Cluster-based Task Scheduling Using K-Means Clustering for Load Balancing in Cloud Datacenters

Geetha Muthusamy, Suganthe Ravi Chandran

Department of Computer Science and Engineering, Kongu Engineering College, India
geetha@kongu.ac.in, suganthe_rc@kongu.ac.in

Abstract

Load balancing is a process of distributing incoming tasks to available resources in cloud datacenters, where a resource exists in terms of a virtual machine (VM). Proper load balancing results in minimizing the computation time and improving the resource utilization rate. Various scheduling algorithms are applied to achieve load balancing in cloud datacenters. Due to the heterogeneous nature of resources in the cloud, greedy approaches are used to schedule the tasks to the VMs. This paper suggests a cluster-based task scheduling framework (CBTS) using K-Means clustering by considering task length and VM capacity. Here, the tasks are clustered based on their length, and the VMs are grouped based on their processing capacity. After clustering, the individual task in each cluster is scheduled to appropriate VM in the VM groups.The proposed system performs dynamic load balancing with an aim in minimizing the makespan and execution time. The experimental results reveal that the proposed method produces better results in terms of execution time, makespan, and deviation in workload than the conventional Min-Min algorithm and the recently developed heuristic algorithms such as Online Potential Finish Time (OPFT), Dynamic Cloud Task Scheduling (DCTS), and Grouped Task Scheduling (GTS).

Keywords: Cloud computing, Scheduling, Load balancing, Clustering, Virtualization

1 Introduction

Cloud computing systems belong to a new class of distributed system that provides computation, storage, and networking capabilities as a service to the users through the Internet. These services are offered on-demand and on a pay-as-you-go model in which the users are charged based on their usage of the resources. Large clouds, predominant today, often have functions distributed over multiple locations from central servers. Cloud computing relies on sharing of resources to achieve coherence and economies of scale. Cloud datacenters provide storage, memory, processing, and bandwidth [2-3, 8], to process the requests or the tasks that are submitted by the users. It also enables convenient, on-demand network access to a shared pool of configurable computing resources through a promising technology called virtualization. Virtualization is the backbone of cloud computing. It is a prominent technology that enables cloud computing by providing computational units in terms of virtual resources called virtual machines (VM) [8]. In the cloud computing environment, resources can be scaled up by adding VM instances or scaled down by removing the VM instances. The independent tasks submitted by the clients are executed by these VMs in the cloud computing environment where VMs run in parallel [2, 12-13]. Thus there is a need for scheduling these tasks in a way that leads to effective resource utilization.

Scheduling [7, 12-19] is a process of allocating resources to the tasks that are submitted by the cloud clients at a specific time. The main objective of scheduling is to minimize the makespan and response time, maximize resource utilization, and to have a balanced load on all the machines. A good scheduling algorithm yields good system performance. Cloud datacenters [2, 8, 23] consist of numerous heterogeneous resources. The cost of executing a task and completion time of a task in the cloud depends on the nature of the resource to which it has been assigned. Hence, scheduling [4-6,18] the tasks to heterogeneous resources in the cloud environment is said to be a challenging issue and it is also known to be an NP-complete problem. Some scheduling algorithms like First Come First Serve (FCFS), Shortest Job First (SJF), Round Robin (RR), Min-Min, Max-Min algorithms exists, but these are not considered as a much better solution to the scheduling problems in cloud computing. The research community has contributed to providing different solutions to the scheduling problem. Some people tried with heuristic methods and some with optimization techniques. Still there is a need for a better scheduling approach to solving the scheduling problem and improve the performance of datacenters by improving the performance metrics. Hence, to provide a solution to the scheduling problem, cluster-based task scheduling (CBTS) framework using the K-

*Corresponding Author: Geetha Muthusamy; E-mail: geetha@kongu.ac.in
DOI: 10.3966/160792642021012201012

Means clustering is proposed which clusters the tasks based on the task length. Tasks in each cluster are scheduled to suitable VMs according to their processing capacity.

The rest of the report is organized as follows: In Chapter 2, related work is reviewed. Chapter 3 depicts the proposed scheduling framework and describes the mathematical model and the implementation details of the proposed system. Experimental evaluation is illustrated in Chapter 4. Finally, in Chapter 5, the conclusion and future work are discussed.

2 Literature Review

Various heuristic-based and optimization-based scheduling algorithms have been proposed in the literature to address the scheduling problem. Task grouping and VM grouping based scheduling algorithms have been designed by Ali et al. [1], Alworafi et al. [20], Marphatia [21], Pawar and Wagh [22], Thomas et al. [23], and Zhang and Zhou [24]. Each of these algorithms differs in the way the tasks are grouped. The scheduling algorithm in [1] divides the tasks in to five different categories inorder to reduce execution time and average latency of tasks and also to balance the workload on the machines. Alworafi et al. [20] grouped the tasks based on the user-assigned priority and the VMs based on their processing speed. The authors did not address the issue related to the tasks with same priority. Also task grouping may not be fair as it considers the priority assigned by the user. Instead of user-assigned priority, the tasks may be ranked based on their length and bandwidth requirements. An optimized version of First Come First Serve (FCFS) algorithm in [21] classifies the tasks based on their requirement: either deadline or execution cost. Tasks in each class are prioritized and then scheduled to the VMs following different scheduling approaches for each class of tasks. If any task belonged to the deadline class, then it is scheduled to a VM which provides less turnaround time. For a task which belongs to the cost class, it is scheduled to a VM which offers minimum execution cost. Thus, tasks in the two classes are scheduled in a parallel manner thereby reducing the task completion time. A scheduling algorithm which makes use of SLA based resource provisioning and online adaptive scheduling strategies has been developed in [22]. This algorithm assigns priority to the tasks based on their earliest start time and latest start time and the schedules them to the VMs in a preemptable manner. Here, the VMs are grouped based on the type of resource they provide. None of the above algorithms considers the task length to classify the tasks. As the smaller tasks with high priority may cause the longer tasks to wait for a long time, task completion time gets increased and hence the makespan. In [23], the tasks are grouped based on their length and user-assigned priority. Initially, task length difference has been calculated by finding the difference between the average length and the actual length. Based on the difference and user priority, the algorithm assigns credits to the tasks. The dynamic cloud task scheduling (DCTS) algorithm proposed in [24] classifies the tasks and creates the necessary VMs in advance, based on the scheduling history. By this way, it reduces the time to create the VMs and hence reduces the completion time of the tasks. This algorithm makes use of the Bayes classifier to classify the tasks.

