A Heuristic ACO Based Technique for Balancing the Load in Cloud Computing

Venkateshwarlu Velde, B. Rama, T. Sudha

Abstract—Cloud computing is associated with huge shared pool of resources with on-demand access to public in pay as you go fashion. Of late, due to plethora of benefits from cloud, individuals and organizations started using cloud in one way or other. Thus there is almost exponential growth demand for cloud services. The rationale behind this is the availability of infrastructure, software and platform services including storage and computing over Internet. There is paradigm shift in the way computing is taking place with the emergence of cloud computing technology. Task scheduling is one of the fundamental issues in cloud computing. It is associated with load balancing as well. Many meta-heuristic approaches came into existence to solve the problem of load balancing with optimized task scheduling. Task scheduling needs to be optimized to have better strategies to withstand diversified tasks and ever changing environments. In this paper we proposed an Ant Colony Optimization (ACO) based algorithm known as Enhanced Heuristic-Ant Colony Optimization (EH-ACO) for load balancing through better task scheduling. ACO is a random search approach which can solve optimization problems like load balancing by efficiently mapping incoming jobs to VMs. Key role of the paper lies in reducing makespan of given set of tasks besides balancing load. The ability to balance load is improved as the load balancing factor is associated with job finishing rate. The proposed algorithm has underlying strategy for effective scheduling of cloud tasks. The proposed algorithm is implemented using "CloudSim" toolkit. Empirical outcomes revealed that our algorithm provides comparable performance improvement over other ACO based approaches.

Index Terms – Cloud Computing, Load Balancing, Task Scheduling, Load Balancing, CloudSim, ACO, MACO, Round Robin.

1 INTRODUCTION

ACCORDING to National Institute of Standards and Technology (NIST) cloud computing is a novel model of computing with certain characteristics such as on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service. It has services models pertaining to software, infrastructure and platform and deployment models in terms of private cloud, public cloud, community cloud and hybrid cloud [11]. In other words cloud computing is a distributed Internet computing that can be integrated with Information Technology (IT) systems of organizations across the globe for obtaining on-demand services in pay-per-use fashion [12].

With respect to cloud computing, load balancing and cloud task scheduling are both related tasks that are important and fundamental for successful cloud computing ecosystem. Many researchers contributed towards making algorithms to achieve load balancing. One such algorithm is ACO which is a random search algorithm as explored in [13]. A modified ACO algorithm was proposed in [14]. From the state of the art, it is understood that balancing the in cloud computing is an optimization problem that needs further investigation.

In this paper - we proposed an ACO based optimization technique that considers load balancing factor and achieves it through optimized cloud task scheduling.

- *Venkateshwarlu Velde, Department of Computer Science, Kakatiya University, Warangal, TS, India. E-mail: veldevenkat@gmail.com.*
- *B. Rama, Asst. Professor, Department of Computer Science, Kakatiya University, Warangal, TS, India. E-mail: rama.abbidi@gmail.com.*
- *T. Sudha, Professor, Department of Computer Science, Sri Padmavati Mahila Visvavidyalayam, Tirupati, AP, India. E-mail: thatimakula_sudha@yahoo.com*

The reminder of the paper is structured as follows. Section 2 provides prior works related to load balancing. Section 3 provides details of the proposed approach. Section 4 presents CloudSim overview. Section 5 provides experimental results while section 6 throws light into deriving conclusions and providing recommendations for future work.

2 RELATED WORKS

Load distributed strategies can be traced back to 1990s where locally distributed systems are widely used for balancing load. Such systems had mechanisms to know the level of load on locally distributed systems and make necessary decisions to ensure the load is balanced. Such systems adapted different algorithms such as sender-initiated algorithms, receiver-initiated algorithms, symmetrically initiated algorithms, and adaptive algorithms [3]. Chaczko et al. [4] explored two aspects of cloud computing which represents load balancing and availability. The former refers to balance of load in cloud computing while the latter reflects the service availability in cloud computing. Both are important for successful adaptation of cloud services. They explored the two factors using a case study based approach.

