A Performed Load Balancing Algorithm for Public Cloud Computing Using Ant Colony Optimization

Awatif Ragmani(*), Amina El Omri, Noreddine Abghour, Khalid Moussaid, Mohammed Rida
Modeling and Optimization for Mobile Services Research Team
Faculty of Sciences Ain chock
University Hassan II Casablanca, Morocco
(*)ragmaniawatif@gmail.com

Abstract— **In the space of a few years, Cloud computing has experienced remarkable growth. Indeed, its economic model based on demand of hardware and software according to technical criteria (CPU utilization, memory, bandwidth…) or package has strongly contributed to the liberalization of computing resources in the world. However, the development of Cloud computing requires the optimization of performance of the different services offered by Cloud providers in order to ensure a high level of security, availability and responsiveness. One major aspect for dealing with performance issues in Cloud computing is the load balancing. In fact, an efficient load balancing contributes to the decrease of costs and maximizes availability of resources. Through this paper, we study various research which has treated the load balancing in the Cloud computing. In a second step, we propose an improved load balancing architecture and algorithm for Cloud computing which aims to allow a better response time.**

Keywords— Cloud computing; Load balancing; Scheduling; Ant colony optimization

I. INTRODUCTION

Cloud computing is a highly promising technology concept. With its fundamental concept of resource use on demand, Cloud computing has greatly contributed to the liberalization of IT resources. Thus, wide kinds of organizations as well as individuals become able to access IT resources that were previously beyond their means. The concept of pay-as-you-go applied by Cloud providers fits in both economic and technological trends of different organizations. Particularly, the use of virtualization facilitates the sharing of resources and the administration of the whole Cloud system. In despite of the reluctance of some actors to adopt the Cloud, all economic trends predict strong growth of Cloud applications within the coming years [1]. However, the development of Cloud cannot be done without the continuous improvement of its performance including response time of Cloud services and the energy consumption of data centers. Indeed, it was noted that each increase in the average response time induces a decrease of users' number and revenue [2].

The excitement experienced by the Cloud was the initiating factor of several studies conducted by researchers as well as the giant Cloud providers like Amazon, Google, IBM and Microsoft. These studies have as key objectives to develop the area of applications covered by the Cloud, optimizing performance and improving security. During this paper, we will focus particularly on the optimization of Cloud performance. Specifically, we analyze the impact of load balancing algorithms on Cloud performance. In fact, an optimal response time will have the advantage of offering users the ability to work on almost identical way of remote resources to local resources. The study of performance in the Cloud can be done through different aspects as identified in the research works [1]–[4].

In summary, we note that Cloud is a distributed system, which shares thousands of computing resources. The Cloud uses Internet links to connect users to providers. The distribution of workloads between different nodes within the Cloud network is a crucial step in the process of optimizing the overall workloads. Hence the importance of load balancing to allocate efficiently the various available resources including Cloud network links, central processing units, disk drives, or other resources. Particularly, using a unique hardware load balancer for central administration of hundreds of processors proves costly in terms of time and money. In addition, oversized hardware remains underutilized if traffic declines [2].

The objectives of our study include the realization of comparative analysis of the different load balancing algorithms within the Cloud in order to propose a more efficient solution. The remainder of the paper is organized as follows. The main works on the load distribution problem in the Cloud are discussed in Sections 3 and 4, Section 5 introduces a load balancing system for Cloud computing and we will conclude with Section 6.

II. CLOUD COMPUTING

The Cloud computing is defined as an advancement of the previous distributed model like Grid Computing. The main goal behind its expansion is to transform the computing resource to service. Cloud computing is based on the concept of offering to users hardware, software applications, data management, and storage as a service. In other words, the users do not need to know the location of servers or any other technical aspects of the computing infrastructure. Cloud computing aims at scalability by using load balancing. In Cloud computing, services offered to users is not limited to processing power and include storage, website hosting, database support and more others [5].

A. *Layers of Cloud Computing*

Cloud computing can be defined as set of services organized on three layers (see Fig. 1). The Software-as-a-Service (SAAS) layer which offer to users the possibility to run applications remotely from the Cloud.

The second layer is the Infrastructure-as-a-Service (IAAS) which refers to computing resources as a service. This includes virtualized computers with defined processing power and bandwidth for storage and Internet access.

Finally, the Platform-as-a-Service (PAAS) layer which is similar to IAAS, but includes in addition, operating systems and required services for a particular application [6].

B. *Types of Cloud Computing*

There are four types of Cloud computing. The most popular is the public Cloud where computing resources are shared over the Internet via web applications and services. Public Clouds services and applications are provided by third parties, and Cloud computing resources are shared between different customers.

The second kind is the Private Cloud which refers to Cloud computing on private networks within a specific organization. Particularly, the Private Clouds are implemented on behalf of a particular user in order to maintain the full control over data, security, and quality of service. Private Cloud can be developed and administrated by the internal IT department of the organization or by an external Cloud provider. The third configuration is a hybrid Cloud environment which includes both public and private Cloud [6].

