Egyptian Informatics Journal 21 (2020) 201–208

Contents lists available at ScienceDirect

Egyptian Informatics Journal

journal homepage: www.sciencedirect.com

An Improved Task Allocation Strategy in Cloud using Modified K-means Clustering Technique

Vrajesh Sharma [a,*], Manju Bala [b]

[a] I. K. Gujral Punjab Technical University, Kapurthala Punjab, India
[b] Khalsa College of Engineering & Technology, Amritsar, Punjab, India

ARTICLE INFO

Article history:
Received 8 November 2019
Revised 4 January 2020
Accepted 4 February 2020
Available online 20 February 2020

Keywords:
Priority Based Scheduling
Multiple Credits Based Scheduling
Scheduling Strategies in Cloud Computing
Modified K-Means
Categorization of Virtual Machines
Categorization of Tasks
Improving Makespan in Cloud
Reduction in Total Computational Cost in Cloud
QoS
High performance network virtualization
Task Allocation Strategies
Virtualization
Efficient Task Scheduling Strategies
Clustering Method
Four Credits based Scheduling
Scheduling Algorithm in Cloud
Efficiency Improvement in Cloud
Rearrangement of Jobs in Cloud
Mapping Cloudlets to VMs
Reducing Prediction Error in Cloud Computing

ABSTRACT

In the present era, cloud computing has earned much popularity, mainly because of its utilities and relevance with the current technological trends. It is an arrangement which is highly customizable and encapsulated for providing better computational services to its clients worldwide. In cloud computing, scheduling plays a pivotal role in the optimal utilization of resources. Prevalent priority based job scheduling strategies are silent in deciding scheduling scheme for tasks with the same priority and strive hard in appropriately allocating jobs to virtual machines. In the recent years, despite of much research in this field, these scheduling algorithms are unable to provide optimal solution and are lacking in one way or the other in their performance and efficiency. Work pertaining to the use of four criteria/credits for deciding priority, with modified K-means clustering technique is scant. Therefore, to eliminate the drawbacks of the prevalent or existing system and to enhance the performance and efficiency of cloud computing, a new credits based scheduling algorithm has been rendered. The proposed system considers four real time parameters/factors namely Task-Length, Task-Priority, Deadline and Cost, as credits and uses Modified K-means Clustering technique for categorizing the cloudlets and virtual machines (VMs). Results indicate that the suggested scheduling algorithm has excelled existing priority-based scheduling strategy and it has been empirically proven with experimental/simulated results in this paper. CloudSim 3.0.3, a Cloud Simulation Tool has been used to implement and test the proposed algorithm.

1. Introduction

Cloud computing has brought utility oriented IT services as a new trend in the enterprise business for users, worldwide. Millions of users, from all over the world, share cloud services by submitting their millions of computing tasks to the cloud computing environment. The most applaudable feature of cloud computing is "pay-as-you-use". Virtualization is the key technology to enable the deployment of efficient and cost-effective cloud computing platforms [1,2]. It acts as an abstraction layer over the physical resources and creates many virtual instances/objects of the operating systems (OS) for managing different applications to run on the same hardware/computing resources, simultaneously.

* Corresponding author at: H No 656, Street No 6, Narayan Nagar, Hoshiarpur 146001, Punjab, INDIA.
E-mail addresses: vrajeshsharma@gmail.com, vrajesh.sharma@pu.ac.in (V. Sharma).

Peer review under responsibility of Faculty of Computers and Information, Cairo University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.eij.2020.02.001
1110-8665/

Optimal management of the available resources is proving to be a big challenge in cloud computing and it is apparent that scheduling plays a pivotal role in improving the utilization of computational devices. Scheduling can be understood as the succession of strategic decisions for the allocation of available resources to jobs and customers on-time. By mapping the tasks aptly to suitable Virtual Machines (VMs), this arrangement not only increases the resource utilization but also increases the throughput of the system. As cloud computing is stretching its dimensions day by day, the task scheduling for the optimal resource utilization is becoming more and more convoluted [3,4].

Task scheduling is essentially a major task which is handled by datacenter brokers. Broker plays a central role in managing the mapping of tasks to the VMs [5] and also keeps the record of available list of VMs while maintaining their Quality of Service [6]. User sends request to Broker and it is Broker's responsibility to appropriately allocate the VM to the specific job/task without violating Service Level Agreement (SLA) or compromising Quality of Service (QoS). In a cloud computing environment, high QoS is assured by the high performance of VMs. When a Low Performing VM is allocated to a High Performance job, it underutilizes the available resources resulting in weak performance and throughput which principally violates Service Level Agreement (SLA) [5]. So, it has become the need of the hour to deploy highly efficient scheduling strategies at the broker level and it has further paved the path for more research in the area of task scheduling algorithms for cloud computing.

2. Literature survey

There is a lot of scope for researchers in devising new scheduling strategies for cloud computing environment. In recent years several scheduling strategies have been proposed to augment the performance of network operations, memory and processors. A traditional cost based scheduling has been proposed by Selvarani et al. [7], by allocating appropriately desired resources it reduces the overall execution cost. As per their processing capabilities, various tasks/cloudlets have been segregated and are allocated with apt resources to meet the minimum total task completion time and minimum cost. This approach can be advanced to handle concurrent tasks, considering more complex scenarios with dynamic parameters and real time factors. To improve overall QoS and throughput of datacenter, Moses et al. [8] have proposed a shared resource monitoring strategy for understanding the usage of resources of each Virtual Machine (VM) on each platform. It gathers resource usage information across various platforms while migrating the VMs which are resource-constrained. The author has tried to incorporate the concept of priority assignment based on the QoS, by taking into account various constraints such as execution time and cost of application. Detailed profiling of VMs is necessary to steer scheduling strategies along with VPA (Virtual Platform Architecture) monitoring to monitor cache on VMs.