List-based scheduling algorithms namely performance-effective task scheduling (PETS) and predict earliest finish time (PEFT) have been devised by Thambidurai and Ilavarasan [25], and Arabnejad and Barbosa [26] respectively to schedule workflow tasks in heterogeneous environments. These algorithms differ in the way the ranks are computed and assigned to the tasks. PETS made use of three metrics namely Average Computation Cost (ACC), Data Transfer Cost (DTC) and the Rank of Predecessor Task (RPT) to compute the rank, whereas PEFT employed optimistic cost table (OCT). Both methods produced better results than heterogeneous earliest finish time (HEFT) algorithm. QoS-Aware Scheduler named Paragon has been proposed by Delimitrou and Kozyrakis [27]. Based on the learning from the previously loaded tasks, Paragon makes use of collaborative filtering technique to classify the new incoming tasks. It also identifies the tasks which interfere with the incoming tasks and the level of tolerance with the interference. A greedy algorithm is used to assign the classified tasks to the server which has the best configuration and with a workload which has high tolerance with the incoming task. Thus this kind of schedule would improve resource utilization which is evident from the experimental analysis. Post-processing scheduling algorithm [11] developed by Tae-Young Choe generated shortest schedules by allocating parent tasks to different processors. This algorithm tries to reduce the number of processors required to execute the tasks. The tasks are represented as directed acyclic graphs (DAG) which satisfies Darbha's Condition. The algorithm also adapted a maximum matching algorithm to find the maximum edge cover in the graph to reduce the number of processors.

Dubey et al. [28] developed a management system comprising of the ideal distribution algorithm (IDA) and enhanced IDA (EIDA) to service multiple organizations in the community cloud. IDA has been developed to reduce execution cost and makespan in processing the workflow applications submitted by the organizations. This has been done using a suitable VM allocation policy in IDA. The deadline and cost of executing the applications have been posed as the constraints during VM allocation. Apart from minimizing the makespan and execution cost, the authors also tried to balance the workload on the VMs

using. In EIDA, the children of smaller tasks are executed before the parent task so that the communication cost and execution delay will be reduced. Those child tasks are allocated to the VMs which can complete them before their deadline. The result analysis shows that IDA and EDA perform better than HEFT, Min-Min, and Max-Min algorithms. Jiayin Li et al. [29] suggested an optimized resource allocation mechanism that incorporates two dynamic scheduling algorithms namely dynamic cloud list scheduling (DCLS) and dynamic cloud min-min scheduling (DCMMS). It works well when there is a severe conflict among the tasks for acquiring the resources. Scheduling is done in a preemptive manner and the information about the resources is updated dynamically. The tasks with high priority are scheduled using advanced resource allocation scheme while the other tasks are scheduled by following the best-effort approach.

In [30], the authors proposed a comparison matrix technique to perform a pair-wise comparison of tasks during scheduling for selecting the most important task. The selected task will be the first to be scheduled to speed up the task completion time. The authors also used the analytic hierarchy process (AHP) to determine the weight of tasks to assign the tasks to suitable resources. Etminani et al. [31] developed a selective algorithm that uses the Max-Min and Min-Min algorithms. It determines to select one of these two algorithms, depending on the standard deviation of the expected completion times of the tasks on each of the resources. A comparison of this algorithm with the FCFS algorithm shows that this algorithm is more efficient in minimizing the makespan. QoS Guided Min-Min Heuristic proposed by He et al. [32] is a new scheduling mechanism that is based on the traditional Min-Min algorithm. The QoS considered here is the bandwidth requirement of the incoming requests or the tasks submitted by the users. The algorithm first computes the completion time of all the tasks on all the hosts and then schedules those tasks whose QoS request is high. This algorithm works only when the bandwidth requirement of the tasks varies. El-Kenawy et al. [14] proposed an improved version of the Max-Min algorithm named Extended Max-Min Scheduling using Petrinet for load balancing. It works on the expected execution time rather than the completion time as a selection basis. They used Petrinets which are well suited for modeling the concurrent behavior of distributed systems. The result shows that this algorithm achieves schedules with a lower makespan than the original Max-min.

Nasr et al. [33-36] developed four different scheduling frameworks called ant colony optimization simulated annealing (ACOSA), water pressure change optimization (WPCO), online potential finish time (OPFT), and highest priority first execute (HPFE) as solutions to the scheduling problem. ACOSA and WPCO are optimization-based frameworks while HPFE and OPFT are heuristic-based frameworks. ACOSA is a hybrid approach that improves the quality of the solution and reduces the time complexity of the algorithm by combining the best features of ACO and SA. During each iteration, new solutions are obtained using ACO and the solutions are updated using SA. ASOCA uses SA to reduce the number of iterations thereby reducing the time complexity of the algorithm. In WPCO, due to the change in water pressure, both the volume and density get changed. Being inspired by this behavior of water, the authors framed the scheduling algorithm to distribute the tasks to suitable VMs. This reduces the makespan and also balances the workload on the VMs. HPFE is yet another solution to perform load balancing and also to minimize makespan. The framework consists of two scheduling algorithms namely over-scheduler and fewer-scheduler to utilize the VMs efficiently. Over-scheduler is used when the number of VMs is less than or equal to the number of submitted tasks. The scheduling framework makes use of fewer-scheduler when the number of free VMs is more than the number of tasks. The extra VMs are used to execute the next job in the arrival queue so that the execution time can be reduced. Experimental analysis proves that both ACOSA and HPFE outperform the existing Min-Min, HEFT, Minimum Completion Time, ACO, and SA algorithms. Finally, OPFT is another heuristic that schedules the independent tasks in such a way that enhances the QoS metrics such as schedule length, execution cost, response time, balance degree, and resource utilization.

