**Journal of Computer Science 10 (7): 1156-1165, 2014**
ISSN: 1549-3636

doi:10.3844/jcssp.2014.1156.1165 Published Online 10 (7) 2014 (http://www.thescipub.com/jcs.toc)

# A LOAD BALANCING MODEL USING FIREFLY ALGORITHM IN CLOUD COMPUTING

**[1]A. Paulin Florence and [2]V. Shanthi**

[1]Research Scholar, Department of CSE, Sathyabama University, Chennai, India and
[1,2]Department of Computer Applications, St. Josephâ€™s College of Engineering, Chennai, India

Received 2013-10-07; Revised 2014-01-04; Accepted 2014-02-15

## ABSTRACT

Cloud computing is a model that points at streamlining the on-demand provisioning of software, hardware and data as services and providing end-users with flexible and scalable services accessible through the Internet. The main objective of the proposed approach is to maximize the resource utilization and provide a good balanced load among all the resources in cloud servers. Initially, a load model of every resource will be derived based on several factors such as, memory usage, processing time and access rate. Based on the newly derived load index, the current load will be computed for all the resources shared in virtual machine of cloud servers. Once the load index is computed for all the resources, load balancing operation will be initiated to effectively use the resources dynamically with the process of assigning resources to the corresponding node to reduce the load value. So, assigning of resources to proper nodes is an optimal distribution problem so that many optimization algorithms such as genetic algorithm and modified genetic algorithm are utilized for load balancing. These algorithms are not much effective in providing the neighbour solutions since it does not overcome exploration and exploration problem. So, utilizing the effective optimization procedure instead of genetic algorithm can lead to better load balancing since it is a traditional and old algorithm. Accordingly, I have planned to utilize a recent optimization algorithm, called firefly algorithm to do the load balancing operation in our proposed work. At first, the index table will be maintained by considering the availability of virtual servers and sequence of request. Then, load index will be computed based on the newly derived formulae. Based on load index, load balancing operation will be carried out using firefly algorithm. The performance analysis produced expected results and thus proved the proposed approach is efficient in optimizing schedules by balancing the loads. The average time obtained for the proposed approach is 0.934 ms.

**Keywords:** Cloud Computing, Optimization, Load Balancing, Firefly Algorithm

## 1. INTRODUCTION

Cloud computing points at the streamlining on the requiremnt on provisioning of software, hardware and data as amenities and supplying end-users with supple and scalable services available through the Internet (Armbrust *et al*., 2009). The cloud computing is assumed to facilitate computing as the value to meet the everyday needs of the community (Buyya *et al*., 2009). A cloud computing is theoretically a dispersed and flexible system where properties will be spread through the network (Cloud). The total resources of the system must be united to reply to a client request, which requires intercommunication through numerous components of the system to design a component or subset of components to deal with the request. This can lead to tailbacks in the network and an excessive charge in a distributed system where some modules

**Corresponding Author:** A. Paulin Florence, Research Scholar, Department of CSE, Sathyabama University, Chennai, India and Associate Professor, Department of Computer Applications, St. Josephâ€™s College of Engineering, Chennai, India

will be overcharged while others will be not or less charged (Khiyaita *et al*., 2012). To solve this problem, load balancing technique is commonly developed and used by recent researchers.

However there is a magnificent future of Cloud Computing, so much of important complications still need to be solved for the realization of cloud computing. Load balancing is triggered as one major problem among those; it shows a very significant role in the realization of Cloud Computing (Mondal *et al*., 2012). Load Balancing is a procedure to mark operational resource consumption by reallocating the total load to the discrete nodes of the joint organization and to recover the response time of the job. For emerging tactics for load balancing, the imporatnt points to be considered are approximation of load, assessment of load, stability of different system, performance of system, interaction between the nodes, nature of work to be transferred, selecting of nodes (Alakeel, 2010). This load measured can be associated as CPU load, amount of memory used, delay or Network load (Mondal *et al*., 2012).

In swift, the load balancing mechanism in cloud computing settings requires three conditions as follows: (i) on the condition that the local load case is not hefty, it can make the local self-organization in order to reduce the information exchange (ii) The cloud computing balancing system could be utilized in the heterogeneous environment. (iii) In order not to affect the system average response time, cloud computing load balancing mechanism should increase the system throughput as much as possible (Yao and He, 2012). To solve those issues, a substantial amount of work has been performed recently on load balancing among the nodes in a dynamic network. load balancer based load balancing (Ranjan *et al*., 2010) and optimization algorithm-based load balancing (Kotoulas *et al*., 2010; Mondal *et al*., 2012; Xin *et al*., 2012).

