International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 7 (2015) pp. 18433-18443

http://www.ripublication.com

Load Balancing Of Tasks In Cloud Computing Environments Using Modified Artificial Bee Colony (MABC) Algorithm

R. S. Vishnudurai and Dr. A. Grace Selvarani

PG Scholar Department of CSE (PG)
Sri Ramakrishna Engineering College Coimbatore
Professor & Head Department of CSE (PG)
Sri Ramakrishna Engineering College Coimbatore

Abstract

Scheduling of tasks in cloud computing with available resources is a challenging task. An important aspect of task scheduling in cloud is the load balancing of non-preemptive independent tasks on Virtual Machines(VM). The scheduling process is done by the scheduler so that the available resources are fully utilized. Currently, the load balancing methods like Artificial Bee Colony(ABC) are good at exploration but poor at exploitation. To overcome this limitation, we propose an algorithm named Modified Artificial Bee Colony Algorithm(MABC). The MABC algorithm introducesthe best-so-far solution, inertia weight and acceleration coefficients to change the search process for available resources. It also improves the exploitation process. The evaluation process shows that the proposed algorithm is more feasible and efficient.

Keywords: Virtual Machine, Honey Bee, Load Balancing, Exploitation, Exploration.

1. Introduction

Cloud Computing is a approach in which all applications and files are hosted on a cloud which consists of thousands of complexly interlinked computers. It is an internet based approach which provides on demand service of sharing of resources such as hardware, software and information to computers based on parallel and distributed computing. These emerging distributed systems follow a "Pay as you Use" [5] model. It is not necessary to buy the software or computation platform. With the

help of internet, the customer can use the computation power or software resources and pay money only for the duration he/she uses.The pay as you use model avoids the unnecessary cost deployed in purchasing the full software.

The expectation of the customer is to reduce the overall execution time of tasks. In cloud, Virtual Machines(VMs) are referred as processing units. The expectation of business professional is that the virtual machines should complete the execution of tasks within a limited time and also the VMs should run in parallel. Due to this reason, the available resources are not sufficient to schedule the tasks. In order to maximize the utilization ofavailable resources, the scheduling should be done efficiently. The VMs are assigned with more than one task which can be executed simultaneously. The main aspect of this kind of environment is to ensure that the loads are well balanced in all VMs. In this case, the scheduler should take care of the load balancing across the machines [10]. In order to improve the response time and utilization of the resources, load balancing algorithm is used.

The main goal of load balancing method is to increase the execution of applications on resources whose load varies dynamically [3].Load balancing techniques are widely discussed in heterogeneous environments such as cloud. The load balancing techniques are divided into two categories:(i)Static and (ii) Dynamic [11].

Static algorithms are best suited for nodes with less variation in the load. Therefore, these algorithms are not suited for dynamic environments such as cloud. To overcome this limitation, Dynamic load balancing algorithm is introduced. However, additional cost is incurred in collection and maintenance of load information.Dynamic techniques providebest solution for load balancing of tasks on heterogeneous resources [15]. The proposed algorithm is a dynamic technique which takes into account the priorities of tasks in the waiting queues of VMs in addition to load balancing. In cloud computing environments, heavily loaded VM with multiple tasks has to be removed and submitted to the under loaded VMs of the same data center. When the tasks are submitted to the under loaded VMs, it should be submitted in such a way that there will be good mix of priorities.

The previously used load balancing algorithms like Artificial Bee Colony(ABC) algorithm provides good exploration but poor exploitation[6,7,9]. The exploration process refers to the ability of seeking for global optimum in the solution space of various unknown optimization problems, while the exploitation process refers to the ability of applying the knowledge of previous solutions to look for better solutions[14]. The proposed Modified ABC(MABC) overcomes the limitation of ABC.

Rest of the paper is organized as follows: Section 2 discusses about the existing load balancing techniques. Section 3 describes the proposed Modified ABC(MABC) algorithm. Section 4 describes the implementation. Section 5 presents the evaluation results of the experiments. Finally section 6 describes the conclusion and future work.

