A Bee Colony based Multi-Objective Load Balancing Technique for Cloud Computing Environment

Ashish Soni
Research scholar
Computer Science and Engineering Department
Samrat Ashok Technological Institute Vidisha (M.P.),

Gagan Vishwakarma
Assistant Professor
Computer Science and Engineering Department
Samrat Ashok Technological Institute Vidisha (M.P.),

Yogendra Kumar Jain,
Ph.D
Head of Department
Computer Science and Engineering Department
Samrat Ashok Technological Institute Vidisha (M.P.),

ABSTRACT
With the recent development of open cloud systems a surge in outsourcing assignments from an internal server to a cloud supplier has been seen. The Cloud can facilitate its clients enormous resources hence even during heavy load conditions. Since the cloud needed to be handle multiple clients workload at same time and each client may have different resource requirements hence choosing proper resources for given workload in such a system, in any case, is a difficult problem. This paper addresses this streamlining issue in a cloud system with different client's priority groups and resource requirements and proposes a bee colony based Multi-Objective load balancing technique, to attain efficient load scheduling over virtual machines under cloud. The proposed algorithm assigns the workload on the virtual machines in such a way that it minimizes the total processing cost in cloud without sacrificing priority of tasks and load management performance.

General Terms
Cloud Computing, Load Balancing.

Keywords
Cloud Computing, Scheduling, Load Balancing, Bee Colony Optimization

1. INTRODUCTION
Cloud computing holds a guarantee to provide large scale accessibility of resources which help its client organizations to choose appropriate resources depending upon their requirements on pay per use or depending upon agreement policies. Since the client does take resources from cloud it saves the buying cost as well as it eliminates the hassle of maintenance and placement. The model's appeal organizations essentially because of the flexibility it provides. Because of such advantages the number of suppliers conveying IT Infrastructure as a Service (IaaS) has expanded rapidly in last few years [1]. For example, Amazon, one of the bigger players in this field has expanded the quantity of cloud sorts from one to eight in under three years. Each one cloud system separates itself from the others in terms of value, number of virtual machines, accessible memory furthermore I/O data transfer capacity and costs charged for distinctive assets (system, memory, CPU) and performance. Nonetheless, in spite of these benefits, previous research has demonstrated that there is a confound between characteristics of cloud environment and clients necessities, especially for high performance computing applications which are strictly hard coupled between resources, And perform frequent communication between processor to processor or processor to other resources and needed greater synchronization [2]. The insufficient network performance because of improper load balancing or utilization of resources is a major bottleneck in cloud, and has been widely studied. These challenges gets further complicated in heterogeneity and multi-accessing environment however these consideration are not as mush studied so far. Clouds needed to evolve its configuration in processors, memory and network overtime, for properly operating in heterogeneous and multi-accessing environment [3]. The multi-accessing is also intrinsic of cloud and leads to numerous sources of interference due to frequent time sharing of CPU, cache, memory access, and their interconnections. For strictly-coupled applications which require dedicated resource requirements, heterogeneity and multi-accessing can result degradation of quality of service and even un-predictable cloud performance, since one slow resource can slow down the entire application.

2. RECENT WORK
Due to the rising interest in cloud computing every field related with it getting attention of researchers and the load balancing in cloud is one of them which are most widely studied. In this section some of them are presented. Shu-Ching Wang et al [4] presented a two-phase scheduling which combines OLB (Opportunistic Load Balancing) and LBMM (Load Balance Min-Min) scheduling algorithms for proper load balancing which give improved performance as compare to single one. A Dynamic Load Balancing approach for high performance computing in cloud is presented in [5], it provides the analysis of static hardware heterogeneity placed in virtualized environments, and also addresses the dynamic heterogeneity caused by the interference arising as a result of multi-accessing. Their proposed load balancer adapts to the dynamic variations in cloud resources by continuous live monitoring, instrumentation, and also a periodic refinement of task distribution to VMs. Ant colony based optimization approach for load balancing in cloud is discusses in [6]. In this they calculate optimal solution for achieve load balancing using Ant colony optimization. A comparison based on various parameters like performance, scalability, associated overhead etc. between existing load balancing techniques in cloud is presented in [7]. It also discusses these techniques from energy consumption and carbon emission perspective. Cost-Optimal Scheduling for Deadline Constrained Workloads in Hybrid IaaS Clouds is presented in [8], they proposed a binary integer program formulation for hybrid IaaS scheduling problem and evaluate the computational costs with respect to the problem's key parameters. While evaluating the performance they concluded that this approach provides an acceptable solution for scheduling the public cloud, but becomes much less feasible in a hybrid cloud setting due to very high solve time variances.