In this study, a load balancing based scheduling is derived for every resource based on several factors such as, memory usage, processing time and access rate. Based on the newly derived load index, the current load will be computed for all the resources shared in virtual machine of cloud servers. Once the load index is computed for all the resources, load balancing operation will be initiated to effectively use the resources dynamically with the process of assigning resources to the corresponding node to reduce the load value. So, assigning of resources to proper nodes is an optimal distribution problem so that many optimization algorithms such as genetic algorithm and modified genetic algorithm are utilized for load balancing. These algorithms are not much effective in providing the neighbour solutions since it does not overcome exploration and exploration problem. So, utilizing the effective optimization procedure instead of genetic algorithm can lead to better load balancing since it is a traditional and old algorithm. Accordingly, I have planned to utilize a recent optimization algorithm, called firefly algorithm to do the load balancing operation in our proposed work. At first, the index table will be maintained by considering the availability of virtual servers and sequence of request. Then, load index will be computed based on the newly derived formulae. Based on load index, load balancing operation will be carried out using firefly algorithm.

The main contributions of the approach:

- A scheduling method is designed by giving importance to balancing loads on nodes
- A firefly algorithm is used to optimize the scheduling queue

The rest of the study is organized as the second section contains the literature review of some recent researches on cloud computing. The problem description is given in the section three. The fourth section gives the overview of load balancing in cloud computing and firefly algorithm. In the fifth section, complete review of the proposed approach is given and the experimentation conducted on the proposed approach is plotted in section six. The conclusion of the research is plotted in the seventh section.

## 1.1. Related Researchers: A Brief Review

Despite a plenty of works available in the literature, a handful of significant research works are reviewed here. Initially, Min *et al.* (2010) have presented an adaptive load balancing optimization scheduling, which means that the cluster system select and distribute the server not only take the previous performance and the present performance of the system into account, but also estimate the future load of the system before these concurrent requests are distributed. A mathematical model of load balancing was improved and an adaptive load balancing optimization scheduling based on genetic algorithm was presented, analyzed and simulated. Empirical results showed that the algorithm was reduced effectively the average execution time of all requests and speed up the average response time.

Mohamed *et al*. (2011) have introduced an innovative method to parallelize file downloads from multiple

replicas on Cloud servers without adding high coordination overhead. DDFTP used the concept of processing the files two different directions. Consider there were two replicas of a file on two servers one server will send blocks strat initially from the file, while the second will start from the end. They are nonstop until they met and the client then asked them both to stop. This basically allowed for automatic load balancing allowing each server to work based on its conditions yet both finishing at the same time. The method was extended to apply to more than two servers and we used several experiments to show the performance gains of DDFTP using two, four and eight servers and compared it to regular FTP, concurrent FTP and DADTM on a similar number of servers under the same conditions.

Mondal *et al*. (2012) have developed a soft computing based load balancing approach. A local optimization approach Stochastic Hill climbing was used for allocation of incoming jobs to the servers or Virtual Machines (VMs). Performance of the algorithm was analyzed both qualitatively and quantitatively using Cloud Analyst. Cloud Analyst was a CloudSim-based Visual Modeller for analyzing cloud computing environments and applications. A comparison was also made with Round Robin and First Come First Serve (FCFS) algorithms.

Dong *et al*. (2012) have presented a dynamic and Adaptive Load Balancing algorithm (SALB) which was totally based on a distributed architecture. In order to understand the network transmission, SALB on the one hand assumed an adaptively adjusted load collection threshold in order to reduce the message swapped for load collection and on the other hand it employed an on-line load estimation model with a view to reducing the decision delay caused by the network transmission latency. Moreover, SALB employed an optimization model for selecting the migration candidates so as to balance the benefits and the side-effects of each dynamic file migration.

Ardagna *et al*. (2012) have developed capacity allocation algorithms able to coordinate multiple distributed resource controllers operating in geographically distributed cloud sites. Capacity allocation solutions were integrated with a load redirection mechanism which, when necessary, distributes incoming requests among different sites. The overall goal was to minimize the costs of allocated resources in terms of virtual machines, while guaranteeing SLA constraints expressed as a threshold on the average response time. They proposed a distributed solution which integrates workload prediction and distributed non-linear optimization techniques. Experiments showed how the proposed solutions improved other heuristics proposed in literature without penalizing SLAs and their results were close to the global optimum which was obtained by an oracle with a perfect knowledge about the future offered load.

Xin *et al*. (2012) have developed a model program which was multi-objective optimization and load balancing of cloud resource schedule. They have taken real-time load parameters (CPU occupancy rate, memory occupancy rate, network bandwidth, the process occupancy rate, service response time) from the server cluster nodes as decision variables of resources scheduling model and used the improved adaptive genetic algorithm to search the optimal solution, in order to realize the load balancing scheduling of cloud resource and make the each index change smoothly. The experimental result showed that, using the improved load balancing scheduling strategy to solve the problems of the load balancing scheduling of cloud resource, not only makes the each index of the system change smoothly, but also improved the performance of the system efficiently.

