A Superlative Approach to Improve the QoS by Load balancing in Cloud Computing

Vrajesh Sharma[a*], Manju Bala[b]
[a]*Ph.D. Research Scholar, I.K.Gujral. Punjab Technical University, Kapurthala (Punjab), India.*
[b]*Director, Khalsa College of Engineering & Technology, Amritsar , (Punjab), India.*

***Abstract**: Scheduling of tasks and resources is a big problem in cloud computing, as there are many factors such as priority, cost, quality of service and deadline which need to be taken care of before devising any scheduling strategy. Efficient job scheduling algorithm enables the optimal utilization of resources in cloud computing platform. Sometimes while scheduling, some virtual machines (VMs) get over-loaded and some remain under-loaded which produces adverse effect on the throughput of the system. The quest to balance the load during scheduling of cloudlets paves the path for the research in the load balancing mechanisms. 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. A Credits based task scheduling algorithm was rendered using modified K-means for clustering of jobs and VMs but it was observed that for providing optimized performance, this arrangement further needed some load balancing strategy to balance the load. Therefore, Honey Bee Foraging behaviour inspired Load balancing technique was roped in for load balancing. Work pertaining to the use of Honey Bee Foraging Load Balancing Algorithm coupled with credits based scheduling and modified K-means clustering technique is not available. Results indicate that the proposed scheduling algorithm has excelled existing priority-based scheduling strategy and it has been empirically proven with experimental/simulated results in this paper.*

***Keywords:** Honey Bee, Load Balancing, Cloud Computing, Honey Bee Foraging Behavior, Virtual Machine Scheduling, Scheduling Credits Based Algorithm with Modified K-means, Priority Based Scheduling, Modified K-means, Quality of Service in Cloud Computing, QoS, Swarm Intelligence.*

1. Introduction

Cloud Computing has revolutionized the concept of computing and has brought the new trend in the enterprise business by bringing in the idea of utility oriented IT services to users worldwide. Due to its global market and services, millions of jobs are submitted for execution at a time, which makes the scheduling in cloud an np hard problem. Virtualization has provided the much needed technological base to allow multiple virtual machines (VMs) on the top of a hardware computing resources and has brought in the idea of cost effective cloud computing platforms [1, 2]. Optimal Utilization of available computational resources has always been a challenge in cloud computing and to design and deploy an efficient task scheduling strategy is a major area of interest for many researchers [3, 4]. Datacenter broker handles and manages the crucial task of Mapping of Cloudlets to Virtual Machine and it also maintains and keeps the record of available Virtual Machines and ensures Quality of Service during execution of tasks [5, 6]. Good performance by VMs promises good quality of service but 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]. It was observed that despite the categorization of jobs and VMs, some machines were found underloaded while other were overloaded with tasks. Whenever a VM is overloaded with tasks, it needs to be shifted to VMs which are lying idle or are underloaded at that given time. This rearrangement should be done in a way that brings in good mix of priorities when no task needs to starve or wait for long to get processed. Load balancing aims to optimize the use of available resources, improves the makespan time and minimizes the latency. There are basically two types of load balancing techniques viz (1) Static and (2) Dynamic. Static algorithms work well when the tasks and VMs have lesser variation in terms of real time parameters but in cloud environment where load would be varying at various times, Dynamic load balancing algorithms are proven advantageous. Due to the changing workload dynamics, conventional load balancing algorithms have many drawbacks in cloud environment. To address these challenges, Swarm Intelligence algorithms (SI), such as ant colony optimization (ACO), and artificial bee colony (ABC), were rendered in recent decades [7]. They achieve a great progress in the dynamic situations of cloud computing. Therefore, many researchers have experimented with Swarm Intelligence algorithms such as foraging for food and used them to balance load in cloud environment.

