ISSN (Online): 2230-7893
www.IJCEM.org

Deadline and Length Based Dynamic Load Balancing Algorithm

Sangeeta[1], Suman[2]

[1,2] Computer Science Department, DCRUST, Murthal
Sonepat, Haryana 27031, India

Abstract

Cloud computing is rapidly growing due to the enormous benefits it offers over the traditional approach. Earlier, lot of things like buying server, managing traffic and maintenance needs to be managed individually leading to increase in cost and overhead for users. Cloud offers a less expensive and easy way of managing things. There are various challenges for cloud server provider are keeping low makespan time, good service, maintaining the priority by clients, increasing successful tasks, better recourse utilization. To handle this dynamic scheduling algorithm is required. In this paper we have proposed such a dynamic algorithm sufferage, which works on minimum completion times of tasks on different machine. Deadline and length of task is considered as priority of tasks. Results if algorithm are analyzed over different number of tasks (100-250) and 10 resources are taken. Computational Results show that the modified algorithm shows better results in context of makespan, resource utilization and increase tasks to meet the deadline.

Keywords: *Cloud Computing, Load balancing, dynamic algorithms, task scheduling, sufferage, min-min.*

1. Introduction

Cloud computing is the use of remote servers on the internet to store, manage and process data rather than a local server or on your personal computer. With cloud you can store the data, manage the data using databases or can process data by renting a server which has larger processing capability, by this we can do our work faster. Cloud is popularly being used in many scientific and business applications. Cloud provides facilities with minimum maintenance which makes it easy to use. Services provides by cloud are on dement you pay per use, no extra cost is paid for services. There are different services provides by cloud on different platforms like infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (SaaS). These services differently provide works on all the services provides by cloud which can be understood by the figure.1 given below. There are different kind of cloud present like private cloud, public cloud and hybrid cloud. Private cloud services are limited to one company only where in public cloud services are provided by third party over internet which can be shared with host of different people. Hybrid cloud is combination of public and private cloud, in this type, public and private clouds are as per requirements. Like, companies can use their own infrastructure and when requirements are high public cloud services can be opt.

The number of cloud users is increasing day by day which is resulting in increasing work load on cloudlets. Load balancing helps in distribute load among resources to use resource more efficiently and execute tasks in a better manner. To handle this load a load balancing algorithm is required which can distribute load among all resources evenly in a way that can help in improve load balance level.

Load balancing focus on utilizing the resources in such a way that can increase different factors that affect load balance level. An efficient algorithm gives minimum makespan time and also improves performance of the system. In cloud computing load balancing is a challenging area which focus in balancing the workload. To achieve good load balance level there are mainly two steps a good tsk scheduling approach and observing the resources. The proposed algorithm helps in improving the makespan time and resource utilization.

The paper is organized as follows: section 2 describe related work of load balancing, problem formulation is discussed in section 3 where different notations are described with formulas used in proposed algorithm, results of proposed algorithm and results describes in section 4 and 5.

2. Related Work

There are different static and dynamic algorithms are present for load balancing. Static algorithms are easy as compared to dynamic algorithms but they are not suitable for. Various static [1-2] and dynamic [3-13] algorithms are discussed in this section. A comparison between two algorithms is performed in [1] where min-min and max-min are used. User requirements here are based on deadline provided by them. Comparison is done in two manners, space shared manner and time shared manner which is done using simulator CloudSim. In this comparison max–min algorithm has shown better results than min-min.

To improve resource utilization and processing time a new improved IBA algorithms is proposed in [2]. This algorithm works with balanced spiral for better QoS

parameters. Priority is considered foe tasks with better resource utilization. This priority based IBA algorithm with EASY backfilling provides better results than IBA and other backfilling algorithms.

A honey bee inspired algorithm for load balancing is proposed in [3] which focus on load of VMs. Rescheduling is performed when there is situation of underload. Honey bees behaviour is followed to balance load in cloud Computing. Honey and food sources in honey bee algorithm are conceptualized as resources and load. Under loaded VMs are paid more attention in this particular algorithm than overloaded resources.