## 1.2. Problem Description

A cloud computing is conceptually a distributed and elastic system where resources will be distributed through the network (Cloud). The full resources of the system must cooperate to respond to a client request which requires intercommunication between various components of the system to design a component or subset of components to deal with the request. This can lead to bottlenecks in the network and an imbalanced charge in a distributed system where some components will be overcharged while others will be not or light charged. Among the major challenges faced by the cloud computing systems, load balancing is one tedious challenge that plays a very important role in the realization of Cloud Computing. For developing strategy for load balancing, the main points to be considered are estimation of load, comparison of load, stability of different system, performance of system, interaction between the nodes, nature of work to be transferred, selecting of nodes. So, in order to tackle the problems related to load balancing in the cloud network, an effective system should be developed.

The primary intention of the proposed approach is to design a load balancing model using firefly algorithm. Here, a main objective is to maximize the resource utilization and provide a good balanced load among all the resources in cloud servers. Initially, a

load model of every resource will be derived based on several factors such as, memory usage, processing time and access rate. Based on the newly derived load index, the current load will be computed for all the resources shared in virtual machine of cloud servers. Once the load index is computed for all the resources, load balancing operation will be initiated to effectively use the resources dynamically with the process of assigning resources to the corresponding node to reduce the load value. So, assigning of resources to proper nodes is an optimal distribution problem so that many optimization algorithms such as genetic algorithm and modified genetic algorithm are utilized for load balancing. These algorithms are not much effective in providing the neighbour solutions since it does not overcome exploration and exploration problem. So, utilizing the effective optimization procedure instead of genetic algorithm can lead to better load balancing since it is a traditional and old algorithm. Accordingly, I have planned to utilize a recent optimization algorithm, called firefly algorithm to do the load balancing operation in our proposed work. At first, the index table will be maintained by considering the availability of virtual servers and sequence of request. Then, load index will be computed based on the newly derived formulae. Based on load index, load balancing operation will be carried out using firefly algorithm.

### 1.3. Load Balancing in Cloud Computing Network

In the cloud computing environment, the cloud resource scheduling controller, through the changing load of real-time monitor servers nodes, assigns resources to the corresponding node dynamically to meet some needs, such as high utilization of resources, excellent performance and fast response. The topology of the server group nodes that involved in the cloud resources scheduling can be described in the following **Fig. 1**.

The above topology can be described as an G (V,E) undirected graph, G represents the node set in the chart, E represents the set of connection between these nodes. $C_i$ represents cluster control server node. $CN_i$ represents node controller, which is a separate physical machine. Each $C_i$ node manages m node controller $CN_i$, you can run k virtual machines on each $CN_i$ and use $V_i$ to represent virtual machine, So the node which will be mentioned in this study is $V_i$. In the cloud service environment, application, DBMS and other software are deployed in several $V_i$ nodes to run. Difficult issues of cloud resource scheduling need to be addressed is, when the user requests the use of cloud resources services, cloud resources scheduling controller should adopt what kind of resources scheduling model and strategy, this makes each node of the whole system deal with tasks in load balance way and prevent scheduling tilt of the cloud services platform system resources.

In the cloud computing system, when the user makes a request, the cloud controller receives the request and then it will arrange the request in the queue of the server cluster. There have more than one cluster controller below the cloud controller, cluster controllers real-time monitor the running load parameters of every virtual resource pool of this cluster, such as CPU occupancy rate, memory occupancy rate, the network bandwidth and process occupancy rate and so on, Therefore, this is a multi-objective problem which searches for the optimal solution. According to monitoring the multi-objective parameter, cluster controller will calculate the fitness of each $V_i$ node, then for the whole cluster, use Improved Adaptive Genetic Algorithm to search the best solution in the multi-objective question, which is the „lighter" node by use algorithm to solve. Finally, according to cloud resource scheduling strategy of load balancing, service resource will be assigned to the best server node.

### 1.4. Firefly Algorithm

Now we can idealize some of the flashing characteristics of fireflies so as to develop firefly-inspired algorithms. For simplicity in describing our new Firefly Algorithm (FA), we now use the following three idealized rules: (1) all fireflies are unisex so that one firefly will be attracted to other fireflies regardless of their sex; (2) Attractiveness is proportional to their brightness, thus for any two flashing fireflies, the less brighter one will move towards the brighter one. The attractiveness is proportional to the brightness and they both decrease as their distance increases. If there is no brighter one than a particular firefly, it will move randomly; (3) The brightness of a firefly is affected or determinedby the landscape of the objective function. For a maximization problem, the brightness can simply be proportional to the value of the objective function. Other forms of brightness can be defined in a similar way to the fitness function in genetic algorithms. Based on these three rules, the basic steps of the Firefly Algorithm (FA) can be summarized as the pseudo code and shown below. In certain sense, there is some conceptual similarity between the firefly algorithms and the Bacterial Foraging Algorithm (BFA) (Qiao and Bochmann, 2009).