Although many contributions have been made by the research community towards providing a solution for the scheduling problem in cloud computing, still there is a need for new solutions to reduce the makespan further and also to perform load balancing. Hence, a new scheduling technique namely cluster-based task scheduling (CBTS) is proposed to minimize the makespan, execution time, and deviation in workload among the VMs. Major contributions of the proposed system are:

1. Cluster the tasks based on their task length using the K-Means clustering algorithm
2. Group the VMs based on their capacity and schedule the task clusters to appropriate VM groups
3. Verify the efficiency of the algorithm by testing it in terms of the QoS metrics such as makespan, execution time, and deviation in load.

3 System Implementation

3.1 Scheduling Framework

Figure 1 depicts the CBTS framework which comprises of the following components:

- Cloud client

- Task manager
- Load balancer
- Resource manager at two levels: Global and Local
- Datacenter with multiple servers

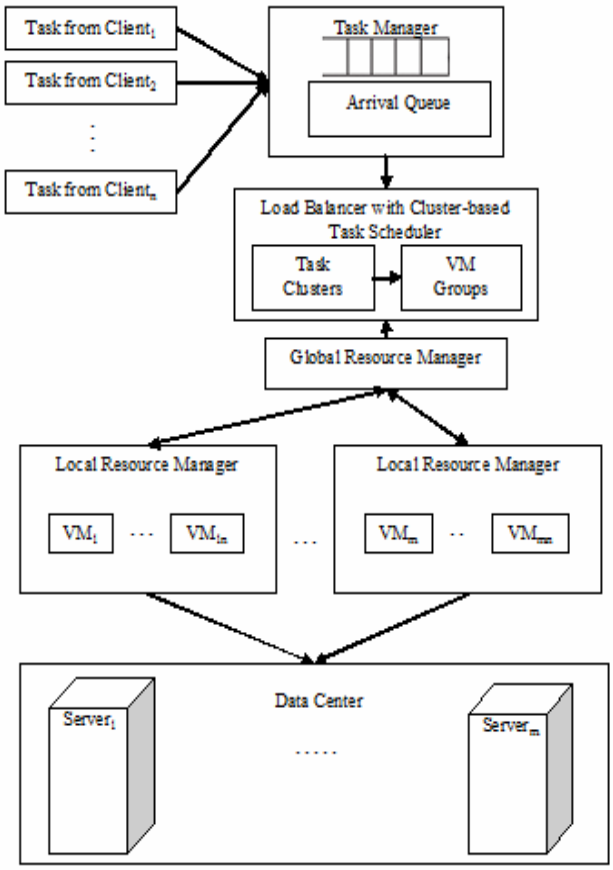

Figure 1. Proposed Scheduling Framework

When a client submits tasks to the cloud provider, the tasks are received by the task manager and kept in the task queue. A data center consists of multiple physical servers that provide multiple VMs as the resources to compute and complete those tasks. Each server has a local resource manager to maintain information regarding the VMs such as the number of active VMs, and their processing, storage, and bandwidth capacity. There is also a global resource manager which receives information from the local managers and maintains a database about the VMs that run in all the physical servers. The load balancer collects information about the tasks and VMs from the task manager and the global resource manager respectively. After collecting the information, it passes the same to the scheduler which generates task clusters and VM groups. The scheduler then allocates tasks in the task clusters to VMs in the VM groups.

3.2 Mathematical Model

A mathematical model for the research problem is proposed with the following assumptions:

- There are n independent heterogeneous tasks to be scheduled to and executed in m heterogeneous machines.
- Each task is of length L in terms of number of instructions (Million Instructions - MI)
- Each machine has a processing capacity in terms of million instructions per second (MIPS)

The scheduling problem is formulated such that n tasks need to be scheduled to m machines in such a way that workload on the machines should be in a balanced state or all the machines should have equal workload according to their capacity. Also,the total completion time of the tasks should be reduced. Hence, the objective of this scheduling problem is to distribute the tasks to the VMs in such a way that there will be a minimum deviation in the workload on all VMs. The objective function is thus formulated as given in Equation (1)

$$F(x) = \min(\sigma) \tag{1}$$

where σ is the standard deviation in workload. Equations (2-5) are devised to calculate σ.

$$C = \sum_{j=1}^{m} C_j \quad \forall j \in 1, 2, 3, ..., m \ VMs \tag{2}$$

$$L_j = \sum_{i=1}^{n} TL_i \quad \forall i \in 1, 2, 3, ..., n \ \ tasks \tag{3}$$

where TL_i is the length of i^{th} task that is assigned to j^{th} VM

$$\sigma = \sqrt{\frac{1}{N}\sum_{j=1}^{n}\left(L_j - \bar{L}\right)} \tag{4}$$

$$PT_{ij} = \frac{TL_i}{C_j} \tag{5}$$

$$CT_{ij} = PT_{ij} + WT_{ij} \tag{6}$$

The notations used in the above-mentioned equations are as follows:

- C_j denotes the CPU capacity of j^{th} VM in terms of MIPS. Since the tasks are compute-intensive, CPU resource is given more importance than the other resources.
- TL_i represents the length of i^{th} task
- PT_{ij} is the time taken to execute task i by VM j.
- CT_{ij} is the time taken by VM j to complete task i
- WT_{ij} is the time task i has been waiting to get the response from the CPU of VM j.

3.3 Task Scheduling using K-Means Clustering

Data clustering [38-41] is an important and widely used technique in data analysis and data mining. The prime objective of clustering is to split the elements in a dataset into subsets where elements of the same group are more similar to each other than the elements

from different groups. Various clustering techniques are available in data mining, among which K-Means clustering is a simple and efficient approach. Compared to the other data clustering algorithms, K-Means clustering works efficiently in handling datasets with a single attribute. It is also fast and easy to implement. Hence, the proposed system makes use of K-Means to form task clusters to reduce makespan and perform a fair distribution of workload among the VMs.

The dataset considered here is the meta-task set which consists of a batch of tasks that are to be scheduled to the resources. Tasks are clustered based on the distance between the task lengths. K in K-Means refers to the number of clusters, which is to be defined beforehand. The algorithm starts by randomly choosing a centroid value for each cluster. As the tasks are to be clustered in the proposed system, task lengths are the data points. Tasks are clustered and scheduled to suitable VMs based on the processing capacity of VMs. This leads to a balanced workload on all the VMs. Algorithm 1 summarizes the procedure to form task clusters using K-Means clustering algorithm. Algorithm 2 presents the steps involved in the proposed CBTS scheduling.

