J Inf Process Syst, Vol.14, No.3, pp.569~589, June 2018
https://doi.org/10.3745/JIPS.01.0028

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

Load Balancing in Cloud Computing Using Meta-Heuristic Algorithm

Youssef Fahim*, Hamza Rahhali*, Mohamed Hanine*, El-Habib Benlahmar*, El-Houssine Labriji*, Mostafa Hanoune*, and Ahmed Eddaoui**

Abstract

Cloud computing, also known as "country as you go", is used to turn any computer into a dematerialized architecture in which users can access different services. In addition to the daily evolution of stakeholders' number and beneficiaries, the imbalance between the virtual machines of data centers in a cloud environment impacts the performance as it decreases the hardware resources and the software's profitability. Our axis of research is the load balancing between a data center's virtual machines. It is used for reducing the degree of load imbalance between those machines in order to solve the problems caused by this technological evolution and ensure a greater quality of service. Our article focuses on two main phases: the pre-classification of tasks, according to the requested resources; and the classification of tasks into levels ('odd levels' or 'even levels') in ascending order based on the meta-heuristic "Bat-algorithm". The task allocation is based on levels provided by the bat-algorithm and through our mathematical functions, and we will divide our system into a number of virtual machines with nearly equal performance. Otherwise, we suggest different classes of virtual machines, but the condition is that each class should contain machines with similar characteristics compared to the existing binary search scheme.

Keywords

Bat-Algorithm, Cloud Computing, Load Balancing, Pre-scheduling, Virtual Machines

1. Introduction

Cloud computing allows users to access different platforms, infrastructures, and software as services, allowing one to take advantage of all these technologies without requiring extensive information technology (IT) expertise. The cloud can reduce the information system's cost through virtualization technologies [1]. Cloud computing is a new computer model whose purpose is to suggest the IT services into requested services (on-demand). The goal is to allow services to be accessible from anywhere, anytime, and by anyone (country as you go). The real novelty of the cloud is its systematic approach. One of the cloud's drawbacks, however, is that the cloud users' data are stored on the cloud provider. Consequently, there could be unauthorized access to data [2]. With cloud computing technology, users

Manuscript received July 12, 2017; first revision September 27, 2017; second revision November 20, 2017; accepted December 12, 2017.
Corresponding Author: Youssef Fahim (Fahimyoussef87@gmail.com)
* Laboratory of Information Technology and Modeling, Faculty of Sciences Ben M'sik, Hassan II University of Casablanca, Casablanca, Morocco (fahimyoussef87@gmail.com, rahhalihamza@yahoo.fr, {mohamedhanine24, h.benlahmer}@gmail.com, labriji@yahoo.fr, mhanoune@gmail.com)
** Dept. of Computer Sciences, Shaqra University, Riyadh, Saudi Arabia (ahmed_eddaoui@yahoo.fr)

can have access to many services without being forced to manage the underlying infrastructure, which is often complex. The adoption of this model raises many challenges including quality of service (QoS) about the provided services, particularly concerning the load balancing between virtual machines [3]. Task processing and allocation division between a data center's virtual machines are made through load balancing algorithms, statics, and dynamics [4,5] under the supervision of a data center's controller. The data center is an aggregation between multiple physical servers, and it receives cloud service clients' requests in order to execute it by using its own virtual machine (VM) in a sequential or parallel way. Sharing resources is among the strengths of data centers in cloud computing. VMs are virtual units that have a specific performance determined by the cloud provider, such as the capacity of memory storage or its processor's performance for tasks processing [6]. The load balancing algorithm are classified it into two types—static and dynamic. They are used to distribute the work load among existing VMs under the direction of a data center controller [7]. The cloud providers face challenges regarding the quality service due to the following causes:

- The complexity of the cloud infrastructure
- The weaknesses of load balancing algorithms
- The variety of stakeholders whose role is to execute client requests [8,9].

This article will suggest a load balancing improvement between a data center's VMs in order to ensure a greater QoS and to maximize the use of provided resources.

Our improvement has two main phases. The first phase is the classification of tasks into levels ('odd levels' or 'even levels') in ascending order based on the meta-heuristic "Bat-algorithm", and the second phase focuses on the tasks' allocation according to the provided levels by the Bat-algorithm. Through our mathematical functions, we will divide the data center into a number of VMs with similar performance.

Our new load balancing approach allows the cloud service's providers to maximize the use of their equipment resources and software, but it also serves to avoid most problems caused by the older static and dynamic load balancing algorithms [2,9].

In the second section of this article, we will present a state-of-the-art advancement on load balancing algorithms and meta-heuristic algorithms. The third and fourth sections will present our proposed model and the results obtained, respectively. Finally, a conclusion will be made in the fifth section.

2. State-of-the-Art

Following our recent research performed on the various static and dynamic load balancing algorithms [10], as subsequent results of the extension of the study presented in [2], we try to demonstrate the weaknesses and disadvantages of existing load balancing, especially when they increase [1]. Thus, our proposal is to design and model a new approach of load balancing by running together one of the meta-heuristic algorithms presented in this state-of-the-art model. In our model, the meta-heuristic algorithms and some load balancing algorithms from the two categories (static and dynamic) will be presented in order to choose the meta-heuristic that suits with Phase 1 of our implementation.

2.1 Load Balancing Algorithms

Load balancing algorithms are used to distribute new requests of users in a data center between the

VMs, according to each technical [7,11], in order to guarantee an equal number of tasks allotted to each machine. According to the study presented on these algorithms in our model, we find that there are two types of algorithms: the static and the dynamic [12,13]. Before assigning tasks to the emphasized VMs, these algorithms remain uninterested in the task characteristics and, therefore, the degree of load unbalance between the VMs as the load increases.

2.1.1 Static algorithms

The statics load balancing algorithms is an assignment from a set of tasks to a set of resources which can take either a deterministic or a probabilistic form [4]. In addition, static approaches are defined usually in the design or implementation of systems [13]. Their main drawback is that the current state of the system is not considered when making the decisions and therefore it is not a suitable approach in systems such as distributed systems where most states of the system changes dynamically. We will present some of these algorithms as follow.

- Throttled load balancing algorithm

This algorithm ceases the state of the virtual machine to find out whether or not it is available for the new allocation. The algorithm then sends the acknowledged VM identifier (ID) to the data center controller for a new allocation and the achievement of the task. When the task processing is fulfilled, the virtual machine sends the result to the data center controller, which notifies the algorithm for an allocation cut-off [14].

- Central manager algorithm

The central load manager assigns tasks to a VM in each new allocation with a minimum load compared to the other machines. From time to time, it updates the system's load status as it experiences changes. This status allows the data center to make the correct load balancing decision when it is creating new tasks [10].

