International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-9 Issue-1, November 2019

Support Vector Regression based Mapreduce Throttled Load Balancer for Data Centers

C.R. Durga Devi, R. Manicka Chezian

Abstract: ***The process of analyzing big data and other valuable information is a significant process in the cloud. Since big data processing utilizes a large number of resources for completing certain tasks. Therefore, the incoming tasks are allocated with better utilization of resources to minimize the workload across the server in the cloud. The conventional load balancing technique failed to balance the load effectively among data centers and dynamic QoS requirements of big data application. In order to improve the load balancing with maximum throughput and minimum makespan, Support Vector Regression based MapReduce Throttled Load Balancing (SVR-MTLB) technique is introduced. Initially, a large number of cloud user requests (data/file) are sent to the cloud server from different locations. After collecting the cloud user request, the SVR-MTLB technique balances the workload of the virtual machine with the help of support vector regression. The load balancer uses the index table for maintaining the virtual machines. Then, map function performs the regression analysis using optimal hyperplane and provides three resource status of the virtual machine namely overloaded, less loaded and balanced load. After finding the less loaded VM, the load balancer sends the ID of the virtual machine to the data center controller. The controller performs migration of the task from an overloaded VM to a less loaded VM at run time. This in turn assists to minimize the response time. Experimental evaluation is carried out on the factors such as throughput, makespan, migration time and response time with respect to a number of tasks. The experimental results reported that the proposed SVR-MTLB technique obtains high throughput with minimum response time, makespan as well as migration time than the state -of -the -art methods.***

Keywords: Big data processing, Support Vector Regression, hyperplane, Throttled Load Balancing, MapReduce function

I. INTRODUCTION

Load-balancing in big data centers is the process of frequently distributing more data between the servers. Load balancing helps to achieve the faster response, and also minimize the overload problems while handling a large number of tasks. Therefore, the appropriate load balancing method is needed to improve the throughput by balancing the load. In general, two types of load balancing are performed such as static and dynamic. The static load balancing considers previous information of the system resources at the time of execution. The Dynamic Load Balancing considers the current state information and assigns the tasks to the system during run time and this information is varied as the situations changes. There are several techniques has been developed to balance the workload among the virtual machine. The major contribution of the SVR-MTLB technique is summarized as follows, To increase the throughput and minimize the makespan, SVR-MTLB technique is introduced. The support vector regression used in the load balancing algorithm for finding the current resource status of the virtual machine in the index table. The support vector regression uses hyperplane as a boundary for identifying the overloaded and less loaded virtual machine. The load balancer assigns the tasks to the less loaded virtual machine.

To minimize the migration time, SVR-MTLB technique effectively identifies the current load of the virtual machine. The load of the virtual machine is identified through the capacity of the virtual machine, bandwidth and memory. The load balancer sends the less loaded virtual machine ID to the data center controller. The controller decides the migration of tasks from the overloaded virtual machine to less loaded virtual machine. To minimize response time, the SVR-MTLB technique uses the MapReduce function for effectively calculating the resource capabilities of the virtual machine for predicting the less loaded VMs and minimizing the waiting time as well as service time of the virtual machine.

This paper is organized into five different sections as follows. Section 2 provides a related research works in load balancing. Section 3 provides the process of SVR-MTLB with neat diagram. Experimental evaluation of SVR-MTLB technique and state-of-art methods is presented in section 4. Section 5 provides the experimental results and discussion with certain parameters. Finally, the conclusion of the paper is presented in section 6.

II. RELATED WORK

A dynamical load-balanced scheduling (DLBS) technique was introduced in [1] for achieving the higher throughput while handling the large number of tasks. The designed DLBS technique significantly minimizes the transmission delay but it failed to consider the makespan minimization.

Revised Manuscript Received on November 30, 2019.
***** Correspondence Author
C.R. Durga Devi *, Ph.D Research Scholar, Department of Computer Science, NGM College, Pollachi, India Email: deviswe@gmail.com
Dr. R. Manicka Chezian, Associate Professor, Department of Computer Science, NGM College, Pollachi, India Email: chezian_r@yahoo.co.in

A Resource-Aware Load Balancing Algorithm (RALBA) was designed in [2] to balance the workload based on resource capabilities of a virtual machine. The algorithm increased the throughput as well as minimized the makespan but the performance of task migration was not improved.

A load-balancing algorithm was designed in [3] using switch migration based on response time. The algorithm quickly balances the overloaded virtual machines but the throughput was not improved. Hybrid workload migration mechanism was introduced in [4] to transfer the workload and avoid the latency of client requests. The designed mechanism failed to minimize the time for workload migration. A modified Central Load Balancer (MCLB) algorithm was introduced in [5] for balancing the workload between the virtual machines by avoiding overloading and underloading of virtual machines. The algorithm failed to consider the migration, processor capacity as well as memory utilization. Collaborative agent-based problem-solving method was developed in [6] for balancing the workloads between the servers with the help of virtual machine migration. The method failed to predict the current resource usage of the virtual machine.

A Flow Distribution-Aware Load Balancing (FDALB) technique was introduced in [7] to minimize the flow completion times and reduces the overhead of the controller. The technique did not perform the migration for minimizing the response time. A geographical load balancing (GLB) technique was developed in [8] for balancing the workload between the data centers. The technique failed to minimize the request processing delay within data centers and more types of workloads.Fat-tree data center virtualization framework was developed in [9] for balancing the global load using migration of the virtual machine. But the framework failed to improve the throughput. A dynamic scheduling technique was presented in [10] for balancing the workload among all the virtual machines using task migration. Though the technique reduces the makespan, the throughput was not improved. A new heuristic technique called Load Balancing based on Bayes and Clustering (LB-BC) was designed in [11]. The technique achieved higher throughput and minimum makespan but the performance of task migration remained unsolved. A Distributed Stochastic Geographical Load Balancing (DGLB) technique was introduced in [12]. The technique failed to perform the migration and does not consider the resource availability of the virtual machine for minimizing the response time. Virtual Machine based federate migration technique was introduced in [13] to minimize the tasks execution time. But the designed technique was not flexible for balancing the load with minimum response time. A hierarchical load balancing strategy based on a generalized neural network (HLBSGNN) was introduced in [14] for minimizing the load balancing overhead and makespan. The strategy failed to improve network throughput. A distributed and scalable load balancing method was introduced in [15] using game theory to minimize the workload among the datacenters. But the method failed to effectively reduce the makespan. An OpenFlow-based Dynamic Load Balancing approach was developed in [16] for increasing the throughput. But the designed approach did not minimize the response time and makespan.