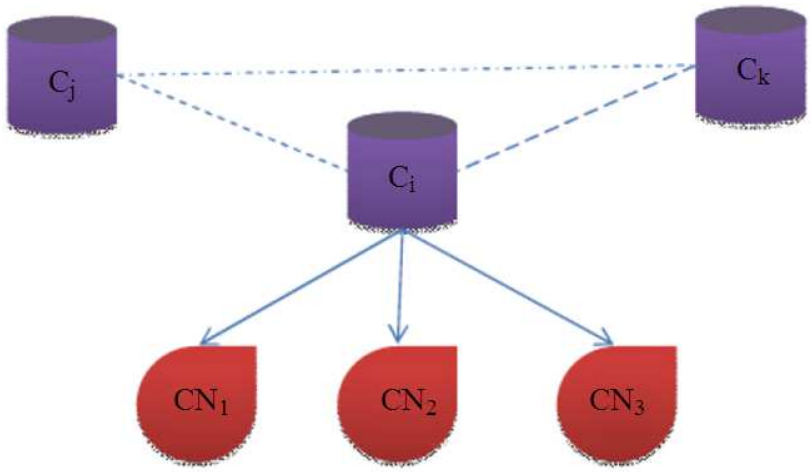


**Fig. 1.** A simple cloud system

In BFA, the attraction among bacteria is based partly on their fitness and partly on their distance, while in FA, the attractiveness is linked to their objective function and monotonic decay of the attractiveness with distance. However, the agentsin FA have adjustable visibility and more versatile in attractiveness variations, which usually leads to higher mobility and thus the search space is explored more efficiently.

Objective function f(x), x(x1,…,xd)T
Generate initial population of fireflies xi (i = 1,2,…,n)
Light intensity Ii at xi si determined by f(xi)
Define light absorption coefficient
while (t< Max_Generation)
fori = 1: n all n fireflies
forj = 1: i all n fireflies
if (Ij>Ii), move firefly i towards j in d-dimension; end if
Attractiveness varies with distance r via exp [-r]
Evaluate solutions and update light intensity
end for j
end for i
Rank the fireflies and find the current best
end while
Postprocess results and visualization

The peculiar characteristic of the firefly algorithm is adapted in the proposed approach to consider the firefly algorithm as a load balance scheduling algorithm. The attractiveness defined in the firefly algorithm helps to generate scheduling index and the distance calculation serves to find the closely associated nodes in the cloud network.

## 1.5. Proposed Load Balance Scheduling Based on Firefly Algorithm

The proposed approach triggers a method for generating an effective load balancing strategy in the cloud network to schedule the nodes. The scheduling process is carried out by the firefly algorithm. The scheduling process is preferred to the set of nodes with least amount of load possession. In other words, the nodes with least load are preferred for engaging extra process by the cloud network. In the proposed approach, we consider virtual machine with three servers and each server contain three nodes. Each node possesses attributes defined by the scheduling process.

The **Fig. 2** shows the basic over view of the virtual machine considered by the proposed approach. The proposed approach contains three main step to generate the scheduling for the proposed virtual machine by balancing their nodes. The different step can be listed as:

- Population generation
- Scheduling index calculation
- Least node selector

## 1.6. Population Generation

The term population is termed as the group of nodes served by the cloud system as per the request from the users to the server. As the request comes, the server fetches the node, which is free and severed to the user. This process is mapped on the basis of the nine nodes and according to their processing time a scheduling list is generated. The generated scheduling list is considered as the initial population for the proposed approach.