2. Literature Survey

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. Some of the research work has been presented in the inference table given below:

TABLE 1. INFERENCE TABLE

Sr. No	**Author**	**Technique**	**Description**	**Future Scope**
1.	Lakra, A. V. et al. [5]	To aim at multiple objectives while keeping Quality of Service as a primary concern/criterion	In this algorithm, the jobs having higher priority are allotted with lower QoS and 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 using some good method for categorization of VMs & Tasks and considering more factors/parameters for QoS. Some optimized load
2.	Thomas, A. et al. [6]	An improved credit based scheduling algorithm considering user priority and task length as two credits	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 priority. There is a scope for balancing the load on VMS by deploying some efficient load balancing algorithm
3.	Selvarani , S. et al. [7]	A traditional cost based scheduling	Depending upon their processing capabilities, various tasks/cloudlets were grouped and allocated to suitable processing resources (VMs) to meet the minimum total task completion time and minimum cost	can be advanced to handle concurrent tasks, considering more complex scenarios with dynamic parameters and real time factors
4.	Moses, J et al. [8]	A shared resource monitoring strategy for understanding the usage of resources of each	Gathers resource usage information across various platforms while migrating the VMs which are resource-constrained.	Detailed profiling of VMs is necessary to steer scheduling strategies along with VPA (Virtual Platform Architecture) monitoring to

		Virtual Machine (VM) on each platform	Incorporated the concept of priority assignment based on the QoS, by taking into account various constraints such as execution time and cost of application.	monitor cache on VMs
5.	Ghanbari, S et al. [9]	A priority-based-job-scheduling technique (PJSC)	Multi-criteria-decision-making (MCDM) model considering multiple attributes, complexity and finish time. Based on Analytical Hierarchy Process (AHP)	Can be enhanced to achieve even lesser makespan time (finish time) by considering other QoS parameters for scheduling
6.	Xiao, J et al. [10]	A priority based strategy for discovering the optimal choices	found that priority based method has more advancements over FCFS technique and is far more beneficial too	More information is gathered on regular fashion of its usage this strategy can be advanced even further.
7.	Yang, L et al. [11]	A class based weighted fair scheduling (CBWFQ)	Existing fair queuing scheduling algorithm performs well in data applications but does not guarantee a fair service in real time application. This class based weighted fair scheduling (CBWFQ) technique improves the network performances, time delay and fairness.	This algorithm is also silent in deciding scheduling strategy where the job priority is same.
8	Wang, W.-J et al. [12]	Algorithm is based on adaptive scheduling and Quality of Service (Adaptive-Scheduling-with-QoS-Satisfaction) and named it AsQ	Devised for hybrid cloud environment for calculating the estimated execution time of the submitted jobs. 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	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
9.	Gupta, G et al. [13]	Deals with Preemptive scheduling of jobs.	It is an Earliest-Deadline-First, priority based scheduling method. Authors have coined waiting queue concept to process the preempted jobs.	This algorithm is silent on the use of any static or dynamic load balancing algorithm.
10.	Ibrahim, E et al. [18]	An enhanced task scheduling policy where the available VMs are allocated to the requesting tasks based upon their processing powers	For effective estimation and calculation of the execution cost/price, authors have used Amazon EC2 and Google pricing models.	It can be further improved by considering dynamic workflow scheduling and dependent tasks.

		and price of execution.		
11.	Kaur, S. et al. [19]	A Support Vector Machine algorithm using hybrid K-means technique	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.	It can be used in series with other categorizing or grouping techniques.
12.	L.D., D. B et al [20]	A dynamic load balancing technique based on Honey Bee foraging behavior	In this method priorities of the tasks have been considered along with load balancing.	It can be further improved by taking other QoS factors and more real time parameters.

From the above Table 1, it is evident that in the real-time scenario, scheduling strategies based on single criteria are far from providing an efficient solution; moreover, there is a strong need to balance the load in the dynamic environment of cloud where millions of users submits their millions of jobs worldwide [5]. Literature survey also indicates that the work pertaining to the use of multiple (four) criteria/credits for deciding priority with modified K means clustering technique coupled with Honey bee Foraging load balancing technique is not available; so, it was decided to consider four parameters for assigning priority credits to Tasks, Modified K-means for clustering of tasks & VMs and Honey Bee Foraging inspired Load Balancing technique to balance the load, to improve Quality of service (QoS) and for optimal utilization of resources. Paper structure is as follows, Section-3 starts with discussing the existing system and elaborates in detail the proposed work. Section-4 apprises the necessary arrangements and Experimental Setup and also discusses the Simulation Results. Section-5 concludes the proposed work and suggests the future scope of the said research work.
s

3. Proposed Work

The existing credits based scheduling system has task length and task priority as two credits [6]. In this arrangement, while sorting the tasks, there may arise many such cases in which two or more tasks may have the same priority [2, 9]. Secondly, to allocate the tasks to appropriate VMs without compromising throughput of the system is a major conern/issue. It was observed that in real time systems, cost and deadline are very important factors which affect the priority of a task extensively and without which we cannot handle real time jobs in time constraint.