Algorithm 1. Task Clustering using K-Means Clustering Algorithm

Input: Meta-task set with task length
Output: Task clusters TG
Step 1: Select k random instances as the centroids
Step 2: Until clustering converges or other stopping criteria:
Step 3: For each instance Tl:
Step 4: Find the Euclidean distance d between each Ti and centroids Sj of all the task clusters $d\left(T_1, S_j\right) = \sqrt{\left(T_1 - S_j\right)^2}$
Step 5: Assign T_l to the task cluster TC_j such that $d\left(T_1, S_j\right)$ is minimal
Step 6: For each task cluster TC_j, find the mean as $S_j = \mu\left(TC_j\right)$ where μ is the mean of all task lengths in cluster j

Algorithm 2. CBTS Algorithm

Input: task set tSet and VM list vList
Output: Schedule with tasks to VMs
Step 1: With tSet, form task clusters using K-Means clustering and store each cluster in tgList
Step 2: Calculate the average task length of each task group
Step 3: Sort the task clusters in descending order based on their average task length
Step 4: Divide the VMs in vList into k groups and store each group in vgList
Step 5: Compute the average processing speed of the VMs in each VM group
Step 6: Sort the VM groups in descending order based on their average processing speed
Step 7: Count the number of VMs in each VM group and store the result in the array nVMs
Step 8: for each task cluster in tgList
Step 9: Initialize i and j to 0
Step 10: for each task T_i in TC_k
Step 11: Assign $T_j$to $VM_j$of VG_k
Step 12: if (j > = nVMs[k])
Step 13: j=0
Step 14: end if
Step 15: Increment i and j by 1
Step 16: end for
Step 17: end for

Figure 2 shows the task clusters along with the task length, task id, and average length of each cluster. Subsequent to task clustering, VMs in the datacenter are divided into k groups. Here, VMs are grouped instead of forming VM clusters. The reason is that the distance between the VMs will be large so that the cluster formation will not be efficient. Inefficient clusters will increase the execution time and makespan. Hence, VMs are just divided into k groups, where k represents the number of task clusters. For example, if there are 15 VMs, each group will have 5 VMs. If there are 17 VMs, two groups will have 6 VMs each and the other one will have the remaining 5 VMs. Figure 3 shows the VM groups and the VMs in each group along with their processing capacity and average processing capacity of each VM group.

Cluster 1 (TC_1)		Cluster 2 (TC_2)	Cluster 3 (TC_3)	
T7-91000	T54-125000	T12-712500	T1-15000	T50-59000
T9-101000	T55-105000	T22-525000	T2-83000	T52-59000
T14-113000	T58-109000	T46-900000	T3-63000	T53-47000
T18-127000	T62-127000	T47-712500	T4-63000	T57-45000
T21-115000	T64-107000	T56-712500	T5-65000	T59-67000
T23-117000	T65-135000	T69-525000	T6-15000	T60-61000
T24-105000	T66-101000	T84-712500	T8-85000	T61-67000
T29-91000	T67-95000	T97-525000	T10-40000	T63-87000
T30-105000	T68-127000		T11-40000	T70-51000
T31-117000	T71-91000		T13-59000	T74-63000
T34-337500	T72-91000		T15-87000	T75-51000
T35-129000	T73-93000		T16-53000	T77-79000
T37-109000	T76-91000		T17-47000	T78-77000
T38-109000	T79-91000		T19-45000	T81-85000
T39-135000	T80-111000		T20-27500	T82-71000
T41-101000	T83-125000		T25-65000	T85-49000
T42-95000	T89-125000		T26-77000	T86-47000
T43-125000	T91-337500		T27-47000	T87-40000
T44-127000	T93-93000		T28-89000	T88-87000
T49-101000	T95-113000		T32-51000	T90-75000
T51-105000	T96-115000		T33-15000	T92-61000
			T36-47000	T94-49000
			T40-89000	T98-55000
			T45-83000	T99-65000
			T48-83000	T100-79000
Average Length				
120548		665625	60190	

Figure 2. Task Clusters (TC)

VG_1	VG_2	VG_3
VM1-3000	VM7-1000	VM13-2000
VM2-2500	VM8-2500	VM14-500
VM3-1000	VM9-1000	VM15-2500
VM4-4000	VM10-3000	VM16-1000
VM5-3500	VM11-1500	VM17-500
VM6-2500	VM12-2000	VM18-1500
		VM19-4000
		VM20-2000
Average Capacity		
2750	**1833**	**1750**

Figure 3. VM Groups (VG)

After clustering the tasks and grouping the VMs, the task cluster with maximum average task length is assigned to the VM group with maximum average capacity. Here, TC2 has the highest average length and VG1 has the highest average processing capacity. Hence, TC2 is assigned to VG1, TC1 is assigned to VG2, and TC3 is assigned to VG3. Consider TC2 and VG1 where the individual task needs to be assigned to a suitable VM in VG1. Figure 4 describes the scheduling of each task in TC2 to a suitable VM in VG1.

$\{T_{12}, T_{84}\} \rightarrow VM_1$
$\{T_{22}, T_{97}\} \rightarrow VM_2$
$T_{46} \rightarrow VM_3$
$T_{47} \rightarrow VM_4$
$T_{56} \rightarrow VM_5$
$T_{69} \rightarrow VM_6$

Figure 4. Schedule of Tasks in TC2 to VMs in VG1

4 Experimental Evaluation

4.1 Performance Metrics

The performance metric is a standard definition of a measurable quantity that indicates some aspect of performance. It should be measurable and consistent with the performance goals of the scheduling problem. Any solution to the scheduling problem should try to minimize the makespan and execution time of the tasks. Hence, the proposed method was evaluated using these metrics.

Makespan: Makespan is defined as the maximum time taken by a VM to complete the tasks in the task queue. It is denoted as the maximum of the completion time of all the tasks which is given by equation (7).

$$\text{MCT} = \max\left\{CTij \middle| i \in 1, 2, 3, .., m,\ j \in 1, 2, ..., n\right\} \quad \textbf{(7)}$$

Execution time: It is the amount of time taken by a VM to run or execute a task. Here, average execution time defined in equations (8) and (9) is used to measure the performance of the datacenter.

$$ET_{ij} = \frac{T_i}{L_j} \quad \textbf{(8)}$$

$$\overline{ET} = \frac{\sum_{j=1}^{m}\sum_{i=1}^{n} ET_{ij}}{n} \quad \textbf{(9)}$$

Standard deviation: Standard deviation (σ) presented in equation (4) is a metric that is used to find the deviation of workload among the VMs. L_j in equation (6) indicates the load on VM_j and $\overline{L}$ represents the average load on the VMs.