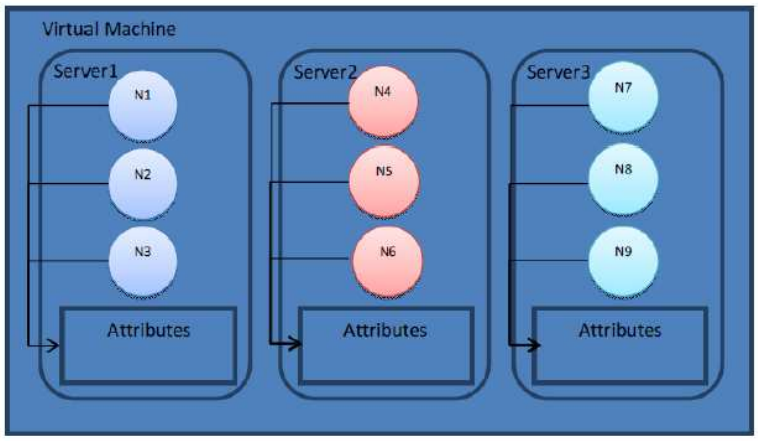


**Fig. 2.** Cloud system overview

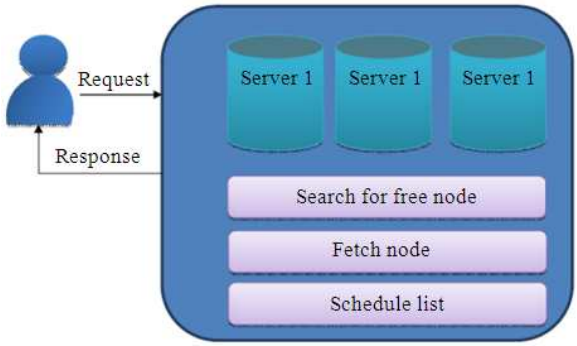


**Fig. 3.** Population generation

The **Fig. 3** shows the request and response process of the cloud system to generate the initial scheduling list. The server on receipt of a request, subjects a search among the nodes to find the free nodes. Once the node is obtained it plotted in the schedule list and when the top element of the queue is processes the free node is allocated as the response to user's request. The initial schedule list constructed by the virtual machine by processing a complete cycle based on the processing time of each nodes and their availability. The nodes list or the schedule list can be represented as table, with each row contains the nodes and the columns contains the server name.

The **Table 1** represented above contains the node list, which is considered as the initial population to the firefly algorithm. The processing in the firefly algorithms are applied in this particular population to generate an effective scheduling strategy for the cloud system by giving priority to load balancing.

**Table 1.** Initial schedule list

| Time (ms) | Server1 | | | Server2 | | | Server3 | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | N1 | N2 | N3 | N4 | N5 | N6 | N7 | N8 | N9 |
| 2 | N1 | N3 | N2 | N9 | N6 | N7 | N5 | N4 | N8 |
| … | | | | | | | | | |
| … | N3 | N9 | N1 | N4 | N7 | N6 | N5 | N8 | N2 |
| n | … | … | … | … | …. | … | … | … | … |

**Table 2.** Node and attributes

| | Attributes | | |
|---|---|---|---|
| Node | CPU rate | Memory rate | Processing time |
| N1 | C1 | M1 | P1 |
| N2 | C2 | M2 | P2 |
| …. | C3 | M3 | P3 |
| $N_n$ | $C_n$ | $M_n$ | $P_n$ |

## 1.7. Scheduling Index Calculation

The scheduling index is one of the prime factors affecting the scheduling process. So, before going into the scheduling index calculation, let us discuss about the decision parameters for the initial population. According to the definition of the firefly algorithm, there should an attraction between the fireflies i.e., the nodes. The attraction is based on the affinity possessed by node to the request. The attraction is controlled by the decision parameters defined by the proposed approach. The proposed approach defines decision parameters for each node and the parameters are considered as the attributes for the nodes. The attributes defined by the proposed approach for each node are listed as follows.

The **Table 2** represents the node and attributes defined by the proposed approach for the scheduling process.

**Table 3.** The updated schedule list

| Time (ms) | Server1 | | | Server2 | | | Server3 | | | SI |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | N1 | N2 | N3 | N4 | N5 | N6 | N7 | N8 | N9 | SI1 |
| 2 | N1 | N3 | N2 | N9 | N6 | N7 | N5 | N4 | N8 | … |
| . | … | …. | …. | …. | …. | …. | …. | …. | …. | … |
| … | N3 | N9 | N1 | N4 | N7 | N6 | N5 | N8 | N2 | … |
| n | … | … | … | … | …. | … | … | … | … | SIn |

Eventually the CPU rate (represented as C), memory rate (represented as M) and processing time (Represented as P) are also considered as the loads to the nodes. So the scheduling parameter should be designed such a way that, the node will be selected with least load weightage. Thus, according to the definitions of firefly algorithm, the equation to the attraction is defined:

$$\text{attr}\left(n_i\right) = \frac{p_i}{\text{cpu}_i + \text{mem}_i}$$

Here, attr($n_i$) represents the attraction between the node and the request as the node will be considered for the request if the attraction is high. $P_i$ represents the processing time for the particular node, $cpu_i$ represents the cpu rate of the node and $mem_i$ represents the memory rate of the nodes. The scheduling index is derived from the above formulae:

$$SI = \sum_{i=1}^{n} \frac{p_i}{\text{cpu}_i + \text{mem}_i}$$

The SI is the scheduling index and it is the total sum of the nodes in a particular scheduling queue. According to the equation all the scheduling queues in the scheduling list is calculated and again a scheduling list is formed.

The update schedule list is presented in the above **Table 3**. The updated schedule list is formulated in such way that the scheduling queue with highest scheduling index will be listed at the top of the list. The next process of the proposed approach is the calculating of least load carrying nodes.