We have observed that some jobs need immediate attention as they cannot sustain for long or are time bound in a system therefore they are needed to be catered earlier than the others making task priority an essential concern. To handle this problem, researchers have used job Priority as a main constraint and nodded that it must be marked as main factor/parameter for effective scheduling strategies. Ghanbari et al. [9] have anticipated a priority-based-job-scheduling technique (PJSC) by considering Analytical Hierarchy Process (AHP) as an underlying concept. It is a multi-criteria-decision-making (MCDM) model considering multiple attributes, complexity and finish time. The proposed algorithm can be enhanced to achieve even lesser makespan time (finish time) by considering other QoS parameters for scheduling. Priority scheduling is a much referred conventional technique for assigning and selecting a task from a batch by using QoS parameters. On the contrary when priority is assigned statically to a task, it lays out many problems and difficulties. In this paper Xiao et al. [10] have rendered a priority based strategy for discovering the optimal choices for the existing scenario. It has been found that priority based method has more advancements over FCFS technique and is far more beneficial too. If more information be gathered on regular fashion of its usage this strategy can be advanced even further. Moreover, in this paper it was proposed to bargain with users for their waiting time and termed it as profit for others. The existing fair queuing scheduling algorithm performs well in data applications but does not guarantee a fair service in real time application like voice or interactive videos. For this Yang et al. [11] have proposed a class based weighted fair scheduling (CBWFQ) technique which improves the network performances, time delay and fairness. This algorithm is also silent in deciding scheduling strategy where the job priority is same.

Wang et al. [12] have proposed an algorithm based on adaptive scheduling and Quality of Service (Adaptive-Scheduling-with-QoS-Satisfaction) and named it AsQ. This was referred as a multi-choice knapsack problem where a fast scheduling strategy was discussed using MAX_MIN strategy without wasting time on decision making. This approach can be further advanced for private clouds by devising a better workload shifting technique and taking into account various parameters like energy efficiency, operation cost and execution time. To allocate the resources depending upon the user's demand and to assign priority to every task (based on their parameters) Gupta et al. [13] have proposed the preemptive scheduling of jobs and these preemptive jobs are then fed to a single and common queue for newly arriving jobs and preempted jobs. To process preempted tasks, the concept of a waiting queue was also proposed in this technique. Tasks in the waiting queue which were preempted earlier are migrated to other virtual machines. It is an Earliest-Deadline-First, priority based scheduling method. Authors have coined waiting queue concept to process the preempted tasks which would not only minimize the waiting time but it would improve the completion time of the jobs which were preempted.

Many researchers/scholars are studying and researching in the fields of Scheduling for Virtual Resources (Non-dominated Sorting Genetic Algorithm II) [4], Context Aware Scheduling [14], Cost Based Scheduling [15], Dynamic Slot Based Scheduling [16] and Energy Efficient Optimization Methods [17] to enhance the optimal usage of resources with minimal cost, by improving the scheduling strategies in cloud computing.

For improving the overall productivity and total throughput of the datacenter, Lakra et al. [5] have suggested to aim at multiple objectives while keeping QoS as a main criterion, so that allocation of task priority relies on the Quality of Service (QoS). In this algorithm, the jobs which are allotted with lower_QoS are the jobs having higher priority actually, while the tasks allotted with Higher_QoS are the task having lower priority value. The whole focus of this technique is to minimize the total execution time taken by a task. It can be further improved by employment of a better arranging/categorizing technique for the VMs & Tasks and using more factors/parameters for QoS. This experimental setup has been made/simulated using CloudSim simulator.

Cloud Computing is a real time scenario where algorithms based on single criteria fail to provide an acceptable task scheduling strategy, instead, more real time parameters should be included for providing an optimal solution. Thomas et al. [6] have proposed an improved credit based scheduling algorithm considering user priority and task length as two credits. No special importance is given to any high priority task, when it arrives; instead every task is allocated with combined credits which are obtained

as a product of task length credit and priority credit. This cumulative total of credits becomes the base for the actual scheduling of the tasks. The proposed scheme is silent on real time systems where deadline is the bottleneck and hence needs research for delving out more real-time parameters and factors for tasks. To reduce the execution cost and makespan time, Ibrahim et al. [18] have proposed, an enhanced task scheduling policy where the available VMs are allocated to the requesting tasks based upon their processing powers and price of execution. For effective estimation and calculation of the execution cost/price, authors have used Amazon EC2 and Google pricing models. This system can be further improved by taking into account the dynamic workflow scheduling and dependent tasks. For improving the prediction as well as overall accuracy, Kaur et al. [19] have proposed a Support Vector Machine algorithm using hybrid K-means technique. In traditional k-means clustering technique the initial centroids are selected randomly which is not a case in this modified and improved strategy. Here, the data is first partitioned in to several 'p' equal parts and then in the second step, the centroid point is calculated by taking the arithmetic mean of the each 'p' part. Distance between each data object and all k cluster centers, is calculated till no change is there in the center of clusters. This algorithm shows very promising outcomes and accuracy levels.