3. CLOUD COMPUTING

The cloud computing, or the cloud, is a term frequently used in computer science to express a computing concept that involves a number of interconnected computing resources through a real-time communication network [9]. In general, cloud computing is similar to distributed computing over a network and means the ability to run a program on many connected computers at the same time. In more practical way cloud computing is described as utilization of computing resources (hardware and software) which can be provided as a service over a network.

Cloud computing enables us to pay for computing resources what we require. These services are provided over the internet, on a consume-based pay-as-you-use model means pay only for how much users use, with short-term contracts and without other expenditure. Whether we realize it or not, we're most likely already using cloud-based services. Facebook and Google are two recognized companies offering cloud-based software as a free online service to billions of users across the world. **Google**, for example, hosts a combination of online productivity tools and applications in the cloud.

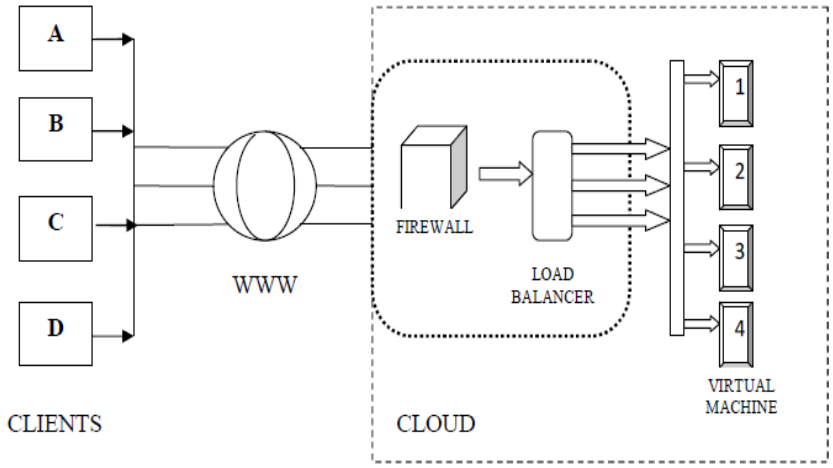

Fig 1: Cloud Computing

3.1 Load Balancing

It is a process of assigning the total load to the respective nodes of the shared system to obtain resource utilization effective. Load balancing also use to improve the response time of the job, concurrently removing a state in which some of the nodes are over loaded whereas some others are under loaded. Load balancing algorithms have two types of nature. First type is dynamic nature, it does not consider the previous state or behavior of the system, it's only depends on the present condition of the system. Second is static in nature. The important term to acknowledge while developing such types of algorithm are evaluation of load, comparison of load, stability of distinct system, performance of system, nature of work to be transferred, interaction between the nodes, selecting of nodes and many other ones [10]. This load considered can be in terms of CPU load, amount of memory used, network delay or Network load. Load balancing is a technique that manages resources of a node for their better utilization and user satisfaction. It also distributes workload evenly across two or more computers for fast processing and better performance. Load on a node can be calculated on the basis of various parameters such as cost, response time, makes span and number of connections.

3.2 Types of Load Balancing Algorithm

Static Load Balancing- In this approach of load balancing, we consider static information of system to choose the least loaded node. It performs better in terms of complexity issue but compromises with the result as decision is made on statically gathered data. It is further classified as Distributed and Centralized. In distributed static Load balancing approach, there are many decision makers but final decision is made by comprehending decision of all individuals while centralized static load balancing technique has a centralized controller that incorporates decisions of all decision makers. Distributed policies on the next level bifurcated to co-operative and non co-operative policies.

In cooperative policies decision makers cooperate with each other while making decision as they have common goal to achieve like minimization of response time, cost incurred for processing requests and maximization of throughput.

In non co-operative policies all decision makers have different objectives to achieve so they take independent decisions to reach an optimal solution for their defined goals. In global static load balancing, there is only one decision maker that optimizes the expected run time of entire system for all jobs.