Lastly, the Community Cloud is a platform used by different organizations which are motivated by shared requirements. This kind of Cloud allows multiple entities to share specific business applications or rent in common hardware resources.

C. *Cloud Computing Feature*

The main strength of Cloud computing lies in the new characteristics it introduces compared to previous computing model like resources sharing and IT cost optimization. Briefly, the first feature of Cloud computing is the scalability by offering resources and services for users on demand. The resources are scalable over several data centers.

The second feature of Cloud can be summarized as user-centric interface which means that Cloud interfaces and user's location are independent and can be used through web services. Otherwise, the Cloud computing systems can be presented as autonomous systems and are administrated transparently to users. The quality of service like CPU performance, bandwidth and memory capacity are guaranteed by a service level agreement (SLA).

Finally, the pricing model of Cloud services offers the users the opportunity to optimize their IT investment. Indeed, the implantation of Cloud computing does not require any particular capital expenditure because users pay only for services and capacity they use [6].

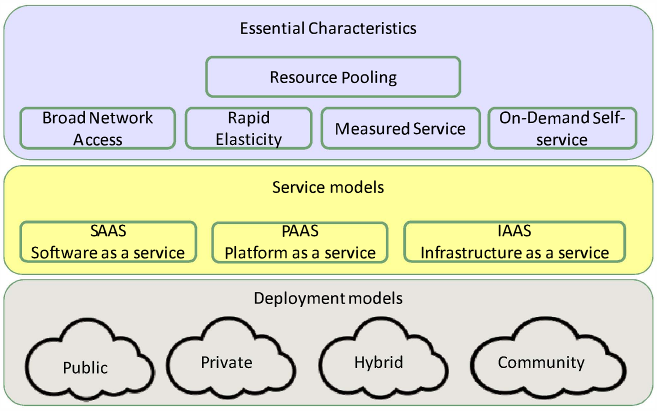

Fig. 1. Visual model of NIST working definition of Cloud Computing [7]

D. *Virtualization*

The virtualization is the key technology which makes the Cloud computing possible. This concept can be defined as a global framework and process of sharing the resources of computer hardware into multiple execution environments. Thus, the virtualization includes multiple concepts such as hardware and software partitioning, time sharing, machine simulation, emulation and quality of service. Particularly, the virtualization has a crucial role on system optimization by reducing and simplifying the administration of server infrastructure or being a mean of modifying the global management approach of data center [8].

There are two types of virtualization in case of Cloud. The first category is the full virtualization where an entire installation of one machine is performed on another machine. As consequence, the virtual machine obtained will have all the software present in the actual server. Full virtualization has been doing well for different reasons like sharing a computer system among multiple users and separating users from each other. The second type of virtualisation is the Para virtualization where the hardware allows multiple operating systems to operate on unique machine by efficient use of system resources such as memory and processor [9].

III. Related Work

Due to the importance and complexity of load balancing within the Cloud system, several researchers have attempt to find out special techniques for optimizing load balancing and distributing workload. Those tools and methods are mostly inspired from the previous techniques applied within distributed system.

In [10], the authors stressed, the importance of choosing the appropriate node in the Cloud for the execution of a task. The authors have also introduced the use of agents in collecting available information such CPU capacity and remaining memory. Finally, they proposed a scheduling algorithm gathering the qualities of two algorithms the opportunistic load balancing (OLB) and the load balance Min-Min. The first algorithm is used to allocate the jobs and divide tasks into sub tasks as three-tier architecture. The second algorithm in turn, serves to improve the execution time of each node.

Another research study [11] have coped with the analysis of various load balancing algorithms applied in the Cloud in order to define a new more powerful algorithm. The objective of this study is to propose a hybrid algorithm combining the properties of biased random sampling algorithms and honeybee foraging algorithm.

In [12], the authors summarized the role of load balancing in the Cloud. They note that load balancing algorithms must satisfy two classifications based on the way that load is distributed and how processes are allocated to nodes and depending on the status of the nodes. In the first case it is presented as centralized vision, distributed or hybrid and in the second approach it is defined as static, dynamic or adaptive approach.

Through the research study [13], the authors presented an algorithm entitled Join-Idle-Queue (JIQ) for load balancing in distributed systems which introduce different advantages. Particularly, the authors announce that JIQ algorithm incurs no overhead communication between dispatchers and processors at job arrivals. Finally, the authors indicate that an extension of the basic algorithm JIQ should cope with very high loads using only local information of server load.

In [14], the authors presented a load balancing model for public Cloud. Thus, the model divides global Cloud into several Cloud partitions in order to simplify load balancing of the large and complex Cloud environment. A main controller chooses the suitable partitions for arriving jobs while the balancer for each Cloud partition decides the best load balancing strategy.