Ant colony optimization with particle swarm (ACOPS) was introduced in [17] for balancing the load between the virtual machine by considering the memory, CPU utilization and disk utilization. Though the optimization technique reduces the average makespan, the network throughput was not maximized. A novel load balancing approach using Genetic Algorithm (GA) was introduced in [18] to balance the load of cloud infrastructure and lessen the makespan of the incoming tasks. The approach failed to improve load balancing efficiency. A novel parallel programming model–Map-Balance-Reduce (MBR) was introduced in [19] for balancing the workload with minimum task execution time. The model failed to consider the efficient load balancing algorithm to achieve higher throughput with minimum response time. A proactive workload management scheme was introduced in [20] for reliable workload prediction with the minimum time. The scheme did not consider the current status of the virtual machine for minimizing the workload. Fuzzy neural network classification was introduced in [21] to handle server clusters and hybrid genetic- cuckoo search algorithm is used for fitness evaluation. Caching mechanism is used to optimize memory. Centralized hierarchical cloud-based multimedia system was introduced in [22].This method overcomes the load balancing problem.

The conventional load balancing frameworks has a few limitations for example, high makespan, less throughput, more response time and migration time, failure to consider the resource of the systems and so on. Motivated by the above-said limitations, a new technique called Support Vector Regression based MapReduce Throttled Load Balancing (SVR-MTLB) is introduced to maximize the network throughput by dynamically balancing the workload. The major issues are identified from the above-said literature are overcome by introducing an SVR-MTLB technique. The load balancing across the virtual machine using SVR-MTLB technique is described in the next section.

III. METHODOLOGY

Load Balancing is the process of migrating the user requested tasks from a heavy loaded virtual machine into a less loaded virtual machine at a run time by using the current status of resources**.**

Load balancing aims to minimize the makespan and maximize the throughput. Resource Optimized Traffic Aware Gradient Boosting Classification (ROTAGBC) technique was introduced in [23] for traffic aware geo distributed big data analytics. The task assigner performs priority task classification of incoming tasks using gradient Boosting ensemble classifier. Task assigner distributes the classified tasks over the multiple data centers at different locations with minimum task completion time and optimal resource utilization. Gradient Descent firefly optimized Round Robin Scheduling is applied in [24] to find the resource optimized virtual machine among the number of the virtual machines based on light intensity.

The virtual machine which utilizes the minimum resource is chosen for handling high priority task.

There are several techniques has been designed for balancing the workload across several data centers with available resources.

But the algorithm failed to handle a large number of tasks with minimum makespan is still challenging issues. Therefore, the Support Vector Regression based MapReduce Throttled Load Balancing (SVR-MTLB) technique is developed to improve the throughput with minimum task completion time. The architecture diagram of proposed SVR-MTLB technique is illustrated in figure 1. Initially, the cloud server collects the number of requested tasks $UT_1, UT_2, UT_3, \ldots, UT_n$ from the user. After that, the load balancing algorithm uses the MapReduce function to send the user requested tasks to the less loaded virtual machine with minimum time for migrating the tasks. The loading capacity of the virtual machine is identified through the support vector regression.

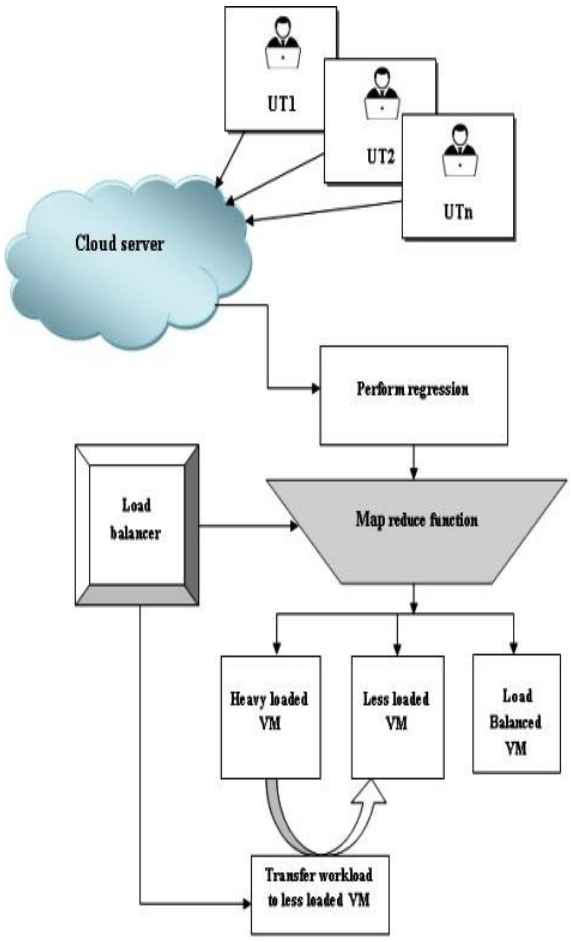

Figure 1 architecture diagram of SVR-MTLB technique

Figure 1 illustrates the architecture diagram of proposed SVR-MTLB technique for increasing the load balancing with higher throughput and minimum task completion time. The detailed process of proposed SVR-MTLB technique is explained as follows.