Yongfei Zhu et al. [4] proposed am algorithm based on particles swarms optimization. This is used along with red black tree for load balancing. New improved algorithm shows better results in terms of tasks solving and time than PSO.

Mahapatra at el. [5] proposed a heuristic based ant colony optimization algorithm which focuses on delay, network load and CPU load. The pheromone is updated by incoming ants travelling from source to destination. However fault tolerance factor is not considered in this algorithm. An improved ACO algorithm is proposed in [5], which apart from original algorithm taking cost and time of tasks execution as main factors. Pheromone and inspired factor are improved in proposed algorithm with improving time, resource utilization and less cost For heterogeneous environment, a heuristic task scheduling algorithm named HABC is proposed (heuristic Artificial Bee Colony Algorithm) in [6]. In this algorithm large tasks are given priority over small tasks which have shown better use of resources. Data is distributive in two ways, normal distribution and data distribution. This algorithm has shown better results even if number of tasks increased.

A dynamic load balanced algorithm is proposed in [7] with constraints like elasticity and deadline in cloud. Aim of this dynamic algorithm is to minimize makespan time and improve the number of task meet the deadline specified by client. To make this happen task are first been sorted on the basis of deadline. In each interval number of tasks which are not meeting deadline numbers of virtual machine increased. Increment or decrement depends on overload and under load situation of resources. The results are compared with min-min, FCFS and SJF algorithm where proposed algorithm showing better results.

A hybrid of two algorithms SLA aware decentralized and JIQ algorithm is proposed in [8]. This algorithm focuses on balancing load between virtual machines. Through iterations response time of virtual machines is calculated and a threshold value is created using user request and number of VMs. SLA is created by response time and comparison is performed on RSA response time and VMs. If response time of VM is less than RSA then a compatible list is created. Task is assigned to resources with the use of JIQ basic on their availability. If VMs are not available they further balanced on the basis of response time.

Genetic algorithm works on natural selection approach. Number of tasks performed in GA is selection, crossover, and mutation. Several genetic algorithms are discussed in [9-11]. A combination of genetic algorithm with double fitness adaptive algorithm is proposed called JLGA. This algorithm takes short jabs first for scheduling. For population analysis greedy algorithm is used [9]. A comparison of genetic algorithm and JLGA is also performed through simulation. But priority is not set with this algorithm. Genetic algorithm with time as priority is used in [10] and population initialization is also done based on time. Time calculation is based on the length of the task. An enhanced generic algorithm is proposed in [11] which focus on makespan time. Load variation is comparatively less as fitness function is used for allocation of resources. Results are compared with ACO, PSO where GA shown better results.

An improved ACO algorithm is proposed in [23], which apart from original algorithm taking cost and time of tasks execution as main factors. Pheromone and inspired factor are improved in proposed algorithm with improving time, resource utilization and less cost

A heuristic based scheduling algorithm is proposed in order to handle the load for IaaS cloud. This algorithm is called HBLBA. This algorithm focuses on minimizing makespan time which directly reflects on cost and resource utilization for load balancing. This algorithm has been divided in two phases, one is server configuration and other is VM mapping. The strategy of this algorithm is based on whether to choose low capacity VM or high capacity VM for any task. This helps in maintain the makespan time.

3. Definitions and Problem Formulation

Load balancing in cloud environment requires an efficient scheduling algorithm that can distribute load on various virtual machines in such a way that cloud users can perform their different tasks with minimal makespan time, better resource utilization and more succeed tasks that failure tasks. Algorithm used for load balancing should be capable of giving good load balance level. In any load balancing algorithms users wants to execute their tasks within time (deadline) while service providers expect for proper resource utilization. Cloud service provider get n number of tasks T1, T2, T3……TN, requests which are independent of each other. Every task has tasks length and deadline which are as TL and DL. Cloud service provider

has M number of resources R1, R2, R3…..RM which are similar to each other in aspects of their processing speed, memory and bandwidth. Task scheduler assign different tasks Tj to different resources Rk and hen tasks gets assigned to virtual machines they are marked as assigned and if not than unassigned. Some tasks might get discarded in the process and gets executed later.

The main objective here is to increase the makespan time of scheduling algorithm with increasing number of succeed tasks considering length and deadline of tasks. Here are some definitions which will be used to describe the proposed algorithm.