- Active monitoring load balancer

The active monitoring loan balancer is an algorithm that counts the minimum number of tasks assigned to each virtual machine and sends its ID to the data center controller, which notifies the algorithm to adjust the allocation and incrementation in its table with the new number of tasks assigned to the machine owning the identified ID [14].

- Round robin algorithm

This task allocation order is based on the FCFS technical [15]. Requests are distributed among VMs in turns using a data center controller regardless of the different allocated tasks characteristics [16]. This algorithm is known for its fair allocation of the VMs to all nodes [17], but it doesn't have control over the workload distribution.

- Weighted round robin algorithm

This algorithm is similar to the 'round robin' in a sense that the manner by which requests are assigned to the nodes is still cyclical, except that the node with the higher specifications will be given a greater number of requests [18].

- Threshold algorithm

In this algorithm, the processes are assigned immediately to hosts that are being locally created. While keeping a copy of the system's load, the processor's load can either be underloaded, medium, or overloaded. The main drawback of this algorithm is the fact that the tasks are allocated locally, which means that there can be processors that are overloaded while others are underloaded. That issue will cause a significant increase in the execution of the tasks [10].

- Opportunistic load balancing algorithm

In this approach, each unexecuted task is appointed randomly to an available node [19]. The main goal of this approach is to keep the node's load full [17]. Even if it provides load balancing, the main drawback of this algorithm is its inability to calculate the current execution time of the node.

- Min-Min algorithm

In this approach, the task having the minimum execution time will be the first to be assigned to a VM that can complete the task [20]. This algorithm outperforms other algorithms when there are more tasks that have a small execution compared to the number of tasks that have a long execution time [17]. The task having the maximum execution time will stay in the queue for an undetermined time period. This way of processing will only result in a poor utilization of the VMs [21].

- Opportunistic load balancing algorithm + Min-Min algorithm

This approach is a combination of the opportunistic load balancing (OLB) algorithm and the Min-Min algorithm [22]. The OLB algorithm is used in order to keep every node's workload full, and the LBB algorithm is used to minimize the execution time of the tasks assigned to the nodes. This approach is known for its better resource utilization and the enhancement of the work's efficiency, but it does not tolerate error [17].

- Improved weighted round robin

This algorithm is based on the 'weighted round robin'. It is similar in regard to the cyclic tasks allocation, except that it considers the priority and the length of the task to choose the most suitable VM [23].

A general disadvantage of all static schemes is that the final selection of a host for process allocation is made when the process is created and cannot be changed during process execution to make changes in the system load.

2.1.2 Dynamic algorithms

Unlike static algorithms, dynamic algorithms consider the actual load of the VMs [24]. The goal of the dynamic algorithms is to decrease the number of errors caused by the static algorithms [11]. Some of these algorithms are presented below.

- Central queue algorithm

The data center controller has a main queue in which tasks are classified in first-in, first-out (FIFO) order. If a virtual machine goes into the "underload" status, this main queue sends a request for a new task allocation to the data center controller, which deletes the task from the queue and sends it directly to the specified recognized machine [10].

- Local queue algorithm

Under this algorithm, all VMs get local queues, and, when they go into underload mode, it looks for other tasks to allocate from the other further away VMs. The advantage of this algorithm is the dynamic migration and efficient allocation of all the tasks loaded in the data center controller to the VMs [25].

- Mini time processing load balancer

The architect of this algorithm has improved the design of the 'Efficient response time load balancer' algorithm in a new one called 'Processing time load balancer', which considers the current state of the VM workload through the processing time, which is a main metric of this algorithm [26].

- Token ring algorithm

In this algorithm, tokens are moved around the system in order to minimize the system cost. Heuristic approach can be used to remove the drawback of token routing algorithm. The Token Ring algorithm provides fast and efficient routing decisions. Hence, no communication overhead is generated in this approach [27].

- Least connections

This algorithm is based on the selection of the service with the least number of active connections to ensure that the load of the active requests is balanced on the services. This method is the default method because it provides the best performance [28].

- Weighted least connections

This algorithm introduces a "weight" component based on the respective capacities of each server. Just like the 'weighted round robin', each server's "weight" must be specified beforehand. A load balancer that implements the 'weighted least connections' algorithm considers two things: the weight capacities of each server and the current number of clients currently connected to each server [29].

- Trust management algorithm

This algorithm proposes a trust value management for the cloud Infrastructure-as-a-Service (IAAS) parameters. The main objectives of this algorithm are to increase the resource utilization and to decrease the request response time of the system. It also enhances the QoS based on the proposed model, but it does not provide a formal description of the expressed model [30].

- Cloud friendly load balancing

This algorithm aims at reducing the energy consumption, timing penalty, and execution time for a virtualized environment [31].

- Two-phase load balancing

This algorithm is shown in service nodes, service managers, and request managers [32]. This algorithm uses the request manager, which is a bottleneck. It also has a lack of simulation, and it does not provide the performance metrics.

- Stochastic hill climbing

Stochastic hill climbing (SHC) is a variant of Hill Climbing that deals with the bottleneck problem. Unlike other algorithms, the SHC gives a better performance than the FCFS and 'round robin' algorithms, but it faces a lack of resources utilization [33].

- L3B algorithm

This algorithm is placed between users and cloud nodes. L3B improves the cloud performance and reduces power consumption and customer cost through a mechanism that initializes a suitable VM if the incoming workload exceeds beyond a certain threshold and switch-off VM if the workload decreases in compression with a certain threshold [34].

- Cloud partition load balancing

Cloud partition load balancing aims at improving the efficiency in the public cloud environment. The algorithm uses algorithms with low complexity for underloaded situations in partitions. Unfortunately, there are neither simulations nor implementations of this solution [35].

- VM-based two-dimensional load management

This algorithm reduces system overhead by reducing migration. However, it is considered only for applications with seasonal attribute change, which is not extendable for a real system. This algorithm also suffers from the absence of resources utilization [36].

- DAIRS

This approach aims at balancing the workload in data centers by relying on three parameters: CPU, memory, and network bandwidth. It also uses four queues: waiting queue, requesting queue, optimizing queue, and deleting queue. This approach is not suitable for cloud systems due to the fact that it is centralized [37].

- TOPSIS method

This method chooses which VM should be selected for migration to a new physical machine and which physical machine should host the selected VM [38].

- EuQoS

The EuQoS system for scheduling VMs consists of two parts: load balancer and agent-based monitor. The load balancer module provides three mechanisms: balance triggering, EuQoS scheduling, and VM control. The distribution of workload system is done by weighted round-robin load balancing algorithm [39].

- Ant colony optimization

This algorithm, as its name states, is based on the behavior of ants to detect the location of underloaded or overloaded nodes. It then updates the resources utilization table [40]. This algorithm is known for its scalability but has low throughput [17].

- Bee-MMT

This approach uses the artificial bee colony algorithm with the feature of minimal migration time [41].