Randles et al. [6] studied various techniques used for distributed the load in cloud with service oriented architecture (SOA) in place. They investigated three distributed solutions namely active clustering, biased random sampling, and honey bee foraging behavior. Penmatsa & Chronopoulos [7] defined a load balancing solution built on game-theoretic approach for distributed systems. Nash Bargaining Solution (NBS) was the

underlying mechanism in their solution. Grosu et al. [8] also used game-theoretic approach in the form of cooperative games for balancing the work load in distributed environment. Incidentally, their solution is also based on NBS. Aote & Kharat [9] proposed yet another model for balancing the load in distributed environment. Nishant et al. [5] proposed an approach that considers load balancing of nodes in cloud computing using ACO. They employed an enhanced approach to ACO for focusing on load balancing of nodes involved in computing. The algorithm was aware of peak usage hours and performs load balancing accordingly.

Xu et al. [10] proposed cloud partition based approach for load balancing where partition states such as "IDLE", "OVERLOADED" and "NORMAL" are used to balance the load. All the existing load balancing solutions provide different approaches to solving the problem. However, we believe that load balancing and cloud task scheduling are optimization problems that need further research. In this paper we investigated the problem of load balancing by computing load balance factor in the context of task scheduling using CloudSim toolkit.

3 Proposed Heuristic Approach For Load Balancing

We considered cloud load balancing as combinatorial optimization problem. We found that ACO technique can be used to solve such optimization problem as far as there is possibility to state problem depiction, heuristic desirability, constraint satisfaction method, pheromone update rule, and probabilistic update rule. The problem is represented as a graph which is denoted as G = (N, E). N represents a set of nodes associated with virtual machines while E denotes a set of edges that define connections tasks and virtual machines as shown in Fig. 1.

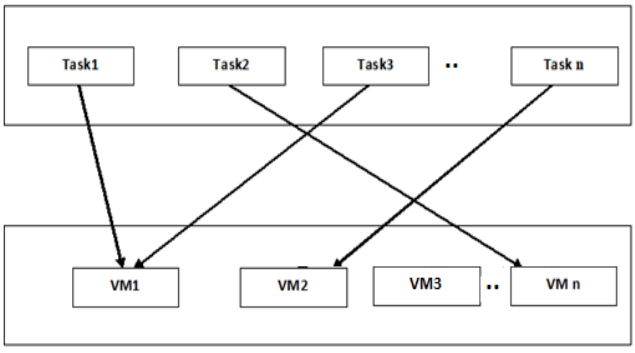

Fig. 1. Problem representation in ACO

Tasks are associated with virtual machines. This is the scenario in which load balancing needs to be optimized in terms of good job scheduling approach.

A simple solution is described, that is the inverse of expected execution time of i^{th} task on j^{th} VM. The constrained satisfaction method used is short term memory that holds visited VM to avoid repeated visits in an ACO procedure. Pheromone updating rule is required by typical ACO. On all edges pheromone is evaporated. Ants deposit pheromone on all visited edges. This behavior is related to influencing quality of the solution provided by ants in ACO.

The basic ideas are obtained from ACO for the proposed algorithm in this paper. The algorithm makes use of different aspects such as rule for choosing next VM for scheduling, computing load balancing factor, and so on. The rule of choosing VM is as presented in Equation (1).

$$P_{ij}^{k}(t) = \begin{cases} \arg \max_{s \in allowed_k} \{(\alpha * \tau_{is}(t)) + (1-\alpha) * n_{is}\} \\ +LB_s \\ ifq \leq q_0 \\ Jifq > q_0 \end{cases} \quad (1)$$

Here $T_{ij}(t)$ represents pheromone concentration at given time t on the path between VM j and task i. LBs is the load balancing factor of VMs. Load balancing factor is computed as in Equation (2).

$$LB_s = 1 - \frac{Tots - LT_{avg}}{Tots + LT_{avg}} \quad (2)$$

The average execution time of virtual machines is represented as LTavg. The expected execution time of the tasks is computed as in Equation (3).

$$Tot_j = sum_{i \in IJ}(d_{ij}) \quad (3)$$

The value of j is computed as shown in Equation (4).