2. Related Works

The task load balancing strategy presented byYagoubi B, and Slimani Y[13] minimizes the average response time of applications submitted to Grid computing.Their main contributions are twofold. First they propose a dynamic tree-based model to represent Grid architecture in order to manage workload. This model is characterized by three main features: (i) it is hierarchical; (ii) it supports heterogeneity and scalability; and (iii) it is totally independent from any Grid physical architecture. Second, they develop a hierarchical load balancing strategy and associated algorithms based on neighborhood propriety. The major challenges of this approach areheterogeneity, scalability and adaptability.

The machine learning innovative approach proposed by Ashish Revar, Malay Andhariya, Dharmendra Sutariya, Prof. Madhuri Bhavsar [1] is based on the study of dynamic load balancing techniques in Grid environment. The rules generated by data mining techniques are used for migrating jobs for load balancing. Load balancing should take place when some change occurs in the load situation. The major limitations of this approach are the decision time for load balancing is not minimal and the performance was not up to the expectation.

The generic model for load balancing proposed by Belabbas Yagoubi and Yahya Slimani [12] is based on hierarchical structure. The three levels in the hierarchical model are Intra site load balancing, Intra cluster load balancing, Intra grid load balancing.The major limitations are spanning of administrative domains, potential performance degradation and high failure rate.

The work proposed by Michael E. Houle , Antonios Symvonis , David R. Woodfor load balancingconsiders dimension-exchange algorithms for load balancing on trees with finitely-divisible loads (token distribution) [8].The first contribution of this paper is improved analysis of an existing dimension-exchange protocol for arbitrary trees. Previous analysis of this protocol on trees has been for the complete binary tree only. For a given tree T, the worst case distribution on T under this protocol is determined. The second contribution is a new dimension-exchange algorithm which produces a distribution with discrepancy at most one, which is of course optimal. For trees of bounded degree, the rate of convergence is shown to be optimal in the worst-case. Unfortunately, this algorithm assumes that each node has knowledge of the number of nodes in the tree. This is the first known algorithm for single-token single-port load balancing which achieves optimal discrepancy.

The load balancing scatter operations for grid computing presented byGenaud S, Giersch A.Vivien Fpresent solutions to statically load-balance scatter operations in parallel codes run on Grids [4].The load balancing strategy is based on the modification of the data distributions used in scatter operations. The major challenge of this approach is that the load balancing depends on the processor speed and network link bandwidth.

3. Proposed System

Scheduling of the customer tasks within the available resources is a challenging task. The scheduler should do the scheduling process efficiently in order to utilize the

available resources fully.

In the proposed work, a new artificial bee colony algorithm can be used. However, ABC is good at exploration but poor at exploitation. Its convergence speed is also an issue in some cases. To overcome this deficiency, we can propose a Modified ABC algorithm (MABC). The exploration and exploitation are extremely important mechanisms in ABC. In ABC algorithm the exploration process refers to the ability of seeking for global optimum in the solution space of various unknown optimization problems, while the exploitation process refers to the ability of applying the knowledge of previous solutions to look for better solutions.

The three major modifications are made by introducing the best-so-far solution, inertia weight and acceleration coefficients to change the search process. Consequently, to improve the exploitation, the modification forms of the employed bees and the onlooker ones are different in the second acceleration coefficient. The operation process can be modified in the following form:

$$V_{ij} = X_{ij}W_{ij} + 2\left(\phi_{ij} - 0.5\right)\left(X_{ij} - X_{kj}\right)\Phi_1 + \varphi_{ij}(X_j - X_{kj})\Phi_2 \quad (1)$$

