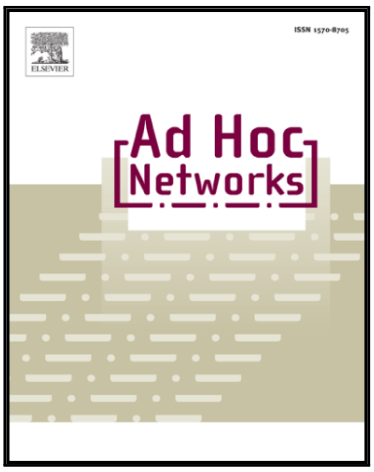

Probabilistic Modeling to Achieve Load balancing in Expert Clouds

Shiva Razzaghzadeh , Ahmad Habibizad Navin ,
Amir Masoud Rahmani , Mehdi Hosseinzadeh

PII: S1570-8705(17)30001-X
DOI: 10.1016/j.adhoc.2017.01.001
Reference: ADHOC 1510

To appear in: *Ad Hoc Networks*

Received date: 2 June 2016
Revised date: 14 December 2016
Accepted date: 4 January 2017

Please cite this article as: Shiva Razzaghzadeh , Ahmad Habibizad Navin , Amir Masoud Rahmani , Mehdi Hosseinzadeh , Probabilistic Modeling to Achieve Load balancing in Expert Clouds, *Ad Hoc Networks* (2017), doi: 10.1016/j.adhoc.2017.01.001

Probabilistic Modeling to Achieve Load balancing in Expert Clouds

Shiva Razzaghzadeh[a], Ahmad Habibizad Navin[b]*[1], Amir Masoud Rahmani[a], Mehdi Hosseinzadeh[a]

[a]Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran
[b]Department of Computer Engineering, Islamic Azad University, Tabriz, Iran

Abstract

Expert Cloud as a new class of Cloud computing systems enables its users to request the skill, knowledge and expertise of people without any information of their location by employing Internet infrastructures and Cloud computing concepts. Effective load balancing in a heterogeneous distributed environment such as Cloud is important. Since the differences in the human resource (HRs) capabilities and the variety of users' requests causes that some HRs are overloaded and some others are idle. The task allocation to the HR based on the announced requirements by the user may cause the imbalanced load distribution among HRs as well. Hence resource management and scheduling are among the important cases to achieve load balancing. Using static and dynamic algorithms, the ant colony, and the method based on searching tree all are among the methods to achieve load balancing. This paper presents a new method in order to distribute the dynamic load based on distributed queues aware of service quality in the Cloud environment. In this method, we utilize the colorful ants as a ranking for making distinction among the HRs capabilities. In this paper, we perform the mapping among the tasks and HRs using allocating a label to each HR. We model the load balancing and mapping process based on Poisson and exponential distribution. This model allows us to allocate each task to the HR which is able to execute it with maximum power using the distributed queues aware of the service quality. Simulation results show that the expert Cloud can reduce the execution and tardiness time and improve HR utilization. The cost of using resources as an effective factor in load balancing is also observed.

Keywords: Expert Cloud, human resource, Cloud computing, load balancing, Poisson distribution, quality of service.

1. Introduction

Human resources (HRs) are the important components of the Societies and organizations so that the Success of each organization depends on its HRs. The organizations achieve their goals by means of Knowledge, experience, strength and skills of human beings. Since the HRs are geographically distributed, it is necessary to establish an infrastructure to share the Knowledge, skills and experience of human beings. This new platform is named Expert Cloud [1].

According to the definition of National Institute of Standards and Technology(NIST) Cloud computing is Internet based computing in which the numerous groups of servers have been networked to allow the sharing of data-processing tasks, centralized data storage and online access to the computer services or resources. In other words, Cloud computing relies on

[1]*[b]Corresponding author: Ahmad Habibizad Navin .Tel.: +98 9144125973;fax: +98 4113817221.
E-mail addresses: shiva.razzaghzadeh@srbiau.ac.ir (SH.Razzaghzadeh), a.habibizad@srbiau.ac.ir (A. Habibizad Navin)*, rahmani@srbiau.ac.ir (A. Masoud Rahmani), hosseinzadeh@srbiau.ac.ir (M. hosseinzadeh)

resource sharing to achieve coherence and economies of scale, similar to a utility over a network. It concentrates on the maximization of the resource sharing effect [2].

A typical distributed system such as Expert Cloud includes the number of distributed HRs. These HRs can be connected to each other to achieve the high performance and execute the task that one HR is not able to do it by itself. To reduce the task execution time by the HR, the workload should be distributed based on the HRs strength. This makes the load balancing necessary. The main purpose of the load balancing is that it facilitates networks and resources by providing a maximum throughput with minimum response time. Dividing traffic among users, and as a result data can be sent and received without major delay [3, 4].

Load balancing and resource management algorithms can be static, dynamic or hybrid [5]. Static algorithms which often use FCFS (First-Come-First-Served) policy devote tasks to the resource based on simple information system such as the function of processor, memory capacity etc. It distributes the tasks among the resources by presenting formulae. In this method, the system doesn't need to collect the information from the environment continuously. The dynamic policies such as genetic algorithm use the present state of system for task allocation and finally the hybrid algorithms use the combination of static and dynamic ones. The hybrid algorithm switches to each of them at the appropriate time. Although these algorithms supply the load balancing, they do not pay attention to the resource capabilities. They do not consider the requirements announced by the user for task execution as well. This means the lack of maximum resource utilization. The load balancing algorithms can be divided into two groups. These are centralized and distributed. In the centralized method only a node is responsible for the management and load distribution through the whole system. This method is simple but its problem is bottleneck. In the distributed method, each node collects the load information of other nodes individually. The load balancing decision is local in this method. The method is suitable for the distributed environment such as Cloud. The method has overhead problem [6].

In this paper, we present a new method for load balancing in Expert Clouds. We utilize the dynamic load balancing and present a mechanism to improve some of the previous methods (such as Branch and Bound, FCFS, and etc). In spite of the fact that most of the existing methods have presented the suitable algorithms to achieve load balancing, they ignore the following cases: the task priority, task characteristics given to the system, the suitable mapping among the tasks and resources and also the system scalability. Of course, most of these methods' purpose is to find an idle virtual machine. It should be mentioned that they don't pay any attention to the effective allocation and processing power of the virtual machines in order to execute the tasks. Our proposed method tends to improve the execution time by considering these problems, but our new viewpoint is that our proposed algorithm models the load distribution being aware of resource service quality and based on Poisson process. Our method allocates the tasks by considering the queue length and calculation of the allocation rate, and so it obviates the existing method problems. This process in this way that this method ranks HRs by inspiring the colorful ants in nature. It shows the HR capabilities in the form of a label. In continuation, the super-peer nodes in each site perform the mapping among the HRs and tasks based on Poisson and exponential distribution model. The super-peer nodes utilize the Poisson distribution and probability mass function distributed queues length for mapping to allocate the tasks to the HRs which are able to execute them with high power and the minimal load. In

addition to allocation, our method allows us to provide load balancing. Simulation result proves the performance of our proposed method.
The rest of the paper is organized in such a way that we can introduce the related work in section 2. In section 3, we discuss the new method that we have presented. We express the simulation and the result of the proposed method in section 4 and finally in section 5 we conclude the paper.