A. Support Vector Regression Based Mapreduce Throttled Load Balancing

The large numbers of user tasks are allocated to the several virtual machines in data centers based on their current load. The processing speed, capacity, bandwidth and memory are considered as major resources of the Virtual Machine (VM) for processing the large number of tasks which results in minimizes the workload across the datacenters. The conventional Round Robin algorithm distributes the incoming tasks into the VMs in the order of circular manner. The idea of Round Robin is effective that the entire VMs have similar processing capacity. As for data centers there are large VMs having more processing capability. In this case, the round-robin scheduling algorithm not effectively minimizes the response time, but support vector regression based MapReduce throttled load balancing minimizes the response time when the number of tasks increased. In SVR-MTLB technique, the load balancer uses the MapReduce function for finding the current status of the virtual machine i.e. less loaded, heavy loaded, balanced load using support vector regression. The MapReduce is a programming model for processing the big data (i.e. user requested tasks) in a parallel manner. The Map phase performs the regression analysis to find the heavy loaded virtual machine among the number of the virtual machines. After finding the status of the virtual machine, the load balancer migrating the incoming tasks from the heavy loaded into the less loaded virtual machine. The reducer phase executes after map phase, and minimizes the workload across the data centers. This load balancer helps to dynamically distributes the workload across the virtual machines. The flow process of the load balancing algorithm is shown in figure 2.

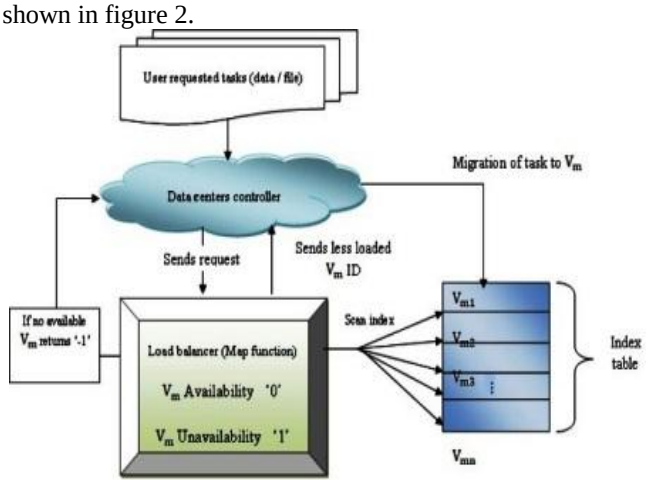

Figure 2 Flow process of load balancing

Figure 2 illustrates the flow process of a load balancing algorithm which comprises the data center controller (DC_r), a load balancer (LB_r) and the number of virtual machines $V_{m1}, V_{m2}, V_{m3}, \ldots V_{mn}$. Initially, the numbers of user requests (i.e data or file) are sent to the cloud data center. Then the cloud data center controller transmits the requests to load balancer.

$$DC_r \overset{UR}{\Rightarrow} LB_r \qquad (1)$$

In (1), DC_r is the data center controller, denotes a load balancer, UR is the user requested tasks. After receiving the user requested data or file from the data center, the load balancer maintaining the entire virtual machine list using an index table for identifying the current status of the virtual machines.

$$\mathrm{LB_r} = \begin{cases} 1; Status\ of\ V_m\ 's\ is\ not\ available \\ 0;\ Status\ of\ V_m\ 's\ is\ available \end{cases} \quad (2)$$

From (2), the status of V_m 's are identified through the two binary codes namely '1' and '0'. Initially, the data center sends a request to identify the status of the virtual machine. Then the load balancer sends the virtual machine availability to data center controller. Then the load balancer starts to find the current resource status of the virtual machine by searching the index table based on the regression analysis. The support vector regression is a statistical measurement used to determine the relationship between the dependent variable (i.e. less loaded, heavy loaded, balanced load) and independent variables (i.e. number of virtual machines $\mathrm{V_{m1}, V_{m2}, V_{m3}, \ldots . V_{mn}}$). The support vector regression is the machine learning method which uses one Hyperplane and two marginal Hyperplane for finding the current resource status of the virtual machine. The regression determines the capacity of the virtual machine, memory and bandwidth. The capacity of the virtual machine is calculated as the product of the number of processing element used and the processing speed. It is mathematically calculated as follows,

$$V_m\ (c) = Number\ of\ p_e * p_s\ (V_m) \quad (3)$$

In (3),$V_m\ (c)$ represents the capacity of the virtual machine, p_e denotes a processing elements which performs arithmetic and logic operations. $p_s\ (V_m)$ denotes a processing speed of the virtual machine in terms of Million Instructions Per Second (MIPS). After that, the current memory capacity of the virtual machine is calculated based on the difference between the total memory and the consumed memory which is given below,

$$M_{V_m} = M_t - M_C \quad (4)$$

From (4), M_{V_m} represents a memory of the virtual machine and M_t denotes a total memory space and M_C denotes an consumed memory space.The bandwidth of the virtual machine is measured as the difference between the total bandwidth and consumed bandwidth. The current status of the bandwidth is mathematically estimated as follows,

$$Bw_{V_m} = Bw_t - Bw_C \quad (5)$$

In (5), Bw_{V_m} denotes a bandwidth of virtual machine, Bw_t represents a total bandwidth and Bw_C denotes a consumed bandwidth. Based on the above said parameters, the support vector regression finds the loading capacity of the virtual machine.