4.2 Performance Evaluation

The proposed CBTS method is demonstrated using CloudSim 3.0 [9], a cloud simulation toolkit that supports both the system and behavioral model of cloud system components such as data centers, VMs, and resource provisioning policies and also implements generic application provisioning techniques that can easily be extended. Table 1 presents the experimental setup for the demonstration. The simulation environment is a 32-bit Windows 7 operating system with core i5 and 8 GB RAM. Google cluster workload is used in conducting the experiments. Analyzing the Google cluster workload traces, a realistic Google-like workload [37] is generated using Monte-Carlo simulation [10]. The analysis reveals that smaller tasks were in the majority and there were few big tasks. The task size ranges from 15,000MI to 900,000MI and the same has been classified as given in Table 2. In CloudSim, submitCloudlets() in a member function of DataCenter class. This method has the code to assign the tasks to VMs and execute the tasks in the VM to which they have been assigned. Hence, to implement the proposed scheduling model, submitCloudlets() method is modified. In addition to that, the Cloudlet class is extended to include the cluster information of the tasks. The results obtained through the experiments are compared with the recently developed task grouping-based GTS algorithm, dynamic cloud task scheduling (DCTS) algorithm, and Online Potential Finish Time (OPFT) algorithm and also with the conventional Min-Min algorithm.

The results are analyzed based on the performance metrics which have been discussed in the previous section. Eight different experiments are carried out by varying the number of VMs and tasks. The experiments are conducted by keeping the VM count as 30 and 50 and the task count as 500, 1000, 1500, and 2000. Here, both the tasks and the resources exhibit heterogeneous characteristics.

Table 1. Experimental Setup

Entity	Quantity
Datacenter	1
Physical machines (Hosts)	4
Processing elements (PE) in Host	4-10
Processing capacity of each PE in the hosts	20000-35000 MIPS
Memory (RAM) capacity of Hosts	8/16/32 GB
VMs in Datacenter	30 - 50
PEs in each VM	1
Processing capacity of each PE in the VMs	500-4000 MIPS
Memory capacity of VMs	512-4196 MB
Task length	15000-900000 MI
Task size	60-3000 KB
Number of tasks	100-1000
Number of VMs	30, 50
Number of task clusters and VM groups (k)	3-5

Table 2. Task Classification from the Generated Workload

Task Size	Task Type
15,000-55,000 MI	Small
59,000-99,000 MI	Medium
101,000-135,000 MI	Large
150,000-337,500 MI	Extra-Large
525,000-900,000 MI	Huge

At first, average execution time is measured and the resultant values are given in Table 3 and Table 4 for 30 VMs and 50 VMs, respectively. CBTS reduces the average execution time by 22%, 24.7%, 29.2%, and 33.1 when compared to OPFT, DCTS, GTS, and Min-Min algorithms, respectively. In CBTS, after task clustering, tasks with huge length are scheduled to the VMs which are of high capacity and those with a smaller length to the VMs of less capacity. This reduces the average execution time which in turn reduces the makespan. Hence, using the proposed method makespan is reduced by 18.6%, 22.8, 42.1%, and 46.8% than OPFT, DCTS, GTS, and Min-Minscheduling, respectively.

Table 3. Average Execution Time (sec) for 30 VMs

No. of Tasks	CBTS	OPFT	DCTS	GTS	Min- Min
500	599	622	635	736	941
1000	1058	1335	1382	1472	1587
1500	1549	2107	2159	2301	2450
2000	2453	3319	3367	3413	3546

Table 4. Average Execution Time (sec) for 50 VMs

No. of Tasks	CBTS	OPFT	DCTS	GTS	Min- Min
500	370	415	453	498	523
1000	789	893	936	1153	1487
1500	838	1438	1459	1359	1745
2000	1625	2255	2304	2486	2713

Figure 5 and Figure 6 depict the improvement obtained for makespan. The last and important metric is the deviation in workload that exists among the VMs. Any scheduling mechanism should try to distribute the workload among the VMs based on their capacity. The proposed method has also tried to balance the workload by clustering the tasks and scheduling those tasks to appropriate VMs. Hence, it shows an improvement of about 10.3%, 13%, 54.8%, and 62.3% in balancing the load on the VMs when compared to OPFT, DCTS, GTS, and Min-Min algorithms, respectively. Figure 7 and Figure 8 represent the deviation in load for the existing and the proposed methods. The results show that when the number of tasks increases there is a huge deviation in the workload in Min-Min and CBTS. But, CBTS, OPFT, and DCTS maintains a balanced workload on the VMs even the number of task increases. It is also found that this increase also increases the makespan due to the limited capacity of the VMs.

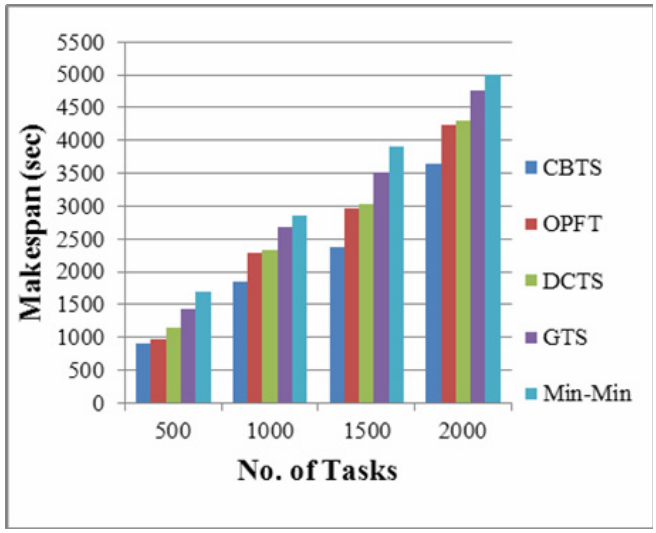

Figure 5. No. of Tasks vs Makespan (30VMs)