2. Related work

The load balancing is the mechanism that causes the tasks to be transmitted from overloaded resources to underloaded ones. This leads to the effective resource utilization and decreases the response time. Main load balancing algorithms can be static, dynamic, and those based on resource management and scheduling as well. In this section, we are going to introduce the existing algorithms in these topics.

- Static methods: CLBDM (Central Load Balancing Decision Model) is a centralized and static load balancing method. In fact, it is the improved Round Robin. Round Robin method [7] sends the requests to the node with the minimum connection but CLBDM [8] computes the relation between client and each node in the Cloud. If it is much greater than threshold, the connection ends, and the task is transmitted to the other node via Round Robin algorithm [8]. Our proposed algorithm allocates the task to the HRs with the minimum load. OLB (Opportunistic Load Balancing) is a centralized and static method, and its purpose is to keep the node busy in the Cloud [9]. It does not pay attention to the execution time in the node. This may cause the task processing to be slowly done. The load balancing algorithm in the private Cloud is presented in [10] which maps the virtual machine to the physical one. The architecture of this algorithm includes a central scheduling controller and resource monitoring. The controller unit determines that which resource is able to execute the task, and the resource monitoring unit collects the resource information. The task mapping process delivers the task to the resource with the highest score after four phases including task submission, virtual machines (VM) requests, receiving the resource information and resource capability computation.
- Dynamic methods: LBMM (Load Balance Min-Min) [11] algorithm called the minimum load balancing utilizes the opportunistic load balancing manner. It improves the OLB by adding three layers to OLB architecture. The first layer of the architecture is the manager that is responsible for receiving the task. This layer delivers the task to the service manager in the second layer. The second layer divides the request into subtasks in order to increase the processing speed. Then this layer delivers the task to the third layer i.e. the node layer. They (nodes) are responsible for task execution [11]. The algorithms inspired the nature have also been proposed for load balancing in Grid and Cloud. One of these algorithms is the HBB-LB algorithm (Honey Bee Behavior Inspired Load Balancing) [4]. In this method, the tasks and resources are considered honey bees and food respectively. The pioneer honey bees are searching for the food. They announce the distance to the food store and the amount of the food to the rest of the honey bees by waggle dance after finding the certain location. In this way, the tasks are transmitted to the resources. In this method running out of the food in one food resource means that the resource is overload. Finding the

underloaded resources, the load balancing is made between these two kinds of resources [4]. The load balancing presented in [12] inspired the ant behavior. Ants act as a request. When a request is submitted, the ants begin to move from the head node and secrete pheromone. The forward movement means the movements form the overload node toward the next one. If the next node is under-loaded, the load transition to this node is done. If the node is overloaded the ants move backward. When the ant finds the target node, it commits suicide [12]. The next strategy is based on intelligent agent [13]. In this method each agent is the indicator of a local resource. In the higher level, agents cooperate to provide the global load balancing. In this method, local load balancing is provided by using FCFS (Fist-Come-First-Served) and genetic algorithms. These algorithms present the appropriate solutions for dynamic scheduling, but they are not efficient in the large environments. So, in these conditions the global load balancing is used. This method acts based on the service discovery and advertisement. It forms the ACT (Agent Capability Table) and presents the necessary information at the disposal of agents to make load balancing. This method never faces the bottleneck. The algorithm proposed in [14] is a dual direction download for FTP (DDFTP) server. This algorithm named DDFTP is used for load balancing. It divides the file with M size into M/2 units. Two servers, individually, begin to download. For example, the first server from the zero block and the second one from the M block begin to download. The download ends when two servers get to the consecutive blocks. At this time, the total file in the best form is delivered to the customer from each server. This method decreases the network connections between clients and nodes. It also decreases the overhead. An optimized task scheduling model and load balancing is presented in [15]. In this algorithm the CTS (Central Task Scheduler) process is introduced. Its purpose is finding the resources which are able to execute the tasks which are transmitted from the overloaded resources. In order to achieve the optimization of task migration the algorithm has considered the bandwidth as a variable for minimizing the task transmission time. Our proposed method like this one is a dynamic algorithm. Among the other studies, regarding the load balancing, one of them is the public Cloud partitioning [16]. In this method, the public Cloud is divided into several units. A main controller determines that the submitted task enters what part of the Cloud. Load balancer selects the best load balancing strategy. Of course, this unit collects the state of the nodes. The controller unit directs the task to the idle or normal units via this information. One of the existing methods has presented dynamic load balancing which considers server throughput and resource load as well. In this algorithm, it is less probable for a server to execute users' request. This method uses virtual web servers and physical servers to monitor the platforms. In this way, it considers the server load, throughput, and service priority and it distributes the tasks based on resource load. This method makes balance the load with highly scalable performance. It increases the average response time [17]. One of the load balancing algorithms utilizes the nature-inspired optimization approach. This algorithm discovers the best tasks as candidates for migration, and it also discovers the best computation resources to receive these tasks. This method is the improved version of the external optimization (EO) [18]. Its main difference with EO is that there is the computation node selection is done using the stochastic selection in which the probability is guided by some knowledge of problem [18]. This algorithm improves the load balancing service quality, but its scalability is low and its

response time is high. In order to increase the resource utilization and also to delete the system bottleneck, another method has presented a framework for load balancing and resource management. In this framework, a workload monitoring has been designed which discovers overload and underload nodes in the cluster. In order to make load balancing among the nodes, the split, merge, and pair algorithms have been applied to regulate the physical machines and resource allocates algorithm has been used to regulate the VMs. This method increases the system performance and utilization but it can increase the system hardware cost [19]. One of the other methods considers the technical factors of load balancing and also the network structure for execution of load balancing algorithms in order to improve the efficiency of algorithms [20]. This method facilitates the load balancing efficiency in heterogeneous environments through considering each node capacity and calculating the effective load average of each node. In this way, it causes the load balancing algorithm to be effectively executed among the constructed overlay network. This method increases the system throughput, but its scalability has not been matched with Cloud environment requirements [20].

- Resource management methods: One of the existing methods to provide the resource is to distinguish the VMs amount which have been dynamically bought, in order to reduce the total computation cost and to keep the service quality. This algorithm has proposed an online overload probable estimation model which it doesn't need prior knowledge of load. It causes the quick and instant response to the load changes. The architecture proposed by this method includes customers, Iaas provider(s), and service provisioning engine (SPE) in which Iaas provider presents the basic computation resources in the form of VMs. SPE also receives the request from the customers, and it analyzes the characteristics of task load by the consideration of requested queue. It estimates the overload probability of VMs. If this probability is high, a large number of VMs should be activated and if this probability is low, the number of these VMs should be shut down to decrease the cost. Finally each task is allocated to the suitable VM with suitable scheduling and comparing the VMs service quality. This algorithm adjusts the service quality for the future task load by using mathematical formula and by adjusting the active VMs [21]. One of the existing methods regarding the load distribution presents a mechanism in order to distribute the users' request in Cloud CDN (Content Delivery Network). It can propose community based video replication. The purpose of this algorithm is to reduce the operational system cost and guarantee average service delay. This method clusters social communications which are interested in watching similar videos which can be found in a close geographical condition; of course it presents this condition in the form of weighed graph, follower, and folllowee (such as Twitter). It presents a framework for the extension of community based video replication. In this method the requests are scheduled based on community. In this way, the users of a community can access a set of CDN corresponding nodes related to regions covered with this community. This algorithm presents the services with the least time delay to the user by sending several queries to the related CDN nodes [22]. One of the existing algorithms has presented the dynamic resource provisioning scheme for transcoding with quality of service (QoS) guarantee in online video sharing services. The purpose of this method is to provide heterogeneous QoS requirements with maximum resource utilization and reducing resource consumption. In this method, the users and video content