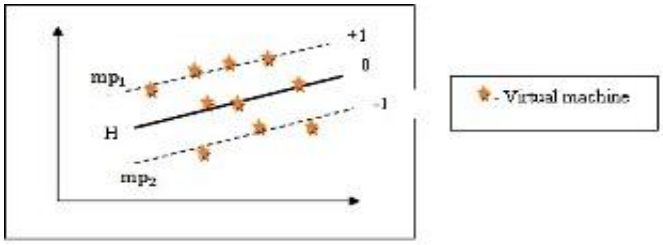

Figure 3 support vector regression

Figure 3 shows the support vector regression using optimal separating hyperplane (H). The region surrounded by the two marginal hyperplanes are represented as $\mathrm{mp_1}$ and $\mathrm{mp_2}$. The regression coefficient is determined using the following equations,

$$\rho = \omega . x + b \quad (6)$$

In (6),ρ denotes a support vector regression coefficient, ω denotes a weight vector of the input samples x (i.e. virtual machine), b represents the bias. The support vector regression coefficient analyzes the current resource status of the each virtual machine and provides the output results from -1 to +1. The virtual machine which has the minimum capacity, bandwidth and memory at a run time is identified as an overloaded. Otherwise, the status of the virtual machine is below the threshold (i.e. 0) is identified as a less loaded.

$$Y = \begin{cases} \omega . x + b = 0 & , load\ balanced\ V_m \\ \omega . x + b = +1 & ,over\ loaded\ V_m \\ \omega . x + b = -1 & ,less\ loaded\ V_m \end{cases} \quad (7)$$

In (7), Y denotes a output of the regression analysis. Based on the regression coefficient result, the load balancer identifies the heavy loaded virtual machine among the number of the virtual machine in the index table with the help of map function resulting in minimizes the migration time of the load balancer. After that, the load balancer transmits a less loaded virtual machine ID to the data center controller.

$$\mathrm{LB_r} \xrightarrow{\mathrm{Vm\ ID}} \mathrm{DC_r} \quad (8)$$

In (8),$\mathrm{LB_r}$ denotes a load balancer sends ID of the virtual machine Vm to datacenter controller $\mathrm{DC_r}$. After receiving the ID, the data center controller migration of the user requested task from the heavy loaded virtual machine to the less loaded virtual machine. The $\mathrm{DC_r}$ acknowledges the $\mathrm{LB_r}$ of new allocation and update the index table. Followed by,

the load balancer updates the VM into the busy Index and wait for the new request from the Data Center Controller. If the status of the virtual machine is unavailable (i.e. empty), then the load balancer returns a value of '-1' to the Data Center Controller. Then the data center controller arranges the incoming user request into the queue.

After processing the user request, DC_r receives a response from the efficient virtual machine, and it informs the LB_rthat the virtual machine has finished the job. As a result, the reduce phase displays the output results and minimizes the workload across the data center as well as traffic level. Therefore, the MapReduce function in SVR-MTLB technique minimizes the response time while processing a large number of user tasks. The algorithmic process of the proposed SVR-MTLB technique is described as follows.

Algorithm 1 describes the support vector regression based MapReduce throttled Load Balancing with minimal makespan and higher throughput. For each incoming tasks, the load balancer calculates the resource status of the virtual machine using support vector regression function. The regression function used to determine the less loaded, overloaded and balanced load of the virtual machine. Then the map phase identifies the less loaded virtual machine. Then the load balancer sends the ID of the less loaded virtual machine and makes the decision to decide the migration of task from an overloaded VM to a less loaded VM at a run time. Based on the decision of the load balancer, the data center controller migrates the workload to the less loaded virtual machine with minimum time. As a result, minimizes the workload across the data centers. As a result, the MapReduce function effectively handling a large number of incoming tasks which results in minimize the traffic level. Therefore, the less loaded virtual machine selection in the proposed technique minimizes the response time.

ALGORITHM 1

Input: Number of users requested tasks $UT_1, UT_2, UT_3, \ldots . UT_n$, number of virtual machines $V_{m1}, V_{m2}, V_{m3}, \ldots . V_{mn}$

Output: Maximize throughput and minimize makespan

Begin

1. **Send users requested tasks**UR_i **to** DC_r
2. DC_r sends request to LB_r
3. LB_r sends the status of the virtual machine to DC_r
4. **If the status of** V_m **is available**
5. LB_r returns the value '0'
6. **For** each V_m
7. Compute the current status of resources $V_m\ (c), M_{V_m}, Bw_{V_m}$
8. LB_r uses the map function to perform regression
9. **If** $(\rho = -1)$ **then**
10. Status of the virtual machine is less loaded
11. **else if** $(\rho = +1)$ **then**
12. Status of the virtual machine is overloaded
13. **else if** $(\rho = 0)$ **then**
14. Balanced load across virtual machines
15. **End if**
16. **End for**
17. LB_r finds less loaded V_m
18. LB_r sends Vm ID to DC_r
19. DC_r migration of UR_i to less loaded V_m
20. DC_r acknowledges LB_r of new allocation and alters the index table
21. LB_rupdates the VM into the busy Index
22. DC_r receives a response from specified V_m and notify to $LB_r$is stopped
23. **else**
24. LB_r returns the value '-1' to DC_r
25. **End if**

End

IV.EXPERIMENTAL SETTINGS

Experimental evaluation of proposed SVR-MTLB technique and existing methods novel dynamical load-balanced scheduling (DLBS) [2] and Resource-Aware Load Balancing Algorithm (RALBA) [2] are implemented

using Java language with CloudSim network simulator. Personal Cloud Datasets (http://cloudspaces.eu/results/datasets) is taken for the experimental evaluation. The main objective of the dataset is to transfer the workload. The dataset comprises 17 attributes and 66245 instances. The 17 attributes are row id, account id, file size (i.e. task size), operation_time_start, operation_time_end, time zone, operation_id, operation type, bandwidth trace, node_ip, node_name, quoto_start, quoto_end, quoto_total (storage capacity), capped, failed and failure info. Among the 17 attributes, two columns are not used such as time zone and capped. The above columns are considered for efficient load balancing among the multiple virtual machines using big data in the cloud. Experimental results and discussion of proposed SVR-MTLB technique and existing DLBS [1] and RALBA [2] are described with various performance metrics such as throughput, makespan, migration time and response time with the number of tasks (i.e. user requested tasks).