Where V_{ij} is the new possible solution that is a modified possible solution depending on its previous solution X_{ij}. W_{ij} is the inertia weight which controls impacts of the previous solution X_{ij}. X_j is the jth parameter of the best so-far solution Φ_{ij} and φ_{ij} are random numbers between [0, 1], Φ_1 and Φ_2 are positive parameters that could control the maximum step size. On the other hand, if the global fitness is very huge, bees are far away from the best values. So a big correction is needed to search the global optimum solution and then w, $\Phi_1 and \Phi_2$ should be bigger values. Conversely, only a small modification needed, then W, $\Phi_1 and\ \Phi_2$ must be smaller values. In this investigation, inertia weight and acceleration coefficients are defined as functions of the fitness in the search process of ABC. They are proposed as follows:

$$W_{ij} = \Phi_1 = 1/(1 + \exp\left(-\frac{Fitness(i)}{ap}\right)) \quad (2)$$

$\Phi_2 = 1$, if a bee is employed one

$\Phi_2 = 1/(1 + \exp\left(-\frac{Fitness(i)}{ap}\right))$, if a bee is onlooker one

Where ap is the *Fitness* (1) in the first iteration. In order to further balance the process of the exploration and the exploitation, the modification forms of the employed bees and the onlooker ones are different in the acceleration coefficient Φ_2. The main advantages of this method are to achieve a fast convergence speed and to discover a best solution.

In initialization process, MABC like ABC starts by correlating all employed bees with arbitrarily produced food sources. After initialization, the population of the food sources is subject to repetitive cycles of the search processes of the employed bees, the onlooker bees and the scout bees. In this algorithm, an employed bee firstly works out three new solutions by three different solution search equations, and then

selects and decides the best one as the candidate solution. Because of the computation of the candidate solution before the employed bee is selected where they should go to explore, the procedure of computing new food position is known as 'predict'. After the bees 'predict' new candidate solution by three dissimilar solution search equations, they decide the best one from the three solutions as the candidate solution. If the fitness values of the candidate solution are better than the best fitness value accomplishes so far, then the employed bee's moves to this new food source and synchronously abandons the old one, otherwise it leftovers the previous food source in its mind. When all employed bees have completed this process, they divide the fitness information with the onlookers, each of which chooses a food source based on the probability. As in the case of the employed bee, an onlooker 'predicts' three modification on the position in her memory, and then chooses the best one as the candidate source and verifies the fitness value of the candidate source. Providing that the fitness value of the candidate source is better than that of the previous one, the bee would memorize the new position and forget the old one. In MABC, the three solution search equations are separately computed, but influence each other by the chosen best solution.

Pseudo code for MABC Algorith is

1: Load training samples
2: Generate initial populationZ_i< i = 1…SN
3: Evaluate the fitness fitness(i) of the population
4: Set iteration to 1
5: **repeat**
6: **FOR** each employed bee and onlooker bee**{**

Produce new solution by using

$$V_{ij} = X_{ij}W_{ij} + 2(\phi_{ij} - 0.5)(X_{ij} - X_{kj})\Phi_1 + \varphi_{ij}(X_j - X_{kj})\Phi_2$$

If $\Phi_2 = 1$, then it is a employed bee

If$\Phi_2 = 1/(1 + \exp\left(-\frac{Fitness(i)}{ap}\right)$then it is a onlooker bee

Calculate inertia weight and acceleration coefficient using $W_{ij} = \Phi_1 = 1/(1 + \exp\left(-\frac{Fitness(i)}{ap}\right))$**}**

7: **If** there is an abandoned solution **then** replace it with the new solution
8: Memorize the best solution so far
9: iteration=iteration+1
10: **until** cycle=MCN

Where MCN is the Maximum Cycle Number

4. Implementation

The algorithm MABC can be implemented by using a CloudSim. This simulator is generalized simulation framework that allows modeling, simulating and experimenting the cloud infrastructure and the application services[2].The MABC algorithm is implemented based on the parameters Priority, CPU, Disk I/O and B/W.