$$j = \begin{cases} \frac{(\propto * \tau_{ij}(t)) + ((1-\alpha) * n_{ij}) + LB_j}{\sum s \in allowed_k (\propto * \tau_{ij}(t)) + ((1-\alpha) * n_{ij}) + LB_s} & ifj \in allowed_k \\ 0, & otherwise \end{cases} \quad (4)$$

The imbalance among virtual machine is computed as in Equation (5).

$$DI = \frac{T_{max} - T_{min}}{T_{avg}} \quad (5)$$

The proposed algorithm EH-ACO which is based on ACO is as follows.

Input: Incoming VMs and cloudlets

Output: Optimized scheduling

Initialize cloudletList and tempCloudletList to null

On arrival of cloudlets, put them into cloudletList

Do while cloudletList is no empty

Compute size of VM list to n

IF size of cloudletList > n THEN

Copy first arrived cloudlets to tempCloudletList

Compute Load Balance Factor

ELSE

Copy all cloudlets to tempCloudletList

Execute ACO procedure for tempCloudletList and n

Compute Load Balance Factor

End Do

Print "Optimized Cloud Task Scheduling with Load Balance Factor"

Stop

The algorithm employs ACO optimization for enhanced performance in load balancing through optimized cloud task scheduling using load balancing factor.

4 Overview Of CloudSim

CloudSim is a simulation toolkit which is used to model and simulate systems pertaining to cloud computing. In other words it provides simulated cloud computing environment with support for both system and behavioral models of cloud system components like data centers, VMs and so on [2]. As it is not possible and not feature to gain access to real cloud environments like Microsoft Azure, Amazon EC2, and Google App Engine for research purposes. Therefore a simulation environment was needed. This gap was filled by CloudSim. The main features of CloudSim include support for modelling and simulation of data centers, virtual machines, visualized server hosts, allocation of resources to VMs, user defined strategies, simulation of federation clouds, and dynamic resource provisioning [1].

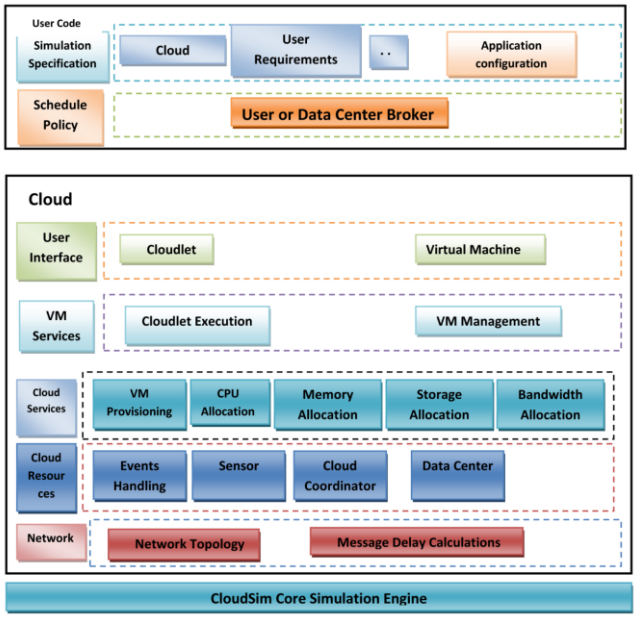

Fig. 2. Architecture of CloudSim [1]

As shown in Fig. 2, it is evident that the CloudSim based applications need three layers. At the bottom CloudSim core simulation engine lies. In middle CloudSim and at the top user code lies which contains simulation specification and scheduling policy. CloudSim provides user interface structures, VM services, cloud services, cloud resources and network related services. There are different classes supported by CloudSim. Some important classes used in this paper are shown in Fig. 3.

