I.J. Information Technology and Computer Science, **2016, 4, 1-10**
Published Online April 2016 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2016.04.01

Real Time Efficient Scheduling Algorithm for Load Balancing in Fog Computing Environment

Manisha Verma
Department of Computer Science and Technology, UPTU University, Lucknow, U.P., India
E-mail: soni2185@gmail.com

Neelam Bhardwaj
Department of Computer Science and Technology, Hindustan Institute of Tech. & Mgmt., Agra, U.P., India
E-mail: 8.bhardwaj@gmail.com

Arun Kumar Yadav
Department of Computer Science and Technology, ITM University Gwalior, M.P., India
E-mail: arun26977@gmail.com

Abstract—Cloud computing is the new era technology, which is entirely dependent on the internet to maintain large applications, where data is shared over one platform to provide better services to clients belonging to a different organization. It ensures maximum utilization of computational resources by making availability of data, software and infrastructure with lower cost in a secure, reliable and flexible manner. Though cloud computing offers many advantages, but it suffers from certain limitation too, that during load balancing of data in cloud data centers the internet faces problems of network congestion, less bandwidth utilization, fault tolerance and security etc. To get rid out of this issue new computing model called Fog Computing is introduced which easily transfer sensitive data without delaying to distributed devices. Fog is similar to the cloud only difference lies in the fact that it is located more close to end users to process and give response to the client in less time. Secondly, it is beneficial to the real time streaming applications, sensor networks, Internet of things which need high speed and reliable internet connectivity. Our proposed architecture introduced a new scheduling policy for load balancing in Fog Computing environment, which complete real tasks within deadline, increase throughput and network utilization, maintaining data consistency with less complexity to meet the present day demand of end users.

Index Terms—Cloud Computing, Fog Computing, Load balancing, Reliability, Throughput.

I. INTRODUCTION

In today's world Cloud computing [1, 2] has made the utilization of resources, commercial applications and sharing of database through internet faster in academia and industry. Cloud Computing is a web of millions of computers which are clustered together in a distributed and complex manner to a central remote server so as to maintain data and applications. It is economical to end users as they have to pay according to the service or resource usage in utility computing basis. It is a blend of technologies, implementing the concept of virtualization, resource allocation, bandwidth utilization, distributed computing, web computing, load balancing, networking and software to provide dynamic scalability for different categories of data and applications. It provides elasticity, fault tolerance, high availability, reduced overhead in multiple technologies. Cloud computing remains independent of the software, operating system and web browser which clients are using on their end, the only requirement is the terminal connected to the internet. It is gaining boom and popularity as a small organization and IT industry, without investing huge amount of money in hardware or software can acquire all the facilities to survive in this competitive business world. All the maintenance cost, storage capacity, computing power, updating of software and other web services are managed by cloud service provider. In Cloud Computing, all the processes executes in parallel manner on Virtual machines thereby making execution faster on available resources. It delivers resources as a service, according to different service models depending on user needs, like Infrastructure as a service known as IAAS , Platform as a service known as PAAS, Software as a service known as SAAS. They all are stacked on each other and in a combine way it is known as "XAAS" anything as a service. It is basically an SOA (service oriented architecture) consisting of four deployment models 1).Public Cloud 2).Private Cloud 3).Hybrid Cloud 4).Community Cloud. Public cloud has made resources available for general public use and it is hosted on service provider site, while on the other hand private cloud supports specific customer that requires high security and it is hosted mainly on Enterprise site. Hybrid Cloud is a combination of public, private and community cloud that are bound together, offering benefits of internal and external hosting. Community Cloud is shared by organizations for similar type of motives.

No doubt cloud computing offers many benefits for enormous applications, though it has certain limitation too. Today, high bandwidth has become essential because of the increase in end users as a result, it led to high latency inhibiting other end users from accessing data faster thus affecting the cost of usage. Cloud Computing is also facing security issues as data has to travel much farther from cloud to end users, which increases the probability of data loss. To overcome the limitations of the cloud computing, a new technology has been introduced known as Fog computing. Here below figure 1 shows the architecture of the cloud and its applications.