Dynamic Load Balancing- In this strategy, current system state plays major role while making decisions. Despite the fact that dynamic load balancing has higher run rime complexity then static one, dynamic has better performance report as it considers current load of system for choosing next datacenter to serve the request. This will surely provide an optimal choice from available ones for that state of system.

Dynamic load balancing is classified as Centralized and distributed. In centralized policy, allocation decisions are made by the central computer that maintains a global state of system based on collected information. Flaw with this approach is the central computer which acts as bottleneck with increase in number of computers. In distributed policy each computer has its own view of global state.

Distributed load balancing approach is further classified as initiated by sender, initiated by receiver and initiated symmetrically. In sender initiated scheme, request from heavily loaded node is sent to lightly loaded node for processing. Sender is identified as a node which if accepts the next request will exceed its threshold level. In receiver

initiated scheme, lightly loaded nodes show their willingness to share some load from heavily loaded nodes by requesting them for the same. In symmetrically initiated scheme, both sender and receiver initiate load balancing process. Actually there is switching between sender initiated and receiver initiated load balancing on the basis of load behavior as it fluctuates between upper and lower threshold.

4. PROBLEM FORMULATION

Workload in the cloud is regularly a multi-objective problem. In this paper we highlighted and paid attention to some of these problem and possible solution, so as to obtain an optimal solution. We expect that every application comprises of a number of slightly parallel tasks. Every application has a strict fulfillment due time. Prior to this due time, all computational assignments in the application must be completely executed with the results conveyed to the client. Our current application model concentrates on random sort of workloads. With two different clients group one with higher resources accessing rights while other group has relatively lower resources accessing rights. A cloud supplier permits it clients to accessibility to one or more virtual machine on its foundation. The capabilities of the virtual machine on which these applications are executed are dictated by their execution capacity. Each task has a related runtime priority and execution capacity requirement by which it can be executed.

The problem formulation with mathematical modeling of the system is presented below:

Let there be two different groups of clients G_1 *and* G_2. the group G_1 clients having greater priority and required dedicated immediate resources allocation.

$$G_1 = \{c_1^1, c_2^1, c_3^1, \dots\dots\dots\dots\dots\dots, c_M^1\};$$

$$G_2 = \{c_1^2, c_2^2, c_3^2, \dots\dots\dots\dots\dots\dots, c_N^2\};$$

Where c_j^i represents the j^{th}client in i^{th} group while M and N are the maximum number of clients in each group.

At any time t the client's generated request is given by

$$g_1 \subseteq G_1, g_2 \subseteq G_2, card(g_1) = m, card(g_2) = n,$$

$$m \leq M, n \leq N$$

The request from each client can be described by the tuple of execution resources requirements and execution priority

$$R_j^i = \{W_j^i, P_j^i\}, i \in \{g_1, g_2\}, j \in g_1 \; if \; i = g_1 \; else \; j \in g_2.$$

However for g_2 clients

$$\forall j: P_j^2 = 1$$

Now the cloud is given by

$$S = \{VM_1, VM_2, \dots\dots\dots\dots\dots\dots.VM_K\}$$

Where K is the maximal number of virtual machines (VM) in cloud environment and each VM can be described by the tuple of execution capacity and execution cost.

$$VM_k = \{E_k, C_k\}, VM_k \in S$$

Now the problem can be stated as

$$\alpha = \sum_{j=1}^{m} \sum_{k \in K_1} \left|E_k - W_j^1\right| + \sum_{j=1}^{n} \sum_{k \in K_2} \left|E_k - W_j^2\right|, K_1 \subseteq K, K_2 \subseteq K, K_1 \cap K_2 = \emptyset$$

$$\beta = \sum_{k \in \{K_1, K_2\}} C_k$$

$$\gamma = \sum_{i_1=1}^{m} P_{i_1}^1 + \sum_{j_1=1}^{n} P_{j_1}^2 \; , i_1 \in \forall j \; which \; satisfy \sum_{j=1}^{m} \sum_{k \in K_1} \left|E_k - W_j^1\right| > 0,$$

$$j_1 \in \forall j \; which \; satisfy \sum_{j=1}^{n} \sum_{k \in K_2} \left|E_k - W_j^2\right| > 0,$$