The authors of the research study [15] introduced a multi-objective ant colony system algorithm in order to ensure an efficient placement of virtual machine (VM). Through this paper, the authors aim to implement solutions which reduce both the total resource wastage and power consumption. Principally, the comparison of performance of the proposed algorithm to the existing multi-objective genetic algorithm and two single-objective algorithms, highlight the fact that the proposed algorithm is more efficient than the previous algorithms.

Another research study [16], tried to find out an optimized load balancing algorithm for the Cloud environment by using ant colony optimization. A heuristic algorithm using ant colony optimization concept has been proposed to implement a service load distribution for the Cloud. The authors highlight the fact that the pheromone update has been proved as an optimized means to balance the load. However, the proposed technique does not address the fault tolerance issues and more works are necessary to fulfill this aspect.

Otherwise, the authors of [17] attend to study the load balancing in Cloud computing by introducing a semi centralized and multi cluster architecture. The main idea of this paper is to find out a system which guarantees a highest level of performance and to keep an efficient load balancing, security and availability.

IV. Load Balancing Algorithm

A parallel program is a set of processes which aims to perform one or more tasks of the program. Particularly, the task is the smallest division of a parallel program. Thus, the development of a parallel program requires at first, decomposing the overall program into a collection of tasks and assigning each task to a process. This step is called the partitioning. Its optimization is based on the balancing of workloads between different processes and reducing inter-processes communication. As the number of processes generated by a program is often different from the number of processor in charge of processing the request, there is a risk of overloading or under loading of a processor by processes. Consequently, the optimization of Cloud resources cannot be done without an efficient load balancing strategy [18]. In summary, the major goal of load balancing is to develop performed algorithms able to achieve both efficiently the partitioning and mapping steps. In other words, the load balancing algorithms should:

- Enhance the performance significantly;
- Ensure a partial or complete backup plan in case of system failed ;
- Keep up the system stability;
- Put up future update of the system.

According to their working strategy, load balancing algorithms are classified on two categories, static or dynamic load balancing (see Table I.) [18].

TABLE I. LOAD BALANCING ALGORITHMS' CATEGORIES

	Static	**Dynamic**
Definition	It distributes processes to processors at compile time, in most cases relying on a priori knowledge about the processes and the system on which they run	Execution of a dynamic load balancing algorithm requires some means for maintaining a coherent view of the system state at run-time and some negotiation policy for process migrations across processors
Advantages	• They will not cause any run-time overhead • They are attractive for parallel programs for which the execution times of processes and their communication requirements can be predicted • In some situations, static load balancing is the only compatible choice	• They incur non-negligible run-time overhead • They are aiming for a limited balanced state between the two extremes representing a certain tradeoff among balancing quality and run-time overhead
Disadvantage	• It is not satisfactory for parallel programs that are of the dynamic or unpredictable kind	• The design and analysis of dynamic's load balancing algorithms is a complex process

There are two main performance indicators of load balancing: stability and efficiency. The stability measures essentially, the capability of an algorithm to force any initial workload distribution into equilibrium state. The efficiency indicates the time necessary to either reduce the variance or arrive at the equilibrium state [19]. The response time, which indicated the time necessary from registering the input to providing a response to it, and throughput, which represents the number of tasks achieved in a given time, represents the most significant key performance indicators in a distributed system [5].

Load balancing algorithms in Cloud can be classified as immediate mode scheduling and batch mode scheduling. The immediate mode scheduling organizes tasks based on its arrival by applying a minimum execution time and minimum completion time algorithms. The Batch mode scheduling organizes tasks in a Meta task set based upon their arrival and they are mapped at prescheduled times to their related machines [19]. Each Load balancing algorithm can be presented by four components as summarized on Fig. 2. Several algorithms of load balancing are cited in the literature [19], [20]. Principally, we introduce:

- Round Robin: this algorithm is an elementary scheduling method which applies the concept of time slices. Thus, the time is separated into various segments then each node is allowed to use a particular time interval. In this time interval the node will execute its tasks. Briefly, this algorithm operates on random selection of the virtual machines. The datacenter controller allocates the tasks to the available virtual machines basing on a rotating process.

- Weighted Round Robin: this algorithm offers an improved resources allocation by defining a weight to each virtual machine. This weight is fixed on the basis of load capacity of each machine. In this case, the data center controller will assign a number of requests to adequate machine based on its weight. The main aspect of this distribution lies in the fact of paying no attention to the advanced load balancing requirement like processing times per individual requests.