professional producers upload their contents for the online video sharing. The processing module divides the content into a set of equal duration chunks using the media stream segmenter. Afterword prediction module considers the arrival rates of various video chunks. This module predicts the arrival rate for the future video chunks. In the following, decision making unit makes decision on resource provisioning using the system information. Provisioning module dynamically adjusts the number of active transcoding instances in order to satisfy the QoS and workload distribution. Finally, the video chunks are dispatched to the all active transcoding instances using the dispatching module. In this way, this algorithm dynamically manages the resources by predicting the arrival rate and paying attention to the QoS [23]. One of the other existing methods presents a systematic framework to survey the following cases: data generation, data acquisition, data storage, and data analytics. This method expresses the challenge of a scalable big-data system as follows: 1. The variety of the data sources and sheer volume of data makes difficult the data integration in the distributed environment. 2. The big-data systems need to store and manage the massive and heterogeneous data sets. 3. Big-data analytics should be effectively mined from the massive data in different levels and real-time. Cloud computing can be used as an infrastructure layer to address these challenges. This layer is also used to provide certain requirements such as: cost-effective, elasticity, and the ability to scale up or down. The infrastructure layer consists of pool of resources which should be allocated to meet the big-data demand. In this process the maximum resource utilization should be achieved. Batch processing is one of the scheduling methods used for data set management in this paper. Data are primarily stored and then analyzed in the batch processing. MapReduce [24] is one of the main models of batch processing. The core idea of the MapReduce is that the data are primarily divided into small chunks. Then these chunks are processed into parallel and distributed forms. And they produce the intermediate results. In the end, the final result is achieved using the aggregation of intermediate results. So the existing layers in cloud present the suitable manners to manage the data sets in big-data [25].

Among the existing problems concerning the mentioned algorithms in this section is that most of them do not pay any attention to the requirements and priorities announced by the user. Most of these methods' purpose is to find an idle VM without any attention to the effective allocation which can lead to defeat the task execution and increase its time. For example some of these algorithms consider the resource power during task allocation, but they do not notice the resource load. The proposed method in [4] takes priority over the tasks but it does not pay attention to the resource power. In the proposed method in [8] the connection time is more. This can be done because there is no load balancing in node, and the shortage of memory space etc. Our proposed method presents a new idea considering the weak points of some of these algorithms; in spite of the existing methods, our proposed mechanism performs load distribution being aware of resource service quality (HRs) and based on Poisson process. According to the task priority, resource characteristics (HRs) considering the queue length and calculating the distribution rate, this method allocates the tasks effectively. Of course this is vividly presented in section 3.

3. Proposed method: concepts and operators

The different algorithms present the suitable methods for load balancing in the Cloud as well; most of them don't pay attention to the resource capabilities and submitted task specifications. In most of these algorithms, the main goal is to find the idle resource for task allocation without paying attention whether the resource is capable to execute the task or not. This paper presents a new method based on the above -mentioned problems. It allocates the tasks to the HRs based on two factors; arrival rate and service time. These factors are calculated by the point estimation. The proposed method models the allocation according to Poisson and exponential distributions (In probability theory, a Poisson process is a stochastic process that counts the number of events in a given time interval and the exponential distribution is the probability distribution that describes the time between events in a Poisson process) [26, 27, 28]. The process is in this way that, at first, we rank the HRs by the colorful ants and put them in three sites. Then, we allocate a label consisting of three parts for each HR. In order to allocate the tasks for the HRs, the super-peer nodes in each site calculate the tasks arrival rate through estimating service time of each node. In this way, the HRs which are not suitable for the submitted tasks are pruned. Finally, by using the distribution rate and surveying the queue length, HRs are selected by the maximum strength and minimum load (algorithm 1).

3.1. The HR ranking by colorful ants

We use the colorful ants' concept for making distinction among the HRs capabilities. In this way, we consider the ants in two colors (red, and yellow). Each of these ants is used for the measurement of some of the HR capabilities. The process is in such a way that each two types of colorful ants is sent towards the HRs in a parallel way. It is hypothesized that the number of each type is *n*, so *n* red ants are sent to the HR as a virus and attacker. If the number of backward ants from the HR is applicable to Eq. (1), it means that the HR has a high protection. In other words, the HR includes a powerful anti-virus and protection.

$$\frac{n}{2} + k \leq backward\text{-}ants \leq n \quad ; \quad 0 \leq k \leq \frac{n}{2} \tag{1}$$

In Eq. (1), *n* shows the number of forwarded ants and *k* is a fixed value. We consider its value two (*k*=2). Of course, the value of n is considered twenty. The yellow ants playing the application role are forwarded in a parallel way to the red ants. These kinds of ants measure the HR speed. For this purpose, a time out is used. If the number of yellow backward ants accords to the Eq. (1), and they come back at the time span less than time out, then the HR is fast. Definition 1 shows this clearly.

***Definition* 1.** If $\forall \; n_{\text{forwarded-ants}} \; \exists \; n_{\text{back warded-ants}} \mid \frac{n}{2} + k \leq backward\text{-}ants \leq n \rightarrow$ *HR is protected and fast*

3.2. Task division and site ranking based on request

We use the HR ranks to make sites. We consider three sites for this purpose which consists of tree structure. The HRs which are fast and have high processing power are set in the first site. The HRs with the highly protective capability are set in the second site. Finally, the HRs which are both fast and highly protective are set in the third site. In order to determine the specifications of each HR in sites, we allocate a label for it. Each label consists of three units. L is the number of existing tasks in the queue at the t_{i-1} moment. T_s is the HR service time and β shows the rank of HR. In addition to HRs, the submitted tasks by the users should be labeled. Indeed, this label determines the type of user request. If the task has to be immediately executed and it can't wait, in other words, this task needs the powerful and fast HR, then the value of label will be one. If the user requests the HR with high protection, the value of task label will be two. At the last step, if a task requires fast and secure HR, the label value will be three.

3.3. Leaf nodes labeling

The specifications of each HR in each site are determined by its label. This label for leaf nodes consists of L_{i-1}, β and $\overline{Ts}$ (Fig.1). L_{i-1} is the indicator of the number of existing tasks in each HR queue at the t_{i-1} moment. The second part of the label is $\overline{Ts}$ or the service time of each node; in fact it shows the average task execution time. The amount of $\overline{Ts}$ is calculated according to the history of each HR in the task execution. For this purpose, we use point estimation. In this method, we hypothesize that $R_1, R_2, \ldots\ldots, R_n$ are random sample of n-array from HRs with considering the service time of $t_1, t_2, \ldots t_m$ whose probability distribution is $f_\theta(t)$. It depends on θ unknown parameter. The purpose of estimation is to calculate the average task execution time by each HR. It acts as an approximation of the unknown θ parameter.

