machines, brokers and cloudlets.
- Creating a Datacenter with Hosts and Characteristics — A datacenter is created with a list of hosts with specific characteristics (CPU, memory, storage, bandwidth).
- Creating a Broker to Manage VMs and Cloudlets — A broker is configured to coordinate the submission and allocation of cloudlets to VMs.
- Creating an Empty VM List — The list of virtual machines (VMs) is initialized as an empty list for dynamically adding VMs.
- Batch Cloudlet Submission — The createCloudlets function is used to submit cloudlets in batches.
- Processing Each Batch of Cloudlets — Adjusting VM resources based on the size of the cloudlets.
- Submitting the VMs to the broker.
- Saving the specifications of the created VMs to an output file.
- Auto-scaling VMs based on average CPU utilization: calculating the average CPU utilization for the VMs, saving average CPU usage to an output file, graphically updating CPU usage, creating a new VM if the average CPU usage exceeds a threshold, and deleting an unused VM if the average CPU usage falls below a threshold.

2.2 Mathematical Model

2.2.1 Assumptions

A VM becomes inactive if it has no more cloudlets to run:

$$I_i = 1 \text{ if } U_i = 0 \quad (1)$$

where U_i is the number of cloudlets executed by VM i. If the sum of assigned cloudlets exceeds the capacity of a VM V_i, additional VMs must be created to distribute the load. The goal is to maximize the efficiency of resource (VM) allocation by minimizing the cost of VM usage and deleting inactive VMs.

2.2.2 Utility of players (VMs)

In this zero-sum game, each player represents a VM, and the objective of each player is to maximize their utility by minimizing their processing cost while executing a maximum number of cloudlets without exceeding their capacity. Each player wants to maximize their defined Gi gain:

$$G_i = A_i - Cu \times Cex \quad (2)$$

Where:

- A_i is the interest provided by cloudlets in VM Vi;
- *Cu* is Cu is the cost of using the VM per unit of time;
- *Cex* is the total operating cost of VM Vi

2.2.3 Conditions for Creating and Deleting VMs

Creation of new VMs:

If Ui > Cex, create VM to contain surplus cloudlets. An additional VM is created if avg_CPU > Cex.

Deletion of inactive VMs:

$$\text{Delete } V_i \text{ if } U_i = 0 \text{ for a duration } \Delta t \quad (3)$$

Equilibrium Solution:

In the Nash equilibrium [13], each VM runs an optimal number of cloudlets so that no VM can increase its utility by unilaterally changing the distribution of cloudlets. Idle VMs are deleted, and new VMs are created only when necessary to meet capacity constraints. The optimization can be formulated as:

$$\min \Sigma_i P_i I_i \quad (4)$$

Under the constraints:

- $\Sigma_i U_i = N$ (total cloudlets) (5)
- $I_i = 1$ if $U_i = 0$; $I_i = 0$ otherwise

Where P_i is the cost of deleting VM V_i. This model incorporates the principles of game theory to optimize VM creation and deletion while ensuring efficient cloudlet distribution.

2.3 Implementation

2.3.1 Hardware

The algorithm was implemented on a computer with following characteristics:

- Processor: Intel(R) Core(TM) i5-2450M CPU @ 2.50 GHz
- RAM: 8 GB
- OS: Microsoft Windows 10 Professional 64-bit

2.3.2 Software

The system implementation required java programming language in cloudsim simulator using Eclipse IDE.