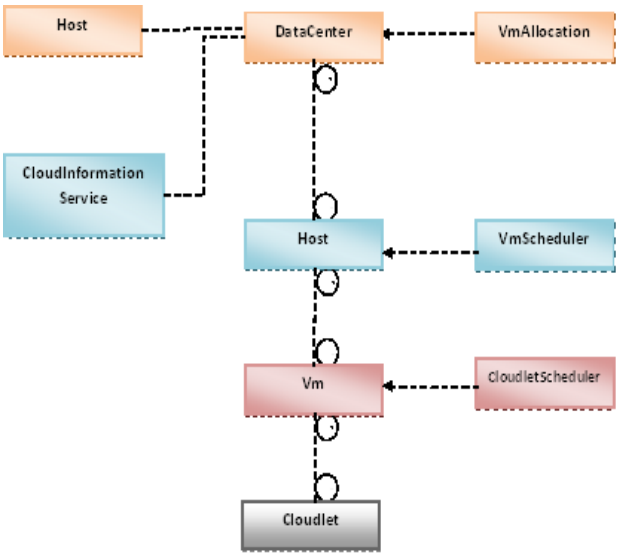

Fig. 3. CloudSim classes used in simulation

Datacenter: It consist a group of computer hosts that may be homogenous or heterogeneous in nature. These hosts work together.
DataCenterBroker: It is a broker which mediates interactions between SaaS and CSPs
CloudInformationService: It is responsible to register data center entities and discover resources.
VmAllocation: It is responsible to allocate VMs to host computers. It is a provisional policy which works at data center level.
VmScheduler: It models scheduling policies such as time-shared and space-shared. This class is required to allocate processor codes to virtual machines. For every host in data center, this runs in data center.
Host: It represents a physical server which exists in data centers.
Vm: This class encapsulates a "Virtual machine". It's run in hosts in cloud-data centers.
Cloudlet: This class encapsulates the cloud-based application services.
CloudletScheduler: It is an abstract class which is implemented by different classes with different scheduling policies such as time-shared and space-shared. It is meant for sharing processing power among various cloudlets used in the simulation.

5 Results Of Simulations

Experiments are done with CloudSim toolkit with the size of data centers 10, virtual machines 50, and tasks from 100 through 1000. The task length is from 1000 through 20000 Millions of Instructions (MI). Task (cloudlet) has parameters such as length and total number of tasks set to 1000-20000 and 100-1000 respectively. VM parameters used include total number VMs, MIPS, RAM used by VM, bandwidth, cloudlet scheduler, and number of PEs needed. These parameters are set to 50, 500-2000, 256-2048, 500-1000, space-shared and time-shared scheduling and 1-4 respectively. With respect to data center, number of data centers is 10, number of hosts is 2-6 and VMscheduler is set to space-shared and time-shared.

Experimentations are made by altering various ACO parameters related with scheduling. α, M, β, ρ, Q and Tmax are the parameters used in Experiment. Where α represents relative weight of pheromone, M represents number of ants, β represents visibility information (distance), ρ represents pheromone update, Q represents random number uniformly distributed and Tmax represents maximum number of iterations allowed. These variables are set with certain default values in each experiment. Some parameters set to fixed values and one varying parameter is observed. The observations are in terms of makespan to find the total time taken by a set of jobs to get executed.

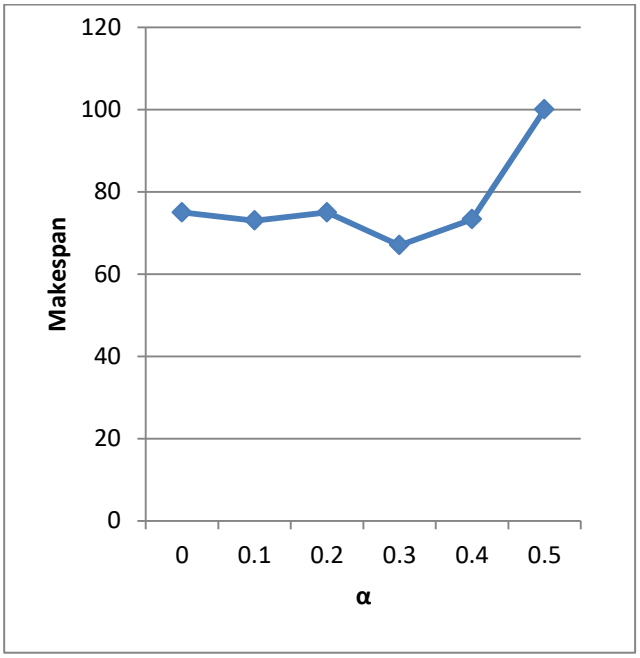

Fig. 4. Performance of ACO against Alpha
(β=1, ρ=0.5 Tmax=150 and M=8)

As shown in Fig. 4, the alpha has its impact on makespan besides other parameters considered for experiments.