Therefore, a new scheduling arrangement was proposed which uses four parameters namely Task Length, Task Priority, Deadline of the Task and the Cost, to decide effective priority of a task. These four parameters are combined to obtain the final credits for the task; the priority thus obtained would not only be closer to real time scenario but would also reduce the chances of same priority occurrence between the two tasks.

Total Credits of the task can be calculated as:

$$\mathrm{T\_Cr_n} = \mathrm{C\_Len_n} * \mathrm{C\_Prio_n} * \mathrm{C\_Dline_n} * \mathrm{C\_Cost_n}$$

Once the credits are assigned to a task, 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 Total Computational Cost are optimized.

Categorization of Tasks & VMs:

1. After assigning total credits to the tasks, these tasks are sorted in descending order of the assigned credits.
2. For the categorization of tasks and VMs modified K-means technique is incorporated which will reduce the prediction error during mapping of the tasks to VMs [19].
3. Task clusters are made on the basis of four parameters i.e. Task_Length, Task_Priority, Deadline and Cost.
4. Clustering at virtual machine side is done on the basis of Bandwidth, RAM, MIPS and Size.

5. After applying Modified K-means clustering technique, the descending sort operation within the clustered groups is performed, which further creates three groups (with priority levels high, medium and low) in VMs and Cloudlets/tasks.
6. Thus, obtained cloudlet/task clusters are allocated/assigned to virtual machine clusters for ultimate execution of the job.

This arrangement shows very good outcomes and accuracy levels except some pointed peaks.

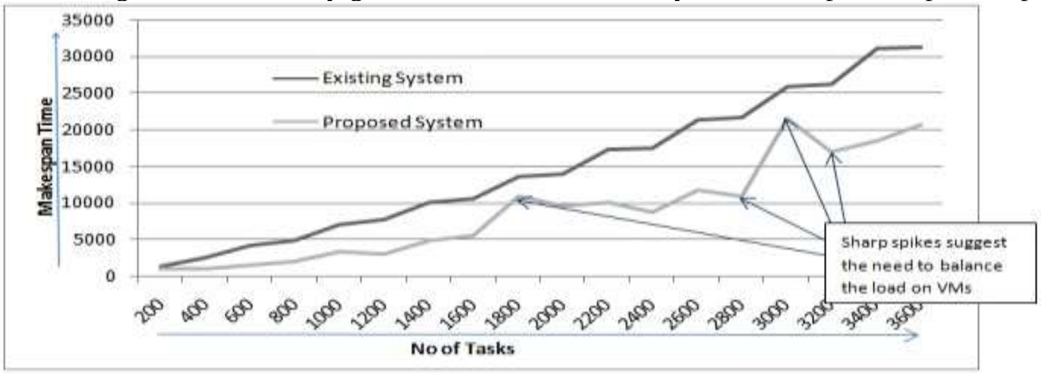


Fig. 1 .Line Graph Representation to show the need to balance the load in the system

As shown in the Fig 1, some sharp spikes were observed while plotting the graphs for proposed system, which suggests the need to balance the load by implementing some superlative technique of load balancing. The proposed system can perform better if some optimized load balancing strategy is applied along with the current arrangement. Therefore, to balance the load and improve the overall stability and efficiency of the system it was decided to deploy Honey Bee Foraging Load balancing algorithm along with the presented arrangement.

3.1 Load Balancing-Honey Bee Foraging Behavior

Load balancing is done at the virtual machine level i.e. intra datacenter level and in this algorithm honey bee foraging behavior is modeled. This algorithm aims to balance the load on the VMs by keeping the priorities of the task in consideration and waiting time of the tasks remains minimal. Tasks are represented as bees and movement of tasks, from one location/VM to another to get executed in minimal time, is regarded as the foraging behavior of Honey Bees. There can be three types of movements as under:

1. Scout Bees: Hunt for the sources of food and when they find the suitable food source they return back to bee hive and intimate others by waggle/tremble/vibration dance. This activity tells the quality and distance of food to other bees.
2. Onlooker bees: keep track of how much food is left and is there at what location.
3. Employee Bees: These are the food supplies and keep neighborhood of supply in their memory.