V. RESULTS AND DISCUSSIONS

Experimental results of proposed SVR-MTLB technique and existing DLBS [1] and RALBA [2] are discussed in this section with different performance metrics such as throughput, makespan, migration time and response time. The comparison results of three methods and their experimental results are explained using table values and graphical representation. For each subsection, the sample mathematical calculation is provided.

A. Performance Analysis of Throughput

Throughput is defined as the number of user-requested tasks is executed per unit time by a virtual machine. The mathematical formula for calculating the throughput is given below,

$$Throughput\ (T) = \frac{Number\ of\ task\ \ executed}{unit\ time\ (s)} \quad (9)$$

By using (9), $Throughput\ (T)$ is calculated which is measured in terms of tasks per second (tasks/sec).

Table 1 Throughput versus number of tasks

Number of tasks	Throughput (tasks/sec)		
	SVR-MTLB	DLBS	RALBA
25	12	10	9
50	23	16	12
75	36	30	25
100	42	36	31
125	52	48	43
150	63	58	51
175	76	65	60
200	83	75	69
225	95	87	79
250	113	100	93

Table 1 describes the experimental results of throughput with respect to a number of user-requested tasks. For the experimental consideration, the numbers of tasks are taken from 25 to 250. The above result shows that the throughput of the proposed SVR-MTLB technique is considerably improved than the other methods DLBS [1] and RALBA [2]. The reported results are plotted in the two-dimensional graph as shown below.

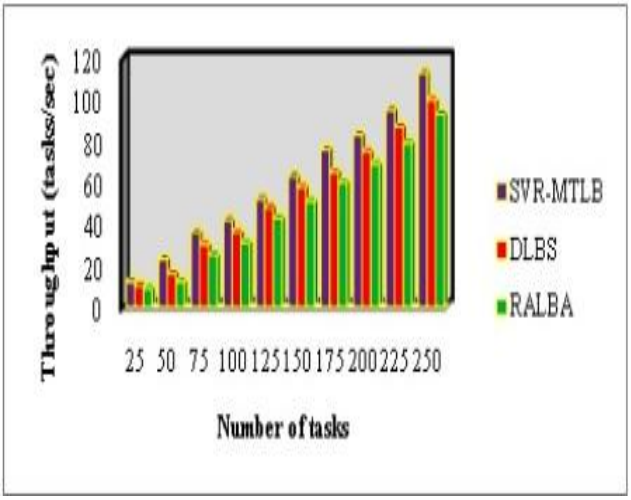

Figure 4 performance analysis of Throughput versus number of tasks

Figure 4 illustrates the experimental results of throughput based on the number of user tasks using three methods namely SVR-MTLB technique, DLBS [1] and RALBA [2]. The numbers of user tasks are taken as input in 'x' direction and the corresponding throughput results are obtained at 'y' direction. The above figure clearly shows that SVR-MTLB technique increases the throughput. This is because of the proposed load balancing algorithm effectively analyzing the resource capacity of each virtual machine using support vector regression. Based on the capacity of the virtual machine, the load balancer identifies the task execution of the virtual machine. If the virtual machine is overloaded, it executes number of tasks per unit time. When the machine is less loaded, it executes less number of tasks per unit time interval. Therefore, the SVR-MTLB technique balances the workload of the overloaded virtual machine resulting in increases the number of tasks execution per unit time.

Let us consider the number of user tasks is 25, 12 tasks executed per second by the virtual machine using SVR-MTLB technique. By applying other two existing DLBS [1] and RALBA [2], the virtual machine executes 10 and 9 tasks respectively. Similarly, the remaining runs are performed with a number of input tasks. Then the proposed results are compared with the existing throughput results. The comparison results clearly show that SVR-MTLB technique maximized the throughput by 17% and 34% than the conventional load balancing techniques DLBS [1] and RALBA [2] respectively.

B. Performance Analysis of Makespan

Makespan is the amount of time taken to complete all the user tasks submitted to the system. The makespan is mathematically computed as the time difference between the starting and finishing the tasks.

$$MS = t_f - t_s \quad (10)$$

In (10),MS represents the makespan, t_f denotes a number

tasks finishing time, t_s represents a tasks starting time. The makespan is measured in the unit of milliseconds (ms).

Table 2 Makespan versus number of tasks

Number of tasks	Makespan (ms)		
	SVR-MTLB	DLBS	RALBA
25	20	24	27
50	23	26	30
75	26	29	31
100	28	31	35
125	31	34	38
150	34	38	41
175	36	41	45
200	38	43	47
225	43	45	48
250	46	49	52

Table 2 describes the performance results of makespan versus a number of tasks. The different makespan results are reported for a different number of user tasks. When the number of tasks increased, the makespan also increased. But the task completion time of proposed SVR-MTLB technique is minimized when compared to other load balancing approaches DLBS [1] and RALBA [2] respectively. The performance results of makespan are shown in the two-dimensional graphical representation.