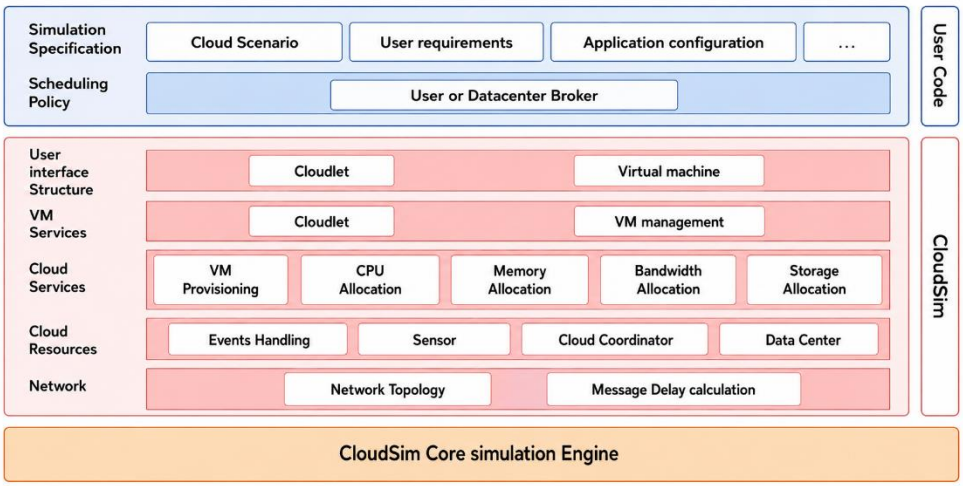


Fig. 5. CloudSim simulator architecture

3 Results and Performance

The graph illustrates the processing time of requests as a function of the VM executing them.

3.1 Experimental Graph on 23 Queries

The processing time for the first set of 23 cloudlets is shown in Figure 6. First, we have observed a regular progression of requests from cloudlet 1 to 10First, we have observed a regular progression of requests from cloudlet 1 to 10, corresponding to the activity of VM1 with an execution time that ranges from 0 to around 30 ms. This execution time is stable and increases linearly, reflecting a uniform distribution of the load on VM1. A first peak appears just after request 10, reaching around 32 ms. The second phase begins at request 11 and runs until request 20, corresponding to the work of VM2, with a linearly increasing and balanced execution time ranging from 32 ms to around 62 ms. A second, less marked peak occurs after request 20, between 62 and 65 ms. Finally, the last three requests (21 to 23) are processed by VM3. These requests are processed between 65 and 70 ms. The time remains short here because the volume to be processed is much smaller, which confirms the effectiveness of dynamic distribution.

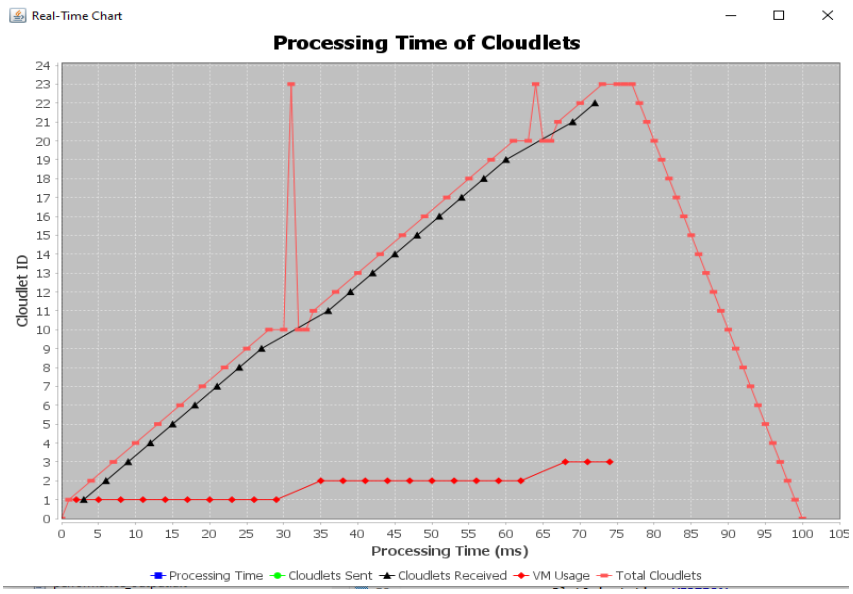
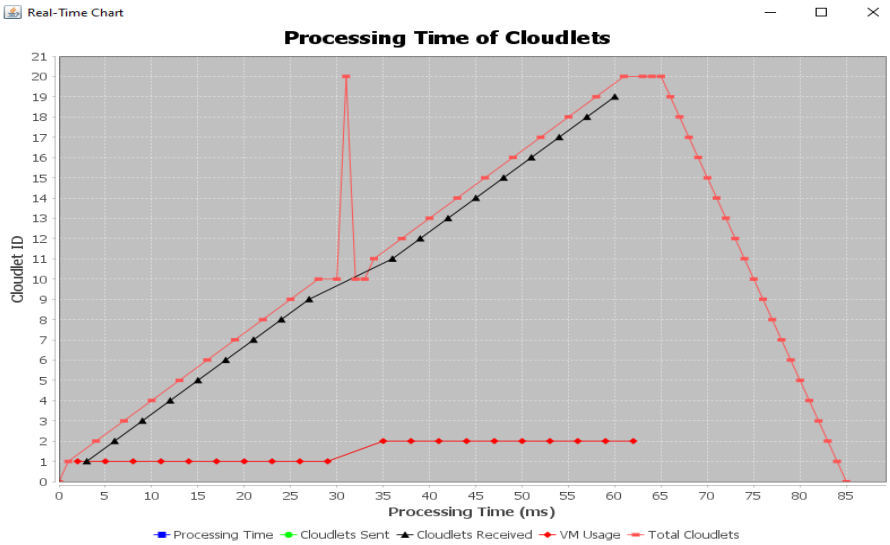