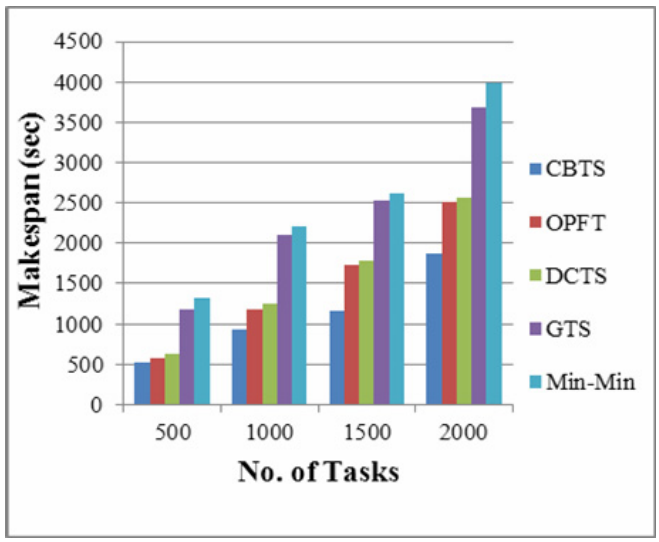

Figure 6. No. of Tasks vs Makespan (50 VMs)

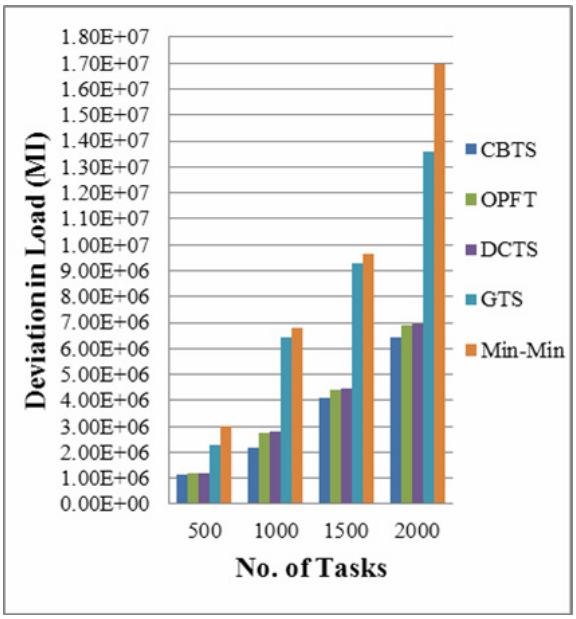

Figure 7. No. of Tasks vs Deviation in Workload (30 VMs)

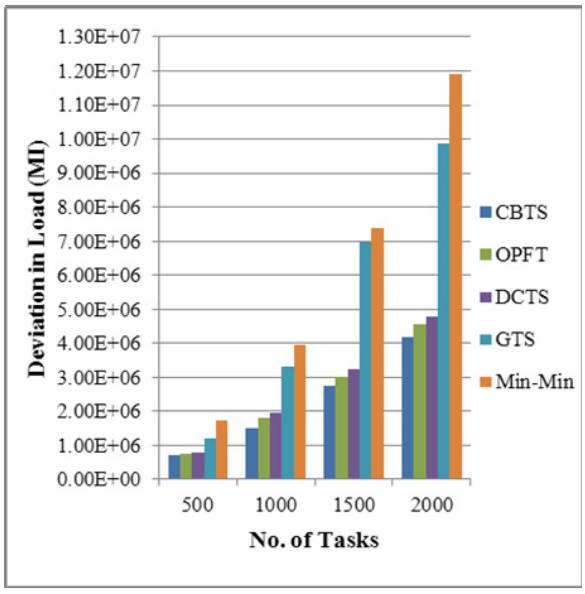

Figure 8. No. of Tasks vs Deviation in Workload (50VMs)

Like CBTS, GTS and DCTS are also grouping based scheduling algorithms. GTS groups the tasks alone into five categories based on the type of user and the task priority that is assigned by the user. Due to user-assigned priorities, some of the shorter tasks need to wait for a long time. This is because the user may assign high priority to longer tasks. This results in scheduling the tasks to inappropriate VMs. GTS executes the tasks in the high priority category first and then moves to the tasks in next category. This increases the waiting time of the tasks which in turn increases the execution time. As GTS makes use of Min-Min algorithm to schedule the tasks in every category, it increases the makespan and the deviation in workload when compared to the proposed CBTS algorithm. DCTS classifies the tasks in to 'n' different types based on the scheduling history of the previously submitted tasks. It also creates the VMs well in advance to reduce the completion time of the tasks. But this leads to few VMs to be not utilized when there are less number of tasks. Also, more number of task types increases the time taken to assign the tasks to suitable VMs. But, CBTS effectively utilizes the available VMs by considering their processing capacity. It also assigns the tasks to best suitable VMs by considering both the task length and VM capacity. Therefore, it is evident from the analysis that CBTS outperforms the existing methods in terms of all the QoS metrics.

4.3 Time Complexity Analysis

This section analyses the running time of the proposed CBTS algorithm with the existing OPFT, DCTS, GTS, and Min-Min algorithms. The CBTS algorithm consists of totally 17 steps to perform the scheduling. Step 1 makes use of the K-Means clustering algorithm to generate the task clusters. The inputs to this step are the number of tasks (n), number of attributes (d) of a task, and number of clusters (k). Hence, the time complexity is O(k*n*d) [19]. Here, k is varied from 3 to 5, d is one, but the size of n is bigger. Therefore, the time complexity is simplified to O(n). The running time of each step in the CBTS algorithm is given in Table 5, where the size of 'n' is bigger than 'm' and 'k'. Therefore the time complexity of CBTS is reduced to O(n). The time complexity of OPFT and DCTS is O(n) whereas it is O(m*n) for GTS and Min-Min algorithms. In comparison with the existing algorithms, CBTS, OPFT, and DCTS result in less time complexity than the other two algorithms. But, from the experimental evaluation it is proved that CBTS enhances the QoS metrics than existing agorithms. Therefore, it is clear that CBTS outperforms the existing methods in terms of the QoS metrics.