5. Experimental Results

The performance of MABC is evaluated based on the parameters of Make span, Response time, Degree of Imbalance and number of tasks migrated.Fig 1 illustrates the comparison of Makespan before and after Load balancing using MABC. The X-axis represents the number of tasks and the Y-axis represents the Makespan (task execution and completion time) in seconds. With dynamic load balancing using Modified Artificial Bee Colony algorithm (MABC), the makespan is reduced considerably. With more number of tasks, the difference in makespan time is quite high and MABC provides the best results. Make span can be calculated by using the formula

$$\text{Makespan} = \max\{CT_{ij} \mid i \in T,\ i = 1, 2, \ldots n \text{ and } j \in VM,\ j = 1, 2, \ldots m\} \quad (3)$$

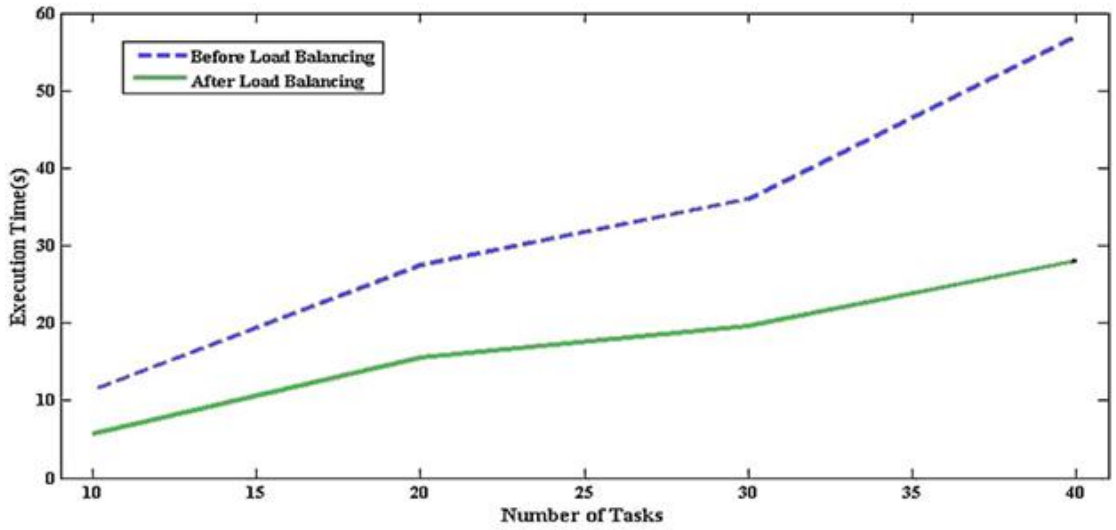

Fig 1 Comparison based on make span

Fig 2 illustrates the response time of VMs in seconds for MABC, DLB, FIFO and WRR Algorithms. The X-axis represents number of tasks and the Y-axis represents time in seconds. It is evident that MABC is more efficient compared with other three methods.