Fig.6. Graph treatment time for 23 nail injections

3.2 Experimental Graph on 20 Queries

The second scenario is made on 20 cloudlets as shown in Figure 7. We observed a linear progression of requests from 1 to 10. This corresponds to the activity of the virtual machine (VM1) with a execution time between 0 and 30 ms. This execution time is stable and increases linearly in the load distribution on VM1.A peak appears after 10 queries corresponding to 32 ms. The second phase starts at the 11th request until the last request (20th request) with a linear progression corresponding to the virtual machine (VM2)'s activity. The execution time is between 32 ms and 62 ms. No VM is more created because



no extra charge. This demonstrates the effectiveness of the system to avoid unnecessary waste of resources

Fig.7. Graph treatment time for 20 nail injections

3.3 Other Results

Figure 8 shows the treatment of all cloudlets injected into the simulator. It specifies the start of the datacenter and the creation of virtual machines that can be used depending on the number of cloudlets received. Cloudlet processing is done in batches of 10 by VM with a delineation of the different VMs (VM1, VM2, VM3). It should be noted that total processing time increases linearly with the number of cloudlets. This shows that our system is stable, predictable at the proposed automatic scale. For temporarily failing transition points, brokers guarantee continuity of services.

```
Starting CloudSim version 3.0
Broker is starting...
Datacenter_0 is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: Trying to Create VM #1 in Datacenter_0
0.0: Broker: Trying to Create VM #2 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenter #3, Host #0
0.1: Broker: VM #1 has been created in Datacenter #3, Host #0
0.1: Broker: VM #2 has been created in Datacenter #3, Host #0
0.1: Broker: Sending cloudlet 0 to VM #0
0.1: Broker: Sending cloudlet 1 to VM #0
0.1: Broker: Sending cloudlet 2 to VM #0
0.1: Broker: Sending cloudlet 3 to VM #0
0.1: Broker: Sending cloudlet 4 to VM #0
0.1: Broker: Sending cloudlet 5 to VM #0
0.1: Broker: Sending cloudlet 6 to VM #0
0.1: Broker: Sending cloudlet 7 to VM #0
0.1: Broker: Sending cloudlet 8 to VM #0
0.1: Broker: Sending cloudlet 9 to VM #0
0.1: Broker: Sending cloudlet 10 to VM #1
0.1: Broker: Sending cloudlet 11 to VM #1
0.1: Broker: Sending cloudlet 12 to VM #1
0.1: Broker: Sending cloudlet 13 to VM #1
0.1: Broker: Sending cloudlet 14 to VM #1
0.1: Broker: Sending cloudlet 15 to VM #1
0.1: Broker: Sending cloudlet 16 to VM #1
0.1: Broker: Sending cloudlet 17 to VM #1
0.1: Broker: Sending cloudlet 18 to VM #1
0.1: Broker: Sending cloudlet 19 to VM #1
0.1: Broker: Sending cloudlet 20 to VM #2
0.1: Broker: Sending cloudlet 21 to VM #2
0.1: Broker: Sending cloudlet 22 to VM #2
267.066: Broker: Cloudlet 20 received
318.774: Broker: Cloudlet 21 received
320.086: Broker: Cloudlet 22 received
632.242: Broker: Cloudlet 2 received
634.798: Broker: Cloudlet 5 received

0.1: Broker: Sending cloudlet 17 to VM #1
0.1: Broker: Sending cloudlet 18 to VM #1
0.1: Broker: Sending cloudlet 19 to VM #1
0.1: Broker: Sending cloudlet 20 to VM #2
0.1: Broker: Sending cloudlet 21 to VM #2
0.1: Broker: Sending cloudlet 22 to VM #2
267.066: Broker: Cloudlet 20 received
318.774: Broker: Cloudlet 21 received
320.086: Broker: Cloudlet 22 received
632.242: Broker: Cloudlet 2 received
634.798: Broker: Cloudlet 5 received
690.156: Broker: Cloudlet 15 received
701.6759999999999: Broker: Cloudlet 14 received
704.9559999999999: Broker: Cloudlet 6 received
708.06: Broker: Cloudlet 10 received
767.9799999999999: Broker: Cloudlet 0 received
777.6999999999999: Broker: Cloudlet 7 received
816.8299999999999: Broker: Cloudlet 13 received
819.194: Broker: Cloudlet 12 received
825.304: Broker: Cloudlet 8 received
841.64: Broker: Cloudlet 1 received
842.6: Broker: Cloudlet 18 received
843.626: Broker: Cloudlet 9 received
855.442: Broker: Cloudlet 19 received
856.714: Broker: Cloudlet 11 received
877.5300000000001: Broker: Cloudlet 4 received
879.258: Broker: Cloudlet 3 received
896.518: Broker: Cloudlet 16 received
897.1560000000001: Broker: Cloudlet 17 received
897.1560000000001: Broker: All Cloudlets executed. Finishing...
897.1560000000001: Broker: Destroying VM #0
897.1560000000001: Broker: Destroying VM #1
897.1560000000001: Broker: Destroying VM #2
Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Broker is shutting down...
Datacenter_0 is shutting down...
Simulation completed.
```