### 1.8. Selection of Node with Minimum Load

The processes of selecting the node with least load are inspired from the firefly algorithm in such way that, the least distinct firefly will possess similar characteristics. Inspired from the theory, we subject a distance calculation between the nodes in the scheduling queues. Before proceeding to the calculation, we find the node with least attr($n_i$) values. The node with least attr($n_i$) values is considered as the pivot point for the queue to calculates least distinct nodes. The distance values of the node is calculated based on the Cartesian distance, which is given by:

$$\text{Dist} = \sqrt{\sum_{j=1}^{k}\left(n_i - n_j\right)^2}$$

The distance is presented using the expression Dist and the $n_i$ is the selected node and $n_j$ is the comparing node. Once all the distance values have been calculated between the node values, the nodes are rearranged according to the least distinct node to the pivot node. As we already sorted the schedule list based on their SI value. The top queue in the schedule list is considered as the most relevant scheduling queue.

## 2. EXPERIMENTAL RESULTS

The proposed scheduling algorithm is based on fire fly algorithm and its application in optimizing the schedule to process requests to the cloud network. The proposed approach is also concerned about balancing the loads in the cloud system. The processing of the proposed approach is explained in the above sections, now, in this section, we plot the experimental analysis of the proposed scheduling methodology by considering a simulated cloud network through CloudSim tool and java programming. A cloud simulation is constructed using the java programming as an application based on CloudSim. The experiments are conducted in a system running on core i5 processor, 4GB RAM and 500 HDD.

### 2.1. Evaluation Criteria

The main evaluation criteria used in the proposed scheduling method over cloud network is the time for executing an effective schedule queue. The main decision parameters that would be taken under consideration are the CPU utility rate and memory usage rate. The simulated dataset is evaluated based on the different parameters to assess the performance of the proposed scheduling techniques. The time is calculated based on the time required to generate an effective scheduling process.

### 2.2. Performance Evaluation

In the performance evaluation, we consider the simulated cloud network as the evaluation system. The variable parameters in the proposed evaluation section are the CPU utility rate and the memory usage rate. Thus the decision parameters mentioned are considered as the load for the nodes. So, we have to consider the performance in generating the schedule by balancing the given loads. In performance analysis, we give two types of analysis.

### 2.3. Analysis Based on CPU Utility Rate

In this analysis, we set the maximum utilization rate of the CPU in different credentials to evaluate performance of the scheduling algorithm. The analysis give an account of, how efficiently the proposed approach performs under different load levels of CPU rate.

Here, we set the maximum utilization of the CPU varying from 60 to 100 and the analysis results are depicted in the **Fig. 4**. The time and taken for scheduling for different CPU rate is represented by the line time and the memory used for the same in memory rate line. The memory rate is calculated as the utilization of the memory per 100 KB. The analysis showed that, the time for scheduling is uniform for different rate of CPU and the memory is also balanced as the rate of CPU increases.

### 2.4. Analysis Based on Memory Rate

Here, we set the maximum memory rate in different credentials to evaluate performance of the scheduling algorithm. The analysis give an account of, how efficiently the proposed approach performs under different load levels of memory rate.

We set the memory rate varying from 60 to 100 and the analysis results are plotted in the **Fig. 5**. The time and CPU usage for scheduling for different memory rate are presented by the line time and CPU rate line respectively. The CPU usage rate is calculated as the utilization of CPU per 100% utilization of CPU. The analysis showed that, the time for scheduling is uniform for different rate of memory and the CPU rate seem balanced.

### 2.5. Comparative Analysis

The above section give the performance analysis of the proposed approach based on the load balanced scheduling through firefly algorithm. To understand the significance of the proposed approach and it has to be analysed with an existing method. Here, we chose the load balancing scheduling method proposed by Xin *et al*. (2012) to evaluate the significance of the proposed approach. The comparative analysis is subjected over the CPU utility rate and memory rate of the proposed approach and existing approach. The analyses are conducted over the nodes in the particular cloud cluster centre of the simulated cloud network. The CPU rate and memory rate of a set of nodes are plotted below.

The **Fig. 6 and 7** represents the comparative analysis of the proposed approach with the existing approach. The values of the existing approach have been taken from the load balanced scheduling methods results from the method proposed by Xin *et al*. (2012). The analysis from the **Fig. 6** shows that, the proposed approach utilizes the CPU rate more efficiently than the existing approach under the load balancing condition. The analysis for the memory graph shows that the average memory utilization is less for approach by Xin *et al*. (2012) than the proposed approach. So, considering load balanced condition and the CPU utility rate the proposed approach has better efficiency over the existing approach.

