Task based approach towards Load Balancing in Cloud Environment

Rajani S. Sajjan
H.O.D., Department of Computer Science & Engg.,
VVP Institute of Engg. & Technology, Soregaon,
Solapur, Maharashtra, India

Rekha Y. Biradar
Department of Computer Science & Engg.,
VVP Institute of Engg. & Technology, Soregaon,
Solapur, Maharashtra, India

ABSTRACT

Load balancing is all time trending topic in a cloud environment. In order to improve system performance and to protect the system against failures, the workload must be distributed among one or more server efficiently and dynamically. In this paper, we propose a Task based Approach towards Load Balancing (TB-LB) in a Cloud Environment based on clustering of virtual machines and heuristic algorithms. The proposed system is used to solve both the population based and non-population based problems. The proposed system combines three heuristic algorithms, namely simulated annealing, particle swarm optimization and genetic algorithm to balance the load and to minimize the makespan of tasks and system performance by considering the task requirement. It also uses k-means clustering approach to organize virtual machines (VMs) into groups to reduce the execution time. The simulation results show that our proposed algorithm improves the system performance by minimizing makespan and execution time

General Terms

Load balancing algorithm

Keywords

Cloud Computing, Load Balancing, Heuristic Algorithms, GA, SA, PSO

1. INTRODUCTION

Cloud computing is a next generation computing that that provides more energetic services dynamically to the users via a network on paid or metered basis [1, 3].Cloud computing provides three types of computing as service namely software, platform and infrastructure. There are three types cloud namely public, private and hybrid cloud through which cloud services can be provided to the user [2]. Organizations from public sector to the private sector are using cloud services to cater the needs of their customers as well as the organization itself. Their by reducing the infrastructure cost. This increasing use of cloud makes the attention towards a load balancing as a hot topic in a cloud environment. Throughput maximization, scalability and improved system performance is ensured by load balancing [2].

Cloud load balancing provides more benefits cost effectively than the traditional hardware based load balancing [4]. Scheduling is very crucial factor to improve the performance of the cloud system. There are many traditional scheduling algorithms that can be used in cloud environment such as round robin, first come first serve, min-max, min-min, etc. these traditional scheduling algorithms are ordinary scheduling algorithms and easy to implement. These algorithms are inappropriate for handling difficult or wide level scheduling problems. Hence, they are inappropriate in getting the best results [5]. There is opportunity to enhance the scheduling algorithms used in a cloud environment.

Heuristic algorithms can be applied to scheduling problems in a cloud environment. Heuristic algorithms use several practical based techniques to solve a scheduling problem. The results produced by heuristic algorithms are better than traditional scheduling algorithms but does not guaranteed to be an optimal [5]. But this result is good enough and fast to achieve a goal of the scheduling problem. To increase the strengths of scheduling algorithms two or more heuristics can be combined into a single heuristic algorithm, called hyper heuristic scheduling algorithm [5]. The proposed algorithm uses clustering approach and also integrates several heuristic algorithms. Hence it is a cluster based hyper heuristic algorithm which considers the task requirements. The basic idea of the proposed algorithm is to solve population based and non population based problems.

The rest of the paper is organized as follows- Section 2 describes related work. Section 3 describes TB-LB system. Section 4 describes experimental setup and result. Section 5 gives Conclusion and future work.

2. RELATED WORK

Become prominent technology in the information technology, the cloud computing makes the availability of computation intensive services dynamically to the user on paid basis [1]. The cloud presents services to the user on demand in speedy, inexpensively and reliably [1, 3]. As the use of cloud is increased tremendously, the cloud is facing new challenges such as availability of services, security, resource management and balancing load, etc. [2]. On demand provision of resources to the user may give rise to challenge of load balancing [2]. Due to this some of the nodes in cloud are lightly loaded or heavily loaded. This may affect on the performance, throughput, response time, etc. [2]. To improve all these criteria, load must be shared efficiently and evenly on cloud nodes. For this sharing, efficient methodology is needed to balance the load. Methodology for load balancing can be static or dynamic [2]. Static methodology can present a good result in non flexible and uniform habitat. Dynamic methodology can present a good result in flexible and non uniform habitat. Some heuristic techniques are also available to further improve the result of load balancing. A heuristic technique discovers a new solution among set of all possible solutions. This new solution is near to optimum solution and can be considered as solution which is enough to solve the given problem [4, 5]. Genetic algorithm, ant colony optimization, simulated annealing, particle swarm optimization and artificial bee colony optimization are some of the examples of heuristic techniques.