Fig. 8. Capture showing total cloudlet processing

3.4 Comparison and Analysis

For the validation of the proposed solution, we used two additional approaches. The first is to compare the results with the theoretical analytical model derived from the Nash equilibrium formulation and the second is to move to a qualitative and quantative comparison with the related literature work..

3.4.1 Validation 1: Comparison with theoretical results

The Nash equilibrium model defines the gain of each VM as

Gi = Ai - Cu × Cex, where Ai represents the advantage of using cloudlets, Cex, the total operating cost of VM Vi, Cu the cost of using. Using typical parameters (Bi=1 unit per cloudlet, Cu=0.05 unit per ms), then (Bi=1 unit per cloudlet, Cu=0.025 unit per ms), We can calculate the gain at the simulation output.

Table 1. Theoretical vs. Simulation results (23-cloudlet scenario, λ = 0.05)

VM	Cloudlets	Sim. time (ms)	G_i theoretical	G_i sim. (est.)	Consistency
VM1	**10**	0 – 30	10 − 0.05×30 = 8.50.	≈ 8.50	✓
VM2	10	32 – 62	10 − 0.05×62 = 6.90	≈ 6.90	✓
VM3	3	65 – 70	3 − 0.05×70 = 2.50	≈ 2.50	✓

A_i = 1.0 unit/cloudlet, Cu = 0.05 cost/ms Cex is the maximum running time for VM

Table 2. Theoretical vs. Simulation results (23-cloudlet scenario, λ = 0.025)

VM	Cloudlets	Sim. time (ms)	G_i theoretical	G_i sim. (est.)	Consistency
VM1	**10**	0 – 30	10 − 0.025×30 = 8.50.	≈ 9.25	✓
VM2	10	32 – 62	10 − 0.025×62 = 6.90	≈ 8.45	✓
VM3	3	65 – 70	3 − 0.025×70 = 2.50	≈ 1.25	✓

A_i = 1.0 unit/cloudlet, Cu = 0.025 cost/ms. Cex est la durée d'exécution maximale pour la VM

The two simulation tests performed allow us to have the theoretical predictions of the Nash equilibrium model for the 3 virtual machines created from the 23 cloudlet processing. The gain obtained on the scenarios decreases from the VM1 machine to the last VM3 machine as the cumulative processing time increases. There is therefore an analytical consistency which validates the first approach.

3.4.2 Validation 2: Comparative study with literature

Recent studies have been carried out to improve work on resource allocation.A study carried out in [14] optimizes the allocation of virtual machines on the basis of reinforcement learning controllers. This study measures convergence rate , load distribution and migration reduction.Another study in [15] addresses an ant-inspired algorithm that focuses on processor usage, memory and bandwidth between different physical hosts.The differences observed between these approaches are shown in Table 3