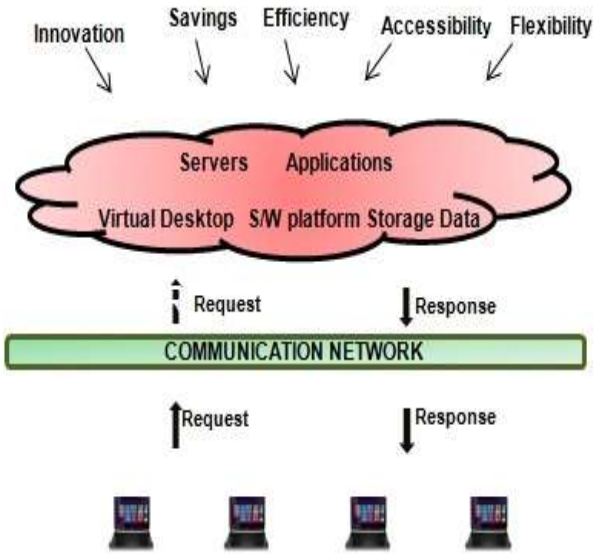

Fig.1. Cloud Computing Features.

Fog Computing [3] is a term proposed by CISCO offering enormous facilities for storage, computation and networking between sensors or end users, along with extending cloud computing data centers to the edge of an enterprise network. Fog computing comes into picture as number of users and connected devices are tremendously increasing at a very rapid rate including Internet of things [4]. IOTs are basically a network of wireless things like Smartphone's, Sensors, Computers, Wearable device, Smart cities, Smart hospitals including everyday devices from medical devices to home appliances. Today, users demand for an easier life and need to access heavy data like facial and voice recognition has lead to improve customer service for better performance.

Fog Computing will inculpates tremendous benefits in areas of agriculture, business process optimization, deep sea exploration, health industry, product prices, real time intelligence, smart dust swarms, smart tattoos, transforming the renewable energy industry, weather forecasting and many more. Fog computing will make it possible as it as it is a highly virtualized technology [5] based on a real time basis and act as a intermediate layer placed between end users and the cloud data centers hosted within the internet. Its main features are low latency, location awareness, distributed geographical distribution and support for mobility and real time interaction. To make fog computing more effective for optimal utilization of bandwidth, and to reduce costs, we have to equally transfer load from clients to all servers, such that no process has to wait for a long time, so here comes load balancing. Although many load balancing algorithms exist with some pros and cons it has become a major challenge in fog computing, as load has to be balanced first between users and fog layer i.e. intermediate layer and then between fog and cloud layer. It attempts to speed up the execution of applications on available resources with proper use of storage to give quick response time to submit user request. In load balancing, we should assure that processing unit, i.e. virtual machines, while running the tasks should not be overloaded or sit in idle condition as system maximum throughput is must.

Load Balancing

Load Balancing [6] is a technique which divides the workload across multiple computing resources such as computers, hard drives and network. In this fair allocation of resources of client request tried to achieve in the best way to ensure proper utilization of resource consumption. It also tries to fix the problem that all the processor in the systems and every node in the network must share equal amount of workload which is assigned to them. It can make feasible through proper hardware or software which can be a multilayer or a domain name system process. The key factors which make efficient load balancing are backup plan in case the system fails a bit, ensuring system stability, throughput, response time, minimum latency, minimum network delay, execution time, low overhead , low delay and scalability.

In a Cloud computing environment different load balancing scheduling exists among which first, is the Batch mode heuristic scheduling algorithm where jobs are queued in a set and collected as batches as they arrive in the system after which they get started after a fixed time period. Its examples are First Come First Serve (FCFS), Min-Max algorithm, Min-Min algorithm and Round Robin (RR) algorithm [7, 8, 9]. Second, one is On-line mode heuristic scheduling here jobs are scheduled individually as they arrive in the system. These algorithms are more feasible in a cloud environment as the systems may have different platforms and execution speed its example is Most fit task scheduling algorithm.

Load balancing algorithm is implemented by first estimating the total load on C.P.U, Memory and Network together. Second, by analyzing how the nodes are interacting with each other, their suitability as the system by estimating load and comparing nature. Third, Load balancer should not be a single point of failure.

It has become a necessity as the popularity of the Internet is increasing because of more use of social networking websites, large databases and E-Commerce as it is leading many businesses to carry out on a daily basis so it demands high bandwidth. It provides single Internet service from multiple servers known as a server farm.

Our proposed algorithm will solve the issues related to latency, bandwidth, deadline and availability of resources in Fog environment. Here in figure 2 we are proposing an

efficient architecture and algorithm for load balancing in Cloud computing environment.

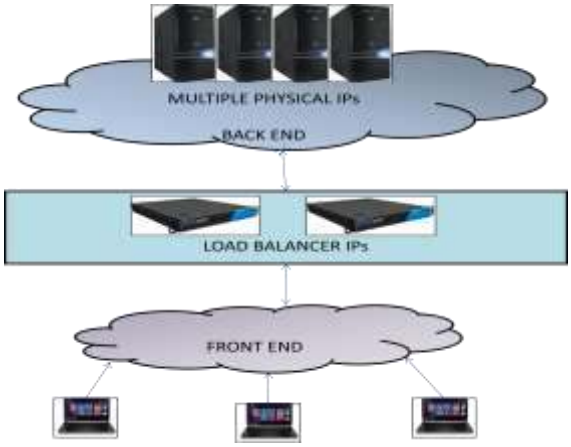

Fig.2. Load balancing in Cloud Computing Environment.