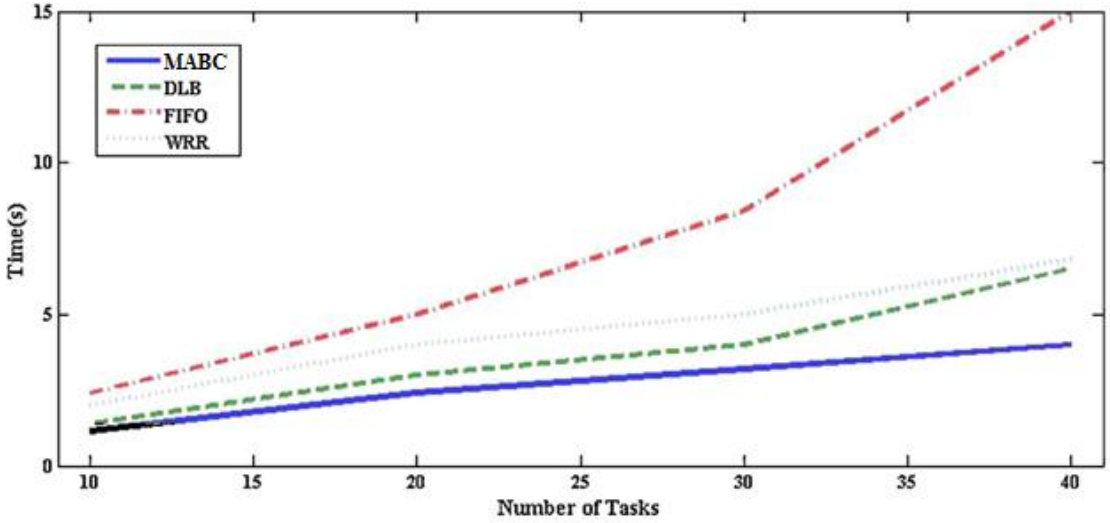

Fig 2 Comparison based on Response time

Fig 3 illustrates the degree of imbalance between VMs before and after load balancing with MABC. The X-axis represents number of tasks and the Y axis represents the degree of imbalance. It is clearly evident that after load balancing with MABC, the degree of imbalance is greatly reduced. Degree of Imbalance can be calculated by

$$DI = \frac{T_{max} - T_{min}}{T_{avg}} \quad (4)$$

where T_{max} and T_{min} are the maximum and minimum T_i among all VMs, T_{avg} is the average T_i of VMs. Our load balancing system reduces the degree of imbalance drastically.

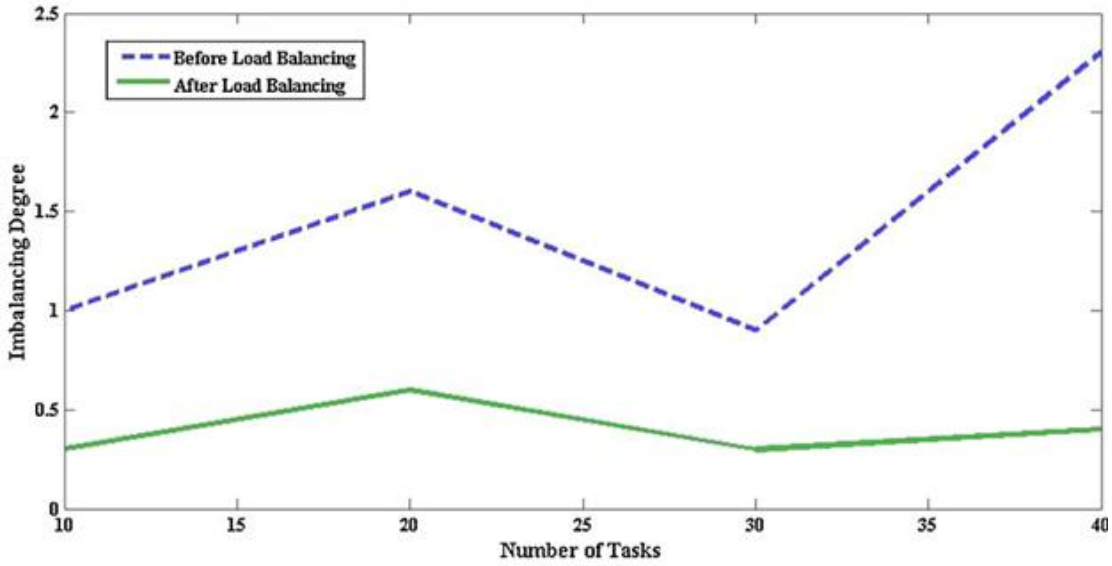

Fig 3 Comparison based on Degree of Imbalance

Fig 4-7 illustrates task migration when numbers of VMs are varied from 4 to 7 for MABC, DLB and HDLB techniques. In all the five cases it is clearly evident that the task migration is very less compared with other two popular techniques irrespective of the number of VMs.