Hence the objective can be expresses as to find the values of $K_1$and K_2 such that it minimizes the $\alpha + \beta + \gamma$

$$obj = \min_{K_1, K_2} (\alpha + \beta + \gamma)$$

5. ARTIFICIAL BEE COLONY OPTIMIZATION

Swarm Intelligence is the part of Artificial Intelligence based on behavior of individuals in various decentralized systems. The Bee Colony Optimization (BCO) is relatively a new member in Swarm Intelligence based Meta heuristic searching. In the algorithm each artificial bees represent agents and also a possible solution of the problem, which collaboratively solve complex combinatorial optimization problem by exchanging the information. The algorithm can be described as follows [11]:

Phase 1: Initialization

1. Set the number of bees in the hive. These bees hold information about passed parameters of all nodes

2. Set the number of productive moves during one forward pass. These moves are used for selecting optimal solution.

At the time of starting all the bees are in the hive.

Phase 2: Execution

1. For every bee estimate all possible productive moves, which are useful for next process.
2. According to estimation, select one move using the roulette wheel.
3. All bees are back to the hive with taking information about passed parameter of nodes.
4. Sort the bees by their objective function (fitness value) value;
5. Every bee concludes randomly whether to continue its own exploration by turn into a recruiter, or to a follower.

6. For every follower, select a new solution from recruiters with using roulette wheel;

Phase 3: Stopping

1. If the objective function value reach to desired value.
2. If the maximum number of iterations reached.
3. If the maximum execution time reached.

Phase 4: Output

1. Output the best result.

Honey bee behavior inspired load balancing (HBB-LB) algorithm [12]:

Cloud computing deals with assigning computational tasks on a dynamic resource pool of virtual machines online according to different requirements from user or the system. In this HBB-LB algorithm these requirements are fulfill by bee colony as we have already discussed in section 4. The service requests(R_j^i) from the clients for diverse applications must be routed to K_1, K_2 virtual machines such that it satisfies the objective function.

6. PROPOSED ALGORITHM

Previous studies shown that the scheduling algorithm for cloud systems does not performs efficiently which result the lower QoS. The Cloud network consist multiple users input with their different requirements which needed to fulfill by efficiently utilizing the available resources. There are different ways to fulfill user's requirement (Like priority). Such as,

1. To allocate overall resources on priority basis from highest priority to lowest priority.
2. To allocate overall resources on the round robin basis.
3. To allocate overall resources randomly.

Although these scheduling and resource assignment schemes could not assure perfect performance for all needs of QoS because each scheduling algorithm has own specific criteria to solve the problem statements which doesn't matches with the generalized requirements of cloud computing [12].

The proposed algorithm is developed to overcome all related problems stated above. The scheduler based on Bee Colony algorithm tunes the system for optimize performance on multi-objective requirements by optimize fitness value of Load, Priority and Execution error (VM's).

An algorithm has proposed as following steps:

Algorithm:

Start:

1. Users send a request about required resources, task priority and task size request to cloud manager;
2. Cloud Manager Store all requests.
3. Now it forms a request table with required resources, task priority and task size request, from all users.
4. Apply Bee-Colony algorithm for all entry;
5. Schedule and Assigns the channel according to Bee-Colony algorithms output.
6. Stop.