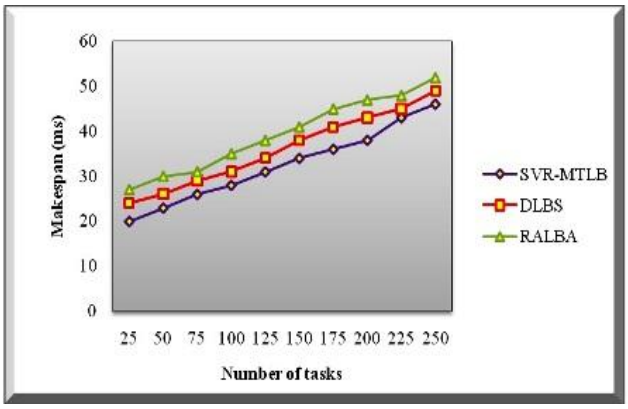

Figure 5 performance results of makespan versus number of tasks

Figure 5 depicts the makespan performance results of three different load balancing algorithms namely SVR-MTLB technique, DLBS [1] and RALBA [2] which are represented by three different colors violet, red and green respectively. The figure clearly shows that the SVR-MTLB technique minimizes the makespan when compared to existing load balancing algorithms. This is because, the data center controller checks the availability of the virtual machine. The load balancer maintains the index table for all the virtual machines. Then the load balancer uses the MapReduce function to map all incoming tasks to all the virtual machine. If any virtual machine overloaded, then the map function finds the less loaded virtual machine through the regression analysis. After that, the reduce phase combines individual output from each virtual machines to produce the final result. By this way, SVR-MTLB technique minimizes the makespan. The average of ten different results shows that the proposed SVR-MTLB technique minimizes the makespan by 10% and 19% when compared to DLBS [1] and RALBA [2] respectively.

C. Performance Analysis of Migration Time

Migration time is defined as an amount of time required to migrate a task from one virtual machine to another. The Migration time is calculated using the following mathematical formula,

$$MT = number\ of\ tasks * T\ (migrate\ one\ task) \quad (11)$$

Where MT denotes a Migration time, T denotes a time taken for migrating the tasks. The migration time is measured in milliseconds (ms).

Table 3 Migration time versus number of tasks

Number of tasks	Migration of tasks	Migration time (ms)		
		SVR-MTLB	DLBS	RALBA
25	5	6	7	8
50	9	7	9	11
75	15	12	15	17
100	25	15	17	19
125	32	16	18	20
150	38	17	19	21
175	43	18	20	22
200	46	19	21	22
225	50	20	22	24
250	52	21	23	26

The performance results of the migration time versus a number of user-requested tasks are described in table 3. It is clear that the migration time using proposed SVR-MTLB Technique is lower than other state-of-the-art works. In table 3, let us consider the 25 tasks taken as input and the migrations of tasks are 5. Then the time taken for migrating the tasks from the overloaded virtual machine to less loaded virtual machine is 6ms using SVR-MTLB Technique.

The migration time of DLBS [1] and RALBA [2] are 7ms and 8ms respectively. The performance analysis of migration time is shown in figure 6.

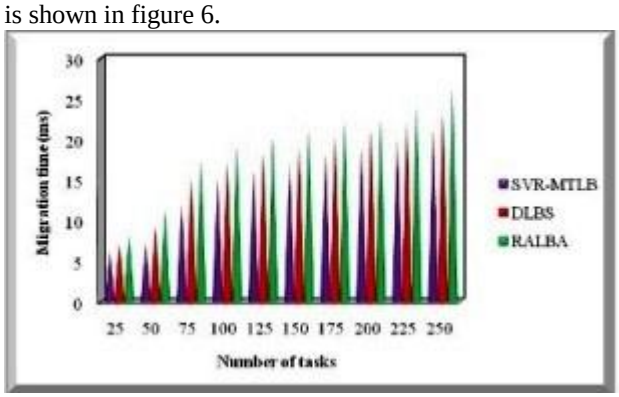

Figure 6 performance analysis of migration time versus number of tasks

Figure 6 depicts the performance results of migration time using a number of user tasks. As shown in figure 6, the results of migration time are minimized using SVR-MTLB technique. This significant improvement of the SVR-MTLB technique uses the dynamic load balancing mechanism and the load balancer decides the migration of task from an overloaded VM to a less loaded VM at a run time.

The proposed dynamic load balancing algorithm uses the support vector regression to identify the current load of the virtual machine. The load of the virtual machine is identified through the capacity of the virtual machine, bandwidth and memory.

As a result, the task migration in SVR-MTLB technique is performed in efficient manner due to extensive dynamic load balancing algorithm discovering the most appropriate VM to each of the tasks. The experimental results clearly illustrate that the SVR-MTLB technique considerably minimizes the task migration time by 13% when compared to DLBS [1]. Similarly, the migration time of the SVR-MTLB technique is compared with the RALBA [2]. The comparison results clearly show that the task migration time is minimized by 22% using SVR-MTLB technique than the RALBA [2].

D. Performance Analysis of Response Time

Response time is defined as an amount of time required by the data center to respond to a task. It is measured as the sum of the transmission time, waiting time and service time. The mathematical formula for calculating the response time is expressed as follows,

$$RT = T_{tr} + T_w + T_s \qquad (12)$$

In (12), RT denotes a response time, T_{tr} denotes a task transmission time, T_w represents the waiting time, T_s denotes a service time. The response time is measured in milliseconds (ms).

Table 4 Response time versus number of tasks

Number of tasks	Response time (ms)		
	SVR-MTLB	DLBS	RALBA
Task 1	1	1.3	1.4
Task 2	1.2	1.5	1.6
Task 3	1.4	1.6	1.8
Task 4	1.3	1.4	1.6
Task 5	1.6	1.7	1.9
Task 6	1.8	1.9	2.1
Task 7	1.2	1.8	2
Task 8	1.7	1.9	2.2
Task 9	1.9	2.1	2.3
Task 10	2.2	2.4	2.6

Table 4 illustrates the response time of each task given to the system. The above table shows that the response time is calculated for each user requested task. For each task, the response time is calculated based on the transmission time of the task, task waiting time and service time of the virtual machine. Totally ten different tasks are considered for calculating the response time which is shown in table 4.