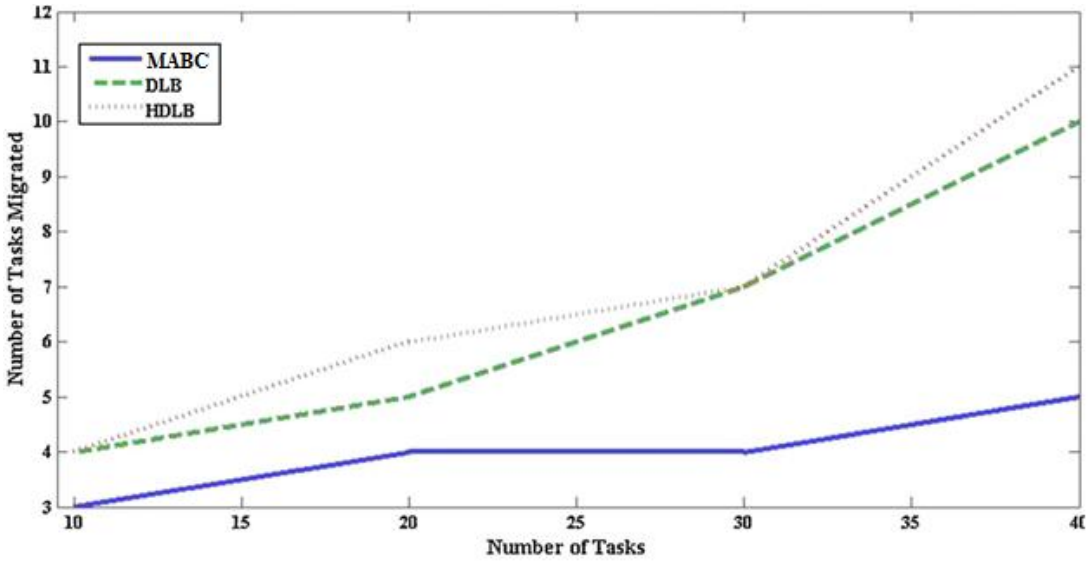

Fig 4 Comparison based on number of task migration when there are 4VMs

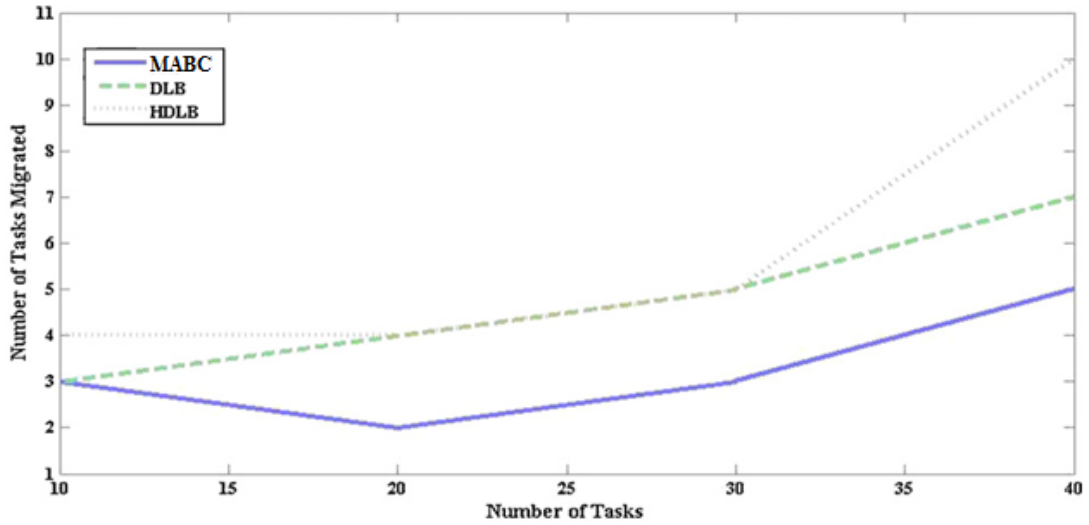

Fig 5 Comparison based on number of task migration when there are 5VMs

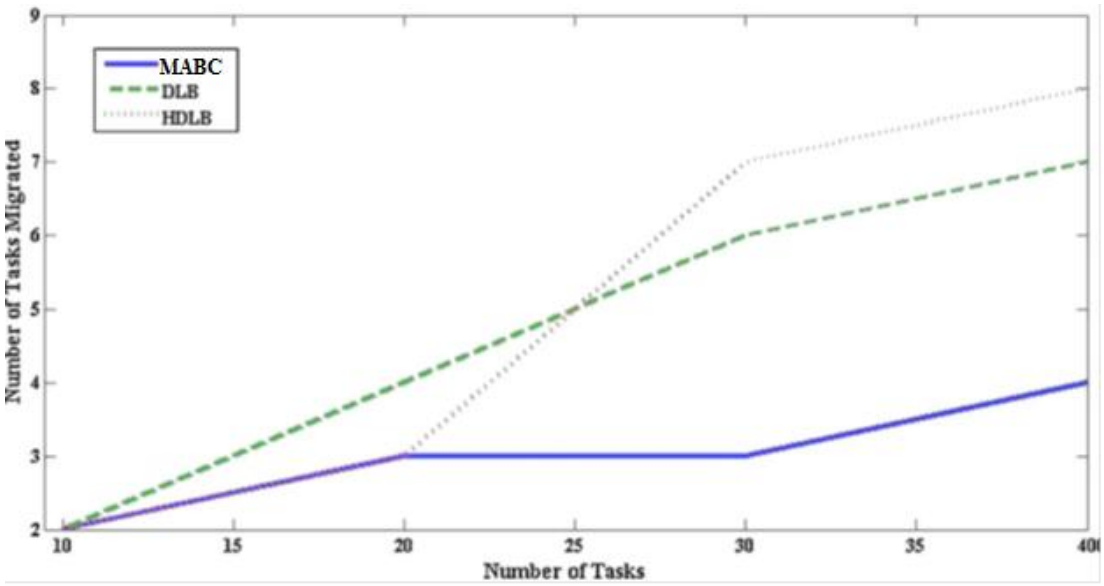

Fig 6 Comparison based on number of task migration when there are 6VMs

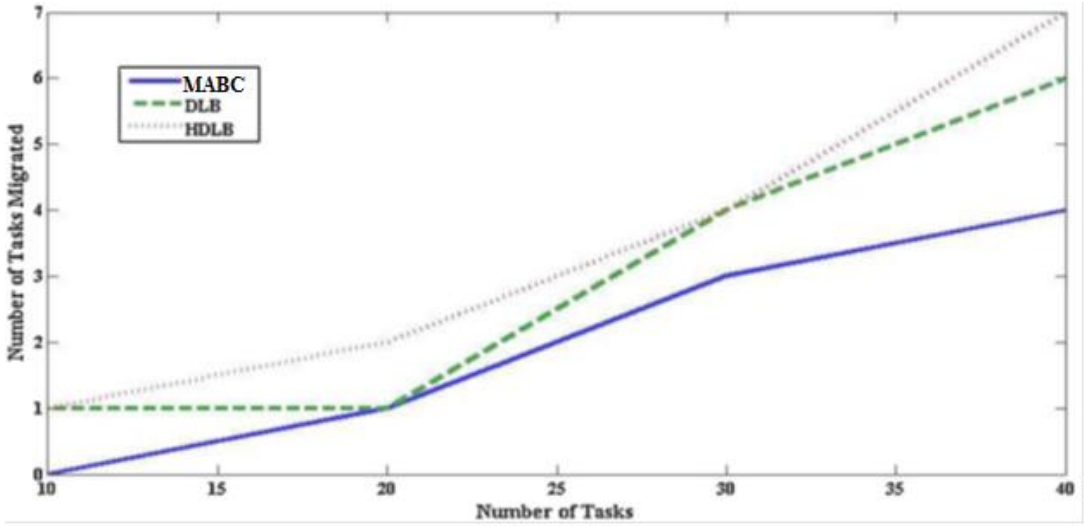

Fig 7 Comparison based on number of task migration when there are 7VMs

6. Conclusion and Future Work

The proposed load balancing strategy is based on the behavior of bee colony. The MABC algorithm is not only good at exploration but also at exploitation.The search process is made efficient by introducing best-so-far solution, inertia weight and acceleration coefficients. The global optimum solution is achieved based on the best global fitness value. The tasks are scheduled based on the founded best solution. The experimental results clearly describe that the proposed MABC algorithm is feasible and efficient.

The future scope of the work is to extend this kind of load balancing for workflows with dependent tasks. This algorithm considers priority as the main QoS parameter. In future, this algorithm can be improved by considering other QoS factors also.

7. References