End

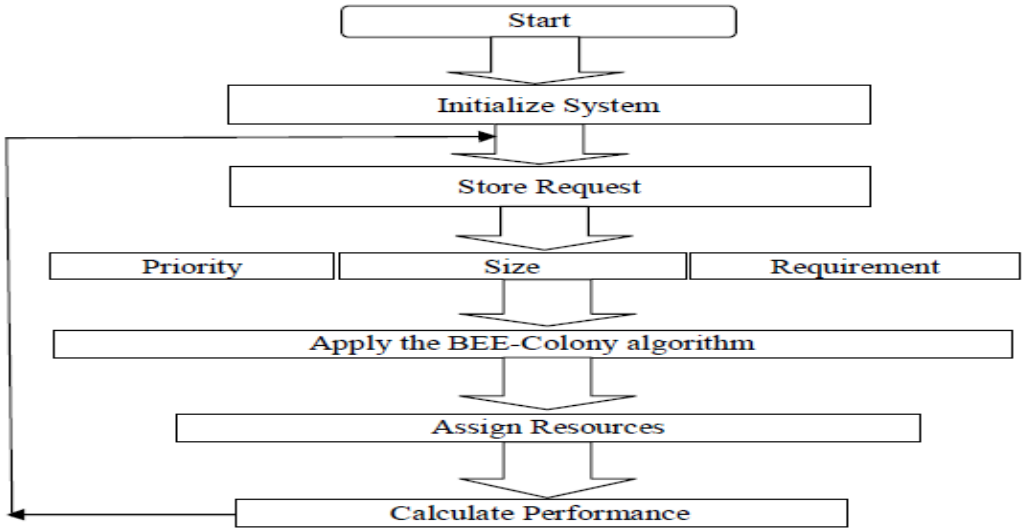

Fig 2: Flow Chart of the Proposed Algorithm

Sub Algorithm 1: This algorithm consists of step (5) of main algorithm

Start:

Scheduler divides available requests;

a.) Primary users;

b.) Secondary users;

End

In the proposed algorithm these fitness value are conclude by minimizing differences of requested load and served load, requested priority and served priority, and also minimizing total execution error. Previous algorithms are not taking account all these parameters. Using these parameters in objective function we get improved performance.

Sub Algorithm 2: This algorithm consists of step (5) of main algorithm

Start:

i. Apply Bee-Colony algorithm with required resources, task priority and task size request.

a.) form objective function which minimizes at best request serving;

b.) Apply cloud constrains (Number of VMs their Processing Capabilities and Availability).

c.) calculates fitness value;

ii. Iterate till best solution (fitness) found.

End

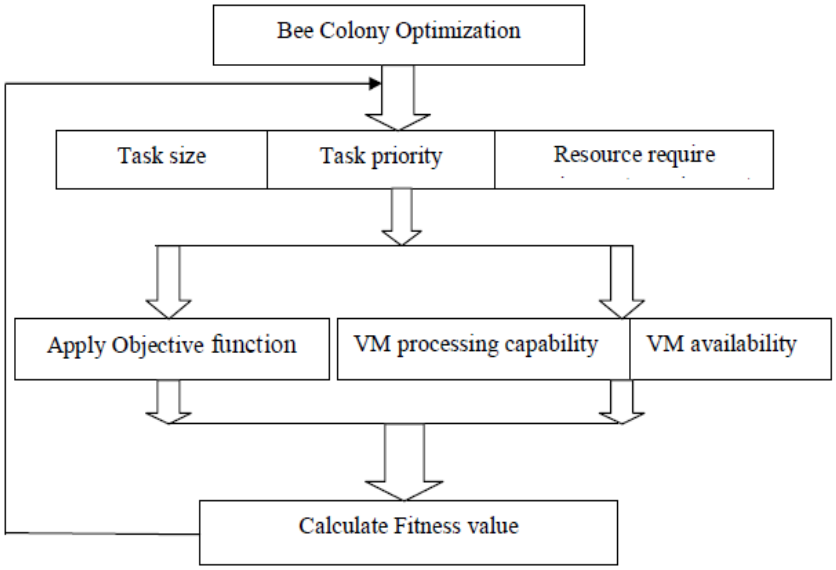

Fig 3: Flow chart of Sub Algorithm

7. SIMULATION RESULTS

The implementation and simulation of the proposed algorithm is performed using MATLAB. The simulation is executed for the configuration given in table 1 and table 2. Finally the simulation results are presented in the form of graphs.

Table 1: the cloud and users configuration

Configuration Variable	Value
Number of VMs	10
Cloud Execution Capacity	10 (MIPS)
Maximum Load (Requested)	10(MIPS)
Number of Paid Users	5
Number of Free Users	20
Request Arrival Probability	0.5
Total Simulation Time	10 Sec.