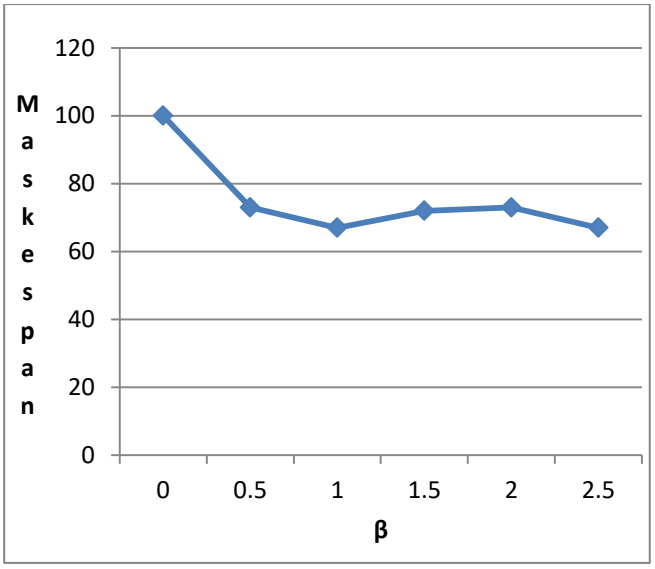

Fig. 5. Performance of ACO against Beta
(α=3, ρ=0.4, Tmax=150 and Q=100 and M=8)

As shown in Fig. 5, the beta has its impact on makespan besides other parameters considered for experiments.

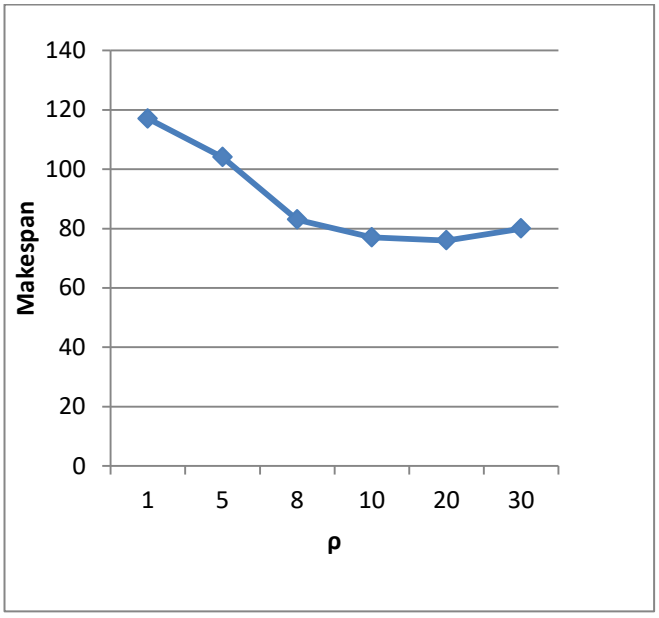

Fig. 6. ACO Performance for Different Values of Rho (α=3, β=1, Q=100, Tmax=150 and M=8)

As shown in Fig. 6, the Rho has its impact on makespan besides other parameters considered for experiments.