As we have already discussed in this paper that Cloud Computing is a real time scenario where algorithms based on single criteria fail to provide an acceptable task scheduling strategy. Literature survey also indicates that the work pertaining to the use of four criteria/credits for deciding priority with modified K-means clustering technique is not available; so, as the outcome of this, it was decided to consider four parameters for assigning priority credits to Tasks using Modified K-means Clustering technique. Paper structure is as follows; Section-III shows the flow chart of the Existing system and elaborates in detail the proposed work. Section-IV apprises the necessary arrangements and Experimental Setup and also discusses the Simulation Results. In Section-V the proposed research has been concluded and it sheds light on the future scope of the said work and in Section VI is the References.

3. Existing system (with two credits) and proposed work (with additional real time credits and Modified K-means Algorithm)

In the existing system [6] (with two credits viz. task length and task priority) as illustrated in Flow Chart Fig. 1, while sorting the tasks, there may arise many such cases in which two or more tasks may have the same priority [2,9], therefore to overcome that same priority issue, an improved scheduling technique/algorithm is rendered, that assigns the priority depending upon the combination of credits given to a task namely Task Length, Task Priority, Deadline

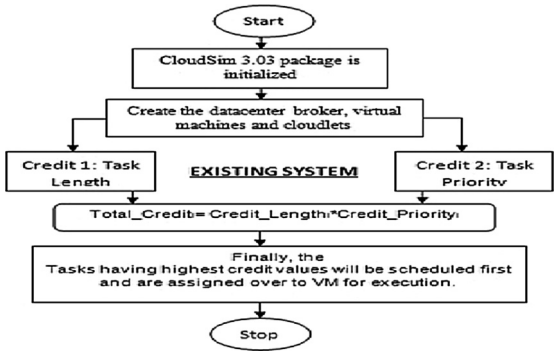

Fig. 1. Flow Chart Representation of the Existing System.

of the Task and Cost. These four parameters are combined to obtain the final credit for the task; this will reduce the chances of same priority occurrence between the two tasks.

Another challenge is to appropriately map/allocate the tasks to the VMs by the broker, in a way that the processing time, makespan time (finish time) and execution cost is optimized. This proposed system is aimed to resolve the priority issues and strives to achieve the appropriate allocation/mapping of jobs to virtual machines while optimizing the efficiency of cloud computing system.

The Proposed (credit based) system uses 04 parameters for assigning priority credits to Cloudlets/Tasks viz.

(i) Task Length (ii) Task Priority (iii) Deadline and (iv) Cost.

3.1. Task length credit

It can be understood as a credit assigned to a task which is obtained from already fed task length values by users, using Procedure 1 (given below). Tasks of variable lengths are fed to cloud system for execution. For providing effective scheduling strategy, in the sorted array of the tasks, it is always advised that the tasks should be taken from both ends of the array i.e. from the front as well as rear. Further, to assign the Task-Credits following steps are taken [6]. Firstly, the length of every task (Tl_i) is calculated. Then, the average length (L_avg) of the all the tasks is calculated. Then, the difference of the length of each task with respect to average length, is calculated

For tasks T1, T2, T3 the Task Length Difference TLD can be represented as TLD = L_avg-Tl_i

Procedure 1: Steps to assign credits (crd) based on the length of task

In this scenario, it is very useful when the tasks are sorted and preferably in ascending order of task length. Here, in this proposed technique, there are five credits which are assigned to every task for different conditions and for allocating these credits, four different values (val_1, val_2, val_3 and val_4) are calculated from the Task Length array. Subsequently, these values (val_1, val_2, val_3 and val_4) set the conditions for assigning credits to the tasks. It is notified that these values cannot be chosen out of the bound therefore these values must be from and within the range of task-length.

val_1 = h_len/5 (i)

val_2 = h_len/4 (ii)

val_3 = val_2 + val_1 (iii)

val_4 = val_3−val_2 (iv)

Where, h_len is the task with the highest length.

For (Tn) // all tasks in the set.

TLD (Task length difference) = absolute value (average length – length of particular task) [6]

if $TLD_n \leq val\_1$

Then set crd = 5

Else if $val\_1 < TLD_n \leq val\_2$

Then set crd = 4

Else if $val\_2 < TLD_n \leq val\_3$

Then set crd = 3

Else if $val\_3 < TLD_n \leq val\_4$

Then set crd = 2

Else $val\_4 > TLD_n$

Then set crd = 1

End For

Where val_1 = h_len/5

val_2 = h_len/4

val_3 = val_2 + val_1

val_4 = val_3−val_2

With the completion of this process, every task would be marked/assigned with a credit depending upon the length of the task (credit_length).

3.2. Task priority credit

Most of the times, various tasks come with different set of priority values marked/assigned on each of them. As, in cloud environment, millions of the tasks are submitted for scheduling and normally the priority value comes out to be same for many tasks. It has been observed that conventional scheduling strategies which are based on task priority, find it difficult to handle and schedule the tasks with same/similar priority. This condition does not erupt in the proposed approach because even when credits are being given to each task based on their priority; still the total credits are calculated or measured on the basis of all the assigned credits values, respectively (Credit_Length, Credit_Priority, Credit_Deadline & Credit_Cost). Moreover, there aren't any default credits set for a task but the value for credits will change with priority which is in turn assigned by the user itself.