Table 2: the Bee-Colony configuration

Number of Bees	16
Number of Rounds	1000

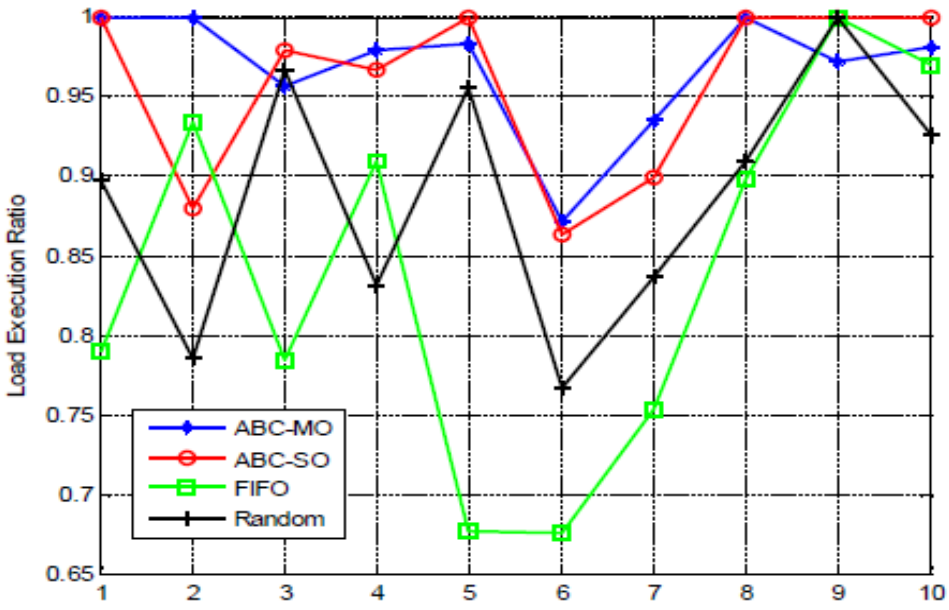

Fig 4: the Load execution ratio comparison for different techniques the graph shows that the proposed technique gives the best execution ratio.

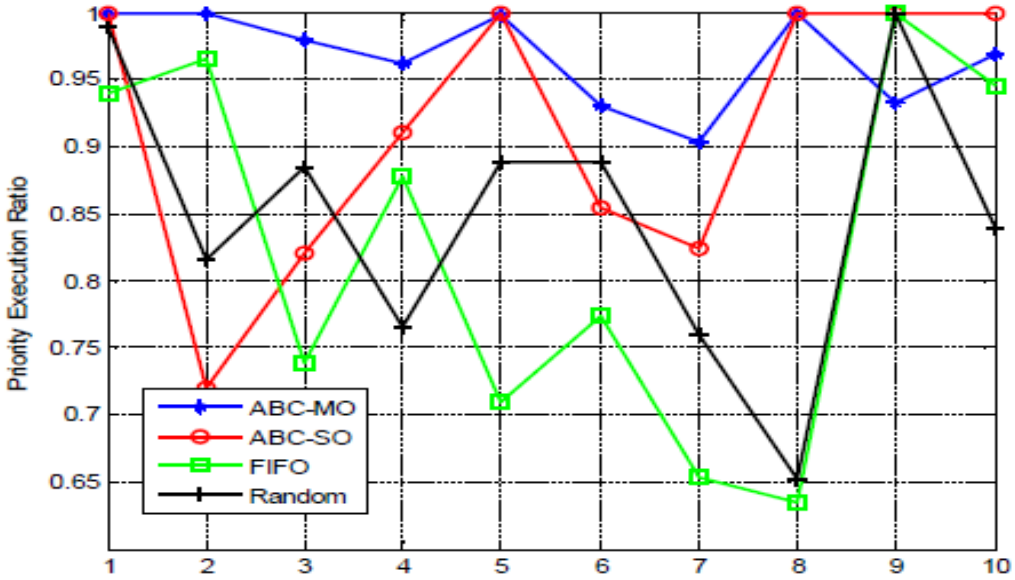

Fig 5: the priority requests execution ratio comparison for different techniques the graph shows that the proposed technique gives the best execution ratio even with best over load execution ratio (Fig (4)).

All the result shows improved performance compare to other existing Load balancing algorithm. Number of paid users shows which users have higher priority. Number of free users shows which users have lower priority. Paid users get resources first then free users. For simulate cloud environment we set Request arrival probability 0.5, which shows that at any instant of time Primary user and Secondary user request arriving probability is 50%. Maximum load 10 mips which set randomly between 0 to 10 mips value and distribute to all users. Set Cloud capacity distributes randomly to all VM. We also set execution error randomly of all VM. Figure shows the performance comparison with respect to time between Artificial Bee Colony-Multi Objective (Load, Priority and Execution Error of VM), Artificial Bee Colony -Single Objective (Load), First-in First-out (FIFO) and Random algorithm.