- Improved ABC

The main purpose of this approach is to suggest a load balancing strategy for cloud computing systems. This way, the system's throughput is improved. The drawback of this method is the lack of resources utilization and its instability [42].

- Dynamic adaptive replica strategy (DARS)

This approach is based around the idea of creating replicas of nodes if they become overloaded and stores them in other nodes in order to release their workload [43]. If it can reduce the number of overloaded nodes, even if it is a decentralized approach, it does not compromise the access delay. One of the main features of DARS is the fact that creating replicas and storing them are up to the nodes themselves, depending on the state of their loads.

2.2 Comparison

The performance of various load balancing algorithms is measured by using the following parameters.

1) Overload rejection

If load balancing is not possible, additional overload rejection measures are needed. When the overload situation ends, then, first, the overload rejection measures are stopped. After a short guard period, load balancing is also closed down [44].

2) Fault tolerant

This parameter shows if an algorithm is able to tolerate tortuous faults or not. It enables an algorithm to continue operating properly in the event of some failure. If the performance of the algorithm decreases, the decrease is proportional to the seriousness of the failure. Even a small failure can cause total failure in load balancing [44].

3) Forecasting accuracy

Forecasting is the degree of conformity of calculated results to its actual value that will be generated after execution. Static algorithms provide more accuracy than dynamic algorithms as in former most assumptions are made during compile time and later during execution [44].

4) Stability

Stability can be characterized in terms of the delays in the transfer of information between processors and the gains in the load balancing algorithm by obtaining faster performance in a specified amount of time [44].

5) Centralized or decentralized

Centralized schemes store global information at a designated node. All sender or receiver nodes access the designated node to calculate the amount of load transfers and also to check that tasks are to be sent to or received from. In a distributed load balancing, every node executes balancing separately. The idle nodes can obtain load during runtime from a shared global queue of processes [44].

6) Nature of load balancing algorithms

Static load balancing assigns load to nodes probabilistically or deterministically without consideration of runtime events. It is generally impossible to make predictions of arrival times of loads and processing times required for future loads. On the other hand, in dynamic load balancing, the load distribution is made during run-time based on current processing rates and network condition. A DLB policy can use either local or global information [44].

7) Cooperative

This parameter shows whether processors share information between them in making the process allocation decision other are not during execution. What this parameter defines is the extent of independence that each processor has in concluding how it should use its own resources. In the cooperative situation, all processors have the accountability to carry out its own portion of the scheduling task, but all processors work together to achieve a goal of better efficiency. In the non-cooperative, individual processors act as independent entities and arrive at decisions about the use of their resources without any effect on the rest of the system [44].

8) Process migration

The process migration parameter shows the capacity of a node to transfer a process to another node. It decides whether to create it locally or create it on a remote processing element. The algorithm is capable of deciding whether to make changes of load distribution during execution of process or not [44].

9) Resource utilization

Resource utilization includes automatic load balancing. A distributed system may have unexpected number of processes that demand more processing power. If the algorithm is capable of utilizing resources, they can be moved to underloaded processors more efficiently [44].

The comparison of some of the load balancing algorithms quoted above is shown in Table 1.

Table 1. Parametric comparison of load balancing algorithms

	Overload rejection	Fault tolerant	Forecasting accuracy	Stability	Dynamic or static	Cooperative	Process migration	Resource utilization	Centralized or decentralized
Round robin	No	No	More	Large	S	No	No	Less	D
Local queue	Yes	Yes	Less	Small	Dy	Yes	Yes	More	D
Central queue	Yes	Yes	Less	Small	Dy	Yes	No	Less	C
Central manager	No	Yes	More	Large	S	Yes	No	Less	C
Threshold	No	No	More	Large	S	Yes	No	Less	D
Least connections	No	No	More	Small	Dy	Yes	No	Less	C
Token Ring	No	No	More	Small	Dy	Yes	Yes	More	D
Trust management	No	No	More	Small	Dy	Yes	No	More	D
Cloud-friendly LB	No	No	More	Small	Dy	Yes	Yes	More	C
Two-phase LB	No	No	More	Small	Dy	Yes	No	More	C
Stochastic Hill Climbing	No	No	More	Small	Dy	Yes	No	More	C
L38	No	No	More	Small	Dy	Yes	No	More	C
Cloud partition LB	No	No	More	Small	Dy	Yes	No	More	C
VM-based 2 dimensioned	No	No	More	Small	Dy	Yes	No	More	C
DAIRS	No	No	More	Small	Dy	Yes	No	More	C
TOPSIS	No	Yes	More	Small	Dy	Yes	Yes	More	D
EUQOS	No	No	More	Small	Dy	Yes	No	More	C
Ant Colony Optimization	No	No	More	Small	Dy	Yes	No	More	C
Bee-MMT	No	Yes	More	Small	Dy	Yes	Yes	More	D

2.3 Meta-Heuristic Algorithm

A meta-heuristic algorithm is formally defined as an interactive generation process that directs the exploration and the use of the research space. Meta-heuristic algorithms are among the techniques that can be used to solve complex problems, including the load balancing which is highlighted in this contribution.

2.3.1. Tabu search

Tabu search is the method that allows tracking of the regions of space solutions that have already been searched. It starts from an initial random solution and moves successively to each neighbor of the current solution. It stands on the concept of taboo list, which is a special short-term memory formed by the previously visited solutions. Indeed, instead of a complete solution, the short-term memory carries only some attributes of every solution, and, then, it gives no access to the considered solutions and avoids making a loop on the optimal solutions [45,46].

2.3.2 Genetic algorithm

Genetic algorithms aim to find solutions to difficult problems. Their basic idea is to generate an initial random population formed with individual solutions to the problem called chromosomes, and then develop it following a number of iterations called generations. During each generation, every chromosome is evaluated by some measurement of aptness. To create the next generation, new chromosomes, called offspring, are formed either by fusing two current generation chromosomes using an operator crossover or by altering a chromosome using an operator mutation. A new generation is formed by selection, according to the correctness values, and, in such way, some of the parents and children are thrown to maintain a constant size of the population. Regulator chromosomes have higher probabilities of being selected. After several generations, the algorithms converge towards the best chromosome, which represents the optimal solution to the problem [47].

2.3.3 Simulated annealing algorithm

The simulated annealing (SA) technique was initially proposed to solve the hard-combinatorial optimization problems through controlled randomization by simulating the temperature falling procedure of particular systems in thermodynamics. It is a technique to find a better solution for an optimization problem by trying random variations of the current solution. The main feature is that a worse variation may be accepted as a new solution with a probability, which results in the SA's major advantage over other searching methods. That is, the ability to avoid becoming trapped at local minima. Theoretically, SA is able to find the global optimal solution with probability equal to 1 [48].