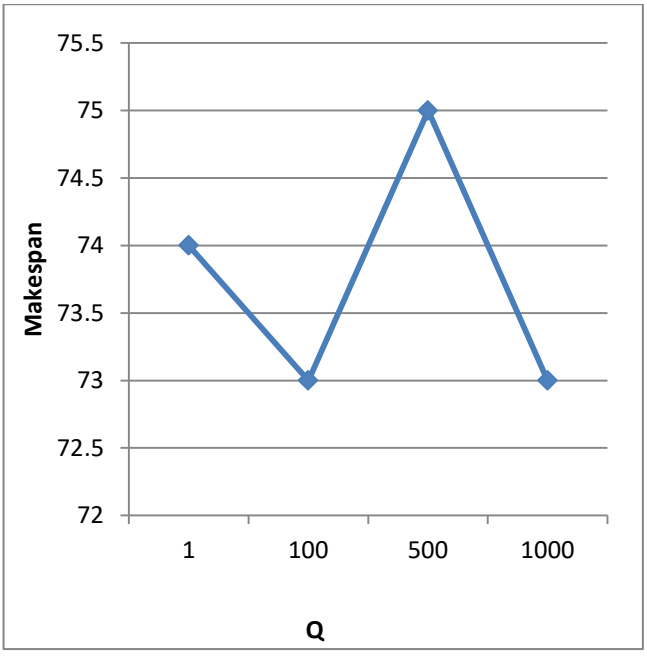

Fig. 7. Performance of ACO for Q values (α=3, β=1, ρ=0.4 Tmax=150 and M=8)

As shown in Fig. 7, the Q has its impact on makespan besides other parameters considered for experiments.

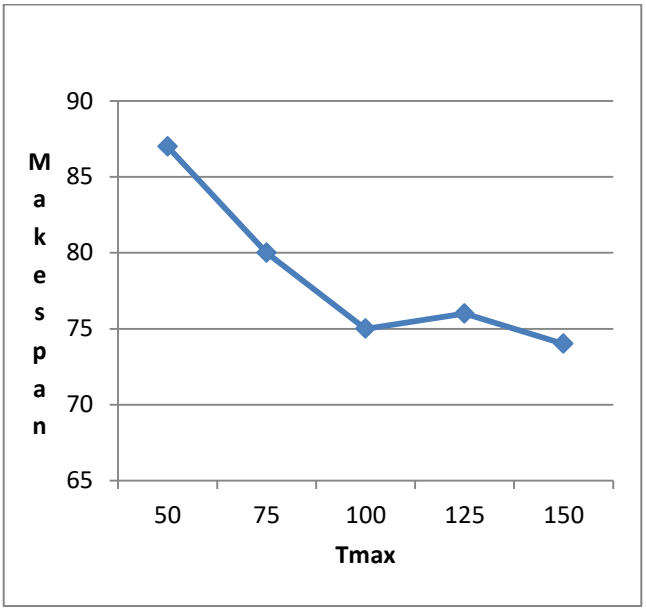

Fig. 8. Performance of ACO for Tmax values (α=3, β=1, ρ=0.4, M=8 and Q=100)

As shown in Fig. 8, the Tmax has its impact on makespan besides other parameters considered for experiments.