The reported results are plotted in the two-dimensional graphical representation. The experimental results confirm that the SVR-MTLB technique minimizes the response time when compared to existing load balancing algorithm

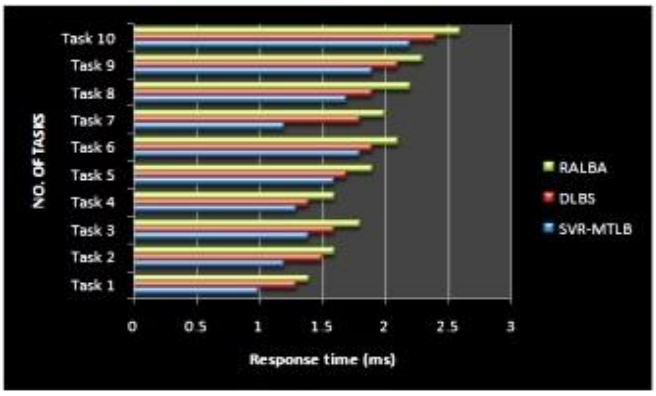

Figure 7 Performance results of response time versus number of tasks

Figure 7 depicts the performance results of response time with respect to a number of tasks. As shown in the figure, different results of response time are obtained for different tasks.. In SVR-MTLB technique, the dynamic load balancing technique called support vector regression based MapReduce throttled Load balancing is employed to find the efficient of VMs under varying load patterns. This helps to achieve the faster response of the user requested tasks from the datacenter. For each iteration, the task transmission time of SVR-MTLB technique is different and the waiting time and service time also minimum in the SVR-MTLB technique.

This is because of the SVR-MTLB technique uses the map function for effectively calculating the resource capabilities of the virtual machine for predicting the less loaded VMs and avoiding the overload on any of the VMs with the help of the regression analysis.

As a result, SVR-MTLB technique effectively balances the workload among the multiple virtual machines in the datacenter resulting increases the response of the incoming tasks. The response time of SVR-MTLB technique is compared with the response time of the existing results. The comparison results show that the response time of SVR-MTLB technique is considerably minimized by 14% and 22% when compared to DLBS [1] and RALBA [2] respectively. From the above results and discussion, SVR-MTLB technique considerably minimizes the workload with maximum throughput and minimal makespan, migration time as well as response time in the big datacenter.

VI. CONCLUSION

An efficient dynamic load balancing technique called, SVR-MTLB technique is introduced to achieve higher throughput and minimum makespan. The SVR-MTLB technique considers the capabilities of all the virtual machine in the index table and assigns the user requested tasks into the most suitable virtual machine. The dynamic load balancer in SVR-MTLB technique considers a load of all its configured virtual machine in the data center. This SVR-MTLB technique finds the three different stages of the virtual machine namely overloaded, less loaded and balanced load to handle a large number of user-requested tasks using MapReduce function. After that, the Map function finds the least loaded virtual machine in the index list from the

regression analysis for processing the user requested tasks. Then the load balancer sends the less loaded virtual machine ID to the data center controller. The controller migrates the tasks to the less loaded virtual machine resulting in increases the throughput and minimizes the makespan. Finally, the reduce phase combines the results of the virtual machine and provides the final output results. Therefore, the SVR-MTLB technique balances the loads uniformly across all the virtual machines and thus minimizes the response time of each task. Experimental evaluation of SVR-MTLB technique and existing methods are carried out and the results performance analysis proved that the proposed algorithm effectively achieves the better load balancing with maximum throughput and minimum makespan, migration time as well as response time than the state-of-art methods.