According to the random sample of n-array namely $R_1, R_2, \ldots\ldots, R_n$, if the given statistics to estimate the unknown parameter is $S= S(R_1, R_2, \ldots\ldots, R_n)$, then the random variable $s= s(R_1, R_2, \ldots\ldots, R_n)$ is called the estimator of θ parameter. $s= S(t_1, t2\ldots, tm)$ is an estimate for θ parameter. Let $R = \{R_i \mid R=1,2,\ldots\ldots..n\}$ be the set of leaf nodes in each site which should process k tasks. We can estimate the average task execution time by each HR. Based on this idea $\overline{Ts}$ is the estimator of the average task execution time in each leaf node. and this is exponential distribution or μ .

$$\overline{Ts_j} = \frac{1}{k}\sum_{i=1}^{k} ts_i \qquad j=1, 2, \ldots\ldots n \quad , \; i=1,2,\ldots..,k \tag{2}$$

$$\overline{Ts_j} = \mu_{time} \tag{3}$$

$$\lambda_j = \frac{1}{\overline{Ts}j} \qquad j=1, 2, \ldots\ldots n \tag{4}$$

In Eq. (2), $\overline{Ts}$ is the same as service time. It is the indicator of k task average execution time by each leaf node (the number of allocated tasks to each leaf node is k and the number of leaves in each site is n). It is also the indicator of exponential distribution. We can acquire the λ which is

the same arrival rate, based on Poisson process, by calculating the μ_{time} (Eq. (4)). The third section of label is β. We hypothesize that the β value is regarded one for the HRs in the first site, two for the HRs in the second site and three for the third one (Eq. (5)).

$$\beta = \begin{cases} 1 & \text{if HR is fast} \\ 2 & \text{if HR is protective} \\ 3 & \text{if HR is fast\& protective} \end{cases} \quad (5)$$

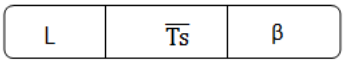

Fig. 1. Leaf node label

3.4. super-peer nodes labeling

The super-peer nodes in each site which take charge of the computation affairs such as considering the calculating the present load and allocation rate in the time unit plays a role in two levels, namely, level one (sp1 or root node) and two (sp2 or the children of sp1) (the clear explanation in next section). The specification of these nodes is shown in a label with three sections (Fig.2). The label includes, $\bar{L}_{i-1}$, $\overline{Ts}$ and β. $\bar{L}_{i-1}$ in level two super-peers (sp2) nodes is the average number of existing tasks in the children queue at t_{i-1} (Eq. (6)). The number of sp2 in each site is m and each sp2 includes h children. The second section of the label is $\overline{Ts}_{sp2}$ which shows the average execution time by the sp2 children; this is the same exponential distribution. Each leaf node executes the tasks with the $\overline{Ts}$ average time and based on the previous section. So, the sp2 node with h children follows these equations:

$$\bar{L}_{sp2_i} = \frac{1}{h}\sum_{j=1}^{h} L_j \qquad i=1,2,\ldots\ldots,m \quad , \quad j=1,2,\ldots\ldots,h \quad (6)$$

$$\overline{Ts}_{sp2_i} = E\left(\sum_{j=1}^{h} \overline{Ts_j}\right) \qquad i=1,2,\ldots\ldots,m \qquad j=1,2,\ldots.h \quad (7)$$

$$E\left(\sum_{j=1}^{h} \overline{Ts_j}\right) = \sum_{j=1}^{h} E(\overline{Ts_j}) = E(\overline{Ts_1}) + E(\overline{Ts_2}) + \ldots\ldots\ldots + E(\overline{Ts_h}) \quad (8)$$

$$E\left(\sum_{j=1}^{h} \overline{Ts_j}\right) = \mu_{time1} + \mu_{time2} + \ldots\ldots\ldots + \mu_{timeh} \quad (9)$$

In Eq. (8), E(x) is the same Mathematical Expectation. In the continuation, the arrival rate for each super-peer node is calculated according to the Eq. (10).

$$\lambda_{sp2_i} = \frac{1}{\overline{Ts_{sp2_i}}} \qquad i=1,2,\ldots.,m \quad (10)$$

The value of $\bar{L}_{i-1}$ and $\overline{Ts}$ for the first level of super-peer node is also calculated similar to the above steps. The difference is that these calculations are calculated by considering the value of $\bar{L}_{i-1}$ and $\overline{Ts}$ of the second level super-peer nodes. The third section of the label is β. The value of β for the all nodes is similar to the leaf nodes.

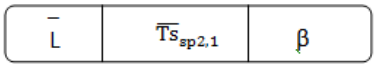

Fig. 2. Supervisor node label

3.5. Proposed load balancing strategy

The proposed load balancing policy is done in four phases. This method is based on tree structure, mathematical model, and is inspired by colorful ants in the nature. In the first phase, we utilize the colorful ants for determining the HR capabilities. In this phase, we rank the HRs. In the next phase, we consider three sites in which the HRs ranking one (powerful and fast HRs) are set in the first site ; the HR ranking two (highly protective HRs) are set in the second site, and finally the HRs ranking three (the secure and fast HRs) are set in the third one. Whenever the user requests the service, primarily, the request is transferred to the manager. The manager devotes the suitable label to it by considering the type of user request. At this time, the available HRs in the sites are also labeled by β, $\bar{L}$ and $\overline{Ts}$ (based on previous sections).This is the third phase. The fourth phase is the mapping and task allocation step. In this step, the manager organizes the submitted tasks, namely those which have the label valued 1 are sent to the first site, those which have the label valued 2 are sent to the second site and finally those valued 3 are sent to the third one. The processing of tasks on a HR cannot be interrupted (assuming that the failure does not occur). In each site, the root node and its first children level are the first and second level super-peers respectively. In fact, the leaf nodes in each site are responsible for the task execution. The super-peer nodes take charge of the computation affairs such as considering the present load, the allocation rate, and also the lack of load balancing among them. Each leaf node (virtual machine), includes a queue. These queues are distributed in each site. Each queue also includes tasks waiting to get service. The super-peer nodes utilize the length of these queues to keep and consider the load balancing. It is hypothesize that, primarily, (the time in which there is no allocation) the queue length is zero, and at this time only the λ is considered for allocation. In this way that whenever the tasks are transmitted to sp1, this node compares the number of the tasks with the value of λ related to each one of sp2 nodes. If sp1 finds the λ value of its children fewer than the number of arrival tasks, it is pruned (diffinition1). At this time, sp1 chooses the nodes with maximum λ among the sp2 nodes, and allocates the tasks to the leaf nodes of this super-peer. In continuation, and after the first allocation it is possible that we face the conditions in which the tasks execution in VMs (HRs) is prolonged; and this leads to form the queue. At this time, in each phase of allocation in addition to λ surveying the queue length should be considered as well. That is to

say that the number of existing tasks in the queue at the t_{i-1} moment should be considered in order to allocate it at t_{i+1} moment; and each sp2 optimal child selected according to Eq. (11).

Definition 2. If $\forall \lambda_{input} \exists \overline{\lambda\imath}$ i= 1,2,3,… $|\overline{\lambda\imath} < \lambda_{input}$ then children of $\overline{\lambda\imath}$ is pruned

$$AR_j = \lambda_j - \bar{L}_{j_{ti-1}} \qquad j=1,2,....,z \tag{11}$$

Eq. (11) shows the allocation rate to each HR in which λ_j is the arrival tasks to the site, and $\bar{L}_{j_{ti-1}}$ is the average of waiting tasks in the queue (z shows the total HRs in the site). If AR_j value is fewer or equal to zero, that is to say the VM (HR) is overloaded and it can 't accept a new task; and if AR_j is more than zero, the allocation will be possible. Each sp2 begins to allocate tasks according to AR_j value to its children after receiving the tasks from sp1. All the mentioned steps are repeated in order to allocate all the tasks. Fig.3.a and Fig.3.b show the proposed environment model and its workflow.