We can understand the implementation of this technique in the following procedure.

3.2 Procedure: Honey Bee Foraging Inspired Load Balancing Method

1. Analyze the Load using Honey Bee Algorithm and check Load and Capacity on VM with

$$\mathrm{Vmload} = (\mathrm{N} * \mathrm{Task}_{\mathrm{length}})/\mathrm{Vm}_{\mathrm{mips}}$$

Here N denotes the number of tasks, $\mathrm{Task}_{\mathrm{length}}$ denotes the length &$\mathrm{VM}_{\mathrm{Mips}}$ is processing speeds of VMs measured in Million Instructions per second.

$$\mathrm{VM}_{\mathrm{Capacity}} = \mathrm{PE}_{\mathrm{Number}} * \mathrm{PE}_{\mathrm{Mips}} + \mathrm{VM}_{\mathrm{BW}}$$

Where processing elements is denoted with $\mathrm{PE}_{\mathrm{Number}}$ and $\mathrm{PE}_{\mathrm{Mips}}$ is Million Instructions per second speed of processing element &$\mathrm{VM}_{\mathrm{BW}}$ is the allocated bandwidth/network speed associated with VM.

2. Calculate the processing time (PT) of VM

$$\mathrm{PT\_Vm}_{\mathrm{i}} = \mathrm{VmLoad}/\mathrm{Vm}_{\mathrm{capacity}}$$

3. Calculate Standard deviation (SD) of the load.

$$SD = \sqrt{1/M \sum_{i=0}^{M} (PT\_Vm_i - \overline{PT\_Vm})^2}$$

Where Processing Time of ith VM is denoted by PT_Vm_i , average Processing Time of all VMs is denoted by $\overline{PT\_Vm}$ & Virtual Machine in VM set is denoted my M.

```
4. Check the need for load balancing by setting threshold value between 0-1
      If   (SD <= Threshold)
          VM's are Balanced or Underloaded, No load balancing is required.
          Exit.
      If  (SD> Average processing Time (Threshold))
           VM's are Overloaded and Load balancing is required.
```

5. Categorize Underloaded and Overloaded VMs in groups.
   a) Tasks in Overloaded VM's are considered as Bees and Underloaded VM's are considered as food sources.
   b) Calculate the supply value of Underloaded VM's & demand value for Overloaded VM's.
6. On the basis of priority Task Transfer takes place from overloaded VMs to Underloaded VMs.
   a) An appropriate Underloaded VM is located for each task in Overloaded VM.
   b) Demand and Supply in VM groups can be calculated as:

$Supply_{Vm}$= Capacity – Load
$Demand_{Vm}$ = Load – Capacity

7. Sort operation is applied on over-loaded & under-loaded VM sets
8. On basis of priority, Sorting of the tasks is done in over-loaded VMs.
9. An appropriate under-loaded VM is searched for each task in over-loaded VM.
10. Update over-loaded & under-loaded VM sets.
11. Repeat steps 6 to 10 till all the VMs are balanced.

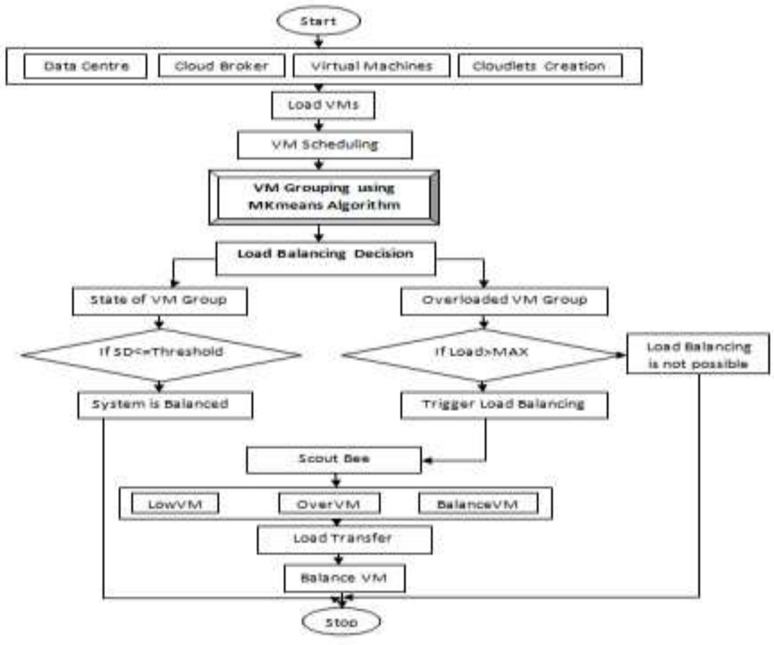


Fig. 2 .Flow Chart Representation of the Proposed System (©self) // name is omitted due to double blinds policy