Definition 1. (Execution time): execution time is the time taken to execute a task. Execution time can be calculated as:

ET=TLTi/m*n,

Where ET (execution time),
TL (task length),
m (processing speed of resources)
n (Number of processors)

Definition 2. (Makespan Time): makespan time is the maximum time to complete any task. Makespan time can be calculated as:

Makespan= max (task.CTi),

where i= 1:n,

CT is the maximum completion time of tasks to complete execution for tasks i. where variable 'i' can be in rage of 1 to n number of tasks.

Definition 3. (Average Resource Utilization): resource utilization is the

AvgRU= ((∑mi=1resui)/m)/ makespan,

Where m is number of tasks, resu is individual task utilization and makespan time is calculated above.

Definition 4. (Load balance level):

ru= ∑_(i=1)^m▒(avg-resu) 2

d= √ru/m

LBL= (1-d/AvgRU)*100

Definition 5. (Succeed tasks): succeed tasks are the task whose value is less than deadline value and executed successfully.

Definition 7. (Failure tasks): failure tasks are the tasks whose value was greater than deadline so discarded.

4. Proposed Algorithm

A dynamic algorithm of task scheduling is proposed in this paper. This developed algorithm decrease the makespan time, increase the succeed tasks and resource utilization. This algorithm is using sufferage algorithm which is modified using deadline as constraint.

4.1 Sorting

Before scheduling the tasks on machines they are sorted. All the tasks are sorted according to their deadline and length. Deadline here is treated as time interval for each tasks in which it should be executed. Deadline of each task is generated randomly every time, as the number of tasks taken are in 100s deadline is not provided manually for each task. Deadline is generated randomly for each task.

After generating deadline all tasks are sorted according to deadline first. The tasks with less deadline value will be given more priority to execute as we want to increase succeeding tasks. So tasks with minimal deadline value are executed first because main motive here is to execute more tasks in less time, so tasks which have less deadline value are given more priority.

To do so all task are sorted in ascending order according to their priority. If two tasks have same priority their length is considered to sort them. In such case whichever tasks is having less length will be executed first. Length here is in MI (Million Instructions).

4.2 Scheduling

Second step after sorting tasks according to their priority is scheduling. In this step a modified version of sufferage algorithm has been used as task scheduler. Sufferage works on MCT (Minimum Completion Time) where for every task execution time is calculated for virtual machines. A sufferage value is calculated which is difference of best machine for scheduling and second best machine. Instead of this virtual machine execution time for each tasks deadline is used as sufferage value. All sorted task are as input are schedule on machines. The task which has high priority get schedule first. If a task is schedule on machine it is marked as assigned task. Otherwise, execution time is calculated for tasks are then assigned. In this proposed algorithm first tasks sorted based on deadline and length are selected. To schedule the tasks suitable resources are selected which can execute that task in minimal time. For each task earliest completion time and second earliest time of task on machine to execute is calculated. If a machine has already assigned tasks new task will be added at the end. As task which are already added have high priority than new once.

Algorithm for task scheduling based on deadline and length

1. Generate M number of virtual machine.
2. We have to schedule N number of Task based on deadline.
3. ∀ Ti€{T1, T2 , T3 .....TN }
4. At the starting number of task assigned to resource is null

$$\phi \qquad RJ \leftarrow \{null\}$$

5. ∀ Rj €{R1 ,R2 ,R3 .....RM }
6. Find machine which gives best completion time

$$CTij=etij+rj$$

7. Sufferage value= deadline of tasks
8. If sufferage value of calculated tasks in less already assigned task
9. Delete tasks from resource
10. Assign task to new resource
11. Update completion time

4.3 Load Balance Level

In this phase we try to balance the load within resources to so this overloaded resources and under loaded resources are calculated based on makespan and maximum execution time. To balance the load between heavily loaded resource and under loaded resource shifting of task is performed. First heavily loaded tasks are found. These resources are found by makespan time of resources. Whichever resource is taking more time to execute tasks means load is more on that resource. Then task is picked from these resources and for this task scheduling is again performed. A suitable resource is founded by calculating minimum completion time. After calculating such virtual machine this tasks is assigned to that recourse. After each shifting if resource completion time of all tasks is calculated and updated. This calculated is performed with makespan time and execution time. Load balancing is done by using these steps:

•Heavily loaded resources are selected
•Pick task from heavily loaded resource
•Assign selected task to new best resource found
•Again calculate completion time

5. Result Analysis

The proposed algorithm is a dynamic algorithm so simulation results are the best way to calculate the performance in this section the simulation results are presented of proposed algorithm. The simulation programs are written in Java. Here comparison results of proposed algorithm as well as other existing algorithms mention above in terms of makespan, succeeded tasks, failed tasks, average resource utilization, load balance level over different data sets.

5.1 Makespan Time

Makespan time is the time difference between start and end sequence of jobs and tasks. Makespan is calculated for different number of tasks (100-170). Experiments are done for four different numbers of tasks i.e. 100, 120, 150 and 170. Results are affected by increasing or decreasing the number of tasks. These variations in results showing because same virtual machine can process the job quickly or can take more time. Simulation results have been run every time when tasks are changed. For these different values of tasks different results has been shown in graph below. In graph x-axis is showing number of tasks while y-axis is showing makespan time in seconds. It has been observed that the proposed algorithm is showing better results than existing algorithms. Makespan graph is made from running programs several times and for different number of tasks. The comparison of makespan time with other algorithms like min-min and max-min is performed. Comparison results showing that proposed algorithm is performing better than existing algorithms.

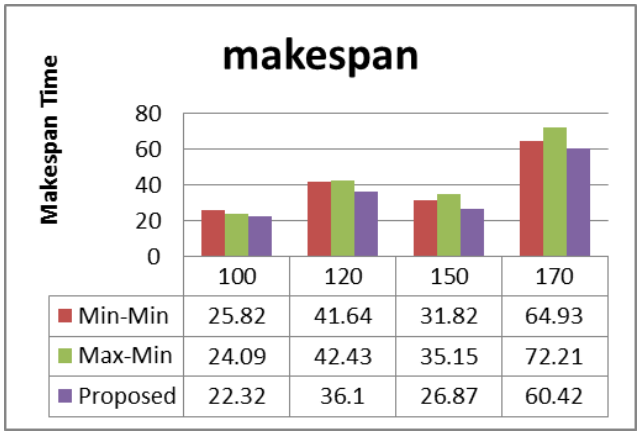
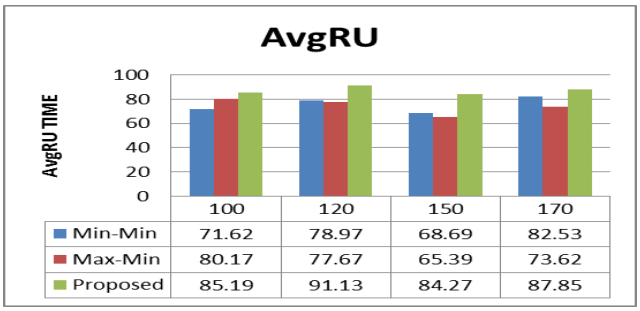


Figure 1 Makespan time Comparison

5.2 Resource Utilization

Resource utilization indicates how many times VMs reschedule to execute different tasks after their creation. Rescheduling of VMs may lead to minimizing the number of VMs. If VM utilization is higher it also contributes in more resource utilization. Average resource utilization (AvgRU) is calculated by formula given in problem and definition section. Simulation is showing different values for different number of tasks. Number of tasks is shown on

x-axis and AvgRU time is on y-axis. Results are varying when number of tasks is changed. For these different values of tasks different results has been shown in fig 2. It has been observed that the proposed algorithm is showing better results for AvgRU than existing algorithms.