Procedure 2: Steps to assign credits based on the priority of task

To assign the Priority-Credits following steps are taken. 1) In the first and the very basic step, highest priority number is sought [6]. 2) In the second step, for each task division factor (div_fac) for finding the priority fraction (Prio_fr) is chosen. Suppose, the highest task priority value is a two digit number then division factor would be chosen as 100 and if it is in 3 digits then the division part would be 1000. 3) In the third step, task-priority-value (prio_val) is divided by division factor (div_fac) to obtain Priority Fraction (Prio_fr), for every task.$Prio\_fr_n = prio\_val_n / div\_fac_n$[6] 4) At last, Priority Fraction (Pri_fr) would be associated to each task as Credit_Priority (Priority Credits).	The process of assigning the priority credits to the tasks is represented in the pseudocode as below. For (Tn) // all tasks in the set. Sought for the highest priority task Choose the divisible factor for priority ($credit\_Priority_n = TaskPriority_n / divisiblefactor_n$) $Prio_fr_n = prio_val_n / div_fac_n$End For

3.3. Deadline of the task as credit

A deadline is the target time-frame signifying the time constraint of a submitted task. In real time applications, mostly, tasks come with a deadline for keeping the pace and saving the task from starving which would otherwise violate the basic feature and utility of real time applications.

Procedure 3: Steps to assign credits based on the deadline of the task

Deadline Credits of the task can be calculated as under. 1) In the First step, the maximum capability of the VM is found depending upon the MIPS of the Virtual Machine List which is available for the execution of the tasks. 2) Then, the Deadline Credits can be calculated as a product of Credit_Length and Credit_Priority divided by Maximum Capability of the Virtual Machine	This process of assigning the Deadline Credits to the tasks is represented in the pseudocode as below. For (Tn) // *all tasks in the set.* Find out the MAX_MIPS of the VM from the Virtual Machine List $Credit_Deadline_n = Credit_Length_n * Credit_Priority_n / MIPS_{MAX}$End For

In the end, every task would be marked/assigned with a $Credit\_Deadline_n$ depending upon the Deadline of the task

3.4. Execution cost of the task

In cloud computing environment Cost of the Task can be simply understood as an amount that has to be paid for using resources and utilities. Cost of a task can be calculated as the sum of the products of $Cost\ per\ Memory_{Size}$ & VM_{RAM} and Cost per Storage & VM_{size}. This can be represented as follows:

Procedure 4: Steps to assign credits based on the cost of the task

For (Tn) // *all tasks in the set.*$Credit_Cost_n = Datacentercharacteristics.getCostperMemory * vm_{Ram} + Datacentercharacteristics.getCostperStorage * vm_size$End For

In the end, every task would be marked/assigned with a Credit_Cost depending upon the Cost of the task.

After performing all these four procedures, various credits are obtained on the basis of all the four parameters for assigning priority credit to Task/Cloudlets i.e. Task Length, Task Priority, Deadline and Cost.

3.5. Total credits of the task can be calculated as

$$Total\_Credit_n = Credit\_Length_n * Credit\_Priority_n * Credit\_Deadline_n * Credit\_Cost_n$$

1) Once, the Totals Credits are assigned to the tasks, the tasks are sorted using descending sort operation on the basis of the assigned credits.
2) To enhance the efficiency of our scheduling strategy, a more efficient scheduling strategy/algorithm is being proposed with the incorporation of modified K-means technique for clustering of Tasks and VMs
3) The next very important step is the clustering of the sorted tasks and virtual machines, using Modified K-means (M-Kmeans) clustering method.
4) Clustering is done at the task/cloudlet side on the basis of four assigned parameters/factors namely (i) Task_Length, (ii) Task_Priority, (ii) Deadline and (iv) Cost, clustering is done at the task/ cloudlet side.
5) Clustering at virtual machine side is done using bandwidth, RAM, MIPS and size.

3.6. Modified K-means algorithm

Modified K-means technique not only enhances accuracy for prediction but also improves the efficiency of the whole system [19]. Unlike, traditional K-means clustering algorithm, where the initial centroids (mean of all objects in each cluster) are randomly picked /chosen, data is first partitioned in to several 'p' equal parts and then arithmetic mean of every part is considered as the centroid point (cpoint), in this modified and improved method. Distance between each data object and all p cluster-centers, is calculated till no change is there in the center of clusters.

The proposed system is aimed to reduce prediction error and Total Computational Time by categorizing/segregating Tasks and VMs using Modified K-Means Clustering Technique.

Procedure 5: Modified K-means Algorithm

M-Kmeans Algorithm as function MKMEANS();Input: Unorganized Tasks/VMsOutput: Clustered Tasks/VMs

1) The dataset is partitioned into p equal parts.
2) centroid point = arithmetic mean of each part p.
3) Repeat;
4) Calculate: distance between every dtpoint. $dtpoint_i = (1 \le i \le n)$& all p cluster centroids $cpoint_i = (1 \le i \le p)$ and map the dtpoint to the nearest cpoint i.e. nearest cluster.
5) For each cluster p $(1 \le h \le p)$, recalculate the *cpoint* until convergence condition reached.

This algorithm shows very promising outcomes and accuracy levels.

After performing Modified K-means clustering, the descending sort operation within the clustered groups is applied/performed which further creates three groups (with priority levels high, medium and low) in VMs as well as Cloudlets/tasks. After this step, all the Cloudlet Clusters are ready to be mapped/assigned to Virtual Machine Clusters and finally, initiates the final execution of the tasks.

3.7. A novel scheduling algorithm for improving efficiency of cloud computing

The course of actions to implement the proposed strategy (discussed in Procedures 1, 2, 3, 4 & 5) has been diagrammatically shown in Fig. 2.1 and been put in a succession, stepwise in Fig. 2.2.