In section II we discuss related work concerned about on our topic. In section III a design model for our Load Balancing algorithm has been presented. We propose our algorithm in section IV followed by experimental result and the comparison in section V. Finally, we give the conclusion in section VI respectively.

II. Related Work

In this section study and review of various Load Balancing algorithms of various authors has been done on the basis of various parameters like bandwidth, cost, deadline, execution time, make span, priority, reliability, scalability, time, task length, trust, throughput. Many efficient load balancing algorithms have been proposed in a cloud environment, but in our proposed work we are implementing algorithm in fog computing environment. The fog computing environment is nothing but an intermediate layer between cloud layer and client layer, to make the work more feasible and efficient with limited resources. It provides access to resources easily with less bandwidth, time and cost as it is closer to end users. Fog has been similar to Cloud in working, it just have been introduced to meet the needs of users, which are increasing at a tremendous rate so that no havoc comes in network traffic.

Here below discussing some of load balancing techniques in a cloud computing environment.

Maximize resource utilization in a cloud computing environment can be achieved for good services by considering both task priority and its length [10]. In this high priority tasks are not given much importance because the tasks are taken from the queue from both first and last indexing to get a more stable system. To schedule tasks total credit system is funded from a combination of both credit_length calculate from task length and credit_priority calculate from task priority. Finally, the tasks having high credit will be executed first. This algorithm has certain shortcomings that if the total credits of some tasks become equal, then FCFS have to be used along with it does not guarantee tasks to get completed before or to its deadline.

In this analog behavior of honey bee is used to implement efficient load balancing algorithm, in a cloud computing environment, taking priority as the main QoS factor, such that no process has to wait for a long time in a queue resulting in maximum throughput and less execution time [11]. In HBB-LB algorithm, there are two type of bees, one is scout bees who searches the food and after finding it out it inform other bees about its quality, quantity and its distance from the beehive, by signaling it through the waggle/tremble/vibration dance, the higher its intensity the more food is available. The other is forager bees which follow the short path used by scout bees to reach the food location, after depleting the food source they inform other bees about how much is still left and thus become scout bee and this process goes on. In the same way tasks are scheduled, here tasks and honey bees represent virtual machines and food sources respectively. VM's here are categorized into three parts, balanced overloaded, high overloaded and low over loaded. The tasks are removed from overloaded VM's and they act as a honey bee which then are submitted to under loaded VM's depending upon how many high priority tasks are executing there. In this only that under loaded VM is first chosen which along with being low overloaded have less number of priority tasks executing on them. After proper assignment of tasks on VM, information is globally updated so that remaining tasks can get the required under load VM. Its enormous benefit lies in the fact that it enhance through proper resource utilization, giving a maximum throughput along with maintaining other QOS factors which are based on task priority. Its limitation is that low priority tasks have to remain idle or wait for a long time in the queue, thus neglecting them thereby unbalancing the balancing workload.

A dynamic and optimized based algorithm approach is used for load balancing [12]. In the centralized load balancing algorithm, the central node takes the decision of distributing the workload with fewer messages, but its disadvantage is that if a central node fails the whole working of the system will collapse thereby degrading the system performance. So here, for achieving better performance getting maximum throughput and optimization can be considered as one of the solution.

It can be done in two ways; first it uses the complete method where valid values are assigned to all variables to find the result or if in case one assigned value get wrong then it will not be considered as a solution.

Second is the incomplete solution where probability is the key factor. The input parameters which give more correct answers are considered as a solution. Its characteristics lie in its simplicity, effectiveness, and faster speed for solving problems. This approach is named as Stochastic Hill Climbing known best for solving the optimization problem. It has a loop which generates random values arranged in increasing order which is analogous to moving the uphill upwards. The loop stops generating values when a unique upper value

or analogously a higher peak is found in uphill which no other uphill is having in its nearby neighborhood. The probability of the optimized solution depends on the steepness of the uphill move. Here, continuously mapping is set to be valid if new values getting mapped to old ones in a set by doing a bit change in old ones fall in some predefined criteria. The best value obtained from the set is used for next assignment and it goes till an optimized solution is found or stopping criteria are met. The key components consist of candidate generator which map solution of one candidate sets to into another based on successor values and next is evaluation criteria which find out the best solution and give ranks to the solution so that it can further improve results to get the best solution. In future, to get good results soft computing approaches can be used in it.