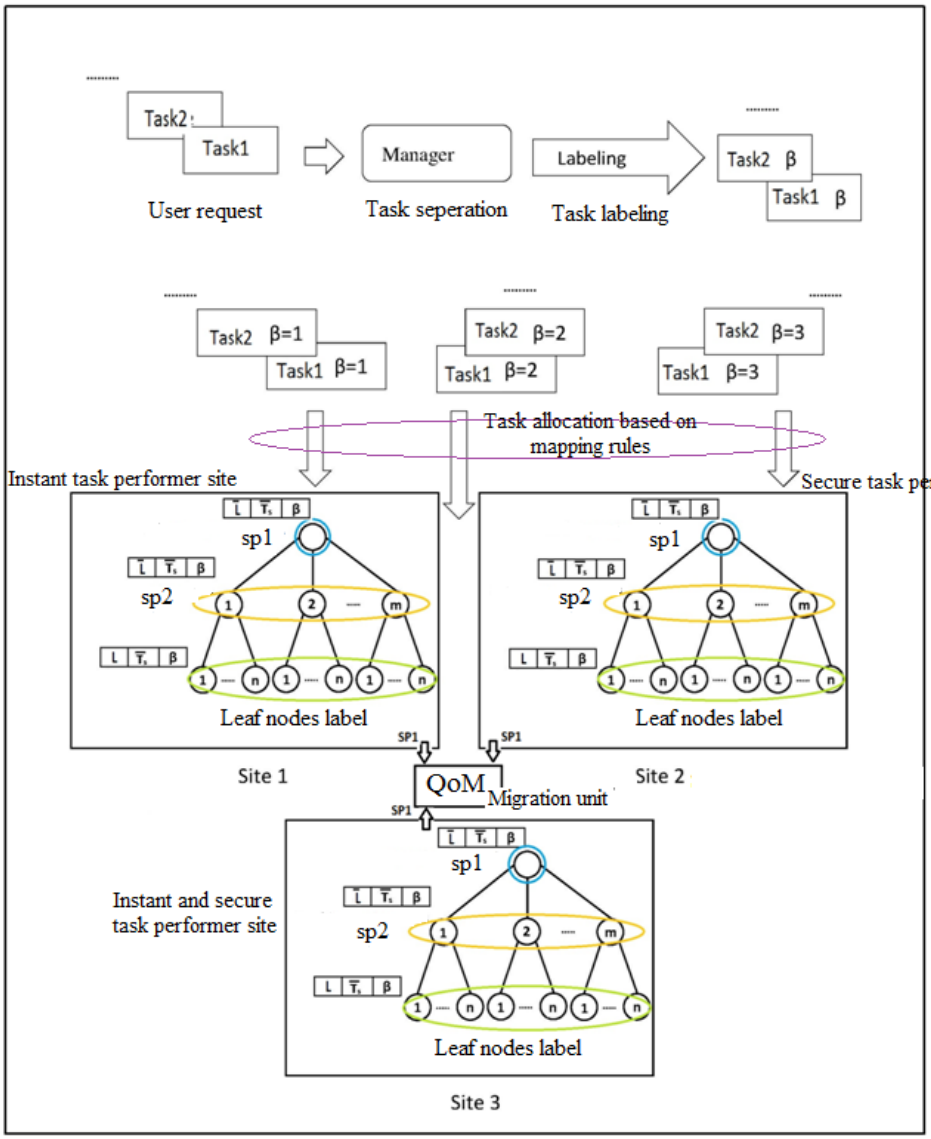

Fig. 3.a) Proposed environment model

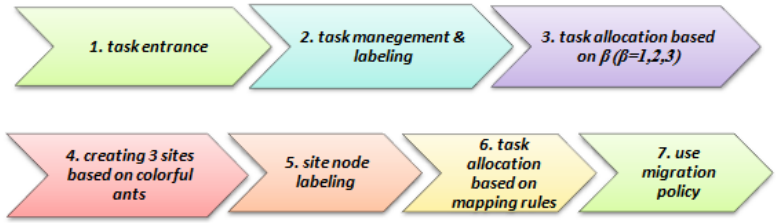

Fig. 3.b) The proposed method workflow

3.6. The tasks migration

We consider the conditions in which sp1 can allocate only a number of the total allocated tasks to the HRs according to AR_j value (when the queues are full); of course, when the number of tasks aren't allocated. At this time the inter-site migration policy is used. The process in this policy is in such a way that sp2 returns the tasks which aren't allocated to sp1. At this time sp1 considers the other sp2 nodes, and it allocates these tasks to the optimal nodes according to λ and AR_j values (according to previous section). Whenever the migration is impossible, namely sp1 doesn't find the child who follows the AR_j *and* λ rules; in this way it uses the transitional phase policy. The basis of this policy is the task migration among the sites. In transitional phase, sp1 levels of all the sites should make relation with each other in order to make a load balancing. This relation is possible through QoM (Queuing- oriented- middleware) (Fig.4). The process in this way that the tasks which are not allocated in each site are allocated to QoM by the sp1 of that site. In this unit, there are three queues with β one, two, and three values. The broker unit existing in QoM which is responsible for mapping the tasks receives the tasks from sp1s ; and it puts the tasks in its given queue based on surveying the β label part, and this section (broker) adds the destination label to each task as well. This is to say that, if the β task value is one, the sites include β with two and three values can give it service; if the β task value is two, then the site includes β with three value is just capable to give it service and finally if the value is three, the site includes β with two value can give it service. The broker is in relationship with storage. This storage includes λ *and* $\overline{L}$ related to sp1 of all sites. The broker considers the tasks label and it can calculate AR_j according to the information of the storage. Finally, the tasks migrate to the sites in which AR_j is more than zero; and also λ value is more or equal to the λ of tasks and its β accords with the tasks label as well. In the end; the tasks migrate as inter-site or outer site, and the load balancing is achieved. The aim of algorithm is to direct the L value to zero in a long time (Fig.4.a). Fig.4.b shows the tasks migration workflow as well.