3.8. Methodology

1. CloudSim 3.03 package is initialized by creating the Datacenter Broker, virtual machines and cloudlets.
2. The list of unallocated Tasks/cloudlets and unassigned VMs are provided to Datacenter Broker.
3. Total credits of the task are calculated on the basis of a combination of credits, given to the task as parameters, namely (i) task length, (ii) task priority, (iii) deadline and (iv) cost.
4. For categorizing cloudlets/tasks, Modified K-means clustering technique is used, which categorizing cloudlets/tasks in to further three groups of priority High, medium and low, in the descending order.
5. Depending upon the MIPS, Size, Bandwidth and RAM, Datacenter Broker (DCB) calculates the processing capabilities of all the Virtual Machines and deploys Modified K-means clustering algorithm to arrange or categorize them into three clusters of priority High, Medium and Low, in the descending order.

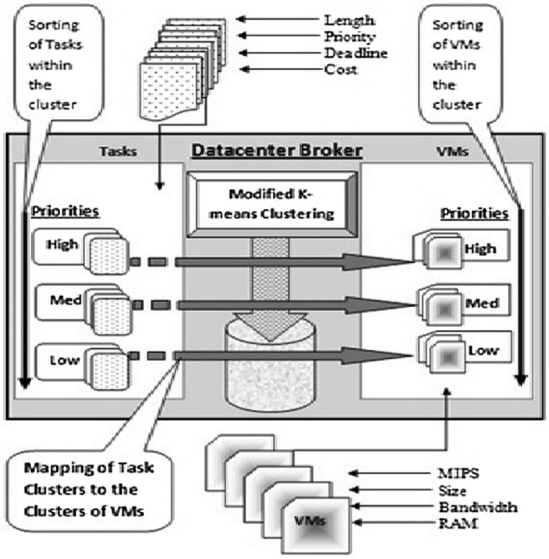

Fig. 2.1. Representation of the Proposed System, diagrammatically.

Input: - (a) Unassigned Cloudlets/Tasks, (b) VMs.
Output: (a) Makespan Time, (b) Processing Time, (c) Processing Cost.

1. Input /Initialize the Tasks (Tn) to the Cloud Simulator (CloudSim 3.0.3)
   (To assigns the priority on the basis of combination of credits given to the task on the basis of 04 parameters)
2. For each n number of Tasks in T. *// To assigns the priority on the basis of combination of credits*
   assigned to the task.
   $Total\_Credit_n = Credit\_Length_n * Credit\_Priority_n * Credit\_Deadline_n * Credit\_Cost_n$
   End For
3. Apply the Modified K-Means Clustering technique on Tasks and segregate them into three clusters.
4. Invoke Function MKMEANS(tasks) *//Call Procedure 5: Modified Kmeans*
5. Initialize/Create Virtual Machines (VMs) in the Cloud Simulator (CloudSim 3.0.3).
6. For each VM v in VMs.
   Get the values of (MIPS, Size bandwidth and ram) processing power, capacity, bandwidth and memory of each VM.
   End For
7. Apply the Modified K-Means Clustering technique on VMs and segregate them into three clusters.
8. Invoke Function MKMEANS(VMs) *//Call Procedure 5: Modified Kmeans*
9. Perform Operation descending sort on Tasks and VMs and divide them in to High, Medium and Low priority Clusters.
10. For each Task/Cloudlet q
    Assign Task/Cloudlet q to VMq of appropriate cluster
    Vindex++
    If Vindex>=VmListSize
    {
    Vindex= =0;
    }
    End For
11. Mapping of the task with the virtual machines using function
    Task.setVMid(m.id)
    sendNow(virtual machine id, Cloudsim Submission Tag, Task);

Fig. 2.2. Representation of *A novel scheduling algorithm for improving efficiency of cloud computing* diagrammatically.

6. In the last step, all these Cloudlet Clusters are assigned or mapped to Virtual Machine (VM) Clusters which leads finally to execution of the tasks.

Through this approach, high-priority jobs are quickly allocated to high-performance Virtual Machines as well as the low-priority and medium-priority jobs are optimally allocated to aptly performing VMs and hence it increases the overall throughput of the system.

4. Experimental setup and Simulation

4.1. Cloud Simulator: CloudSim 3.0.3

The proposed algorithm has been implemented in Java using CloudSim 3.0.3 as a Cloud Simulator. It helps to deploy various cloud applications and allows to create datacenters, virtual

machines and other facilities which can be rapidly generated as per need and with utmost ease. Selecting or Choosing a simulator, majorly, depends upon the nature and kind of research. Many simulators are available for different types of research issues in cloud environment namely CloudAnalyst, GreenCloud, iCanCloud, NetworkCloudSim, EMUSIM, GroudSim, MR-CloudSim, SmartSim, DCSim, SmartSim, SimIC, SPECI, DynamicCloudSim, CloudSimSDN, secCloudSim, CEPSim, PICS etc. These all have their advantages over the other and each one is best suited for its type. Due to its features and popularity, CloudSim is mostly recommended as a general purpose simulator for researchers [20].

Table 1
Basic configurations of datacenters.

Number of Datacenters	5
Number of Hosts under each Datacenter	2
Total Hosts	10
Number of cloudlets/tasks	200–3600
Number of Brokers	1

Table 2
Basic configurations of host.

Number of Virtual Machines (VMs)	80
RAM (MB) value initialized in simulator	20,480
MIPS (Millions of instructions per second)	5000
Storage (MB) value initialized in simulator	1,048,576
Bandwidth (MB/sec) value initialized in simulator	500,000

Table 3
Virtual machines and their Basic Configuration.