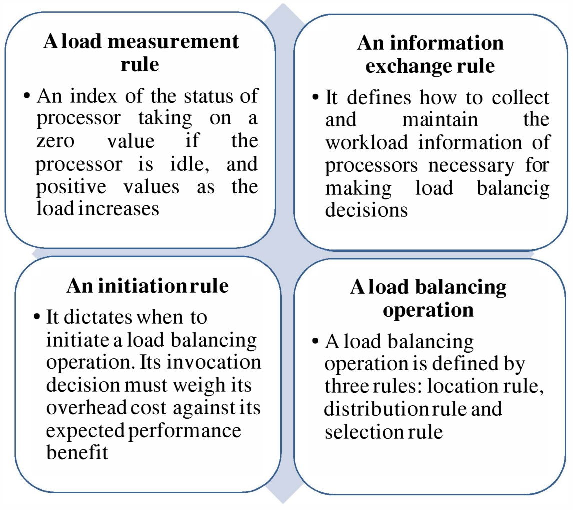
A load measurement rule
• An index of the status of processor taking on a zero value if the processor is idle, and positive values as the load increases
An information exchange rule
• It defines how to collect and maintain the workload information of processors necessary for making load balancig decisions
An initiation rule
• It dictates when to initiate a load balancing operation. Its invocation decision must weigh its overhead cost against its expected performance benefit
A load balancing operation
• A load balancing operation is defined by three rules: location rule, distribution rule and selection rule

Fig. 2. Load Balancing Algorithms components

- Equally Spread Current Execution: this algorithm processes deal with priorities. In other words, the algorithm dispatches the load randomly by analyzing the volume and relocates the load to those virtual machines which is lightly loaded or should process that task easy, need less time to fulfill the task and maximize throughput. The tasks are transferred to the virtual machine manager which maintains a set of the tasks and their resources needed.

- Throttled load balancer: this algorithm is fully based on virtual machine. In brief, through this algorithm the user first communicates with the load balancer to find out an adequate virtual machine for the required operation. If an adequate machine is identified then the load balancer accepts the request of the client and assigns that virtual machine to the client. Otherwise, the request is queued.

- Min-Min: this algorithm copes with a task set which are primarily not allocates to any of the nodes. Firstly, the minimum completion time is estimated for all the available nodes. After the calculation, the completion time minimum is selected and assigned to the adequate node. This algorithm is more adapted for the case where the small tasks are larger in number than the great tasks. Min-Min is a simple and fast algorithm able to guarantee enhanced performance.

- Max Min: this algorithm is similar to the min-min algorithm. Firstly, the available tasks are sent to the system and minimum completion times for all of them are calculated. Secondly, the tasks which have the maximum completion time is chosen and assigned to the related machine. In the case of short tasks are higher in number than the long one, this algorithm performed better than the Min-Min algorithm.

V. Proposed Load Balancing System

A. *Simulation Framework*

Before the definition of an efficient and optimized load balancing algorithm, it is essential to be able to understand the influential parameters of Cloud system. However, the analysis of Cloud resources operating under real and varied conditions seems difficult. To overcome this difficulty we opt for the use of the simulation platform CloudAnalyst [21]. This toolkit is developed on the basis of CloudSim [22] simulator by improving and extending its characteristics. The CloudAnalyst is an efficient toolkit which enables modelers to achieve simulation and modeling of Cloud components such as data centers, virtual machines and resource provisioning policies. This simulator allows firstly, the separation of the programming and experimentation functions and secondly to carry out repeatable experiments by varying simulation parameters. This toolkit provides the ability to model each Cloud by defining one or more data centers which are composed of several hosts. Each physical machine may host multiple virtual machines.

As depicted in Fig. 3, the Cloud components are identified by different parameters (user region, load balance policy, memory, storage…) that could be modified in order to complete the various experiments in economic and easy way. Particularly, the service broker policies parameter allows the control of the data center which performs the user request at a given time. The load balance policy is the parameter that describes request allocation within a data center. The CloudAnalyst apply a round robin algorithm as default load balance policy and proposes also a throttled load balance policy which constraints the requests being processed to a throttling threshold [21].

B. *Analysis of Influential Cloud Computing Factors*

To conduct our study we apply the Taguchi experience plans [23]. This methodology inspired from the industrial sector offers the advantage of organizing and making easier the study of a complex system such Cloud computing. The Taguchi's concept utilizes an optimal number of planned experiments in order to examine the settings of influential process control parameters. For each set of factors, the appropriate experiments are achieved and the performance measurements are recorded. Afterward, this approach use figures and tables to analyze the obtained measurements and to predict the best combination to optimize the effect of factors. Lastly, a few additional experiments are achieved in order to validate the combination predicted.

To fulfill our analysis, we choose the Taguchi L16 array because it has the same number of parameters as the predefined simulation model (see Fig. 3). In other words, the L16 array has 13 factors and each factor has two levels. Each array cell define a qualitative or quantitative factor levels (1 or 2), which determine the type of parameter values for each experiment. Once the combination of control parameter is defined by the Taguchi array, we have proceeded to simulation on the CloudAnalyst platform. The L16 array applied and the achieved measurements are summarized in Table II.