2.1 Genetic Algorithm

Genetic algorithms (GA) are inspired by natural selection strategy [7]. GA solves population-based complex problem. Hence it is considered as population based heuristic algorithm. GA can also be considered as evolutionary algorithms. A new population also referred as solution is generated based on old solution. Newer one will be better than the old one. This process is repeated until the best solution is received [6, 13]. This process includes following three operations: selection, crossover and mutation [6-7, 13]. Genetic algorithm is most suitable algorithm to solve scheduling and optimization problems. Mutation and crossover are evolutionary operations.

- **Selection:** Two good parent populations are selected [6-7, 13] depending on fitness.
- **Crossover**: Exchanging parts of two parents selected in selection operation to build better children [6-7, 13].
- **Mutation:** Mutation is performed by randomly flipping some of the bits. Finally a new child is accepted as best solution [6-7, 13].

Advantages:

1. Easily modified
2. Able to improve the result of existing models

2.2 Simulated Annealing

Simulated annealing (SA) is non population based probabilistic heuristic algorithm. It uses iterative improvement strategy to solve complex optimization problems. It is derived from the process of annealing [8, 13]. Its working similar to hill-climbing but it picks the move randomly. If the solution is improved by picked move then it is accepted [9]. Probability of accepting worse solutions is decreased slowly [8,13]. Simulated annealing algorithm is most suitable algorithm to solve scheduling and optimization problems. The disadvantage is that it takes long time to present best solution.

Advantages:

1. Fit for large search space
2. Provides better results

2.3 Particle Swarm Optimization

PSO is population and swarm based intelligence heuristic algorithm. It is one of the artificial intelligence techniques. This algorithm is encouraged from behavior of a flock of birds [10]. PSO and GA are similar in some ways [10]. But PSO does not incorporate evolutionary operations such as mutation and crossover. Like GA, it is initialized with populations which are random solutions. These solutions are particles and then it searches for best result by updating generations [10-13]. Types of PSO are multi-objective PSO, bi-objective PSO, hybrid PSO, etc. [12, 13]. In PSO, there is no direct re-combination of individuals of the population. Each particle adjusts its flying path based on two best values known as pbest and gbest value. PSO is most suitable algorithm to solve scheduling.

Advantages:

1. Easy to implement
2. Gives best solution quickly

2.4 K-means Clustering Algorithm

Clustering technique is categorized as unsupervised learning problem. Clustering is the task of classifying objects into groups [14, 15]. Members of group are similar in some way. Need to define k centers, one for each group. Procedure starts with initial k centers. These centers can be selected randomly from the data set. K-Means algorithm is partitional clustering algorithms. The algorithm then iterates between two steps [14, 15]:

- **Data assignment step:**
  Each center defines one of the clusters. In this step, each data point is assigned to its nearest center, based on the Euclidean distance.
- **Center update step:**
  In this step, the centers are recomputed. This is done by taking the mean of all data points assigned to that centers cluster.

3. TB-LB SYSTEM

Cloud computing has faced many challenges, including security, efficient load balancing, resource scheduling, scaling, QoS management, etc. Load balancing is one of the main challenges and concerns in cloud environments [2]. It is the process of assigning and reassigning the load among available resources so that we can maximize throughput, while minimizing the cost and response time. Load balancing also boosts the performance, resource utilization as well as energy saving [2]. Therefore, providing the efficient load-balancing algorithms and mechanisms is a key to the success of cloud computing environments.

The existing hyper heuristic system does not incorporate the clustering technique. Due to this execution time is larger. Also existing hyper heuristic system does not provide security and does not consider the task requirements. The proposed system is Task Based approach towards Load Balancing in Cloud Environment (TB-LB). It is load balancing system which combines three heuristic algorithms, namely simulated annealing, particle swarm optimization and genetic algorithm to keep the makespan of tasks to the smallest amount as possible and system performance by considering the task requirement. Makespan is the completion time of tasks. TB-LB system uses k-clustering algorithm to group the VMs according to VMs specification [13].

3.1 Algorithm Overview

The two main components of proposed TB-LB systems are the k-means clustering and proposed algorithm. Proposed TB-LB system follows the following steps [13]:

1. Log in using username and password
2. Load the VMs
3. VMs are grouped using K-means clustering algorithm. All cluster information is stored at Global Load scheduler.
4. Submit the input tasks. Global Load scheduler selects the appropriate cluster and assigns task to the selected cluster for execution.
5. Cluster Load Balancer maintains information about VMs in that cluster. It schedules the task by using proposed load balancing algorithm is applied and input tasks are assigned to appropriate VM.
6. Finally makespan is provided.