2.3.4 Honey bee behavior algorithm

This algorithm aims at realizing a well-balanced load among all VMs in order to maximize the system efficiency. The suggested algorithm equally balances the tasks priorities on machines so that the time spent in the waiting queue may be minimized [49].

2.3.5 Bat-algorithm

Bat-algorithm is a modeling inspired by the method of echolocation. It is used by "micro-bats" to identify the shortest iteration to the "prey". Developers try to improve this meta-heuristic in order to meet the requirements of NP-hard computations [50]:

1. All bats use echolocation to realize the distance and the difference between obstacles and prey.

2. To search for prey, bats randomly fly with a velocity (v_i) at position (x_i), a fixed frequency (f_{min}), and a wavelength and a varying loudness (A_0). They can adjust automatically the wavelength (or frequency) and the emission rate of the pulse (r_2) (0, 1) depending on the closeness to their target prey.

3. Although the loudness can vary in several ways, we suppose that the loudness fluctuates from a big loudness (A_0) (positive) to a constant minimal value (A_{min}).

The pseudo-code of the Bat-algorithm [50]:

-Data:

Initializes bat population position (x_i), velocity (v_i), pulse rate (r_i), loudness (a_i), and frequency (f_i).

-Result:

Optimized Solution

-Begin

Set maximum number of iterations and represent it using max.

while (curr_iter<max)

Generate new solutions by adjusting position, frequency and velocity

If (r and >r_i)

Select best solution among all solutions

Generate a local solution around the selected best solution.

End if

If ((r and <a_i) && ($f(x_i) < f(x^*)$)

Accept new solutions and increase r_i and reduce a_i.

End if

Rank the bats and find the current best solution, x_i.

End while

Post-process the results.

-End

We chose to use the Bat-algorithm to develop our algorithm due to the fact that it is simple, flexible, and easy to implement. It can also solve a wide range of problems efficiently. There is also the fact that it works well with complicated problems and can give an optimal solution in quick time.

3. System Model and Algorithm

Following our research proposed in [2], we noticed that the load balancing based only on the statute of the VM is insufficient to guarantee an ideal balancing. Consequently, we propose in this article a new approach of load balancing based on several parameters which influence normal unfolding of the burden-sharing and the allowance of the tasks between the VMs of a data center. We separated our

modeling in two parts. The first part is the pre-classification of the tasks in an ascending order based on the meta-heuristics "Bat-algorithm". The second is the allowance of the tasks to the numbers of the virtual machines having more or less equal performances according to Fig. 1 in lower part.

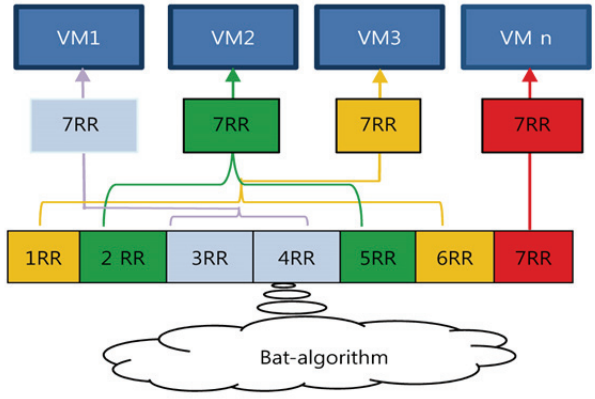

Fig. 1. Load balancing model.

3.1 Classification of the Tasks Based on Bat-Algorithm

For our first part, we chose to collaborate with the research carried out by [50] and [51]. This choice has as objective pre-classifying the spots according to the required levels suitable, and the resources. Let us note that our model is adaptable with all the heuristic preset ones in the state of the art or else. It is used to seek the level of suitable classification according to Fig. 2 in lower parts.

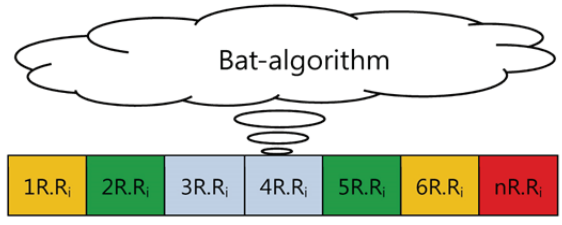

Fig. 2. Pre-classification based on "Bat-algorithm".

Stages of our first classification (Bat-algorithm):

- Initialize the parameters of Bat-algorithm (position, loudness, speed) [15].
- Initialize the levels of classifications: to assign the first task T_i to a level ($R.R_i$).
- Seek second site $R.R_i+1$ of the task T_i+1 containing: $R.R_i+1$ in $\{R.R_i, R.R_i*N, (R.R_i)/N\}$
- Use Bat-algorithm in order to detect the ideal site of the following tasks Ti+1 in the levels of classification
- Identify the number of the box of the Ti+1 task with the N while= $R.R_i+1/R.R_i$
- Launch the selection of the good sites for the following tasks T_i+1 according to the resources requested ($R.R_i+1$), one using path-algorithm
- Repeat the operation for all the tasks of the segment requested
- Pass to the second part of our model of load balancing in lower part

3.2 Allocation of the Tasks to the Virtual Machines

Thanks to the Bat-algorithm, the tasks are now redistributed in different levels. The choice of maximum number of the levels of classification depends on the different required resources of the tasks, which arrive in a data center. The levels presented in the form of a table of temporal scheduling have parameters of alterable classification according to the types of the programs arriving at our model of load balancing (meta-heuristics wished).

Whatever the metric of classification (execution time, processing time, standard of the tasks, resources requested… etc.), our approach successively begins from a lower level up to a higher level (key of our balancing) provided that

$$R.Rs = R.R_i \times NR \quad (1)$$

$R.R_s$ is the metric of a higher-level task (R.R=resource required for the task); $R.R_i$ is the metric of a lower level task and NR is the number of ranking of the higher level.

Example of Fig. 3, if the number of the desired levels is 7, then $R.R_i = 1$ and R.Rs= 7:

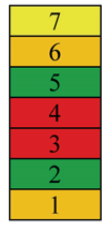

Fig. 3. Levels of classification.

Thanks to our technique of pre-classification presented before, we can

- have an equal distribution of the tasks pre-classified in each level of scheduling.
- obtain a load balancing between the virtual machines provided that:

If the number NR (number of the levels) is even:

Nbr.VMs=Nbr.level/2

If the number NR is odd:

Nbr.VMs=(Nbr.level+1)/2

Nbr.VMs is the number of the virtual machines, Nbr.level is the number of the levels.

In hoping to develop the performance of the load balancing parameters, we undertook a deep study on the different existing algorithms. As a result, we found out that they are not taking into consideration the parameters of the tasks to balance the load. Thus, we developed a hybrid model based on two different, but complimentary, approaches:

- Pre-classifying tasks by using the meta-heuristic Bat-algorithm, and determining the levels by calculating the metric using Eq. (1).
- Determining the number of sufficient VMs by using the levels found previously, and by taking into consideration whether the levels are even or odd.