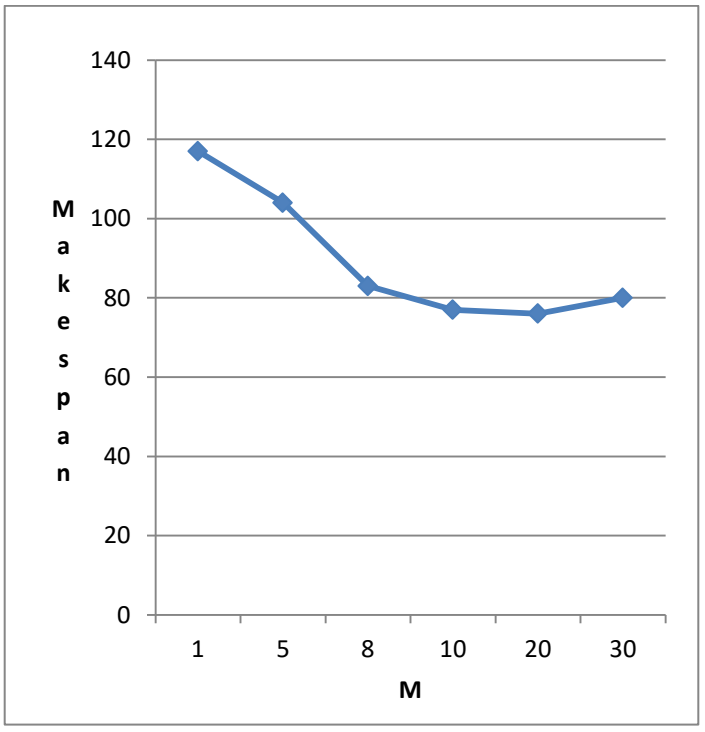

Fig.9. Performance of ACO for different values of M (α=3, β=1, ρ=0.4 Tmax=100 and Q=100)

As shown in Fig. 9, the M ants' numbers has its impact on makespan besides other parameters considered for experiments.

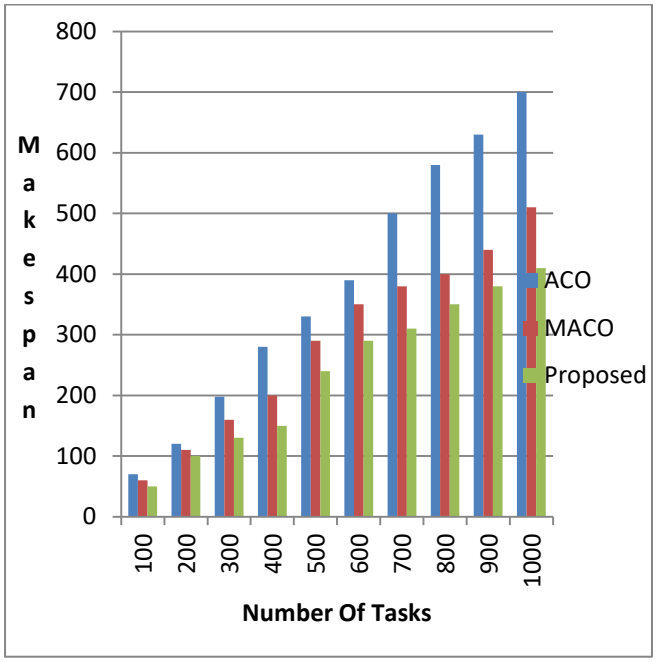

Fig. 10. Average Makespan of ACO, MACO [4], and Proposed EH-ACO

As shown in Fig. 10, the performance of ACO, MACO and proposed EH-ACO based algorithm are compared in terms of makespan.

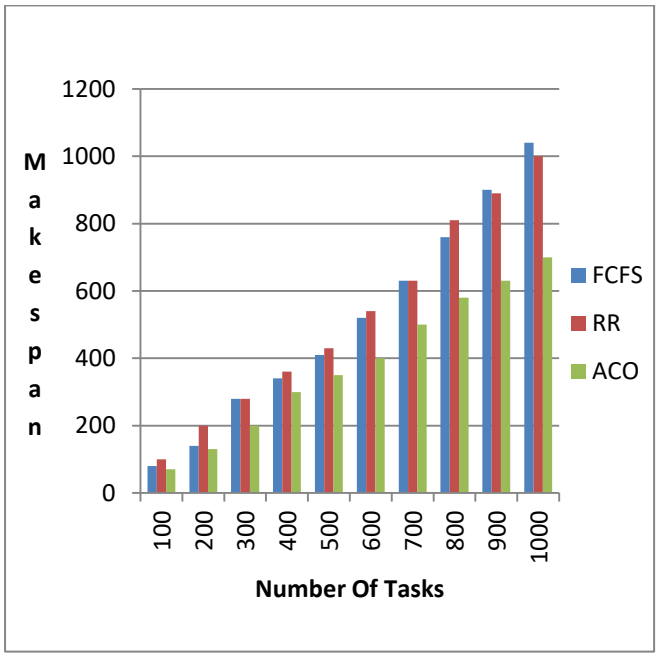

Fig. 11. Average Makespan of FCFS, RR, ACO

As shown in Fig. 11, the performance of ACO, RR and FCFS algorithms are compared in terms of makespan.