Fig 1 shows the flowchart of TB-LB system [13].

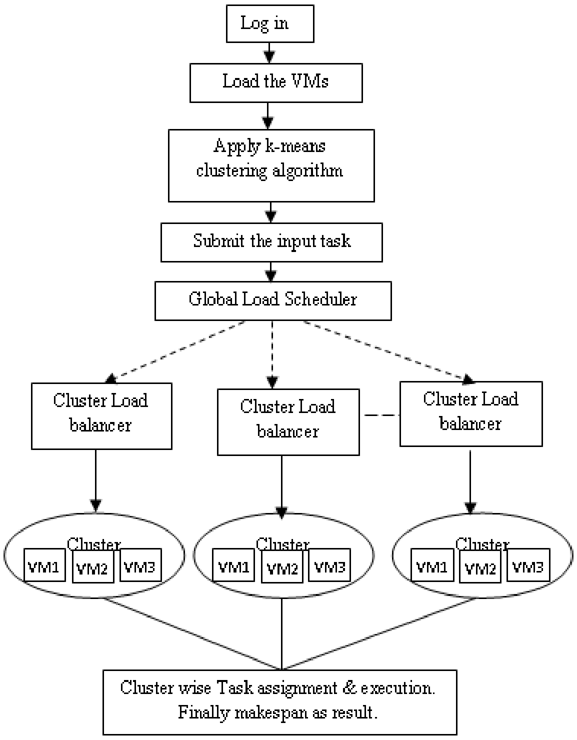


Fig 1: Flowchart of TB-LB System

3.2 Key Operations of Heuristic Algorithm

For each of the iteration, one algorithm is picked as heuristic algorithm from the pool of heuristics algorithms [13]. To do this three operations are performed.

- **Improvement operation:**
  This operation is performed to decide whether to change the scheduling algorithm. It compares makespan with best so far makespan. It returns true for improvement in makespan
- **Diversity detection operation :**
  To decide when to change the scheduling algorithm, diversity detection operation is performed. The threshold value is initialized at first iteration and it is compared with solution received in current iteration. It returns true if current solution is greater than threshold.
- **Perturbation operation:**
  To determine which algorithm is picked from the pool of heuristic algorithms, perturbation operation is performed. When first two operations namely improvement and diversity operation returns true, then there is no need to pick an algorithm from pool. Hence it returns false. When it returns true, then there is need to pick an algorithm from pool. To perform this, first lower priority is assigned to current heuristic algorithm and then high priority heuristic algorithm is picked from the pool.

4. Experimental Setup and Result

The algorithm was implemented and simulated in CloudSim which provide simulation environment of cloud computing using java. Simulation environment is best way to judge the performance of system. The experiment has been carried out by implementing the TB-LB system in java.

VMs considered in the experiment are 50. Comparison of algorithm is taken on the basis of makespan (completion time of task) and execution time. Table 1 shows the simulation parameters.

Table 1. Simulation Parameters

Parameter	Values
Number of VMs	50
Number of clusters	4
Number of tasks	8-100
VM memory	512-2048 MB
VM bandwidth	500-2500 bps

Execution of TBLB system was analyzed for several input tasks, started from 10 tasks to 100 tasks. The result of this is shown in table 2 and 3. Table 2 shows the makespan provided by TB-LB system for specific number of tasks. For example for 60 tasks, the TBLB system has produced makespan of 121 ms. It would be different for another run. The makespan received for same number tasks will plus or minus 2 or 3 ms. As the TB-LB system incorporates both clustering and heuristic approach, makespan provided by TB-LB system is smaller. Table 3 shows the makespan provided by TB-LB system and existing hyper heuristic system. Existing hyper heuristic system does not incorporate clustering approach as well as authentication.

Table 2. Makespan of TB-LB System

No. of Tasks	Makespan in ms
8	17
10	22
20	41
30	65
40	79
50	95
60	121
70	138
80	155
90	184
100	193

Table 3: Makespan of TB-LB and existing hyper heuristic algorithm

No. of Tasks	10	30	60	90
Makespan of Proposed (TB-LB) system in ms	22	65	121	184
Makespan of Hyper heuristic without cluster in ms	1562	1703	3281	3343

Fig 2 shows the makespan comparison of TB-LB system for several tasks. Makespan for number of tasks from 8 to 100 are shown in fig 2. As the proposed TB-LB system incorporates clustering approach and three heuristic algorithms, produced makespan is kept as small as possible.