4. Experiment and Results

4.1 Experiment

In order to evaluate the performance of the suggested load balancing algorithm, we implemented it on CloudSim, which presents the different stages of this process.

We proceeded to the simulation of a data center, initially consisting of several VMs of the same type. Each of these will receive a single task with random metric (execution time, response time, etc.).

In this scenario, we will treat an odd number of tasks (Fig. 4). In each instance, we will have the same number of odd tasks distributed on the VMs.

	T1	T2	T3	T4	T5	T6	T7	T8
Number of tasks	5	5	5	5	5	5	5	5
Input	5	10	5	6	9	30	5	5
	16	20	15	30	24	90	15	15
	40	40	30	18	18	60	30	40
	67	30	40	25	35	150	40	67
	80	50	50	12	40	120	50	80

Fig. 4. Task allocation.

Now, to explain how our pre-classifications method shown above works, we will take the example of time T6 shown in Fig. 4. By performing a task scheduling method, our algorithm will order the tasks in different levels in an ascending order depending on the metric used (execution time or response time). Then, we will select the number of tasks that will take in charge these tasks. Because we have here an odd number of tasks (5), we will follow the rule stated above: Nbr.VMs= (Nbr.levels+1)/2. By following that rule, we will only use three virtual machines to take in charge these tasks in this scenario. The allocation of the tasks is shown in Fig. 5, where the task with the maximal metric value will be given to a VM, then the goal is to give to the other VMs all the tasks whose metric sum will be almost equal to the maximal metric value. In this example, the maximal metric value is 150.

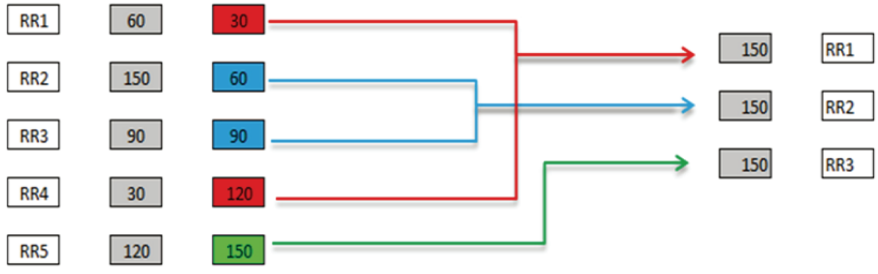

Fig. 5. Allocation of the tasks to the VMs.

This provides a balance in the distribution of the workload on a lower number of virtual machines compared to the initially used machines. The experiments that we conducted consist of executing the algorithm presented above.

Our algorithm adds a new concept of elasticity, which is the calculation of the optimal number of virtual machines (Fig. 6) depending on the received tasks at time T (Fig. 4).

	T1	T2	T3	T4	T5	T6	T7	T8
Number of VMs	8	3	5	3	2	3	5	8
Output without queue management	85	50	55	30	84	150	55	85
		50		31	42	150		
	16	50	55	30		150	55	16
	67							67
			30				30	
	40							40

Fig. 6. Calculation of the optimal number of VMs.

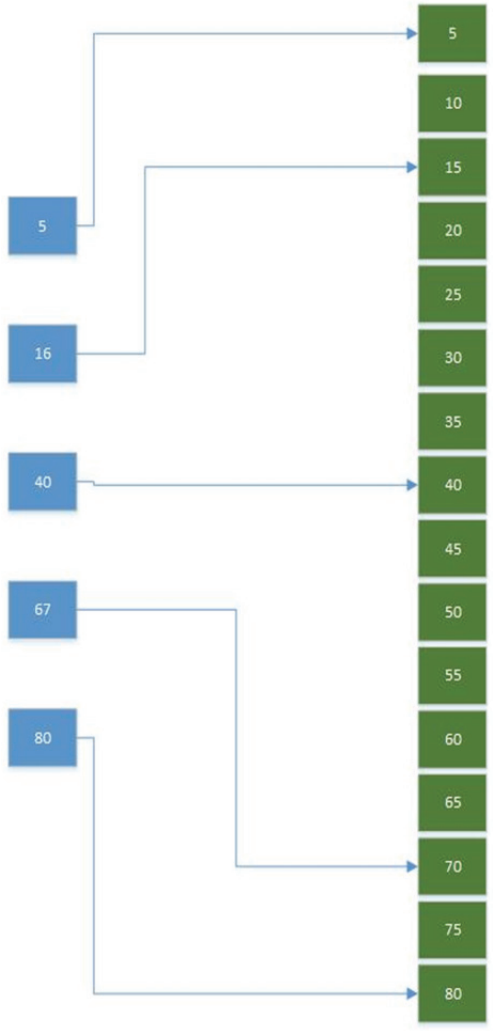

Fig. 7. Allocation of tasks in their respective level.

In this case, we noted that in some scenarios the sum of the metric values of the tasks is not equal to the value of the maximal metric. Next, we will explain this scenario by using the T1 in Fig. 4.

The values of the maximal and minimal metric are 80 and 5, respectively. As a result, we put the task whose metric is 5 in Level 1. Level 2 is then created by respecting the following condition: level n = n* Level 1. This means that Level 2 will only include tasks whose metrics values are around 10. By following this condition, we will have 16 levels (Fig. 7), which means that we will be using 8 VMs: Nbr.VMs=Nbr.level/2.

The new maximal metric that will be used is Mmax-i+ Mmin+i, where *M* is the metric's value of the tasks in the level and *i* goes from 0 to the number of levels minus 1 in this example, the maximal metric will be 85 (Fig. 8).

At this stage, the allocation of the tasks to the VMs will start. If there is no task in a Level X, then its metric will be 0. In this example, the levels that contain the tasks are the Levels 1, 3, 8, 14, and 16. This explains why there are 4 VMs out of 8 VMs that are in standby (T1 in Fig. 6). We then used these VMs in standby to treat tasks from the next segment of tasks (Fig. 9). This allows us to clear the task queue, which will reduce waiting time.

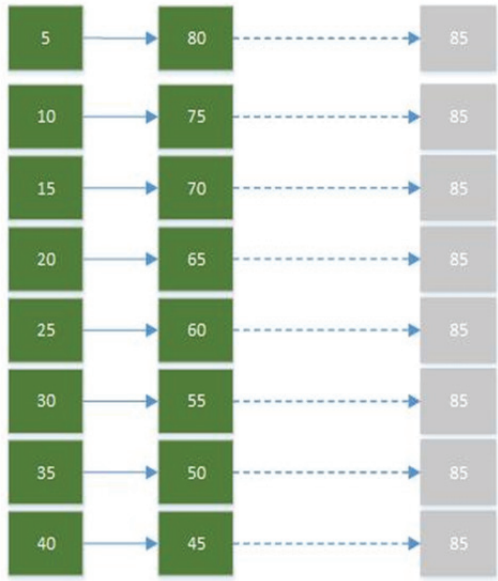

Fig. 8. Determination of the exact process time.