1. Ashish Revar, Malay Andhariya, Dharmendra Sutariya, Prof. Madhuri Bhavsar(2010) , "Load Balancing in Grid Environment using Machine Learning - Innovative Approach", International Journal of Computer Applications, No 10 - Article 6.
2. Calheiros R N, Ranjan R, Beloglazov A, De Rose C.A.F, Buyya R (2011), "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms", Software: Practice and Experience 41 23–50, http://dx.doi.org/10.1002/spe.995.

3. Eager D.L, Lazowska E.D, Zahorjan J (1986), "Adaptive load sharing in homogeneous distributed systems", The IEEE Transactions on Software Engineering 12 (5) 662–675.
4. Genaud S, Giersch A.Vivien F(2003), "Load-Balancing Scatter Operations for Grid Computing" Parallel and Distributed Processing Symposium, ISSN:1530-2075, 22-26.
5. http://azure.microsoft.com/en-us/offers/ms-azr-0003p/
6. Karaboga D (2005), "An idea based on honey bee swarm for numerical optimization", Technical Report TR06, Computer Engineering Department, Erciyes University, Turkey.
7. Karaboga D, Basturk B (2008), on the "performance of artificial bee colony (ABC) algorithm", Applied Soft Computing 8 (1) 687–697.
8. Michael E. Houle , Antonios Symvonis , David R. Wood(2002) , "Dimension-Exchange Algorithms for Load Balancing on Trees", SIROCCO 9, Proceedings of the 9th International Colloquium on Structural Information and Communication Complexity, Andros, Greece.