Hence TB-LB system produces the result within reasonable time. Values of makespan may be different each time for same number of task and for similar task requirements because of k-mean clustering. For example in figure 2, makespan for task 30 is 65 and in figure 3 makespan for task 30 is 67.

Fig 3 shows the cluster wise task scheduling and makespan. This is the result for 60 tasks. Fig 4 shows the makespan comparison among TB-LB system and existing hyper heuristic system implemented without cluster. The Fig 4 is graphical representation of table 3. Makespan for number of tasks from 10 to 90 are shown in fig 4. As the proposed system incorporates k-means clustering approach, performance of the proposed system is increased by reducing makespan. And it is clearly shown in fig 4. Fig 5 shows the performance analysis based on execution time for the result shown in fig 3.

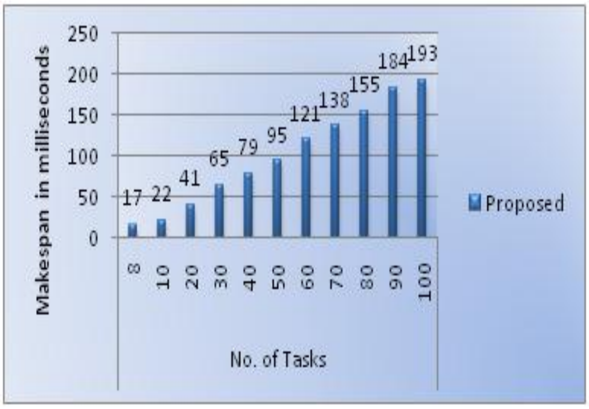
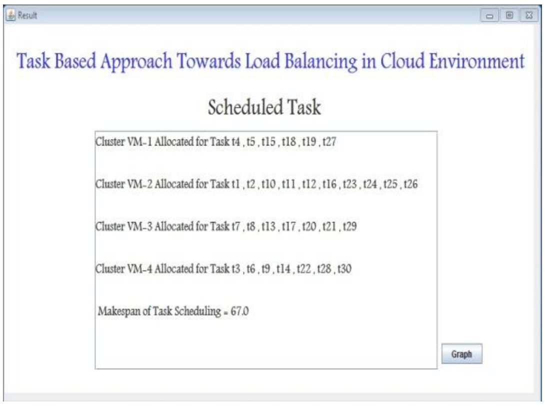


Fig 2: Makespan of TB-LB System



Task Based Approach Towards Load Balancing in Cloud Environment

Scheduled Task

Cluster VM-1 Allocated for Task t4 , t5 , t15 , t18 , t19 , t27

Cluster VM-2 Allocated for Task t1 , t2 , t10 , t11 , t12 , t16 , t23 , t24 , t25 , t26

Cluster VM-3 Allocated for Task t7 , t8 , t13 , t17 , t20 , t21 , t29

Cluster VM-4 Allocated for Task t3 , t6 , t9 , t14 , t22 , t28 , t30

Makespan of Task Scheduling = 67.0

Fig 3: Result of TB-LB System (no. of tasks=30)

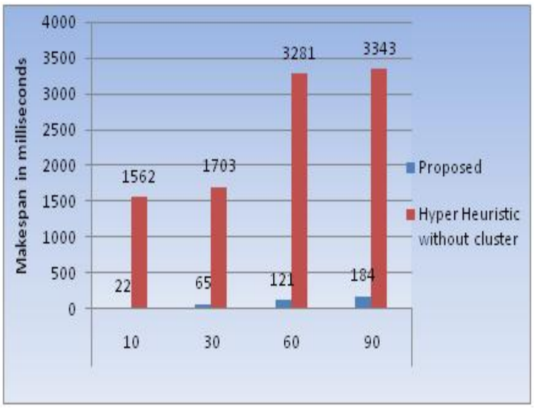


Fig 4: Makespan of TB-LB System and Hyper Heuristic System

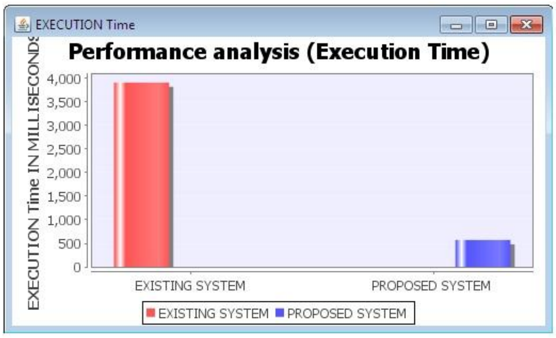


Fig 5: Performance of TB-LB and Existing system

In Table 4 comparison between existing hyper heuristic system and proposed TB-LB system is listed.