6 Conclusion and Future Work

In this paper, we defined an ACO based algorithm known as EH-ACO for optimizing the load in cloud computing by computing the load balancing factor. The defined algorithm achieved it with its underling task scheduling strategy. The algorithm strives to find resource allocation optimally for batch tasks in a dynamic and distributed cloud computing eco system. On the entire cloud computing system, the proposed algorithm also reduces makespan of tasks. It has self-adapting criteria pertaining to control parameters besides having a load balancing intention. Experiments are made using CloudSim toolkit with different number of cloud tasks from 100 through 1000. The toolkit is used to evaluate the functionality of the proposed algorithm. Experiments are used to determine best values for ACO parameters. Afterwards the makespan of proposed algorithm is compared with other ACO based approaches. The simulation results revealed that the defined algorithm provides improved performance over other algorithms. We find some important directions for future work. First, it is interesting to investigate the optimal or fine-grained pricing strategies to bring about equilibrium in the advantages of CSPs and service users. Second, it is also interesting to know the impact of optimized or fine-grained pricing strategies on balancing the load in cloud computing.

References

[1] Peter Mell and Timothy Grance. (2012), "The NIST Definition of Cloud Computing", Computer Security Division Information Technology Laboratory National Institute of Standards and Technology002C pp. 1-7.

[2] Marios D. Dikaiakos,George Pallis,Dimitrios Katsaros,Pankaj Mehra and Athena Vakali, (2009), "Cloud Computing Distributed Internet Computing for IT and Scientific Research", IEEE, pp. 10-13.

[3] Niranjan G. Shivaratri, Phillip Krueger and Mukesh Singhal, (1992), "Load Distributing for Locally Distributed Systems". Pp. 33-44.

[4] Zenon Chacko, Venkatesh Mahadevan, Shahrzad Aslanzadeh and Christopher Mcdermid. (2011), "Availability and Load Balancing in Cloud Computing", International Conference on Computer and Software Modeling, 14, pp. 134-140.

[5] Kumar Nishant, Pratik Sharma, Vishal Krishna, Chhavi Gupta , Kuwar Pratap Singh, Nitin and Ravi Rastogi. (2012), "Load Balancing of Nodes in Cloud Using Ant Colony Optimization", IEEE, pp. 1-6.

[6] Martin Randles, David Lamb and A. Taleb-Bendiab. (2010), "A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing", IEEE, pp. 551-556.

[7] Satish Penmatsa and Anthony T. Chronopoulos, (2011), "Game-theoretic static load balancing for distributed systems", J. Parallel Distrib. Comput. 71, pp. 537-555.

[8] Daniel Grosu, Anthony T. Chronopoulos and Ming-Ying Leung, (2002), "Load Balancing in Distributed Systems: An Approach Using Cooperative Games", IEEE, pp. 1-10.

[9] Shailendra S. Aote and M. U. Kharat, (2009), "A Game-Theoretic Model for Dynamic Load Balancing in Distributed Systems", International Conference on Advances in Computing,

Communication and Control, pp. 235-238.
[10] Gaochao Xu, Junjie Pang and Xiaodong Fu, (2013), "A Load Balancing Model Based on Cloud Partitioning for the Public Cloud", TSINGHUA SCIENCE AND TECHNOLOGY, 18(1), pp. 34-39.
[11] CloudSim, Available online at: http://cyberlingo.blogspot.in/2016/09/cloudsim-simulation-software-for-cloud.html
[12] Buyya, R., Ranjan, R., Calheiros, R.N., "Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities" in Proceedings of the 7th High Performance Computing and Simulation (HPCS 2009) Conference, Leipzig, Germany, 2009.
[13] M. Dorigo, C. Blum, "Ant colony optimization theory: A survey" in Theoretical Computer Science 344 (2–3) (2005), pp. 243–278, 2005.
[14] Medhat A. Tawfeek, Ashraf El-Sisi, Arabi E. keshk and Fawzy A. Torkey, "An Ant Algorithm for Cloud Task Scheduling", in International Workshop on Cloud Computing and Information Security CCIS, 2013.