3.3 An optimized scheduling algorithm for improving efficiency of cloud computing: The course of actions to implement the proposed strategy has been put in a succession (discussed in Procedures 1,2,3,4 & 5) as under.

```
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 credits on the basis of combination of task length, priority, deadline and the execution cost assigned to the task)
2.  For each n number of Tasks in T.    // To assigns the priority on the basis of combination of credits.
        $\mathrm{Total\_Credit_n = Credit\_Length_n * Credit\_Priority_n * Credit\_Deadline_n * Credit\_Cost_n}$
     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. For all $\mathrm{Vm_i}$          //all Virtual Machines in the Set
        $\mathrm{Load_i = \frac{N * Cloudlet\_length}{VM\_MIPS}}$
        // Find the load on the virtual machine, where N is the total number of tasks assigned to a VM, Cloudlet_length is the length of single task and VM_MIPS is the MIPS rate of that VM.
12. Calculate capacity of a particular VM
        $\mathrm{Capacity_i = PE_{Num} * PE_{MIPS} + Vm_{BW}}$
13. Calculate Processing Time
        $\mathrm{PT\_Vm_i = \frac{Load}{Capacity}}$
14. Calculate Standard deviation (SD) of load
        $\mathrm{SD} = \sqrt{\frac{1}{\mathrm{N}}\sum_{\mathrm{i=0}}^{\mathrm{N}}(\mathrm{PT\_Vm_i} - \overline{\mathrm{PT\_Vm}})^2}$
        //Where $\mathrm{PT\_Vm_i}$ is Processing Time and $\overline{\mathrm{PT\_Vm}}$ is the average Processing Time of the virtual machine
```

```
15. Decide the state of VM groups based on load (whether system is overloaded or under loaded)
    If      SD<=threshold.
            System is balanced
    Exit
16. Load balancing Decision
    If      Ld>Max_capacity
            Load Balancing is not possible
    Else
            Trigger: Process Load Balancing
17. Process: Load Balancing   // Find the supply of under loaded VMs and demand of overloaded VMs
            Supply of each VM in LVM is
                $\text{Supply of LVMj} = \text{Maximum Capacity} - \frac{\text{Load}}{\text{Capacity}}$
            Demand of each VM in OVM is
                $\text{Demand of OVMj} = \frac{\text{Load}}{\text{Capacity}} - \text{Maximum Capacity}$
18. Sort the overloaded VM sets based on the Load  in Descending Order
19. Sort the under loaded VM sets based on the Load in Ascending Order
20. Sort the tasks in overloaded VMs based on priority
21. For each task
            Each overloaded VM (find a suitable under loaded VM)
            Update the overloaded and under loaded VM sets
            go to step 11
     End For
22. Mapping of the task with the virtual machines using function
    Task.set VMid (m.id)
    sendNow(virtual machine id, Cloudsim Submission Tag,  Task);
23. Analyse the performance parameters Makespan Time (Finish Time), Processing Time and Total Computational Cost.
```

Load Balancing using Honey Bee Foraging Behaviour coupled with categorised credits based prioitised taks & VMs improves the performance of processor, saves memory allocated during tasks and optimizes time on network operations by dropping datacenter’s workload.

4. Experimental Setup and Simulation

4.1 Cloud Simulator:CloudSim 3.0.3

The simulation of the proposed arrangement/algorithm has been done in CloudSim 3.0.3 cloud simulation tool, using Java. It provides a conducive environment for cloud managing computing applications and helps in creating datacenters, virtual machines and other facilities which can be rapidly generated as per the need, easily. Selecting or Choosing a simulator, majorly, depends upon the nature and kind of research. Many researchers recommend CloudSim as a general purpose simulator due to its features, facilities and ease of use for simulation [21].