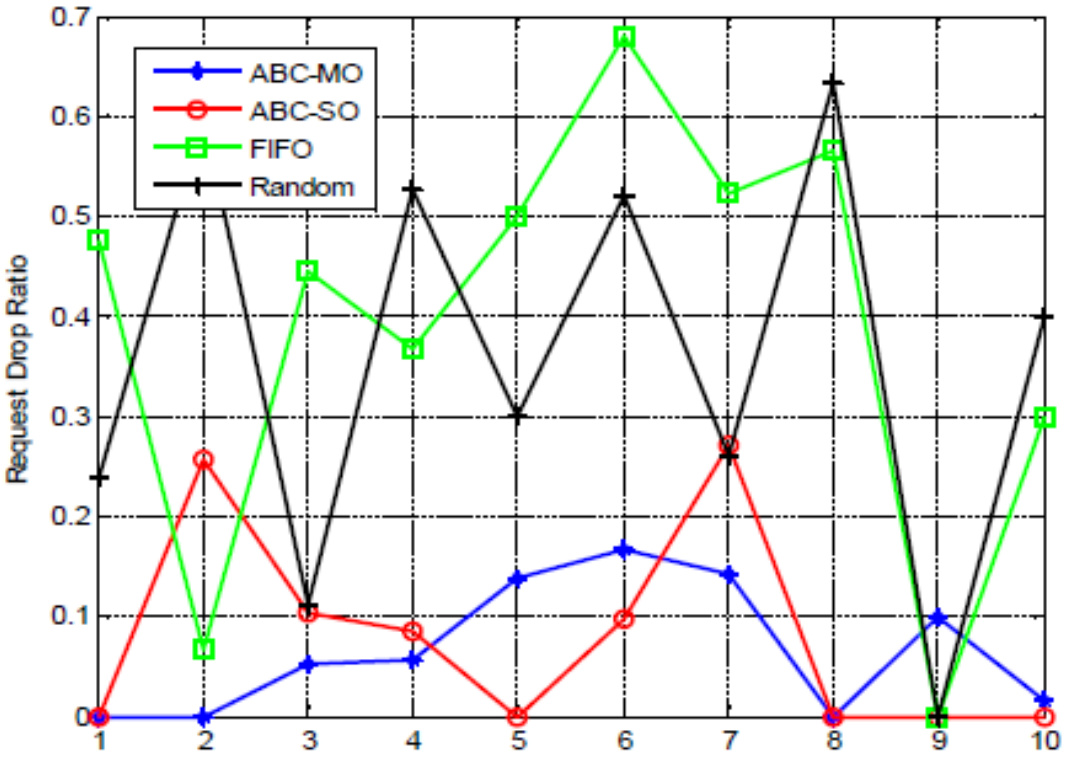

Fig 6: the graph represents the unhandled task ratio or request drop ratio for different techniques the graph shows that the proposed technique provides minimum unhandled task.

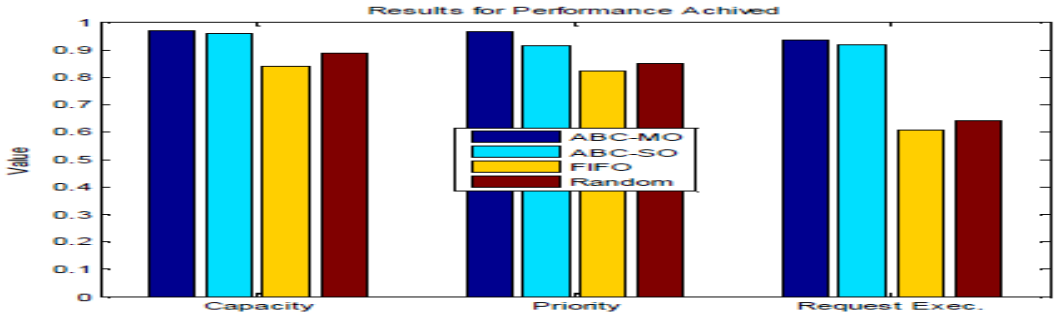

Fig 7: overall performance comparison of different techniques the graph shows that the proposed technique achieves all the objectives.