	T1	T2	T3	T4	T5	T6	T7	T8
Number of VMs	8	3	5	3	2	3	5	8
Output with queue management	85	50	55	30	84	150	55	85
	50	31	42	150				
	16	50	55	30	150	55	16	
	67	67						
	30	30						
	40	40						

Fig. 9. Time management.

4.2 Result

We tested the evolution process of our approach by following three steps:

Step 1: The pre-estimation of the classification metric using the Bat-algorithm. Fig. 10 shows the initial variation of the workload.

Step 2: Adding the elasticity aspect of our approach to the previous aspect. Fig. 11 shows the impact of the determination of the number of VMs based on the levels of the tasks on the workload.

Step 3: Adding the local queue migration aspect to the previous aspects. The result of the workload is shown on Fig. 12.

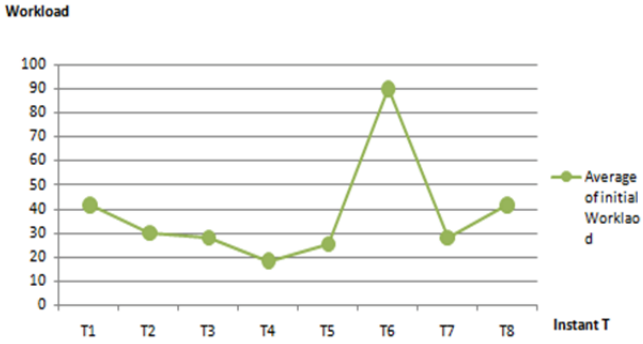

Fig. 10. Initial workload.

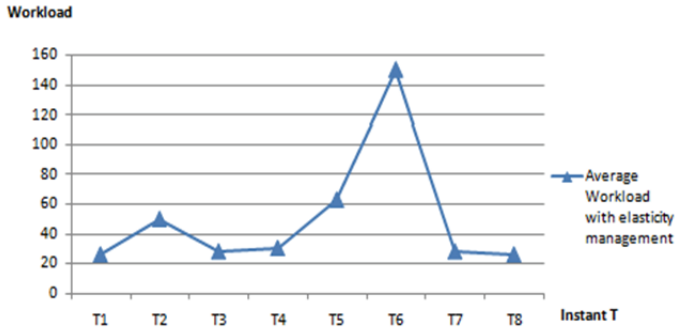

Fig. 11. Impact of the elasticity on the workload.

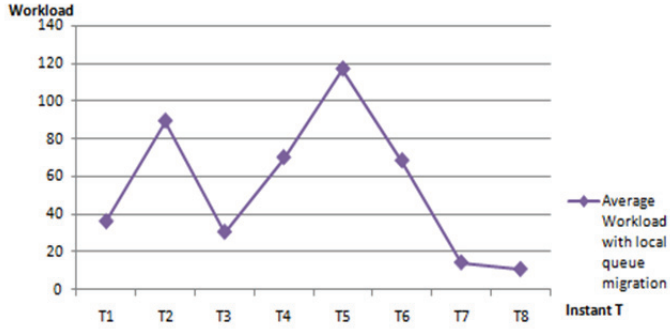

Fig. 12. Workload results of the local queue migration.

The results shown above are resumed by Fig. 13 to show the positive impact of the full utilization of our approach on the workload. They are explained as follows.

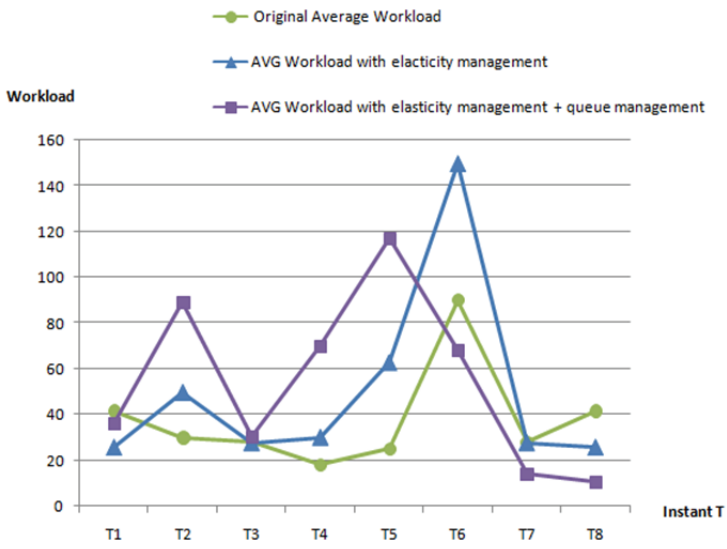

Fig. 13. Results of the positive impact of the full utilization of our approach.

The curves show that our new algorithm provides better results in terms of workload. We notice that it meets the performance criteria of a load balancing algorithm. Indeed, the various spikes that have been recorded at the beginning of the curve in purple reflect the maximization of the use of resources, which fluidizes subsequently the waiting line management. Based on the curves and the data from both Figs. 4 and 6, we notice that the algorithm uses at time T the exact optimal number required in terms of virtual machines.

5. Conclusions

Nowadays, the number of cloud users is growing exponentially. This fast growth leads to many QoS issues regarding load balancing. In an attempt to find a solution which allows better load balancing, we propose a novel load balancing model, which is based on two main parts. The conjunction of these parts allows us to obtain a complete load balancing model. Our approach will pre-classify the entry instructions through any meta-heuristic algorithm, which is dedicated to the pre-estimation of a metric that can be used as a classification parameter. Our approach is adaptable with any meta-heuristic and should give better load balancing results. Here, we chose Bat-algorithm. Our model will propose the most efficient number of VMs to execute all the tasks. Our approach allocates tasks to VMs with the guarantee of equal load distribution and increasing the total number of possible allocations in series or parallel mode, depending on the levels of the tasks. Finally, our approach will manage the local queue. It can also migrate tasks of one segment to a previous one in order to prevent overloaded and standby VMs. Our next work will be a comparative study between our approach and different existing load balancing algorithms.

References

[1] M. Katyal and A. Mishra, "A comparative study of load balancing algorithms in cloud computing environment," *International Journal of Distributed and Cloud Computing*, vol. 1, no. 2, pp. 5-14, 2013.

[2] Y. Fahim, E. B. Lahmar, E. H. Labriji, and A. Eddaoui, "The tasks allocation based on the pre-estimation of the processing time in the cloud environment," *Journal of Theoretical and Applied Information Technology*, vol. 75, no. 3, pp. 350-355, 2015.