Table 5. Time Complexity of CBTS

Step (s)	Input(s)	Input Size	Running Time
Step 1, Step 2, and Steps 8-17	Tasks	n (maximum 2000)	O(n)
Step 3 and 6	Number of task clusters or number of VM groups	k (maximum 5)	O(k)
Steps 4, 5, and 7	Number of	m (maximum 50)	O(m)

5 Conclusion and Future Directions

Scheduling in a distributed computing environment such as cloud computing remains to be a challenging issue, as cloud datacenters comprise of heterogeneous resources. The number of users and their requests use to vary dynamically in the cloud. Hence, there is a need for a scheduling strategy that considers this

heterogeneity while assigning the requests to the suitable resources so that the completion time of the requests is minimized. The proposed scheduling mechanism generates task clusters based on the similarity in the task lengths by making use of K-Means clustering which is a simple and efficient clustering algorithm. Tasks in the clusters are scheduled to suitable VMs based on the VM capacity. From the experimental analysis, it is evident that the proposed method improves the performance of the datacenters by minimizing the makespan, execution time, and deviation in the workload. At present, task length alone is considered to form the clusters. A task may be bound to its deadline and the delay in completion. Therefore, the deadline may also be included as an additional attribute of the tasks while forming the task clusters. Nowadays, the research community has turned their attention towards applying nature-inspired algorithms to their research problems. Hence, the scheduling problem in cloud datacenters can also be addressed with appropriate nature-inspired algorithms to obtain an optimal schedule.

References

[1] H. G. E. D. H. Ali, I. A. Saroit, and A. M. Kotb, Grouped Tasks Scheduling Algorithm Based on QoS in Cloud Computing Network,*Egyptian Informatics Journal*, Vol. 18, No. 1, pp. 11-19, March, 2017.

[2] S. Anousha, M. Ahmadi, An improved min-min task scheduling algorithm in grid computing,in: J. J. H. Park, H. R. Arabnia, C. Kim, W. Shi, J. M. Gil (Eds.), *International Conference on Grid and Pervasive Computing*, Springer, Berlin, Heidelberg, 2013, pp. 103-113.

[3] G. B. H. Bindu, K. Ramani, C. S. Bindu, Energy aware multi objective genetic algorithm for task scheduling in cloud computing, *International Journal of Internet Protocol Technology*, Vol. 11, No. 4, pp. 242-249, October, 2018.

[4] H. Cui, Y. Li, X. Liu, A. Ansari, Y. Liu, Cloud service reliability modelling and optimal task scheduling, *IET Communications*, Vol. 11, No. 2, pp. 161-167, January, 2017.

[5] M. A. Alworafi, A. Dhari, A. A. Al-Hashmi, Suresha, A. B. Darem, Cost-Aware Task Scheduling in Cloud Computing Environment, *International Journal of Computer Network and Information Security*, Vol. 9, No. 5, pp. 52-59, May, 2017.

[6] J. J. Wu, H. J. Chang, Y. F. Ho, P. Liu, Scheduling of variable-time jobs for distributed systems with heterogeneous processor cardinality, *International Journal of Ad Hoc and Ubiquitous Computing*, Vol. 10, No. 2, pp. 112-121, July, 2012.

[7] U. Rugwiro, C. Gu, W. Ding, Task scheduling and resource allocation based on ant-colony optimization and deep reinforcement learning, *Journal of Internet Technology*, Vol. 20, No. 5, pp. 1463-1475, September, 2019.

[8] A. Bala, I. Chana, Multilevel priority-based task scheduling algorithm for workflows in cloud computing environment, in: S. Satapathy, A. Joshi, N. Modi, N. Pathak (Eds.), *International Conference on ICT for Sustainable Development, Advances in Intelligent Systems and Computing*, Springer, Singapore, 2016, pp. 685-693.

[9] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. D. Rose, R. Buyya, CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, *Software- Practice & Experience*, Vol. 41, No.1, pp. 23-50, January, 2011.

[10] Y. Chen, A. S. Ganapathi, R. Griffith, R. H. Katz, *Analysis and Lessons from a Publicly Available Google Cluster Trace*, Technical Report No. UCB/EECS-2010-95, June, 2010.

[11] T. Y. Choe, Task scheduling algorithm to reduce the number of processors using merge conditions, *International Journal on Computer Science and Engineering*, Vol. 4, No. 2, pp. 255-266, February, 2012.

[12] D. B. L. D., P. V. Krishna, Honey bee behavior inspired load balancing of tasks in cloud computing environments, *Applied Soft Computing*, Vol. 13, No. 5, pp. 2292-2303, May, 2013.

[13] T. Dillon, C. Wu, E. Chang, Cloud computing: issues and challenges, *International Conference on Advanced Information Networking and Applications*, Perth, Australia, 2010, pp. 27-33.

[14] E. S. T. El-kenawy, A. I. El-Desoky, M. F. Al-rahamawy, Extended Max-Min scheduling using Petri Net and load balancing, *International Journal of Soft Computing and Engineering*, Vol. 2, No. 4, pp. 198-203, September, 2012.

[15] S. Kumar, A. Mishra, Application of min-min and max-min algorithm for task scheduling in cloud environment under time shared and space shared VM models, *International Journal of Computing Academic Research*, Vol. 4, No. 6, pp. 182-190, December, 2015.

[16] T. Mathew, K. C. Sekaran, J. Jose, Study and analysis of various task scheduling algorithms in the cloud computing environment, *International Conference on Advances in Computing, Communications and Informatics*, New Delhi, India, 2014, pp. 658-664.

[17] S. Mittal, A. Katal, An optimized task scheduling algorithm in cloud computing, *IEEE 6th International Conference on Advanced Computing*, Bhimavaram, India, 2016, pp. 197-202.

[18] K. Shin, M. Cha, M. Jang, J. Jung, W. Yoon, S. Choi, Task scheduling algorithm using minimized duplications in homogeneous systems, *Journal of Parallel and Distributed Computing*, Vol. 68, No. 8, pp. 1146-1156, August, 2008.

[19] M. Geetha, R. C. Suganthe, Task scheduling using artificial bee foraging optimization for load balancing in cloud data centers, *Computer Applications in Engineering Education*, Vol. 28, No. 4, pp. 769-778, July, 2020.