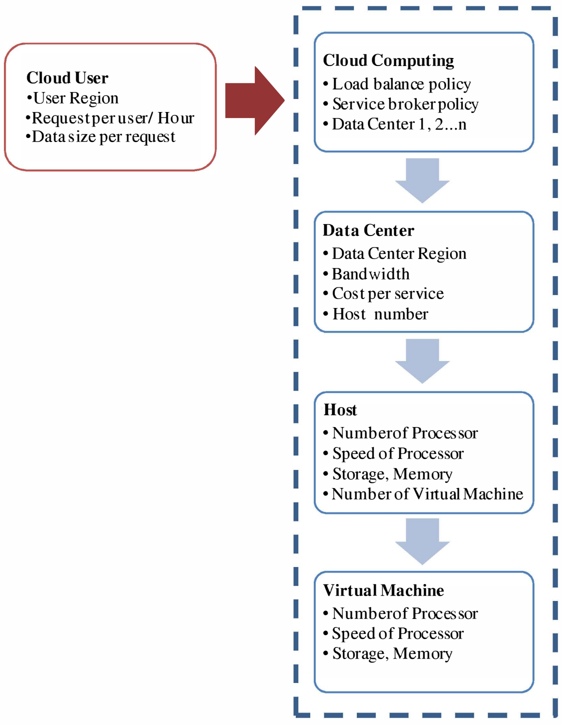

Fig. 3. Cloud Computing simulation model

The Taguchi experiment plans make use of the signal-to-noise ratio (SNR) to analyze the results obtained. In fact, the optimization of outputs can be obtained by minimizing the function defined on (1).

$$SN_i = -10\log\left(\sum_{u=1}^{N_i}\frac{y_u^2}{N_i}\right) \tag{1}$$

Where i: experiment number; u: trial number; Ni: Number of trials for experiment; y_u: performance characteristic measurements per trial.

According to the simulation results summarized on the Table II and the ranking of influential factors presented in Table III, we highlight two facts. Firstly, the geographical location of the Data Center which performs the request and that of the user who initiated the request have a strong impact on the response time.

As mentioned on Table II, it is established that the response time may be multiplied by 10 in the case of two remote locations. Secondly, the factors effect ranks (see Table III) highlight the fact that the most influential factor in optimizing the response time is the request per user followed by the data size of the user's request. In other words, load balancing system should make it possible to affect the user's request to the closest data center and to optimize the data size and frequency of request performed.

TABLE II. TAGUCHI ARRAY APPLIED AND RESULTS OBTAINED

User Region	DC Region	Request User/Hour	Data size	Broker Policy	Balancing Policy	Number Host	Storage	Memory	Number of Processor	Speed of Processor	Number VM	Bandwidth	Response Time (ms)	Processing Time (ms)
1	1	1	1	1	1	1	1	1	1	1	1	1	**50.16**	**0.49**
1	1	1	1	1	1	1	2	2	2	2	2	2	**50.08**	**0.41**
1	1	1	2	2	2	2	1	1	1	1	2	2	**50.70**	**0.87**
1	1	1	2	2	2	2	2	2	2	2	1	1	**50.26**	**0.43**
1	2	2	1	1	2	2	1	1	2	2	1	1	**499.85**	**0.29**
1	2	2	1	1	2	2	2	2	1	1	2	2	**500.25**	**0.66**
1	2	2	2	2	1	1	1	1	2	2	2	2	**500.37**	**0.38**
1	2	2	2	2	1	1	2	2	1	1	1	1	**500.65**	**0.66**
2	1	2	1	2	1	2	1	2	1	2	1	2	**500.25**	**0.30**
2	1	2	1	2	1	2	2	1	2	1	2	1	**500.69**	**0.73**
2	1	2	2	1	2	1	1	2	1	2	2	1	**500.19**	**0.63**
2	1	2	2	1	2	1	2	1	2	1	1	2	**499.90**	**0.35**
2	2	1	1	2	2	1	1	2	2	1	1	2	**50.23**	**0.45**
2	2	1	1	2	2	1	2	1	1	2	2	1	**50.17**	**0.39**
2	2	1	2	1	1	2	1	2	2	1	2	1	**50.12**	**0.46**
2	2	1	2	1	1	2	2	1	1	2	1	2	**50.51**	**0.85**

TABLE III. FACTORS EFFECT RANKS FOR RESPONSE TIME INDICATOR

Level	UserRegion	DCRegion	RequestUser	Datasize	BrkPolicy	BlcPolicy	NbrHost
1	-44.01	-44.01	-34.03	-44.00	-44.00	-44.00	-44.00
2	-44.00	-44.00	-53.98	-44.02	-44.01	-44.01	-44.02
Delta	0.00	0.00	19.96	0.02	0.01	0.01	0.02
Rank	10	11	1	2	7	8	3

Level	NbrProcessor	SpeedProcessor	NbreVM	Bandwidth	Memory	Storage
1	-44.02	-44.01	-44.01	-44.00	-44.01	-44.01
2	-44.00	-44.00	-44.01	-44.01	-44.00	-44.00
Delta	0.02	0.01	0.00	0.02	0.02	0.00
Rank	4	9	13	6	5	12

C. *Proposed Load Balancing Architecture*

According to the above analysis, an efficient load balancing policy should allocate users' request to closest data center and allows optimization of request size. As depicted in Fig. 4, the proposed load balancing architecture is organized on three layers. The first layer includes the main controller and the second controller. The main controller is the first element to receive the users' tasks. The main controller classifies users' tasks by priority, region and technical characteristics and then dispatches users' tasks on regional load balancers as mentioned hereafter (see Algorithm I and Fig. 5). In addition, the main controller communicates regularly with the second controller in order to update information about the state of the system. The second controller has to communicate frequently with the regional load balancers in order to refresh the information about system state. Through different agents, the second balancer updates the load level of the different nodes according to a frequency T. Then it informs the main controller about the modified state of a node. This configuration aims to lighten the main controller load.