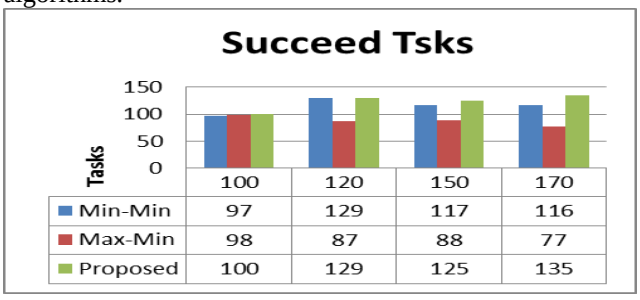


	100	120	150	170
Min-Min	71.62	78.97	68.69	82.53
Max-Min	80.17	77.67	65.39	73.62
Proposed	85.19	91.13	84.27	87.85

Figure 2 AvgRU Time Comparison

5.3 Successful Task

Succeed tasks are the tasks which completed on or before the deadline of tasks. Proposed algorithm considers deadline and length as main factors and selects efficient resource to schedule task which can execute tasks before deadline. Succeed tasks are calculated for different number of tasks (100-170). For these different values of tasks different results has been shown in fig 3. Results are compared with min-min and max-min algorithm. Form figure it can observed that the proposed algorithm is able to execute more tasks meeting than other algorithms.

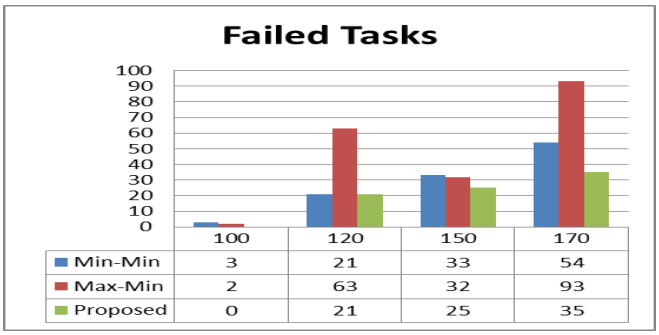


	100	120	150	170
Min-Min	97	129	117	116
Max-Min	98	87	88	77
Proposed	100	129	125	135

Figure 3 Succeed Tasks comparison

5.4 Failed Task

Failed tasks are those which couldn't meet the deadline and their execution time are high than the deadline. Number of tasks taken varies from 100 to 170 and number of virtual machines are 10 here. For these different values of tasks different results produced which has been shown in graph 4 where x-axis is showing number of tasks and y-axis is showing number odf failed tasks. It has been observed that the proposed algorithm is showing less number of failed tasks than min-min and max-min algorithm.

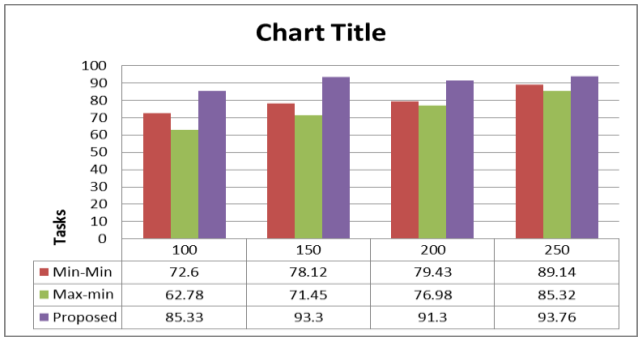


	100	120	150	170
Min-Min	3	21	33	54
Max-Min	2	63	32	93
Proposed	0	21	25	35

Figure 4 Failed tasks Comparison

5.5 Load Balance Level

Number of tasks taken varies from 100 to 170 and number of virtual machines is 10 here. For these different values of tasks different results produced which has been shown in figure 4, where x-axis is showing number of tasks and y-axis is showing number of failed tasks. It has been observed that the proposed algorithm is showing less number of failed tasks than min-min and max-min algorithm.



	100	150	200	250
Min-Min	72.6	78.12	79.43	89.14
Max-min	62.78	71.45	76.98	85.32
Proposed	85.33	93.3	91.3	93.76

Figure 5. Load Balance Level

Conclusions

This paper proposed a dynamic load balancing algorithm. To balance the load among virtual machines fairly algorithm is divided into three phases: sorting, scheduling and load balance level. In the first phase all arriving tasks are sorted according to deadline of tasks and if two tasks have same priority their length is considered for scheduling. And whichever tasks have less length are taken

first. Tasks with grater execution time than their deadline value are considered as failed tasks and discarded. This is done to meet more tasks to deadline and improve number of succeeding tasks. For scheduling modified sufferage algorithm is used. Which based on deadline, schedule task on machine which will gives minimum execution time for that task. Last phase is load balance level where heavily loaded machines and under loaded machines are balanced. And load balance level is maintained. Proposed algorithm is compared with modified min-min and max-min algorithms. Computational results shows better results for proposed algorithm in context of makespan time, AvgRu and succeed tasks.