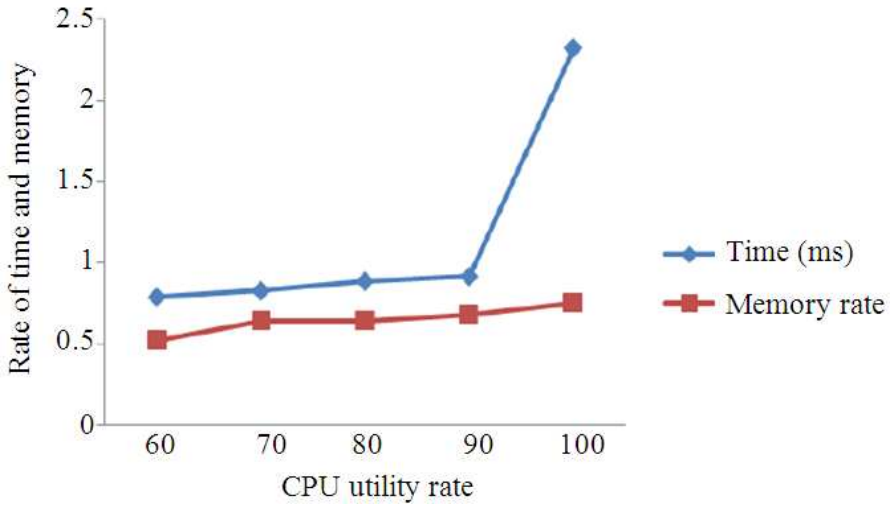


**Fig. 4.** analysis based on CPU rate

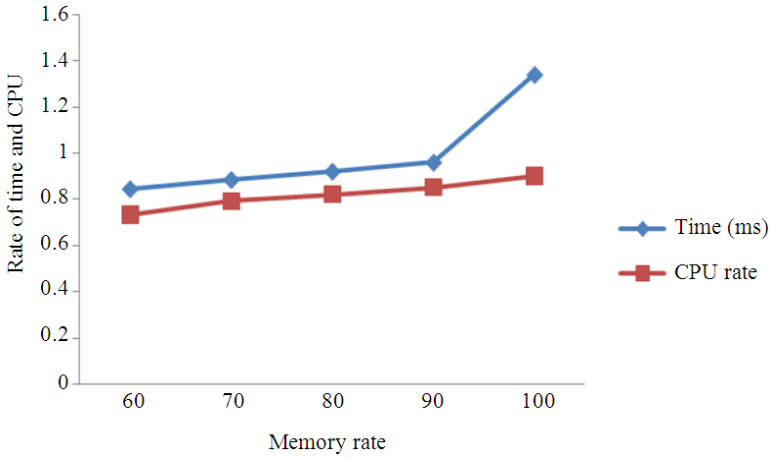


**Fig. 5.** Analysis based memory rate

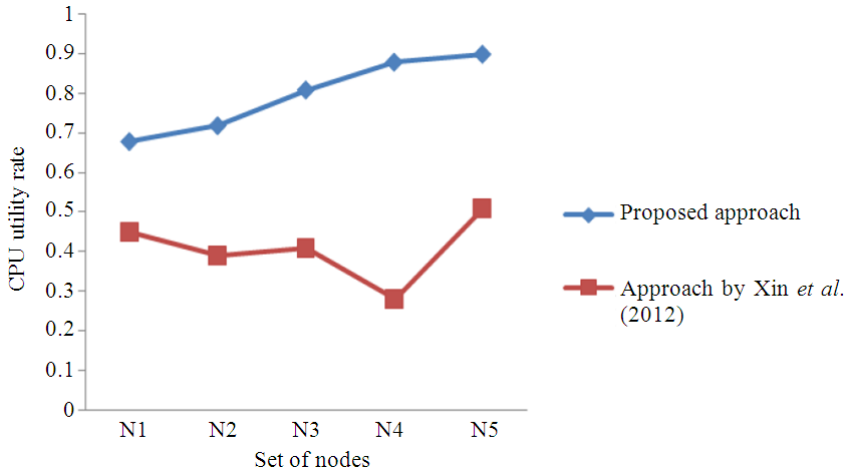


**Fig. 6.** Comparison on CPU utility rate

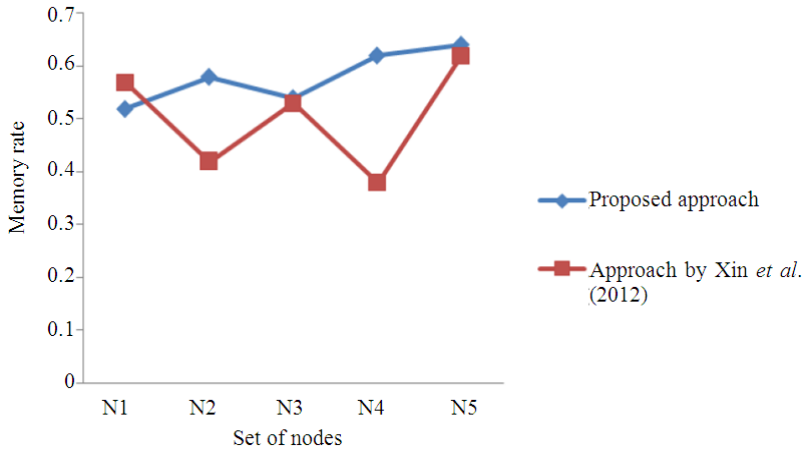


**Fig. 7.** Comparison on memory rate

## 3. CONCLUSION

In this study, we proposed a scheduling algorithm for services in a cloud network by concentrating on balancing the loads. The proposed approach deals with a simulated cloud network with set of requests and servers. The servers are associated with nodes and each node is supplied with some attributes. The attributes are assigned to control the load in each node. A load balancing based scheduling is developed for the proposed approach. The proposed approach is inspired from the firefly algorithm, because of the attracting features of the firefly algorithm. The proposed approach is developed in three steps, initially a population is generated from the cloud network, then a scheduling index calculation is subjected and finally, the schedule list is optimized using the firefly algorithm. The experimentations are conducted on the same simulated cloud network and the performances of the proposed approach are evaluated. The performance analysis produced expected results and thus proved the proposed approach is efficient in optimizing schedules by balancing the loads. The average time obtained for the proposed approach is 0.934 ms.

## 4. REFERENCES


Alakeel, A.M., 2010. A guide to dynamic load balancing in distributed computer systems. Int. J. Comput. Sci. Netw. Security, 10: 153-160.

Ardagna, D., S. Casolari, M. Colajanni and B. Panicucci, 2012. Dual time-scale distributed capacity allocation and load redirect algorithms for cloud systems. J. Parallel Distribut. Comput., 72: 796-808. DOI: 10.1016/j.jpdc.2012.02.014

Armbrust, M., A. Fox, R. Griffith, A.D. Joseph and R.H. Katz *et al*., 2009. Above the clouds: A Berkeley view of cloud computing. University of California, Berkeley.

Buyya, R., C. Yeo, S. Venugopal, J. Broberg and I. Brandic, 2009. Cloud computing and emerging it platforms: Vision, hype and reality for delivering computing as the 5th utility. Future Generat. Comput. Syst., 25: 599-616. DOI: 10.1016/j.future.2008.12.001