The second layer embraces regional load balancers. Those components receive various tasks sent by the main controller and schedule tasks received through the different nodes of the Cloud following an ant colony optimization algorithm. The third layer is composed by two levels of nodes. The first level receives the tasks sent by regional load balancer. Then, they divided the tasks received on elementary subtasks in order to minimize the size of users' requests. This approach is driven by the conclusion of the Taguchi analysis which point out that the "Request size" factor has a high impact on the performance response time.

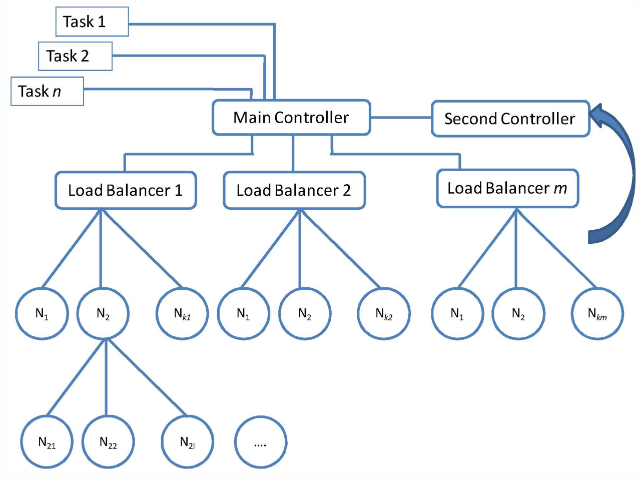

Fig. 4. The proposed load balancing architecture

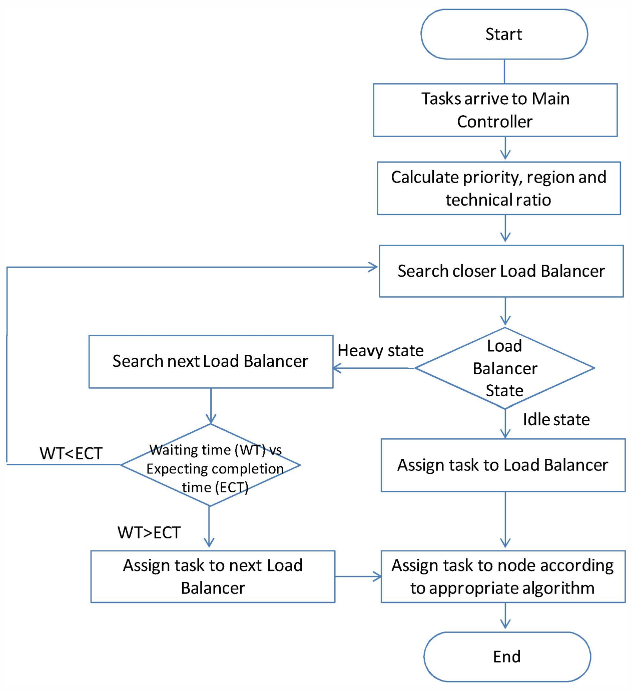
Start
Tasks arrive to Main Controller
Calculate priority, region and technical ratio
Search closer Load Balancer
Load Balancer State
Heavy state
Search next Load Balancer
Idle state
Assign task to Load Balancer
WT<ECT
Waiting time (WT) vs Expecting completion time (ECT)
WT>ECT
Assign task to next Load Balancer
Assign task to node according to appropriate algorithm
End

Fig. 5. Task scheduling schema

Algorithm I: Task scheduling of the Main Controller

```
begin
  while Task do
    CalculateTechnicalratio(Task);
    while (value of Timer ) < T do
      SearchcloserLoadBalancer (Task);
      if RegionalBalancerState == Idle then
        Send Task to RegionalBalancer;
      else
        Search for next RegionalBalancer;
        CalculateWaiting Time (Task);
        CalculateExpectingCompletionTime (Task) ;
        if Waiting Time > Expecting Completion Time then
          Send Task to NextRegionalBalancer;
        else
          Put Task on the RegionalBalancer queue
        end if
      end if
    end while
  end while
end
```

D. *Ant Colony Optimization*

Ant colonies represent distributed systems which has a well structured social organization. Indeed, this organization allows ant colonies to achieve complex tasks which exceed usually the individual capabilities of a single ant. As consequence, the different research tried to study the models resulting from the examination of real ants' behavior, and use these models as a basis for implementing novel algorithms dedicated to the optimization and distributed control problems [24].