8. CONCLUSION AND FUTURE WORK

This paper presents a new optimization approach for the efficient load scheduling while maintaining some client specific objectives, under complex load conditions and the simulation result shows that it effectively utilizes the resources while minimizing latency and without compromising processing speed in cloud. In the proposed algorithm, fitness values are conclude by minimizing differences of requested load and served load, requested priority and served priority, and also minimizing total execution error. Previous algorithms are not taking account of all the parameters that are load, execution error of VM's and priority. Using these parameters in objective function we get improved performance. The simulation results also show that the proposed technique also fulfils the user specific requirements such as priority execution, and specific resource allocation. It also reduces the number of unhandled tasks during heavy load conditions. These results validates that the proposed algorithm is can provide a better solution for cloud systems. Using this algorithm we can effectively perform load balancing and QOS provisioning for Cloud environment. Since this paper considers only two groups of clients which can be further increased and the VM specifications can also be added for more detailed simulation but presently these tasks are leaved for the future work. Also, we can try to add some more parameters for load calculation to improve the results.

9. ACKNOWLEDGMENTS

I would like to thank **Mr. Gagan Vishwakarma** Assistant Professor and **Dr. Yogendra Kumar Jain,** Head, Department of Computer Science and Engineering who has contributed towards development of the template.

10. REFERENCES

[1] Ali Khajeh-Hosseini, David Greenwood, Ian Sommerville, 2010 "Cloud Migration: A Case Study of Migrating an Enterprise IT System to IaaS", Proceeding CLOUD '10 Proceedings of the IEEE 3rd International Conference on Cloud Computing Pages 450-457.

[2] Alexander Lenk, Markus Klems, Jens Nimis, Stefan Tai, Thomas Sandholm, 2009 "What's Inside the Cloud? An Architectural Map of the Cloud Landscape", Proceeding CLOUD '09 Proceedings of the ICSE Workshop on Software Engineering Challenges of Cloud Computing Pages 23-31.

[3] Rajkumar Buyya, Rajiv Ranjan, Rodrigo N. Calheiros, 2010 "InterCloud: Utility-Oriented Federation of Cloud Computing Environments for Scaling of Application Services", ProceedingICA3PP'10 Proceedings of the 10th international conference on Algorithms and Architectures for Parallel Processing - Volume Part I.

[4] Shu-Ching Wang, Kuo-Qin Yan, Wen-Pin Liao and Shun-Sheng Wang, 2010 "Towards a Load Balancing in a Three-level Cloud Computing Network", Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on (Volume:1): 9-11.

[5] Abhishek Gupta, Osman Sarood, Laxmikant V Kale, Dejan Milojicic, 2013 "Improving HPC Application Performance in Cloud through Dynamic Load Balancing", Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on13-16.

[6] Ratan Mishra and Anant Jaiswal, 2012 "Ant colony Optimization: A Solution of Load balancing in Cloud", International Journal of Web & Semantic Technology (IJWesT) Vol.3, No.2.

[7] Nidhi Jain Kansal, Inderveer Chana, 2012 "Cloud Load Balancing Techniques: A Step Towards Green Computing", IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 1, No 1, ISSN (Online): 1694-0814.

[8] Ruben Van den Bossche, Kurt Vanmechelen and Jan Broeckhove, 2010 "Cost-Optimal Scheduling in Hybrid IaaS Clouds for Deadline Constrained Workloads", IEEE 3rd International Conference on Cloud Computing.

[9] Ambika Mishra, Prof. Susheel Jain and Prof. Anurag Jain, 2014 "A Hierarchical Resource Switching and Load Assignment Algorithm for Load Balancing in Cloud System", International Journal of Scientific & Engineering Research, Volume 5, Issue 3.

[10] T. Casavant and J.G Kuhl, 1988 "Taxonomy of scheduling in general-purpose distributed computing systems", IEEE Transaction on Software Engineering, vol. 14, issue 2, pp 141-154.

[11] Dušan Teodorović, 2009 "Bee Colony Optimization (BCO)", Innovations in Swarm Intelligence Studies in Computational Intelligence Volume 248, Pages 39-60.

[12] Dhinesh Babu L.D. , P. Venkata Krishnab, 2013 "Honey bee behavior inspired load balancing of tasks in cloud computing environments", ELSEVIER Applied Soft Computing 13,Pages 2292–2303.