RAM (MB) value	256
MIPS	(MIPS = 250,count = 0; count < 18; MIPS + 5, count++)
Bandwidth (MB/second) value	(bandwidth = 1000, count=0; count < 18; bandwidth + 500, count++)
Number of Cores	1
Size (MB) value	(size = 10000, count = 0; size < 18; size = size + 2343, count++)

4.2. The setup and basic configuration of proposed system

4.2.1. Equations

Makespan: Makespan can be defined as the total amount of time taken for the overall/ total completion of given tasks. Overall-task-completion-time (Total Completion Time) can be represented as T_a on VM_b as CT_{ab}. hence Makespan Time (MS) is represented as following [14]:

$$MS = max\{CT_{ab} | a \in T,\ a = 1,2..n \text{ and } b \in VM,\ b = 1,2..m\} \quad (1)$$

Processing Time: Processing Time can be calculated as length of the task divided by the product of MIPS of a Virtual Machine and Number of processing elements (NumberOfPes).

$$\text{Processing time} = \text{CloudletLength}/\text{vmMips} * \text{vmNumberOfPes} \quad (2)$$

Memory Cost: The cost associated with the memory utilization of a task can be calculated as

$$\text{CostPerMem} * \text{vm.getRam} + \text{CostPerMem} * \text{vm.getSize} \quad (3)$$

Total Computational Cost: It can be calculated by taking the product of Eq. (2) *Eq. (3).

$$\text{Total Computational Cost} = (\text{CloudletLength}/\text{vmMips} * \text{vmNumberOfPes}) * (\text{CostPerMem} * \text{vm.getRam} + \text{CostPerMem} * \text{vm.getSize}) \quad (4)$$

In simulation, we can use any currency value as Units for Processing Cost and Total Computational Cost (Network Cost is not included) Tables 1–3.

4.3. Experimental observations (simulated environment)

Experiment has been conducted on Cloudsim 3.0.3(a Cloud Simulation Tool) and simulated observations of the Existing System/ Prevalent System, which is a Credit Based System considering two parameters as Length and Priority, are recorded in Table 4:

Experimental Results or the simulated observations of the Proposed system which is a Credit Based System considering Task Length, Task Priority, Deadline and Cost as Parameters coupled

Table 4
Simulated observations of the Existing System/Prevalent System.

Existing System (Credit-Based-System having two credits as Task Length & Task Priority)					
S. No	No of Cloudlets/ Tasks	MS Makespan Time *(in milliseconds)*	Processing Time *(in ms)*	Processing Cost *(currency units)*	Total Computational Cost *(currency units)*
1	200	1449.77	4699.86	41204.52	193655475.37
2	400	2549.29	14512.51	81284.4	1179640667.84
3	600	4222.58	28727.09	120145.92	3451442656.97
4	800	5014.42	48137.36	157882.8	7600061181.41
5	1000	7089.1	72082.29	194401.32	14012892324.62
6	1200	7758.62	101166.52	229795.2	23247580696.70
7	1400	10,207	134558.88	263970.72	35519604435.99
8	1600	10747.84	173236.25	297021.6	51454908153.00
9	1800	13611.04	216381.83	328854.12	71158056288.64
10	2000	14016.45	264733.16	359,562	95187984475.92
11	2200	17342.27	317326.17	389051.52	123456228774.28
12	2400	17605.45	375270.64	417416.4	156644119574.50
13	2600	21450.45	437616.87	444562.92	194548233568.46
14	2800	21652.47	505235.29	470584.8	237756047897.59
15	3000	25995.68	577028.95	495388.32	285853402131.86
16	3200	26208.48	654240.52	519067.2	339594794842.94
17	3400	31051.61	735787.4	541527.72	398449273126.73
18	3600	31276.33	822672.92	562863.6	463052641373.71

Table 5
Simulated observations of the proposed system.

Credit Based System with Length, Priority, Deadline and Cost as Credit Parameters with Modified K-means Clustering Algorithm					
S. No	No of Tasks	Makespan Time (in milliseconds)	Processing Time (ms)	Processing Cost	Total Computational Cost
	200	981.87	4365.05	21254.26	92775907.61
	400	971.87	11528.11	44315.79	510877301.86
	600	1420.92	23336.36	63390.41	1479301428.31
	800	1900.96	39632.8	81312.28	3222633330.78
	1000	3368.77	65783.82	116705.01	7677301370.94
	1200	2967.7	79773.95	171230.04	13659696649.46
	1400	4893.38	118925.43	197922.02	23537961334.97
	1600	5478.08	136358.05	229255.48	31260830204.61
	1800	10857.33	248257.67	171005.97	42453543668.29
	2000	9466.71	227881.06	294480.96	67106633314.62
	2200	10068.66	315514.58	243388.06	76792481527.91
	2400	8766.33	325405.87	235253.66	76552921902.98
	2600	11794.6	377968.33	241386.05	91236282203.80
	2800	10849.62	431165.62	278629.12	120135297274.85
	3000	21475.96	627686.63	311169.05	195316652354.80
	3200	16882.25	654375.92	338277.17	221360434333.75
	3400	18368.89	736838.12	355923.67	262258127866.30
	3600	20616.19	736589.93	354695.86	261265398688.69

with Modified K-means Clustering Technique, are mentioned in Table 5:

4.4. Results & discussions

Upon, assessing and evaluating empirically the Makespan Time (Eq. (1)), Processing Time (Eq. (2)) and Total Computational Cost (Eq. (4)) of Cloudlets/tasks in Existing System and Proposed System, it was observed that the proposed System/algorithm shows better results and presents reduced Makespan Time, decreased Processing Time and Total Computational Cost while increasing the throughput of the cloud computing system, which can be analyzed in the Figs. 3 and 5 and Figs. 5.1 & 5.2.