4.2 The Configuration of Proposed Simulated Environment

Basic Configuration for Proposed Simulated Environment is as mentioned in

TABLE 1. BASIC CONFIGURATIONS OF DATACENTERS

Number of Datacenters	5
Number of Hosts under each Datacenter	2
Total Hosts	10
Number of cloudlets/tasks	100-2900
Number of Brokers	1

TABLE 2. BASIC CONFIGURATIONS OF HOST

Number of Virtual Machines (VMs)	80
RAM (MB) value initialized in simulator	20480
MIPS (Millions of instructions per second)	5000
Storage (MB) value initialized in simulator	1,048,576
Bandwidth (MB/sec) value initialized in simulator	76800

TABLE 3. VIRTUAL MACHINES AND THEIR BASIC CONFIGURATION

RAM (MB) value	256
MIPS	(MIPS=250,count=0;count<15; MIPS+5,count++)
Bandwidth (MB/second) value	(bandwidth=1000,count<15;bandwidth+500,count ++)
Number of Cores	1
Size (MB) value	(size=10000,count=0;size<15;size=size+2343,cou nt++)

4.3 Equations

Makespan Time: Makespan is overall task completion time for the jobs assigned for execution [20]. Completion time of task T_p onVM_q as $CT_{pq.}$ .Therefore, the makespan (MS) can be represented as:

$$MS = \max\{CT_{pq} | p \in T, p = 1,2 \ldots n \text{ and } q \in VM. j = 1,2..m\} \qquad \text{(eq. I)}$$

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).

Processing time = CloudletLength / vmMips*vmNumberOfPes (eq. II)

Memory Cost: The memory utilization cost of a task can be calculated as CostPerMem * vm.getRam (eq. III)

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

Total Processing Cost = (CloudletLength / vmMips*vmNumberOfPes) * (CostPerMem * vm.getRam) (eq. IV)

In simulation we can use any currency value as Units for Processing Cost and Total Computational Cost.

4.4 Experimental Observations in Simulated Environment

Experimental Results of the existing Credit Based System having Length and Priority as two credits are recorded in Table 4:

TABLE 4. EXPERIMENTAL RESULTS OF THE EXISTING SYSTEM

Credit Based System with Length and Priority as Credit Parameters					
Sr. No	**Tasks**	**Makespan Time** (ms)	**Processing Time**(ms)	**Processing Cost** (currency units)	**Total Computational Cost** (currency units)
1	100	900.87	1738.30	20742.84	36057296.80
2	300	1902.37	8893.00	61385.04	545897082.37
3	500	3422.48	20954.57	100855.74	2113388657.45
4	700	4657.32	37611.77	139154.94	5233864063.75
5	900	5990.41	59442.53	176282.64	10478685898.92
6	1100	7481.82	85436.72	212238.84	18132990746.31
7	1300	8772.61	117096.99	247023.54	28925712576.10
8	1500	10556.19	152797.73	280636.74	42880656680.18
9	1700	11807.71	194043.94	313078.44	60750974598.88

10	1900	13915.26	239090.43	344348.64	82330462928.07
11	2100	15131.89	290166.15	374447.34	108651942675.25
12	2300	17600.50	344919.18	403374.54	139131615967.62
13	2500	18788.49	405580.85	431130.24	174858169316.44
14	2700	21661.69	469679.62	457714.44	214979146491.28
15	2900	22829.98	540170.81	483127.14	260971176226.24

Experimental Results of the Proposed Credit Based System with Modified K-means Clustering Algorithm coupled with Honey Bee Foraging Load Balancing Algorithm are mentioned in **Table 5:**

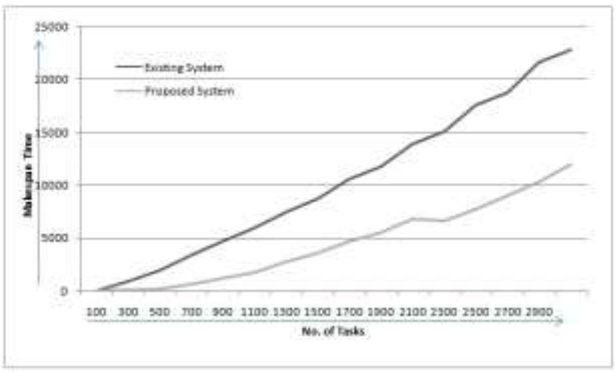
TABLE 5. EXPERIMENTAL RESULTS OF THE PROPOSED SYSTEM