Table 4: Comparison between existing hyper heuristic system and TB-LB system

Hyper Heuristic	Proposed (TB-LB)
Overhead of Scanning the entire list of VM	No overhead of Scanning the entire list of VM
Performance is better due to heuristic algorithms.	Performance is better than existing hyper heuristic system because of clustering and combination of three heuristic algorithms.
Does not consider task/ resource specific demands.	Considers task/ resource specific demands.
Not Secure. No user authentication is provided.	Secure. User authentication is provided.

5. CONCLUSION AND FUTURE WORK

The aim of the load balancing is to improve system performance. Load balancing could be performed by a large number of algorithms. We have implemented TB-LB system as load balancing algorithm using clustering and the features of three heuristic algorithms to keep the makespan as smallest to the possible amount. It also keeps the execution time minimum. By adopting clustering technique, TB-LB system reduces the time required to search suitable VM for the execution of task and improves the performance of the system. In future, this work can be extended to use some other load balancing algorithms which are more efficient so that the performance of proposed system can be further improved. Also security of the TBLB system can be further improved by using multifactor authentication. In addition to this, the TB-LB system can be implemented to work on a real cloud so as to evaluate its performance.

6. ACKNOWLEDGMENTS

Author would like to thank Ms Sajjan R.S. for her constant guidance and her family for their unconditional love and support.

7. REFERENCES

[1] Sajjan R.S, Biradar Rekha Yashwantrao, Torvi Harshal. Enterprise Architecture and Services in Cloud Computing: A Survey. International Journal of Computer

Sciences and Engineering Vol.-4, Special Issue-4, Jun 2016, pp.28-34.

[2] Sajjan R.S, Biradar Rekha Yashwantrao. Load Balancing and its Algorithms in Cloud Computing: A Survey. International Journal of Computer Sciences and Engineering Vol.-5(1), Jan 2017, pp.95-100.

[3] Peter Mell, Timothy Grance. The NIST Definition of Cloud Computing. National Institute of Standards and Technology Special Publication 800-145(September 2011).

[4] Geetinderkaur and Sarabjitkaur. Improved Hyper-Heuristic Scheduling with Load-Balancing and RASA for Cloud Computing Systems. International Journal of Grid and Distributed Computing Vol. 9, No. 1 (2016), pp.13-24.

[5] Deepa.K, Prabhu.S, Dr.N.Sengottaiyan. A Hyper-Heuristic Method for scheduling the jobs in Cloud Environment. Informatics Engineering, an International Journal (IEIJ), Vol.4, No.1, March 2016, pp. 23-31.

[6] http://www.obitko.com/tutorials/genetic-algorithms/ga-basic-description.php

[7] M. Lagwal and N. Bhardwaj, "Load balancing in cloud computing using genetic algorithm," 2017 International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, 2017, pp. 560-565.

[8] Leszek Sliwko and Vladimir Getov, “A Meta-Heuristic Load Balancer for Cloud Computing Systems” ,2015 IEEE 39th Annual International Computers, Software & Applications Conference,pp-121-126

[9] https://in.mathworks.com/discovery/simulated-annealing.html

[10] http://www.swarmintelligence.org/tutorials.php

[11] Mohammad Masdari, Farbod Salehi, Marzie Jalali, Moazam Bidaki,” A Survey of PSO-Based Scheduling Algorithms in Cloud Computing”, Journal of Network and Systems Management, Volume 25 Issue 1, January 2017, PP: 122-158,doi: 10.1007/s10922-016-9385-9

[12] Suraj Pandey, Linlin Wu, Siddeswara Mayura Guru, Rajkumar Buyya, “A Particle Swarm Optimization-based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments”, www.cloudbus.org/papers/AINA2010-PSOSched-Workflow.pdf

[13] Sajjan R.S, Biradar Rekha Yashwantrao. “Load Balancing using Cluster and Heuristic Algorithms in Cloud Domain” , Indian Journal of Science and Technology, Vol 11(15), DOI: 10.17485/ijst/2018/v11i15/118729, April 2018

[14] https://www.datascience.com/blog/k-means-clustering

[15] Surbhi Kapoor and Dr. Chetna Dabas, “Cluster Based Load Balancing in Cloud Computing”,2015 Eighth International Conference on Contemporary Computing (IC3)