Impact of the proposed system on processor, memory and network operations is as follows.

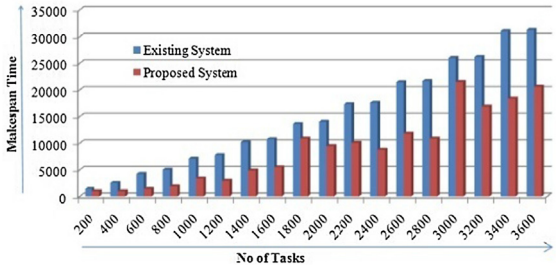

Fig. 3. 3-D Cluster-Column Graphical Representation of the Experimental Results for Makespan Time in the Existing System vs. Proposed System.

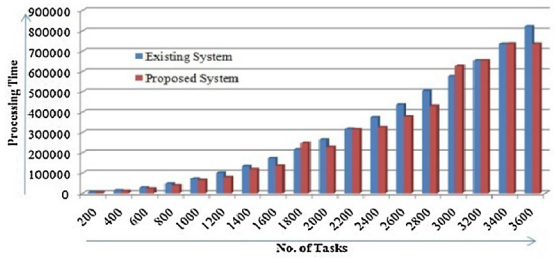

Fig. 4. 3-D Cluster-Column Graphical Representation of the Experimental Results for Processing Time in the Existing System vs. Proposed System.

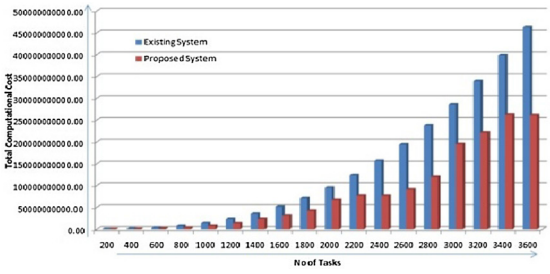

Fig. 5.1. 3-D Cluster-Column Graphical Representation of the Experimental Results for Total Computational Cost in the Existing System vs. Proposed System.

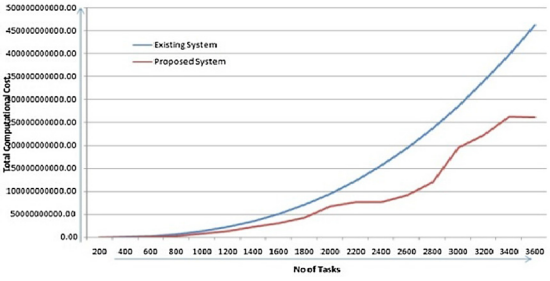

Fig. 5.2. Line Chart Graphical Representation of the Experimental Results for Total Computational Cost in the Existing System vs. Proposed System.

a. *Processor*: Experimental evaluations and results indicate that the proposed approach provides 9.43% better results in Processing Time as compared to the Existing System. Thus it improves the efficiency of the processor while uplifting the performance of the whole system.
b. *Memory*: In task scheduling strategies, the submitted jobs/tasks utilize two types of memories viz. RAM and physical memory for transitional data & storage in virtual machines, respectively It has been observed that longer the task resides in queue for getting executed; it keeps hold of that memory for longer duration. Simulation Results show that the proposed system has gained huge leap in the performance for Makespan Time by providing 44.75% better results than the existing system and has saved physical memory allocated to tasks and RAM by relinquishing the occupied

memory on cloud resources earlier than the existing system, which is now free and available to other jobs/customers for use.

c. *Network Operations*: With the improved Makespan Time and Processing Time the proposed system is saving time on network operations by making the channel free earlier than the existing system. Results indicate that the proposed system is 43.81% better in terms of Total Computational Cost.

5. Conclusion

To overcome that same priority issue and to appropriately map the tasks to the virtual machines, a multiple credits based scheduling algorithm was developed considering four parameters, which incorporates Modified K-means clustering technique for the categorization of Cloudlets/tasks and VMs. CloudSim 3.0.3, a Cloud Simulation Tool has been used to implement and test the proposed algorithm using the framework of Java. The obtained results were compared against the Existing System. Results indicate that the Makespan Time, Processing Time and Total Computational Cost were reduced considerably by using the proposed system/algorithm. Some spikes were observed while plotting the graphs for proposed system, which suggests the need to balance the load by implementing some superlative techniques of load balancing. In future, more Quality of Service factors/parameters and effective load balancing techniques can be researched for enhanced performance and improved efficiency of cloud computing system.

References

[1] Bourguiba M, Haddadou K, El Korbi I, Pujolle G. Improving network I/O virtualization for cloud computing. IEEE Trans. Parallel Distrib. Syst. 2014;25 (3):673–81. doi: https://doi.org/10.1109/tpds.2013.29.

[2] Buyya R, Broberg J, Andrzej G. Cloud Computing: Principles and Paradigms. New Delhi: Wiley India Pvt. Ltd.; 2015. p. 637.

[3] Sharma V., Chhabra N., Bala M.,(2017) An_Approach_to_Improve_Efficiency_of_Cloud_Computing. In: Proceedings International Interdisciplinary Conference on Science Technology Engineering Management Pharmacy and Humanities Held on 22nd – 23rd April 2017, in Singapore, paper 27, ISBN: 9780998900001

[4] Zhao, J., Zeng, W., Liu, M., & Li, G. (2011). Multi-objective optimization model of virtual resources scheduling under cloud computing and it's solution. 2011 International Conference on Cloud and Service Computing. doi:10.1109/csc.2011.6138518.