Dong, B., X. Li, Q. Wu, L. Xiao and L. Ruan, 2012. A dynamic and adaptive load balancing strategy for parallel file system with large-scale I/O servers. J. Parallel Distribut. Comput., 72: 1254-1268. DOI: 10.1016/j.jpdc.2012.05.006

Khiyaita, A., M. Zbakh, H. El Bakkali and D. El Kettani, 2012. Load balancing cloud computing: State of art. Proceedings of the National Days of Network Security and Systems, Apr. 20-21, IEEE Xplore Press, Marrakech, pp: 106-109. DOI: 10.1109/JNS2.2012.6249253

Kotoulas, S., E. Oren and F.V. Harmelen, 2010. Mind the data skew: Distributed inferencing by speeddating in elastic regions. Proceedings of the 19th International Conference on World Wide Web, Apr. 26-30, ACM Press, New York, USA., pp: 531-540. DOI: 10.1145/1772690.1772745

Min, J., H. Liu, A. Deng and J. Ding, 2010. Adaptive load balancing optimization scheduling based on genetic algorithm. Proceedings of the 3rd IEEE International Conference on Computer Science and Information Technology, Jul. 9-11, IEEE Xplore Press, Chengdu, pp: 81-85. DOI: 10.1109/ICCSIT.2010.5564114

Mohamed, N., J. Al-Jaroodi and A. Eid, 2011. A dual-direction technique for fast file downloads with dynamic load balancing in the cloud. Proceedings of the 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, (CGC' 11), Newport Beach, California, USA.

Mondal, B., K. Dasgupta and P. Dutta, 2012. Load balancing in cloud computing using stochastic hill climbing-a soft computing approach. Proc. Technol., 4: 783-789. DOI: 10.1016/j.protcy.2012.05.128

Qiao, Y. and G.V. Bochmann, 2009. A diffusive load balancing scheme for clustered peer-to-peer systems. Proceedings of 15th International Conference on Parallel and Distributed Systems, Dec. 8-11, IEEE Xplore Press, Shenzhen, pp: 842-847. DOI: 10.1109/ICPADS.2009.119

Ranjan, R., L. Zhao, X. Wu, A. Liu and A. Quiroz *et al*., 2010. Peer-to-Peer Cloud Provisioning: Service Discovery and Load Balancing. In: Cloud Computing: Principles, Systems and Applications, Antonopoulos, N. and L. Gillam (Eds.), Springer, London, ISBN-10: 1849962413, pp: 195-217.

Xin, L.U., J. Zhou and D. Liu, 2012. A method of cloud resource load balancing schedulingbased on improved adaptive genetic algorithm. J. Inform. Comput. Sci., 9: 4801-4809.

Yao, J. and J.H. He, 2012. Load balancing strategy of cloud computing based on artificial bee algorithm. Proceedings of the 8th International Conference on Computing Technology and Information Management, Apr. 24-26, IEEE Xplore Press, Seoul, pp: 185-189.