[3] A. Goyal and Bharti, "A study of load balancing in cloud computing using soft computing techniques," *International Journal of Computer Applications*, vol. 92, no. 9, pp. 29-32, 2014.

[4] A. M. Alakeel, "A guide to dynamic load balancing in distributed computer systems," *International Journal of Computer Science and Information Security*, vol. 10, no. 6, pp. 153-160, 2010.

[5] R. P. Padhy and P. Rao, "Load balancing in cloud computing systems," Master's thesis, Department of Computer Science and Engineering, National Institute of Technology, Orissa, India, 2011.

[6] C. Ghribi, M. Hadji, and D. Zeghlache, "Energy efficient VM scheduling for cloud data centers: exact allocation and migration algorithms," in *Proceedings of 2013 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, Delft, the Netherlands, 2013, pp. 671-678.

[7] P. Werstein, H. Situ, and Z. Huang, "Load balancing in a cluster computer," in *Proceedings of the 7th International Conference on Parallel and Distributed Computing, Applications and Technologies*, Taipei, Taiwan, 2006, pp. 569-577.

[8] P. L. Doddini, "Load balancing algorithms in cloud computing," *International Journal of Advanced Computer and Mathematical Sciences*, vol. 4, no. 3, pp. 229-233, 2013.

[9] V. Sakthivelmurugan, A. Saraswathi, and R. Shahana, "Enhanced load balancing technique in public cloud," *IJREAT International Journal of Research in Engineering & Advanced Technology*, vol. 2, no. 2, pp. 1-4, 2014.

[10] S. Sharma, S. Singh, and M. Sharma, "Performance analysis of load balancing algorithms," *World Academy of Science, Engineering and Technology*, vol. 38, no. 3, pp. 269-272, 2008.

[11] Y. Fang, F. Wang, and J. Ge, "A task scheduling algorithm based on load balancing in cloud computing," in *Web Information Systems and Mining*. Heidelberg: Springer, 2010, pp. 271-277.

[12] R. Tong and X. Zhu, "A load balancing strategy based on the combination of static and dynamic," in *Proceedings of 2010 2nd International Workshop on Database Technology and Applications*, Wuhan, China, 2010, pp. 1-4.

[13] A. Khiyaita, H. El Bakkali, M. Zbakh, and D. El Kettani, "Load balancing cloud computing: state of art," in *Proceedings of 2012 National Days of Network Security and Systems (JNS2),* Marrakech, Morocco, 2012, pp. 106-109.

[14] M. Sharma, P. Sharma, and S. Sharma, "Efficient load balancing algorithm in VM cloud environment," *International Journal of Computer Science and Technology*, vol. 3, no. 1, pp. 439-441, 2012.

[15] S. Stattelmann and F. Martin, "On the use of context information for precise measurement-based execution time estimation," in *Proceedings of the 10th International Workshop on Worst-Case Execution Time Analysis*, Brussels, Belgium, 2010, pp. 64-76.

[16] J. Kaur, "Comparison of load balancing algorithms in a cloud," *International Journal of Engineering Research and Applications*, vol. 2, no. 3, pp. 1169-1173, 2012.

[17] A. Aditya, U. Chatterjee, and S. Gupta, "A comparative study of different static and dynamic load balancing algorithm in cloud computing with special emphasis on time factor," *International Journal of Current Engineering and Technology*, vol. 5, no. 3, pp. 1897-1907, 2015.

[18] M. Mesbahi and A. M. Rahmani, "Load balancing in cloud computing: a state of the art survey," *International Journal of Modern Education and Computer Science*, vol. 8, no. 3, pp. 64-78, 2016.

[19] C. L. Hung, H. H. Wang, and Y. C. Hu, "Efficient load balancing algorithm for cloud computing network," in *Proceedings of the International Conference on Information Science and Technology (IST 2012),* Chennai, India, 2012, pp. 28-30.

[20] T. Kokilavani and D. G. Amalarethinam, "Load balanced min-min algorithm for static meta-task scheduling in grid computing," *International Journal of Computer Applications*, vol. 20, no. 2, pp. 43-49, 2011.

[21] P. G. Gopinath and S. K. Vasudevan, "An in-depth analysis and study of Load balancing techniques in the cloud computing environment," *Procedia Computer Science*, vol. 50, pp. 427-432, 2015.

[22] K. Kaur, A. Narang, and K. Kaur, "Load balancing techniques of cloud computing," *International Journal of Mathematics and Computer Research*, vol. 3, no. 7, pp. 1616-1623, 2013.

[23] D. C. Devi and V. R. Uthariaraj, "Load balancing in cloud computing environment using improved weighted round robin algorithm for nonpreemptive dependent tasks," *The Scientific World Journal*, vol. 2016, article ID. 3896065, 2016.

[24] H. Casse and P. Sainrat, "OTAWA, a framework for experimenting WCET computations," in *Proceedings of the 3rd European Congress on Embedded Real-Time Software*, Toulouse, France, 2006, pp. 1-8.

[25] W. Leinberger, G. Karypis, V. Kumar, and R. Biswas, "Load balancing across near-homogeneous multi-resource servers," in *Proceedings of the 9th Heterogeneous Computing Workshop*, Cancun, Mexico, 2000, pp. 60-71.

[26] A. El Mahdaouy and M. Oumsis, "Evaluation et amélioration de performances des algorithmes d'équilibrage de charges dans un environnement Cloud Computing," in *Les 4èmes Journées Doctorales en Technologies de l'Information et de la Communication (JDTIC 2012),* Casablanca, Morocco, 2013.

[27] B. Ananthakrishnan, "An efficient approach for load balancing in cloud environment," *International Journal of Scientific & Engineering Research*, vol. 6, no. 4, pp. 36-40, 2015.

[28] J. Vashistha and A. K. Jayswal, "Comparative study of load balancing algorithms," *IOSR Journal of Engineering*, vol. 3, pp. 45-50, 2013.

[29] R. Lee and B. Jeng, "Load-balancing tactics in cloud," in *Proceedings of 2011 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC),* Beijing, China, 2011, pp. 447-454. IEEE.

[30] P. Gupta, M. K. Goyal, and P. Kumar, "Trust and reliability based load balancing algorithm for cloud IaaS," in *Proceedings of 2013 IEEE 3rd International Advance Computing Conference*, Ghaziabad, India, 2013, pp. 65-69.

[31] O. Sarood, A. Gupta, and L. V. Kale, "Cloud friendly load balancing for hpc applications: preliminary work," in *Proceedings of 2012 41st International Conference on Parallel Processing Workshops,* Pittsburgh, PA, 2012, pp. 200-205.

[32] S. C. Wang, K. Q. Yan, W. P. Liao, and S. S. Wang, "Towards a load balancing in a three-level cloud computing network," in *Proceedings of 2010 3rd IEEE International Conference on Computer Science and Information Technology,* Chengdu, China, 2010, pp. 108-113.