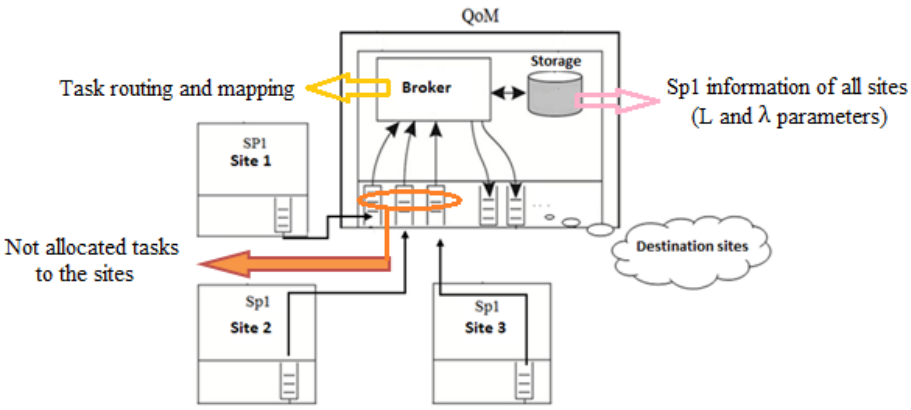

Fig. 4.a) Architecture of Queuing- oriented- middleware

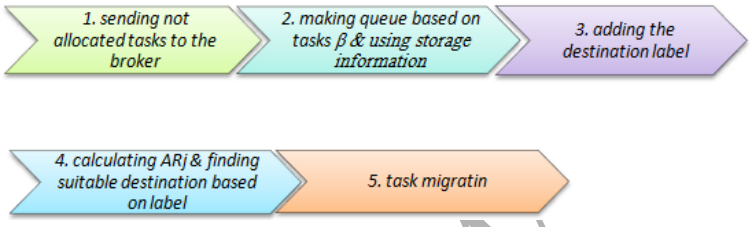

Fig. 4.b) tasks migration workflow

Algorithm 1.Our proposed algorithm at a glance

1. Begin
2. Give the tasks to the manager
3. For i=1 to k do // k is the number of tasks
4. {
5. Add label to the $task_i$
6. }
7. For j=1 to z do // z is the number of all HRs
8. {
9. find the HR_j rank
10. Put the HR_j in the suitable site
11. }
12. For f=1 to n do // n is the number of leaf nodes
13. {
14. $\overline{TS}_f = \frac{1}{h}\sum_{i=1}^{h} TS_i$ i=1,2,…,h //h is the numer of tasks in each leaf node
15. Compute L for each leaf node // L is the number of queued tasks

16. Put $\overline{TS_f}$ in its section in label
17. Put L in the label
18. Allocate 1, 2 or 3 for β section based on its site
19. }
20. For p=1 to m do // m is the number of sp2 nodes
21. {
22. $\overline{Ts_{sp2_p}}$ = E($\sum_{f=1}^{z} \overline{Ts_f}$) //z is the number of sp2 children
23. $\bar{L}_{sp2_p} = \frac{1}{z}\sum_{i=1}^{z} L_i$
24. $\lambda_p = \frac{1}{\overline{Ts_{sp2_p}}}$
25. Allocate 1, 2 or 3 for β section based on its site
26. }
27. For p=1 to m do
28. {
29. If number of input tasks >λ_p then
30. VM_p is pruned
31. Else
32. {
33. $AR_p = \lambda_p - L_{p_{ti-1}}$
34. If $AR_p > 0$ then
35. Allocate the tasks with λ_p rate to the sp2 children
36. }
37. If sme of input tasks can not allocate do
38. {
39. Send these tasks to the sp1
40. Goto line 29
41. }
42. If sp1 not found sp2 in this site do
43. {
44. Go to site 2 and 3 based on destination β
45. Repeat line 29 to 35
46. }
47. }
48. end

4. Simulation results and analysis

In this section, we demonstrate the conclusion of the proposed algorithm with simulation. In this section, we have analyzed the performance of our algorithm based on the results of simulation using Cloudsim and validate it in Amazon EC2 .We have expanded the Cloudsim classes for the simulation of our algorithm. The selected instance type is "t2.micro". We hypothesize that the number of tasks are varied from 10 to 40 with interval 10 according [4]. The task length is varied

from 100 to 2000. These tasks are independent. We do simulation in one data center. The number of VMs (HRs) in each host is 30. The hosts of data center have different specifications and the number of PEs is varied from 1 to 4. In general the environment of this data center is heterogeneous. The code of our algorithm is in such a way that it can be easily varied for the numerous number of VMs and tasks. In this way, the procedure is that after HR labeling, we place the HRs with β=1 in the first site, the HRs with β=2 in second site and finally the HRs with β=3 in the third one. The point which should be taken into consideration is that each site has tree structure; the maximum depth for each site is three. The existing HRs in the environment are different because of their processing power. We use MIPS option in order to make distinction among HRs processing power. In this way, we consider MIPS for HRs: 500, 100 and 300 respectively. Some of the important qualitative parameters for load balancing in Cloud computing are: makespan, tardiness, cost, and total execution time. Because of the importance of these parameters in Cloud, our goal is to focus on these parameters and performing the simulation to calculate their values. Through careful attention to the mentioned quantities, execution time and tardiness are computed as follow [29]:

$$makespan = total\ simulation\ time + overhead\_time. \quad (12)$$

$$tardiness = |granularity\_time - total\ simulation\ time\ |. \quad (13)$$

Eq. (12) shows the execution time. It obtains the sum of simulation time and overhead time. Eq. (13) also shows the delay and obtains the subtraction of granularity time and simulation time. Using these formulas; the Fig. 5 is formed. Taking the Fig. 5, axis x is the indicator of makespan and y axis shows tardiness. This figure shows that the maximum makespan in our proposed method is 70.41 seconds and its minimum is 46.31 seconds.

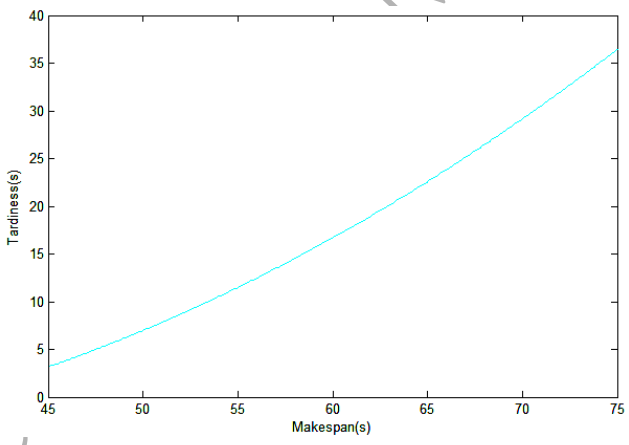

Fig. 5. Relationship between makespan and tardiness in our proposed method

The comparison between the proposed method and some of the other famous existing ones such as FCFS, LIFO, Min-Min, Max-Min, and B&B shows that our method is able to reduce the

execution time more effectively. Figure 6 and 7 makes the comparison between the maximum and minimum makespan among the proposed method and the other mentioned ones. The comparison of the algorithms in Fig.6 shows that the FCFS has maximum makespan and our method has the minimum one. The improvement of our method in comparison with B&B (has the least makespsn among the previous methods) is 2838.59 seconds. The comparison of the algorithms in Fig.7 shows that the FCFS has maximum makespan and our method has the minimum one. The improvement of our method in comparison with B&B is 2787.69 seconds. In our proposed method the effective reduction of makespan has been according to our expectation. This is because of, first, using the standard datasets and validating our method with EC2; secondly we have repeated the simulation 20 times to increase the execution accuracy; thirdly the main idea of our method is to effectively distribute the tasks in order to reduce the execution time. Namely all the tasks are allocated to the resources which have the capability to execute them. Based on this fact whenever the task is performed by awareness of the type of service, each resource services to the tasks suitable to itself. This condition reduces the execution time and prevents reiterating the allocation. So the execution of all the tasks comes to successfully and no task need to re-execution; and in turn it can reduce the execution time. So our proposed method prevents reiterating the allocation and as a result it can reduce the makespan according to the following items: resources capabilities, the awareness of resource service, the survey of queue length prior to allocation, and correspondence of arrival rate with service time during the allocation. Of course it can be done if the existing methods in the figure haven't paid any attention to the above mentioned items. Most of the time their purpose is to allocate the tasks to the idle resources without paying attention to the resource capabilities. This can lead to defeat the tasks execution and increase the makespan.