[20] M. A. Alworafi, A. Al-Hashmi, A. Dhari, Suresha, A. B. Darem, Task-Scheduling in Cloud Computing Environment: Cost Priority Approach, International Conference on Cognition and Recognition, Lecture Notes in Networks and Systems, Springer, Singapore, 2018, pp. 99-108.

[21] A. Marphatia, A. Muhnot, T. Sachdeva, E. Shukla, L. Kurup, Optimization of FCFS Based Resource Provisioning

Algorithm for Cloud Computing, *IOSR Journal of Computer Engineering*, Vol. 10, No. 5, pp. 1-5, March-April, 2013.

[22] C. S. Pawar, and R. B. Wagh, Priority Based Dynamic Resource Allocation in Cloud Computing with Modified Waiting Queue, *International Conference on Intelligent Systems and Signal Processing(ISSP 2013)*, Vallabh Vidyanagar, Gujarat, India, 2013, pp. 311-316.

[23] A. Thomas, G. Krishnalal, V. P. J. Raj, Credit Based Scheduling Algorithm in Cloud Computing Environment, *Procedia Computer Science*, Vol. 46, pp. 913-920, 2015.

[24] P. Y. Zhang and M. C. Zhou, Dynamic cloud task scheduling based on a two-stage strategy, *IEEE Transactions on Automation Science and Engineering*, Vol. 15, No. 2, pp. 772-783, April, 2018.

[25] E. Ilavarasan and P. Thambidurai, Low Complexity Performance Effective Task Scheduling Algorithm for Heterogeneous Computing Environments, *Journal of Computer Science*, Vol. 3, No. 2, pp. 94-103, February, 2007.

[26] H. Arabnejad and J. G. Barbosa, List Scheduling Algorithm for Heterogeneous Systems by an Optimistic Cost Table, *IEEE Transactions onParallel and Distributed Systems*, Vol. 25, No. 3, pp. 682-694, March, 2014.

[27] C. Delimitrou and C. Kozyrakis, QoS-Aware Scheduling in Heterogeneous Datacenters with Paragon, *ACM Transactions on Computer Systems*, Vol. 31, No. 4, pp. 12:1-12:34, December, 2013.

[28] K. Dubey, M. Y. Shams, S. C. Sharma, A. Alarifi, M. Amoon, A. A. Nasr, A Management System for Servicing Multi-Organizations on Community Cloud Model in Secure Cloud Environment, *IEEE Access*, Vol. 7, pp. 159535-159546, October, 2019.

[29] J. Li, M. Qiu, Z. Ming, G. Quan, X. Qin, Z. Gu, Online optimization for scheduling preemptable tasks on IaaS cloud systems, *Journal of Parallel and Distributed Computing*, Vol. 72, No. 5, pp. 666-677, May, 2012.

[30] D. Ergu, G. Kou, Y. Peng, Y. Shi, Y. Shi, The analytic hierarchy process: task scheduling and resource allocation in cloud computing environment, *Journal of Supercomputing*, Vol. 64, No. 3, pp. 835-848, June, 2013.

[31] K. Etminani, M. A. Naghibzadeh, A Min-Min Max-Min selective algorithm for grid task scheduling, *3rd IEEE/IFIP International Conference in Central Asia on Internet*, Tashkent, Uzbekistan, 2007, pp. 1-7.

[32] X. He, X. Sun, G. Von Laszewski, QoS guided Min-Min heuristic for grid task scheduling, *Journal of Computer Science & Technology*, Vol. 18, No. 4, pp. 442-451, July, 2003.

[33] A. A. Nasr, N. A. El-Bahnasawy, G. Attiya, A. El-Sayed, A new online scheduling approach for enhancing QOS in cloud, *Future Computing and Informatics Journal*, Vol. 3, No. 2, pp. 424-435, December, 2018.

[34] A. A. Nasr, N. A. El-Bahnasawy, G. Attiya, A. El-Sayed, Cloudlet Scheduling Based Load Balancing on Virtual Machines in Cloud Computing Environment, *Journal of Internet Technology*, Vol. 20, No. 5, pp. 1371-1378, September, 2019.

[35] A. A. Nasr, K. Dubey, N. A. El-Bahnasawy, S. C. Sharma, G. Attiya, A. El-Sayed, HPFE: a new secure framework for serving multi-users with multi-tasks in public cloud without violating SLA, *Neural Computing & Applications*, Vol. 32, No. 11,pp. 6821-6841, June, 2020.

[36] A. A. Nasr, A. T. Chronopoulos, N. A. El-Bahnasawy, G. Attiya, A. El-Sayed, A novel water pressure change optimization technique for solving scheduling problem in cloud computing, *Cluster Computing*, Vol. 22, No. 2, pp. 601-617, June, 2019.

[37] A. Hussain, M. Aleem, *GoCJ: Google Cloud Jobs Dataset*, Mendeley Data, v1, 2018, http://dx.doi.org/10.17632/b7bp6xhrcd.1#file-e5590a0b-e41b-4c66-a2b9-7a606403d69a.

[38] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, A. Y. Wu, An efficient k-means clustering algorithm: analysis and implementation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, No. 7, pp. 881-892, July, 2002.

[39] K. Wagstaff, C. Cardie, S. Rogers, S. Schrödl, Constrained K-means clustering with background knowledge, *Eighteenth International Conference on Machine Learning*, San Francisco, CA, USA, 2001, pp. 577-584.

[40] Z. Yang, X. Qin, W. Li, Y. Yang, A dynamic clustering algorithm for cloud computing, *Information Technology Journal*, Vol. 12, No. 18, pp. 4637-4641, 2013.

[41] F. Y. Yuan, X. C. Zhang, S. B. Luo, Accurate property weighted K-means clustering algorithm based on information entropy, *Journal of Computer Applications*, Vol. 31, No. 6, pp. 1675-1677, June, 2011.

Biographies

Geetha Muthusamy is a PhD Candidate in Information and Communication Engineering at Anna University, Chennai. She received her ME (CSE) degree Anna University, Chennai. Her research interests include scheduling and load balancing in cloud computing, optimization algorithms and multimedia.

Suganthe Ravi Chandran is a PhD Supervisor in Information and Communication Engineering at Anna University, Chennai. She has published more than 25 research articles in National and International Journals. Her research interests include Wireless Networks, Network Security, Blockchain Technology and Deep Learning.