Credit Based System (Task Length, Task Priority, Deadline & Cost) **with Modified K-means Clustering technique coupled with Honey Bee Foraging Load Balancing Algorithm**					
Sr. No.	**Tasks**	**Makespan Time** (ms)	**Processing Time**(ms)	**Processing Cost**	**Total Computational Cost**
1	100	57.29	1478.95	7961.93	11775303.76
2	300	211.19	6765.16	27347.76	185011953.91
3	500	640.98	15185.53	44980.58	683053848.73
4	700	1191.05	27534.06	61885.35	1703954812.34
5	900	1765.01	42164.44	81852.17	3451251235.62
6	1100	2795.45	63683.70	97626.43	6217212358.72
7	1300	3558.65	80664.45	138589.31	11179229744.62
8	1500	4730.92	102040.31	163989.09	16733496803.94
9	1700	5536.00	128972.08	187010.17	24119090130.63
10	1900	6822.96	159165.05	209598.43	33360745283.21
11	2100	6647.61	212129.56	201530.62	42750603325.20
12	2300	7738.63	257866.85	216971.83	55949841823.81
13	2500	9059.55	312632.45	231402.37	72343890753.57
14	2700	10308.81	358260.80	250676.92	89807714661.06
15	2900	11956.83	410653.12	268361.11	110203326536.39

4.5 Results & Discussions

From the above findings it has been empirically proven that the proposed system has excelled in performance and presented reduced Makespan Time (eq. I), Processing Time (eq. II) and Total Computational Cost (eq. IV) than the existing system while increasing the throughput of the cloud computing system which can be analyzed in the Fig 3, Fig 4, Fig 5 & Fig 6.

After comparing Graphs in the Fig 1 and Fig 3 it is evident that the spikes occurred due to the imbalance in the system have been smoothened out by deploying an optimized Honey Bee foraging behavior inspired Load balancing algorithm and shows the steady behavior of the proposed algorithm.



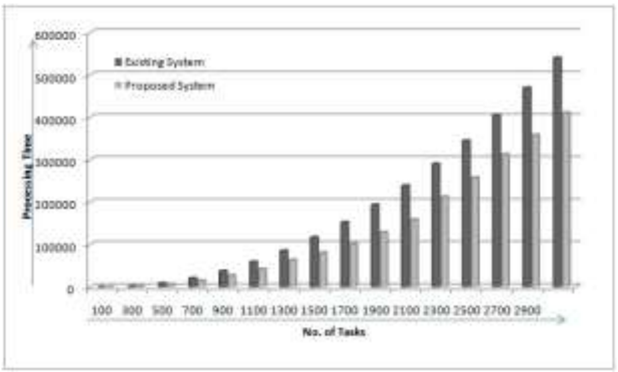
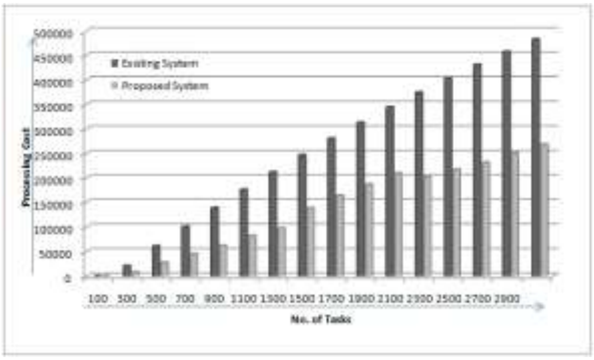


Fig. 3 Line Chart Graphical Representation of the Experimental Results for Makespan Time in the Existing System vs. Proposed System

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



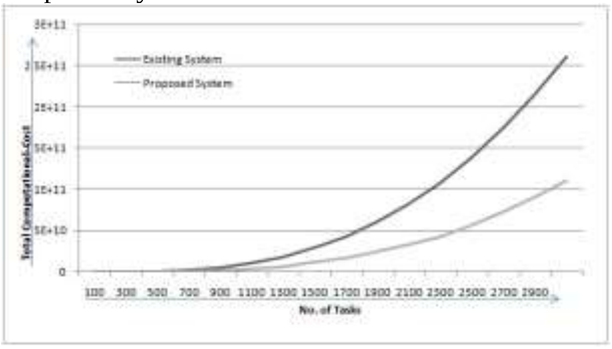


Fig. 5 3-D Cluster-Column Graphical Representation of the Experimental Results for Processing Cost in the Existing System vs. Proposed System