[33] B. Mondal, K. Dasgupta, and P. Dutta, "Load balancing in cloud computing using stochastic hill climbing-a soft computing approach," *Procedia Technology*, vol. 4, pp. 783-789, 2012.

[34] M. Simjanoska, S. Ristov, G. Velkoski, and M. Gusev, "L3B: low level load balancer in the cloud," in *Proceedings of 2013 IEEE EUROCON,* Zagreb, Croatia, 2013, pp. 250-257.

[35] G. Xu, J. Pang, and X. Fu, "A load balancing model based on cloud partitioning for the public cloud," *Tsinghua Science and Technology*, vol. 18, no. 1, pp. 34-39, 2013.

[36] R. Wang, W. Le, and X. Zhang, "Design and implementation of an efficient load-balancing method for virtual machine cluster based on cloud service," in *Proceedings of the 4th IET International Conference on Wireless, Mobile & Multimedia Networks*, Beijing, China, 2011, pp. 321-324.

[37] W. Tian, Y. Zhao, Y. Zhong, M. Xu, and C. Jing, "A dynamic and integrated load-balancing scheduling algorithm for cloud datacenters," in *Proceedings of 2011 IEEE International Conference on Cloud Computing and Intelligence Systems,* Beijing, China, 2011, pp. 311-315.

[38] F. Ma, F. Liu, and Z. Liu, "Distributed load balancing allocation of virtual machine in cloud data center," in *Proceedings of 2012 IEEE 3rd International Conference on Software Engineering and Service Science,* Beijing, China, 2012, pp. 20-23.

[39] J. L. Chen, Y. T. Larosa, and P. J. Yang, "Optimal QoS load balancing mechanism for virtual machines scheduling in eucalyptus cloud computing platform," in *Proceedings of 2012 2nd Baltic Congress on Future Internet Communications,* Vilnius, Lithuania, 2012, pp. 214-221.

[40] K. Nishant, P. Sharma, V. Krishna, C. Gupta, K. P. Singh, and R. Rastogi, "Load balancing of nodes in cloud using ant colony optimization," in *Proceedings of 2012 UKSim 14th International Conference on Computer Modelling and Simulation (UKSim),* Cambridge, UK, 2012, pp. 3-8.

[41] S. M. Ghafari, M. Fazeli, A. Patooghy, and L. Rikhtechi, "Bee-MMT: a load balancing method for power consumption management in cloud computing," in *Proceedings of 2013 6th International Conference on Contemporary Computing (IC3),* Noida, India, 2013, pp. 76-80.

[42] J. Yao and J. H. He, "Load balancing strategy of cloud computing based on artificial bee algorithm," in *Proceedings of 2012 8th International Conference on Computing Technology and Information Management (ICCM),* Seoul, Korea, 2012, pp. 185-189.

[43] S. Sun, W. Yao, and X. Li, "DARS: a dynamic adaptive replica strategy under high load Cloud-P2P," *Future Generation Computer Systems*, vol. 78, pp. 31-40, 2018.

[44] A. S. Milani and N. J. Navimipour, "Load balancing mechanisms and techniques in the cloud environments: systematic literature review and future trends," *Journal of Network and Computer Applications*, vol. 71, pp. 86-98, 2016.

[45] E. Ikonomovska, I. Chorbev, D. Gjorgjevik, and D. Mihajlov, "The adaptive tabu search and its application to the quadratic assignment problem," in *Proceedings of Information Society* (IS), Ljubljana, Slovenia, 2006, pp. 26-29.

[46] G. A. E. A. Said, A. M. Mahmoud, and E. M. El-Horbaty, "A comparative study of meta-heuristic algorithms for solving quadratic assignment problem," *International Journal of Advanced Computer Science and Applications*, vol. 5, no. 1, pp. 1-6, 2014.

[47] F. Neumann and C. Witt, *Bioinspired Computation in Combinatorial Optimization: Algorithms and Their Computational Complexity*. Heidelberg: Springer, 2010.

[48] P. J. Van Laarhoven and E. H. Aarts, "Simulated annealing," in *Simulated Annealing: Theory and Applications*. Dordrecht: Springer, 1987, pp. 7-15.

[49] M. Randles, D. Lamb, and A. Taleb-Bendiab, "A comparative study into distributed load balancing algorithms for cloud computing," in *Proceedings of 2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops,* Perth, Australia, 2010, pp. 551-556.

[50] X. S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature Inspired Cooperative Strategies for Optimization*. Heidelberg: Springer, 2010 pp. 65-74.

[51] S. Sharma, A. K. Luhach, and K. Jyoti, "A novel approach of load balancing in cloud computing using computational intelligence," *International Journal of Engineering and Technology*, vol. 8, no. 1, pp. 124-128, 2016.

Youssef Fahim https://orcid.org/0000-0003-0644-0376

He is currently a researcher on Department of Math & Computer Science at the Faculty of Science Ben M'sik, Hassan II University, Casablanca, Morocco. His current research projects focus on load balancing algorithm and quality of service in the cloud computing.

Hamza Rahhali https://orcid.org/0000-0002-5758-6941

He received a M.Sc. degree in networks and telecommunications from Aix Marseille University, Marseille, France, in 2011. His research interests include load balancing algorithms and scheduling algorithms in cloud computing.

Mohamed Hanine https://orcid.org/0000-0001-5417-3503

He received an engineering degree in telecommunications and networks at University Abdel Malek Essaidi, Tetouan, Morocco in 2015. He is currently pursuing the Ph.D. degree in Information modeling and technologies at the University of Hassan II, Casablanca, Morocco. His research interests include information modeling, cloud computing, load balancing, and beyond.

El-Habib Benlahmar https://orcid.org/0000-0001-7098-4621

Is a professor of higher education at the Faculty of Sciences Ben M'Sik, Casablanca, Morocco. His research interests include metasearch, information retrieval, semantic web, automatic processing of natural language, cloud computing and beyond.

El-Houssine Labriji https://orcid.org/0000-0002-6987-8554

He is currently a Professor in the Department of Computer Science and Mathematics, University Hassan II of Casablanca, Morocco. His research interests are in the areas of computer sciences, data mining, information technology, e-learning and cloud computing.

Mostafa Hanoune https://orcid.org/0000-0001-5047-187X

He is currently a Professor in the Department of Computer Science and Mathematics, University Hassan II of Casablanca, Morocco. His research interests are in the areas of computer sciences, data mining, information technology and cloud computing.

Ahmed Eddaoui https://orcid.org/0000-0003-1321-5396

He is currently an Associate Professor and Chair of the Department of Math & Computer Science at Shaqra University, Riyadh, Saudi Arabia. His current research projects focus on cloud computer and security.