Particularly, ant colony optimization (ACO) is a Meta heuristic which is developed on the basis of the pheromone trail laying and the behavior of a number of ant colonies. Artificial ants in ACO represent the procedure of stochastic solution construction which put up possible solutions for the optimization problem. The artificial pheromone information is updated based on the results of ants' search experience and existing heuristic information. Numerous researches have focused on the implementation of high-performing algorithmic possibilities and successful applications of ACO algorithms to a large variety of computationally hard problems, and the theoretical understanding of properties of ACO algorithms [25].

The standard ant colony algorithm is composed by five main steps [26]. The first step is the initialization of the variables. The second and third steps take into account the different iterations. The fourth step allows the building of solution and the updating of local pheromone. Finally, the fifth step includes the updating of the global pheromone until end condition.

An ACO algorithm can be summarized as the interaction of three procedures. The first procedure administrates a colony of ants that concurrently and asynchronously progress through neighbor nodes. As consequence, ants incrementally put up solutions to the optimization problem. As the solution is being built, the ant estimates the solution that will be used by procedure to decide how much pheromone to deposit. Secondly, ACO apply the second procedure that is the process by which the pheromone trails are updated. Those values can either amplify, as ants deposit pheromone on the elements or connections used, or decline, due to pheromone evaporation.

From a practical point of view, the deposit of new pheromone increases the probability that those elements that were either used by many ants or that were used by at least one ant and which produced a very good solution will be used again by future ants. In a different way, pheromone evaporation implements a useful form of forgetting: it avoids a too rapid convergence of the algorithm toward a suboptimal region, therefore favoring the exploration of new areas of these arch spaces. To finish, the third procedure is applied to put into practice centralized actions which cannot be performed by single ants.

Through this paper, we decide to apply the ACO due to the fact that this algorithm is highly compatible with the behavior of distributed network. Moreover, the ants' search experience can be applied to influence reminiscent of strengthening learning, the solution construction in future iterations of the algorithm. Furthermore, the use of a colony of ants should increase the algorithm robustness and in many ACO applications the collective interaction of a population of agents is needed to proficiently find out a solution to problem.

E. Proposed ACO Algorithm

Let a Cloud be presented by a set of nodes which represent the available virtual machine VMj. Concerning the load balancing in Cloud environment, computing resources are dynamically increased or decreased following the users demand. Each virtual machine VMj estimates a reward per task. This value is equivalent to the quantity of pheromone. The different rewards are saved on a pheromone table. This control system should improve the load balancing system in Cloud environment. Each node VMj of the Cloud system will be identified by:

- Cost and technical capacity
- Probability to be chosen
- Pheromone table

The first step of the proposed algorithm is the initialization of the pheromone per virtual machine VM_j. The initial pheromone value $\tau_j(0)$ of VM_j is calculated as follows:

$$\tau_j(0) = \left(Pe_{capacity} + St_{capacity} + Memory + BW\right) \quad (2)$$

Where $Pe_{capacity}$ is processing capacity, $St_{capacity}$ is storage capacity, Memory is the enable memory and BW is bandwidth ability indicators of the VM_j.

After the initialization step, each ant k select a VM_j for achieving a task with a probability defined as follows:

$$P_j^k(t) = \begin{cases} \frac{[\tau_j(t)]^\alpha [Ct_j]^\beta [LB_j]^\gamma}{\sum[\tau_k(t)]^\alpha [Ct_k]^\beta [LB_k]^\gamma} & \text{if } j\ 1,..,n \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

Where Ct_j is the cost parameter of the VM_j and LB_j is the load balancer indicator for VM_j which allows the algorithm to maintain an optimized level of load balancing.

α, β and γ are respectively the weight of the component of pheromone trial. Otherwise the intensity of pheromone update is given hereafter:

$$\tau_j(t+1) = (1-\rho) * \tau_j(t) + \Delta\tau_j \quad (4)$$

The pheromone value can increase or decrease and value of ρ vary between 0 and 1. The main steps of the ACO load balancing algorithm are summarized hereafter:

Algorithm II: Ant Colony Optimization

```
Begin
  Step 1: Initialize the pheromone
  while (termination condition not met) do
      Step 2: Construct solutions for each ant k
      Step 3: evaluate the fitness of each solution
      Step 4: save the best solution
      Step 5: update the pheromone value
  end
end
```

VI. Conclusion and Future Work

Given the increasing number of users in the Cloud and the geographic scope of data centers, it becomes more and more difficult to maintain a satisfactory level of availability, security and responsiveness through conventional models of load balancing. Through this paper we propose an improved architecture for load balancing by offering to share the functions of the main controller representing the entry point to Cloud into two parts. Firstly, the main controller will keep the function of partitioning users' tasks through different regional load balancers. Secondly, the auxiliary controller will insure the updating state of the system through various agents. Otherwise, we choose to combine different algorithms in order to optimize the partitioning and mapping steps. Briefly, the proposed architecture of the Cloud is organized on three levels and each level has a specific role and specific algorithm. The main controller applies an algorithm based on the geographic location of both user and data center in order to choose the closest load balancer. The regional load balancer applies a second policy based on ant colony optimization algorithm to allocate the appropriate nodes to users' tasks. Through this approach, we aim to optimize the whole response times of services in the Cloud. Following this work, we plan to test the solutions proposed firstly, on the CloudSim simulator and secondly in a real environment and finally to improve the proposed algorithm.