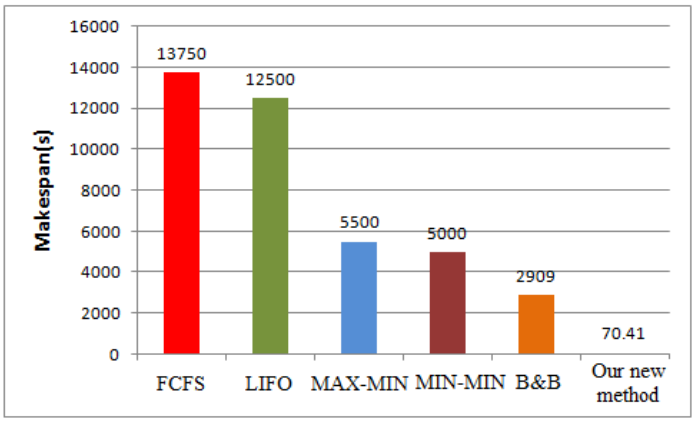

Fig. 6. Maximum makespan comparison between methods.

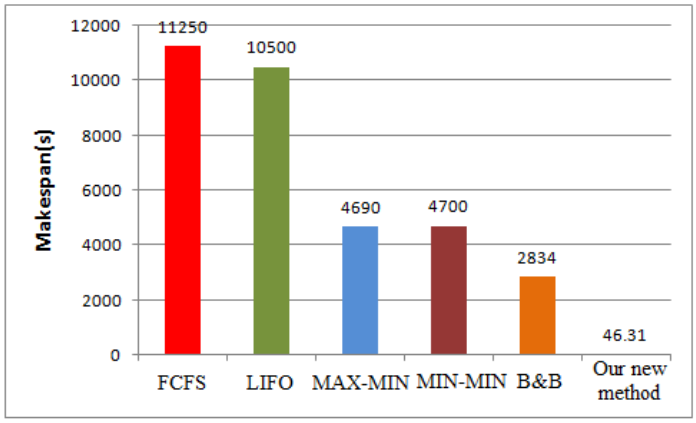

Fig. 7. Minimum makespan comparison between methods.

In continuation we calculate the total execution time for 40 tasks. The result is shown in Fig.8. In this figure, axis x shows the number of tasks and y axis shows total execution time. This figure shows that the maximum total execution time is 40.69 seconds and the minimum is 12.37 seconds. It is clear that with the increasing number of tasks total execution time increases.

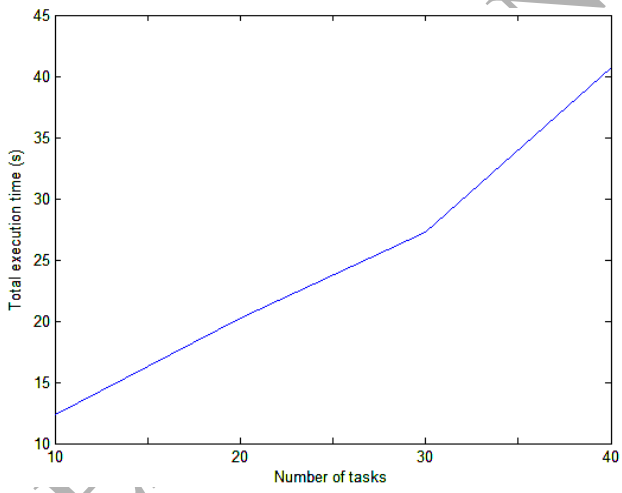

Fig. 8. Total execution time in our method.

In the last step of simulation, we calculate the total cost. Fig.9 shows this. In this figure, axis x shows the number of tasks and y axis shows cost. This figure shows that the maximum cost for execution the user tasks is 337.71$ and the minimum is 290.52$. According to the figure by increasing the number of tasks, the cost also increases. The results obtained from Cloudsim are similar to EC2. Finally, we conclude that our proposed method can improve the mentioned methods and reduce the execution and tardiness time.

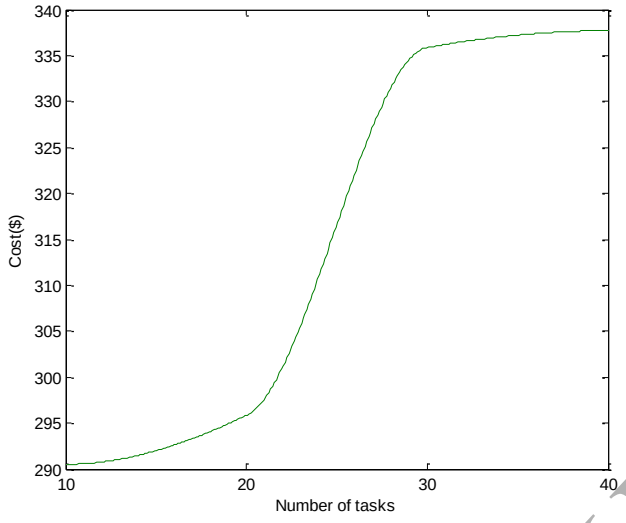

Fig. 9. Total cost of our proposed method.

Comment [M1]: Based on the simulation, the increases in accordance to the number of the tasks. In fact, the cost increases when the number of the tasks increase from 10 to 15. There is no reduction in this interval. Th reduction is because of initial adjustment of "plot fits" on the "cubic option" in the Matlab environment. This adjustment had changed the values to make the cubic figure and i had caused confusion. We re-drew the figure by adjusting the "plot fits" in the normal state. And we changed it with the previous figure, and in this way we removed the previous ambiguity.

5. Conclusion

In this paper we proposed a new method for load balancing in the Expert Clouds. This algorithm not only balances the load but also pays attention to effective task allocation. It allocates each task to the HR with the highest power and the least load based on the mathematical model. This method distributes the dynamic load based on distributed queues aware of service quality in the Cloud environment.

In this algorithm, it has been used from the colorful ants inspired of the nature for HRs ranking and making a distinction between their capabilities. The HRs are set in the sites with tree structure and their specifications are shown in the form of the three-section label. The tasks are labeled as well. The HRs allocation is done by the super-peer levels. These levels perform the mapping based on Poisson and exponential distribution. probability mass function. The proposed method improves the throughput by effective tasks allocation and also reduces the tardiness. We compare our proposed method with some of the existing techniques. The results show that the proposed algorithm in the distributed system such as Cloud works well. It improves the makespan and cost in comparison with the existing methods.

In the future, we plan to extend this type of load balancing for dependent tasks and consider more factors such as fault tolerance for HR label. Solving the single point of failure problem which exists in our method as a challenge can be considered in the future as well. The estimation of the λ and Ts can be calculated by the use of likelihood and interval estimation methods instead of point estimation. Using another mathematical function, the probability and also genetic algorithm to find the best HR are among the methods which we can deal with in the future.

References

[1]. Jack-Ide. I.O., et al. (2013)."Mental health care policy environment in Rivers State: experiences of mental health nurses providing mental health care services in neuro-psychiatric hospital, Port Harcourt, Nigeria", International Journal of Mental Health systems, 7(1): 1-9.