Fig. 6 Line Chart Graphical Representation of the Experimental Results for Total Computational Cost in the Existing System vs. Proposed System

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

a. **Processor**: Experimental results indicate that the proposed approach provides 26.78% better results in terms of 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**: Simulation Results prove that the proposed system has gained huge leap in the performance for Makespan Time by providing 63.60% better results than the existing system. It has optimized memory utilization by relinquishing the occupied memory on cloud resources, which is now free and available to other jobs/customers for use.

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

d. **Quality of Service:** By using, Honey Bee Foraging inspired superlative load balancing technique, to balance the load on the VMs; the quality of service (QoS) has been improved immensely which is vivid from the comparison of Fig 1 and Fig 3. The smooth line in the Fig 6 shows the consistency and reliability of the proposed algorithm with reduced total Computational Cost.

5. Conclusion

In this paper, an optimized approach to improve the quality of service and efficiency of cloud computing has been proposed. For optimal allocation of the tasks, a scheduling strategy was rendered by categorizing VMs & credits based prioritized cloudlets. This arrangement had shown very good results but it had also shown some sharp spikes while plotting the graph for the obtained readings. These peaks were pointing towards an unbalanced load on some VMs and advocated that some superlative load balancing technique was required to balance the load. Therefore an optimized Honey Bee foraging inspired load balancing technique was deployed at VM level i.e. intra-datacenter level. This arrangement has shown excellent results when compared to the existing system and has shown the remarkable improvement in terms of balancing of load on VMs. In future, it can be further improved by using more superlative load balancing techniques and more real time factors can be included for deciding priority of the tasks.

References

[1] Bourguiba, M., Haddadou, K., El Korbi, I., & Pujolle, G. (2014). Improving Network I/O Virtualization for Cloud Computing. IEEE Transactions on Parallel and Distributed Systems, 25(3), 673–681. doi:10.1109/tpds.2013.29

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

[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] Journal, I., Technological, F., Lamba, A., Singh, S., Singh, B., Dutta, N., … Islands, C. (2017). ANALYZING AND FIXING CYBER SECURITY THREATS FOR SUPPLY CHAIN MANAGEMENT. 4(5), 5678–5681.

[5] 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

[6] Lakra, A. V., & Yadav, D. K. (2015). Multi-Objective Tasks Scheduling Algorithm for Cloud Computing Throughput Optimization. Procedia Computer Science, 48, 107–113. doi:10.1016/j.procs.2015.04.158

[7] Thomas, A., Krishnalal, G., & Jagathy Raj, V. P. (2015). Credit Based Scheduling Algorithm in Cloud Computing Environment. Procedia Computer Science, 46, 913–920. doi:10.1016/j.procs.2015.02.162 .

[8] 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

[9] 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

[10] Ghanbari, S., & Othman, M. (2012). A Priority Based Job Scheduling Algorithm in Cloud Computing. Procedia Engineering, 50, 778–785. doi:10.1016/j.proeng.2012.10.086

[11] 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

[12] Yang, L., Pan, C., Zhang, E., & Liu, H. (2012). A New Class of Priority-based Weighted Fair Scheduling Algorithm. Physics Procedia, 33, 942–948. doi:10.1016/j.phpro.2012.05.158

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

[14] Journal, I., Technological, F., Lamba, A., Singh, S., Singh, B., Dutta, N., … Islands, C. (2017). ANALYZING AND FIXING CYBER SECURITY THREATS FOR SUPPLY CHAIN MANAGEMENT. 4(5), 5678–5681.

[15] 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

[16] 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

[17] 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

[18] Shih, H.-Y., Huang, J.-J., & Leu, J.-S. (2012). Dynamic slot-based task scheduling based on node workload in a MapReduce computation model. Anti-Counterfeiting, Security, and Identification. doi:10.1109/icasid.2012.6325318

[19] 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

[20] 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

[21] 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

[22] L.D., D. B., & Venkata Krishna, P. (2013). Honey bee behavior inspired load balancing of tasks in cloud computing environments. Applied Soft Computing, 13(5), 2292–2303. doi:10.1016/j.asoc.2013.01.025

[23] 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