Acknowledgment

The authors would like to put on record their thanks to the anonymous referees for their precious suggestions, which have improved the value of the paper over its initial version.

References

[1] S. K. Garg, S. Versteeg, and R. Buyya, "SMICloud: A Framework for Comparing and Ranking Cloud Services," 2011, pp. 210–218.

[2] Y. C. Tay, "Analytical Performance Modeling for Computer Systems," *Synth. Lect. Comput. Sci.*, vol. 2, no. 1, pp. 1–116, Apr. 2010.

[3] J.-S. Chang and R.-S. Chang, "A performance estimation model for high-performance computing on clouds," in *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*, 2012, pp. 275–280.

[4] S. Mare, D. Kotz, and A. Kumar, "Experimental validation of analytical performance models for IEEE 802.11 networks," in *Communication Systems and Networks (COMSNETS), 2010 Second International Conference on*, 2010, pp. 1–8.

[5] K. Erciyes, *Distributed Graph Algorithms for Computer Networks*. London: Springer London, 2013.

[6] B. Furht and A. Escalante, Eds., *Handbook of Cloud Computing*. Boston, MA: Springer US, 2010.

[7] P. Mell and T. Grance, "The NIST definition of cloud computing," 2011.

[8] D. E. Williams and J. R. Garcia, *Virtualization with Xen: including XenEnterprise, XenServer, and XenExpress*. Burlington, MA: Syngress, 2007.

[9] A. Tripathi and S. Raghuwanshi, "Cloud Task Scheduling in Hybrid Environment using Ant Colony Optimization."

[10] 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 *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, 2010, vol. 1, pp. 108–113.

[11] V. Pathari and K. Calicut, "Load Balancing In Cloud Computing."

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

[13] Y. Lu, Q. Xie, G. Kliot, A. Geller, J. R. Larus, and A. Greenberg, "Join-Idle-Queue: A novel load balancing algorithm for dynamically scalable web services," *Perform. Eval.*, vol. 68, no. 11, pp. 1056–1071, Nov. 2011.

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

[15] 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," *J. Comput. Syst. Sci.*, vol. 79, no. 8, pp. 1230–1242, Dec. 2013.

[16] R. Mishra, "Ant colony Optimization: A Solution of Load balancing in Cloud," *Int. J. Web Semantic Technol.*, vol. 3, no. 2, pp. 33–50, Apr. 2012.

[17] M. Belkhouraf, A. Kartit, H. Ouahmane, H. K. Idrissi, Z. Kartit, and M. El Marraki, "A secured load balancing architecture for cloud computing based on multiple clusters," in *Cloud Technologies and Applications (CloudTech), 2015 International Conference on*, 2015, pp. 1–6.

[18] C. Xu and F. C. Lau, *Load balancing in parallel computers: theory and practice*, vol. 381. Springer Science & Business Media, 1996.

[19] P. P. G. Gopinath and S. K. Vasudevan, "An In-depth Analysis and Study of Load Balancing Techniques in the Cloud Computing Environment," *Procedia Comput. Sci.*, vol. 50, pp. 427–432, 2015.

[20] M. Berwal and C. Kant, "Load Balancing in Cloud Computing," vol. 6, no. 2, pp. 52–58, 2015.

[21] B. Wickremasinghe, R. N. Calheiros, and R. Buyya, "CloudAnalyst: A CloudSim-Based Visual Modeller for Analysing Cloud Computing Environments and Applications," 2010, pp. 446–452.

[22] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw. Pract. Exp.*, vol. 41, no. 1, pp. 23–50, Jan. 2011.

[23] G. Taguchi, S. Chowdhury, Y. Wu, S. Taguchi, and H. Yano, *Taguchi's quality engineering handbook*. Hoboken, N.J. : Livonia, Mich: John Wiley & Sons ; ASI Consulting Group, 2005.

[24] T. S. Marco Dorigo, *Ant Colony Optimization*. London, England: A Bradford Book The MIT Press, 2004.

[25] M. Gendreau and J.-Y. Potvin, Eds., *Handbook of Metaheuristics*, vol. 146. Boston, MA: Springer US, 2010.

[26] S. Banerjee, I. Mukherjee, and P. K. Mahanti, "Cloud computing initiative using modified ant colony framework," *World Acad. Sci. Eng. Technol.*, vol. 56, pp. 221–224, 2009.