Table 3. Table showing a comparison of literary approaches to validation

Criterion	**Proposed work (U_i)**	**[14] Learning automate**	**[15] Ant algorithm**
Optimization method	Game theory (Nash eq.)	Reinforcement learning	Swarm intelligence
VM scaling	Dynamic (auto-scale)	Dynamic	Static placement
Request quota / VM	Yes (10 cloudlets/VM)	No	No
VM deletion (idle)	Yes (if $U_i = 0$)	Partial (migration)	No
Formal model	Nash equilibrium	Markov decision process	Ant colony optimizatio n
Time-based analysis	Yes (ms per cloudlet)	Convergence rate only	No
Simulation tool	CloudSim	CloudSim	CloudSim / custom

The table 3 above presents the distinctive features of the approach proposed by two literature-related studies. Our solution combines only a formal theoretical game model (Nash balance) to make decisions about the virtual machine cycle. It allows you to set the cloudlet quota per virtual machine and an explicit analysis based on the time taken by cloudlets. Thus, these two methods validate the scientific strength of our system of dynamic optimization of virtual machines in a cloud environment.

4 Conclusion

In conclusion, a dynamic cloud environment requires orchestration for virtual machine (VM) management. This orchestration makes it possible to optimize resources and reduce costs. Dynamic supply techniques and methods are a strong point for systems to adapt to changing workloads. They automatically adjust the number of active virtual machines and remove inactive virtual machines. The proposed system allows the balance between performance, availability and profitability to be maintained, taking into account unexpected

changes in workload. Our system was evaluated using 2 additional scenarios in CloudSim. The first involves processing 23 cloudlets and the second 20 cloudlets generating 3 VMs and 2VMs respectively. In addition, we compared related work that addresses the automation of reinforcement learning [14] and algorithms inspired by ants [15]. This proves that our solution is scientifically credible.

Future work will focus on virtual machines that have been removed in case of inactivated. Re-create them automatically for possible use. Consideration should also be given to extending experimentation on a real infrastructure to confirm simulation results.

References

1. [1] R. Aoudjit (wife Lagab), "Charging Decision-Making in Mobile Networks: Load and Energy Threshold Management," Doctoral thesis, Université Mouloud Mammeri de Tizi-Ouzou, 2020.
2. [2] J. Geelan. Twenty-one experts define cloud computing. Virtualization, Electronic Magazine, 2008.
3. [3] R. Buyya, C. S. Yeo, and S. Venugopal. Market-oriented cloud computing: Vision, hype and reality for delivering IT services as computing utilities. In 10th IEEE HPCC'08, pages 5–13. IEEE, 2008.
4. [4] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner. A break in the clouds: Towards a cloud definition. SIGCOMM Comput. Commun. Rev., 39(1):50–55, 2008.
5. [5] P. Mell and T. Grance. The NIST definition of cloud computing (draft). NIST Special Publication, 800(145):7, 2011.
6. [6] Zhang, Q., Cheng, L. and Boutaba, R. 2010. Cloud computing: state-of-the-art and research challenges. Journal of Internet Services and Applications, 1(1):7–18.
7. [7] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic. Cloud computing and emerging IT platforms. Future Generation Computer Systems, 25(6):599–616, 2009.
8. [8] R. Buyya, A. Beloglazov, and J. Abawajy. Energy-Efficient Management of Data center Resources for Cloud Computing. arXiv:1006.0308, 2010.
9. [9] C.-H. Hsu. Dynamic Virtual Machine Resource Provisioning for Energy-Efficient Cloud Computing. In Proc. IEEE Int. Conf. on Cluster Computing, 2012.
10. [10] A. Y. Zomaya and Y.-C. Lee. A Taxonomy and Survey of Energy-Efficient Data Centers and Cloud Computing Systems. arXiv:1007.0066, 2010.
11. [11] Usman M. J., Ismail A. S., Chizari H., et al. Energy-efficient virtual machine allocation technique using flower pollination algorithm in cloud datacenter. Journal of Bionic Engineering, 16:354–366, 2019.
12. [12] Aghasi Ali, Jamshidi Kamal, Bohlooli Ali, and Bahman Javai. A decentralized adaptation of model-free Q-learning for thermal-aware energy-efficient virtual machine placement in cloud data centers. Computer Networks 224 (2023): 109624.
13. [13] Q. Li. Optimization of Resource Scheduling in Cloud Computing. ResearchGate, 2012.
14. [14] Nash Equilibrium-Based Replacement of Virtual Machines for Efficient Utilization of Cloud Data Centers. Academia.edu.
15. [15] Multi-Resource Optimization via Ant-Inspired Algorithm for Virtual Machine Placement in Cloud Data Centers. arXiv:2311.16147, 2023.