[2]. Mell, P., et al. (2011). "The NIST Definition of Cloud Computing", National Institute of Standards and Technology, sp. 800-145: 1-7.

[3]. Chaczko, Z., et al. (2011). "Availability and load balancing in cloud computing", international conference on computer and software modeling 14.

[4]. Babu, D., et al. (2013). "Honey bee behavior inspired load balancing of tasks in cloud computing environments", Applied Soft Computing**13**(5) : 2292-2303

[5]. Yan, K.Q., et al. (2007). "A hybrid load balancing policy underlying grid computing environment", Computer Standards & Interfaces, **29**(2) 161-173.

[6]. Sidhu, A.K., et al. (2013). "Analysis of Load Balancing Techniques in Cloud Computing", International Journal of Computers & Technology, **4**(2): 2277-3061.

[7]. Sotomayor, B., et al. (2009). "Virtual infrastructure management in private and hybrid clouds", Internet Computing, IEEE, **13**(5):14-22.

[8]. Al Nuaimi, K., et al. (2012). "A Survey of Load Balancing in Cloud Computing: Challenges and Algorithms", IEEE Second Symposium on Network Cloud Computing and Applications.

[9]. Sang, A., et al. (2008). "Coordinated load balancing, handoff/cell-site selection, and scheduling in multi-cell packet data systems", in Wireless Networks, **14**(1), pp. 103-120.

[10]. Ni, J., (2011). "Virtual machine mapping policy based on load balancing in private cloud environment", International Conference on Cloud and Service Computing (CSC), IEEE, (2011) 292-295.

[11]. Wang , S-C ., et al (2010). "Towards a load balancing in a three-level cloud computing network", 3rd International Conference on Computer Science and Information Technology (ICCSIT), IEEE.

[12]. Nishant, K., et al. (2012). "Load Balancing of Nodes in Cloud Using Ant Colony Optimization", 14th International Conference on Computer Modeling and Simulation (UKSim), IEEE,(2012)3-8.

[13]. Cao, J., et al. (2005) "Grid load balancing using intelligent agents", Future Generation Computer Systems **21**(1), pp.135–149.

[14]. Al-Jaroodi, J., et al. (2011). "DDFTP: Dual-Direction FTP, 11th IEEE/ACM International Symposium on Cluster", Cloud and Grid Computing (CCGrid), IEEE, (2011)504-503.

[15]. Ramezani, F., et al. (2014). "Task-Based System Load Balancing in Cloud Computing Using Particle Swarm Optimization". International Journal of Parallel Programming, **42**(5), pp.739-754.

[16]. Pathan , A.F., et al. (2014). "A Load Balancing Model Based on Cloud Partitioning for the Public Cloud", International Journal of Information & Computation Technology, 4(2014),pp. 1605-1610.

[17] Chen, S.-L., et al. (2016). "CLB: A novel load balancing architecture and algorithm for cloud services." , In Press, Corrected Proof Computers & Electrical Engineering, 2016.

[18]. De Falco, I., et al. (2015). "Extremal Optimization applied to load balancing in execution of distributed programs.", in Applied Soft Computing, 2015. 30, pp. 501-513.

[19].Wang, Z., et al. (2015). "Workload balancing and adaptive resource management for the swift storage system on cloud." , in Future Generation Computer Systems, 2015. 51, pp.120-131.

[20]. Daraghmi, E. Y., et al. (2015)."A small world based overlay network for improving dynamic load-balancing.", in Journal of Systems and Software, 2015. 107, pp.187-203.

[21]. Ran, Y., et al. (2015). "Dynamic IaaS Computing Resource Provisioning Strategy with QoS Constraint," IEEE Transactions on Services Computing, PrePrints, 2015, no. 1, pp. 1.

[22] Hu, H., et al. (2016). "Joint Content Replication and Request Routing for Social Video Distribution Over Cloud CDN: A Community Clustering Method," in IEEE Transactions on Circuits and Systems for Video Technology, 2016. **26**(7), pp. 1320-1333.

[23]. Gao, Guanyu., et al. (2016). "Dynamic Resource Provisioning with QoS Guarantee for Video Transcoding in Online Video Sharing Service", Proceedings of the 2016 ACM on Multimedia Conference, pp. 868-877.

[24]. J. Dean., et al. (200). "Mapreduce: Simplified data processing on large clusters", Commun. ACM, **51**(1), pp. 107-113.

[25]. Hu, H., et al. (2014). " Toward Scalable Systems for Big Data Analytics: A Technology Tutorial", IEEE Access, pp.652- 687.

[26]. Good, I ., (1986). "Some statistical applications of Poissons work", Statistical Science, 1986, **1**(2): pp. 174-176.

[27]. Papoulis, A., (1984). "Poisson Process and Shot Noise" Ch. 16 in Probability, Random Variables, and Stochastic Processes, 2nd ed. New York: McGraw-Hill, pp. 554-576.

[28]. Ross, S.M., (2009), "Introduction to probability and statistics for engineers and scientists" (4th ed.), Associated Press, pp. 267.

[29]. Khanli, L.M ., et al. (2012). "A new step toward load balancing based on competency rank and transitional phases in Grid networks", Future Generation Computer Systems, **28**(4): pp. 682-688.

Author Biography

Shiva Razzaghzadeh received her B.S. degree in Computer Engineering (hardware branch) from the Islamic Azad University—Ardabil Branch, Iran, in 2008 and M.S. in the Computer Architecture from Islamic Azad University— Tabriz Branch, Iran, in2011. She is Ph.D. student in computer engineering at IAU University. Her current research interests include distributed systems, Load balancing and resource management in Clod and Grid Networks and Routing in Grid Networks.

Ahmad Habibizad Navin was born in 1971. He received his H.N.D. in electronic in1997 and B.Sc. degree in applied mathematics from Tabriz University, Tabriz, Iran, in 1999. He received his M.Sc. degree in computer architecture from Science and Research Branch of Islamic Azad

University, Tehran, Iran, in 2003 and his Ph.D. in computer engineering from Science and Research Branch, Islamic Azad University, Tehran, Iran, in 2007. His research interest includes computer architecture, data-oriented approach, robotic, soft computing and probability and statistic.

Amir Masoud Rahmani received his B.S. in computer engineering from Amir Kabir University, Tehran, in 1996, the M.S. in computer engineering from Sharif University of Technology, Tehran, in 1998 and the Ph.D. degree in computer engineering from IAU University, Tehran, in 2005. He is the postdoctoral researcher at the University of Algarve. He also is associate professor in the Department of Computer and Mechatronics Engineering at the IAU University. He is the author/co-author of more than 80 publications in technical journals and conferences. He served on the program committees of several national and international conferences. His research interests are in the areas of distributed systems, ad hoc and sensor wireless networks, scheduling algorithms and evolutionary computing.

Mehdi Hosseinzadeh was born in Dezfulin 1981. Re-ceived his B.Sc. in Computer Hardware Engineering from Islamic Azad University, Dezful Branch, in 2003. He also received the M.Sc. and Ph.D. degrees in Computer Systems Architecture from Science and Research Branch, Islamic Azad University, Tehran, Iran in 2005 and 2008, respectively. He is currently Assistant Professor in Department of Computer Engineering of Science and Research Branch of Islamic Azad University, Tehran, Iran. His research interests are computer arithmetic with emphasis on residue number system, cryptography, network security and e-commerce.