[5] Lakra AV, Yadav DK. Multi-objective tasks scheduling algorithm for cloud computing throughput optimization. Procedia Comput. Sci. 2015;48:107–13. doi: https://doi.org/10.1016/j.procs.2015.04.158.

[6] Thomas A, Krishnalal G, Jagathy Raj VP. Credit based scheduling algorithm in cloud computing environment. Procedia Comput. Sci. 2015;46:913–20. doi: https://doi.org/10.1016/j.procs.2015.02.162.

[7] Selvarani, S., & Sadhasivam, G. S. (2010). Improved cost-based algorithm for task scheduling in cloud computing. 2010 IEEE International Conference on Computational Intelligence and Computing Research. doi:10.1109/iccic.2010.5705847.

[8] Moses, J., Iyer, R., Illikkal, R., Srinivasan, S., & Aisopos, K. (2011). Shared Resource Monitoring and Throughput Optimization in Cloud-Computing Datacenters. 2011 IEEE International Parallel & Distributed Processing Symposium. doi:10.1109/ipdps.2011.98.

[9] Ghanbari S, Othman M. A priority based job scheduling algorithm in cloud computing. Procedia Eng. 2012;50:778–85. doi: https://doi.org/10.1016/j.proeng.2012.10.086.

[10] Xiao, J., & Wang, Z. (2012). A Priority Based Scheduling Strategy for Virtual Machine Allocations in Cloud Computing Environment. 2012 International Conference on Cloud and Service Computing. doi:10.1109/csc.2012.16.

[11] Yang L, Pan C, Zhang E, Liu H. A new class of priority-based weighted fair scheduling algorithm. Phys. Procedia 2012;33:942–8. doi: https://doi.org/10.1016/j.phpro.2012.05.158.

[12] Wang W-J, Chang Y-S, Lo W-T, Lee Y-K. Adaptive scheduling for parallel tasks with QoS satisfaction for hybrid cloud environments. J Supercomputing 2013;66(2):783–811. doi: https://doi.org/10.1007/s11227-013-0890-2.

[13] Gupta, G., Kumawat, V. K., Laxmi, P. R., Singh, D., Jain, V., & Singh, R. (2014). A simulation of priority based earliest deadline first scheduling for cloud computing system. 2014 First International Conference on Networks & Soft Computing (ICNSC2014). doi:10.1109/cnsc.2014.6906659

[14] Assuncao, M. D., Netto, M. A. S., Koch, F., & Bianchi, S. (2012). Context-Aware Job Scheduling for Cloud Computing Environments. 2012 IEEE Fifth International Conference on Utility and Cloud Computing. doi:10.1109/ucc.2012.33

[15] Yang, Z., Yin, C., & Liu, Y. (2011). A Cost-Based Resource Scheduling Paradigm in Cloud Computing. 2011 12th International Conference on Parallel and Distributed Computing, Applications and Technologies. doi:10.1109/pdcat.2011.1

[16] Shih H-Y, Huang J-J, Leu J-S. Dynamic slot-based task scheduling based on node workload in a MapReduce computation model. Anti-Counterfeiting, Secur. Identif. 2012. doi: https://doi.org/10.1109/icasid.2012.6325318.

[17] Liang Luo, Wenjun Wu, Dichen Di, Fei Zhang, Yizhou Yan, & Yaokuan Mao. (2012). A resource scheduling algorithm of cloud computing based on energy efficient optimization methods. 2012 International Green Computing Conference (IGCC). doi:10.1109/igcc.2012.6322251.

[18] Ibrahim, E., El-Bahnasawy, N. A., & Omara, F. A. (2016). Task Scheduling Algorithm in Cloud Computing Environment Based on Cloud Pricing Models. 2016 World Symposium on Computer Applications & Research (WSCAR). doi:10.1109/wscar.2016.20.

[19] Kaur, S., & Kalra, S. (2016). Disease prediction using hybrid K-means and support vector machine. 2016 1st India International Conference on Information Processing (IICIP). doi:10.1109/iicip.2016.7975367.

[20] Pericherla S Suryateja,"A Comparative Analysis of Cloud Simulators", International Journal of Modern Education and Computer Science(IJMECS), Vol.8, No.4, pp.64-71, 2016.DOI: 10.5815/ijmecs.2016.04.08.

Vrajesh Sharma was born in Vrindavan, Uttar Pradesh, India on June 29, 1980. He graduated from Govt College Hoshiarpur in year 2000, affiliated to Panjab University, Chandigarh and did Masters in Computer Applications (with Hons), from Punjab Technical University Jalandhar, in 2004.He is pursuing Ph.D. from I. K .Gujral Punjab Technical University Jalandhar, India on the Topic of "An Optimal Approach to improve the Efficiency of Cloud Computing". His major field of study is Cloud Computing and Computer Networks. His employment history includes over 15 years' of qualitative technical experience in Industry, College and University. Presently, working as a System Manager in Panjab University Swami Sarvanand Giri Regional Centre, Hoshiarpur, an off-campus establishment of Panjab University, Chandigarh. He has worked as a casual anchor in All India Radio Jalandhar from 2000 to 2004 for Morning Broadcasts of Shows Sajjri Sawer on FM and Navi Sawer on MW. He has Published over 14 papers in reputed Journals, International and National Conferences.

Dr Manju Bala received her Ph.D. degree in Computer Science and Engineering from NIT Hamirpur, Himachal Pradesh, India. She is currently the Director of Khalsa College of Engineering and Technology, Amritsar, India. Her areas of researches include Wireless Sensor Networks and Data Communication.