References

[1] Kalita, R., & Patnaik, H. (2014, May). A novel heuristic resolving deadline-oriented task scheduling in cloud. In Advanced Communication Control and Computing Technologies (ICACCCT), 2014 International Conference on (pp. 1137-1142). IEEE.\

[2] Dubey, K., Kumar, M., & Chandra, M. A. (2015, March). A priority based job scheduling algorithm using IBA and EASY algorithm for cloud metaschedular. In Computer Engineering and Applications (ICACEA), 2015 International Conference on Advances in (pp. 66-70). IEEE.

[3] Babu, K. R., & Samuel, P. (2016). Enhanced bee colony algorithm for efficient load balancing and scheduling in cloud. In Innovations in bio-inspired computing and applications (pp. 67-78). Springer, Cham.

[4] Zhu, Y., Zhao, D., Wang, W., & He, H. (2016, January). A Novel Load Balancing Algorithm Based on Improved Particle Swarm Optimization in Cloud Computing Environment. In International Conference on Human Centered Computing (pp. 634-645). Springer, Cham.

[5] Mahapatra, D., Saini, G. K., Goyal, H., & Bhati, A. ANT COLONY OPTIMIZATION: A SOLUTION OF LOAD BALANCING IN CLOUD.2016.

[6] Kimpan, W., & Kruekaew, B. (2016, August). Heuristic Task Scheduling with Artificial Bee Colony Algorithm for Virtual Machines. In Soft Computing and Intelligent Systems (SCIS) and 17th International Symposium on Advanced Intelligent Systems, 2016 Joint 8th International Conference on (pp. 281-286). IEEE.

[7] Kumar, M., & Sharma, S. C. (2017). Deadline constrained based dynamic load balancing algorithm with elasticity in cloud environment. Computers & Electrical Engineering.

[8] Choudhary, M., Chandra, D., & Gupta, D. (2017, May). Load balancing algorithm using JIQ methodology for virtual machines. In Computing, Communication and Automation (ICCCA), 2017 International Conference on (pp. 730-735). IEEE.

[9] Wang, T., Liu, Z., Chen, Y., Xu, Y., & Dai, X. (2014, August). Load balancing task scheduling based on genetic algorithm in cloud computing. In Dependable, Autonomic and Secure Computing (DASC), 2014 IEEE 12th International Conference on (pp. 146-152). IEEE.

[10] Makasarwala, H. A., & Hazari, P. (2016, June). Using genetic algorithm for load balancing in cloud computing. In Electronics, Computers and Artificial Intelligence (ECAI), 2016 8th International Conference on (pp. 1-6). IEEE.

[11] Sharma, Harshdeep, and Gianetan Singh Sekhon. "Load Balancing in Cloud Using Enhanced Genetic Algorithm." (2017).

[12] Qingbin, N., & Pinghua, L. (2016, October). An Improved Ant Colony Optimization Algorithm for Improving Cloud Resource Utilization. In Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2016 International Conference on (pp. 311-314). IEEE.

[13] Adhikari, M., & Amgoth, T. (2018). Heuristic-based load-balancing algorithm for IaaS cloud. Future Generation Computer Systems, 81, 156-165.

Miss Sangeeta is a M.Tech student of DRUST, Murthal in CSE Dept. she has done her bachelor degree from Kurukshetra University in Computer Science. Her research area for is cloud computing.

Mrs. Suman has been into teaching and research for about 14 years. She did her Ph.D. from Deenbandhu Chhotu Ram University of Science and Technology, Murthal (Haryana) INDIA. Her research areas include Network Security and Heterogeneous Wireless Networks. She received her M.Tech. degree in Computer Science & Engineering from Kurukshetra University, Kurukshetra, INDIA. She has published more than 20 papers in various journals and conferences of repute.