9. Pan Q K, Tasgetiren M F, Suganthan P, Chua T (2011),"A discrete artificial bee colony Algorithm for the lot-streaming flow shop scheduling problem", InformationSciences 181 (12) 2455–2468.
10. Randles M, Lamb D, Taleb-Bendiab A (2010),"A comparative study into distributed load balancing algorithms for cloud computing", in: Proceedings of 24th IEEE International Conference on Advanced Information Networking and Applications Workshops, Perth, Australia, pp. 551–556.
11. Revar A, Andhariya M, Sutariya D, Bhavsar M (2010), "Load balancing in grid environment using machine learning-innovative approach", International Journal of Computer Applications 8 975–8887.
12. Yagoubi B, and Slimani Y(2007), "Task Load Balancing Strategy for Grid Computing," Journal of Computer Science, Vol. 3, No. 3, pp. 186-194.
13. Yagoubi B, and Slimani Y(2007), "Task Load Balancing Strategy for Grid Computing," Journal of Computer Science, Vol. 3, No. 3, pp. 186-194.
14. Yenny Noa Vargas, Stephen Chen (2010), "Particle Swarm Optimization with Resets – Improving the Balance between Exploration and Exploitation", Advances in Soft ComputingLecture Notes in Computer Science Volume 6438, 2010, pp 371-381
15. Zhao C, Zhang S, Liu Q, Xie J, Hu J(2009)," Independent tasks scheduling based on genetic algorithm in cloud computing, wireless communications, Networking and mobile computing", WiCom '09, in: 5th International Conference, pp. 1–4.