REFERENCES

1. Feilong Tang , Laurence T. Yang , Can Tang , Jie Li ; Minyi Guo, "A Dynamical and Load-Balanced Flow Scheduling Approach for Big Data Centers in Clouds", IEEE Transactions on Cloud Computing, Volume 6 , Issue 4 , 2018, Pages 915 – 928
2. Altaf Hussain, Muhammad Aleem, Abid Khan, Muhammad Azhar Iqbal, Muhammad Arshad Islam, "RALBA: a computation-aware load balancing scheduler for cloud computing", Cluster Computing, Springer, Volume 21, Issue 3, 2018, Pages 1667–1680
3. Jie Cui , Qinghe Lu , Hong Zhong , Miaomiao Tian , Lu Liu, "A Load-Balancing Mechanism for Distributed SDN Control Plane Using Response Time", IEEE Transactions on Network and Service Management , Volume 15 , Issue 4 , 2018, Pages 1197 – 1206
4. Jian Liu, Huanqing Dong, Junwei Zhang, Zhenjun Liu, Lu Xu, "HWM: a hybrid workload migration mechanism of metadata server cluster in data center", Frontiers of Computer Science, Springer, Volume 11, Issue 1,2017, Pages 75–87
5. Sweekriti M. Shetty and Sudheer Shetty, "Analysis of load balancing in cloud data centers", Journal of Ambient Intelligence and Humanized Computing, Springer, 2019, Pages 1-9
6. J. Octavio Gutierrez-Garcia and Adrian Ramirez-Nafarrate, "Agent-based load balancing in Cloud data centers", Cluster Computing, Springer, Volume 18, Issue 3, September 2015, Pages 1041-1062
7. Shuo Wang, Jiao Zhang, Tao Huang, Tian Pan, Jiang Liu, Yunjie Liu, "Flow distribution-aware load balancing for the datacenter", Computer Communications, Elsevier, Volume 106, 2017, Pages 136-146
8. Liang Yu ,Tao Jiang , Yulong Zou, "Price-Sensitivity Aware Load Balancing for Geographically Distributed Internet Data Centers in Smart Grid Environment", IEEE Transactions on Cloud Computing, Volume 6 , Issue 4 , 2018, Pages 1125 – 1135
9. Jun Duan and Yuanyuan Yang, "A Load Balancing and Multi-Tenancy Oriented Data Center Virtualization Framework", IEEE Transactions on Parallel and Distributed Systems, Volume 28 , Issue 8 , 2017 , Pages 2131 – 2144
10. Mohit Kumar and S.C.Sharma, "Deadline constrained based dynamic load balancing algorithm with elasticity in cloud environment", Computers & Electrical Engineering, Elsevier, Volume 69, 2018, Pages 395-411
11. Jia Zhao, Kun Yang ,Xiaohui Wei , Yan Ding , Liang Hu , Gaochao Xu, "A Heuristic Clustering-Based Task Deployment Approach for Load Balancing Using Bayes Theorem in Cloud Environment", IEEE Transactions on Parallel and Distributed Systems, Volume 27 , Issue 2 , 2016 , Pages 305 – 316
12. Tianyi Chen , Antonio G. Marques , Georgios B. Giannakis, "DGLB: Distributed Stochastic Geographical Load Balancing over Cloud Networks", IEEE Transactions on Parallel and Distributed Systems ,Volume 28 , Issue 7 , Pages 1866 – 1880
13. Xiao Song, Yaofei Ma, and Da Teng, "A Load Balancing Scheme Using Federate Migration Based on Virtual Machines for Cloud Simulations", Mathematical Problems in Engineering, Hindawi Publishing Corporation, Volume 2015, September 2014, Pages 1-11
14. Jixiang Yang, Ling Ling, and Haibin Liu, "A Hierarchical Load Balancing Strategy Considering Communication Delay Overhead for Large Distributed Computing Systems", Mathematical Problems in Engineering, Hindawi Publishing Corporation, Volume 2016, April 2016, Pages 1-9
15. Hadi Khani, Nasser Yazdani, Siamak Mohammadi, "A self-organized load balancing mechanism for cloud computing", Concurrency and Computation: Practice And Experience, Volume 29, Issue 4, 2017, Pages 1-19
16. Ramona Trestian , Kostas Katrinis, Gabriel-Miro Muntean, "OFLoad: An OpenFlow-Based Dynamic Load Balancing Strategy for Datacenter Networks", IEEE Transactions on Network and Service Management , Volume 14 , Issue 4 , 2017, Pages 792 – 803
17. Keng-Mao Cho, Pang-Wei Tsai, Chun-Wei Tsai, Chu-Sing Yang, "A hybrid meta-heuristic algorithm for VM scheduling with load balancing in cloud computing", Neural Computing and Applications, Springer, Volume 26, Issue 6, August 2015, Pages 1297–1309
18. Kousik Dasguptaa, Brototi Mandalb, Paramartha Duttac, Jyotsna Kumar Mondald, Santanu Dame, "A Genetic Algorithm (GA) based Load Balancing Strategy for Cloud Computing", Procedia Technology, Elsevier, Volume 10, 2013, Pages 340 – 347
19. Jianjiang Li, Yajun Liu, Jian Pan, Peng Zhang, Wei Chen, Lizhe Wang, "Map-Balance-Reduce: An improved parallel programming model for load balancing of MapReduce", Future Generation Computer Systems, Elsevier, 2017, Pages 1-28
20. Hui Zhang , Guofei Jiang , Kenji Yoshihira , Haifeng Chen, "Proactive Workload Management in Hybrid Cloud Computing", IEEE Transactions on Network and Service Management , Volume 11 , Issue 1, 2014, Pages 90 – 100
21. E.Sivaraman, R. Manickachezian, "Unevenness measurement using the support vector machine and dynamic multiservice load balancing with modified genetic algorithm in cloud-based multimedia system", International Journal of Computer Aided Engineering and Technology, Volume 10, Issue 6, 2018, Pages 732-747
22. E.Sivaraman, R. Manickachezian, "Efficient multimedia content storage and allocation in multidimensional cloud computing resources", International Journal of Intelligent Systems Technologies and Applications,, Volume 18 , Issue 1-2, 2019, Pages 20 – 33
23. C.R.Durga Devi and R.Manicka Chezian, "Resource optimized ensemble gradient boosting classifier for traffic aware big data analytics", Journal of Advanced Research in Dynamical & Control Systems , Volume 10 , Issue 3, 2018, Pages 839-852
24. C.R.Durga Devi and R.Manicka Chezian, "Multivariate Logistic Regression based Gradient Descent Firefly Optimized Round Robin Scheduling with Big Data", Journal of Advanced Research in Dynamical & Control Systems , Volume 11 , Issue 1, 2019, Pages 179-192

AUTHORS PROFILE

CR.Durga devi received her Bsc.Computer Technology from Coimbatore Institute of Technology, Coimbatore, India. She had her Master of Computer Applications from Bharathiar University, Coimbatore, India. she holds Mphil in Computer Science from Bharathiar University, Coimbatore, India. She has 12 years of experience in teaching. She is presently working as an Assistant Professor in NGM College (autonomous), under Bharathiar University, Coimbatore, India since 2008. She has published 15 papers in International/National Journal and Conferences Her research interest includes Data Mining,Big Data Analytics. Now she is pursuing her ph.d Computer Science in Dr.Mahalingam center for research and Development at NGM College, Pollachi. Her research focuses on Data Mining, Cloud Computing

Dr.R.Manickachezian received his M.Sc., Degree in Applied Science from P.S.G College of Technology, Coimbatore, India in 1987. He completed his M.S. Degree in Software Systems from Birla Institute of Technology and Science, Pilani, Rajasthan, India and Ph.d degree in Computer Science from school of Computer Science and Engineering, Bharathiar University, Coimbatore, India. He served as a faculty of maths and Computer Applications at P.S.G College Of Technology, Coimbatore from 1987 to 1989.

presently, he has been working as an Associate Professor of Computer Science in NGM college (autonomous), pollachi under Bharathiar University, Coimbatore, India since 1989. He has published 150 papers in International/National Journal and Conferences. He is a recipient of many awards like best Computer Science Faculty of the year 2015, Best Research Supervisor award, Life Time Achievement award, Desha Mithra Award and best paper award. His research focuses on Network Databases, Data Mining, Distributed Computing, Data Compression, Mobile Computing, Real Time Systems and Bio-Informatics