Resource utilization can be done efficiently to improve the throughput, thereby reducing the cost of an application running in a SAAS environment without violating service level agreements [13]. In this algorithm tasks are bound to VM's in such a way that they get executed faster. Firstly, tasks are assigned priorities such that High QoS are given low QoS value and low QoS are given high value, thereby making task with lower value a high priority and a vice versa. Secondly, VM's are assigned QoS values, such that VM's having high MIPS are assigned a high QoS value and low MIPS as low QoS value. Now on the basis of minimum size and minimum QoS value task is selected using function based on sorting. The tasks are sorted in descending order from high MIPS to low MIPS and after this they are assigned to a list of VM's which are also sorted in such a way that the first task from task list is assigned to the first VM in VM list, the second task to second VM in the list and in the same way process of assignment for further tasks goes on to other VM's. Once all tasks have been assigned till last VM in VM list, then the upcoming next task is assigned to first VM and this process goes on. In this algorithm minimal QoS parameters like execution time have been considered, so in future other QoS factors can be added.

In a Cloud computing environment, resources can be made fast available virtually irrespective of the time and location along with maintaining the trust of end users [14]. Trust plays a key role in maintaining integrity of data during data transfer through one location to another in a cloud environment. So here this trust is maintained by using a combination of two approaches BAT algorithm and heuristic search. In BAT algorithm, analog behavior of BATS is used as Bats uses sonar waves in their path to find their prey and avoid obstacles. Sound waves after getting emitted get transformed into another frequency by reflecting back from the obstacles. The time delay from emission to reflection is used for navigation. Secondly, in heuristic search, a perfect state of harmony is found by optimizing meta_heuristic approach in music. The advantage of this algorithm is that it efficiently reduces time to complete tasks.

In a Cloud computing environment, on the basis of priority and admission control based service scheduling policy a good optimization along with maximum throughput can be achieved [15].In this user requests are served in such a way that have to spend less time in a queue thereby fully utilizing available resources. The users which pay higher for cloud services get higher precedence as compare to others. In future for enhancing further performance other features can be included such as of security and hiring resources from other cloud.

In this a framework for global server load balancing has been proposed for managing enormous data by replacing expensive physical network load balancers by virtualized network load balancers [16]. The need of data varies as per customer requirement and network service providers deploy a load balancer dynamically in data centers. It consists of two levels of balancing, first load acts as a master and other acts as a slave which further comprises of load balancer selector and network load balancer. The network load balancer ensures high availability of web services respectively. In this mostly software-type load balancer is preferred over hardware-type load balancer as its cost puts high burden on enterprise users. Its advantage lies in the fact that web connection limitation in single network has been resolved, additionally, users can update algorithm by adding or decreasing the number of load balancers. In future performance can be further improved for large users prevailing in internet by implementing it in hybrid cloud environment.

Different scheduling algorithms with different QoS parameters for different environment have been proposed [17]. Scheduling is done to get the enormous profit and to enhance the efficiency of work load. So for the same different types of scheduling algorithms exist like FCFS algorithm, Round Robin algorithm, Min-Min algorithm and Max-Min algorithm, but the one which proves to be most efficient is a heuristic technique. It comprises of three stages for scheduling in a cloud computing environment, First one, is to find out the resource, then selecting a best target resource and finally submitting task to the target resource.

A different scheduling algorithm based on deadline with the perspective like cost, delay, execution time, response time, resource utilization time and task selection over different tools and environment have been compared[18]. Scheduling is done to minimize response time and fully utilizing the resources. On the basis of deadline different types of scheduling algorithms exist like Priority and admission control based algorithm, Schedule –as–soon-as-possible algorithm, Preemptive scheduling of Online real time service with task migration, Sporadic task approach with deadline constraints, Level based scheduling algorithm, TPD scheduling algorithm. These proposed algorithms, are trying to do cost optimization efficiently.

III. Proposed Architecture

In a cloud computing environment, the load balancing plays a key role for efficient resource utilization, bandwidth and to get good response time. Load balancing can be categorized [19] on the basis of the state of cloud

environment like static and dynamic scheduling, on the type of load balancer like hardware and software based, on the basis of who initiated the process like sender, receiver and symmetric, on the basis of policies like information , resource ,selection, location and transfer. In our proposed architecture, the load balancing algorithm has been proposed in fog computing environment. In fog computing environment, the fog servers are situated near to the clients, thus allowing easy access of resources with minimum response time and enormous economic benefits in factors such as cost. Our proposed architecture has been implemented for solving the issues related to deadlines, execution time, data consistency and proper resource utilization as given in figure 3.

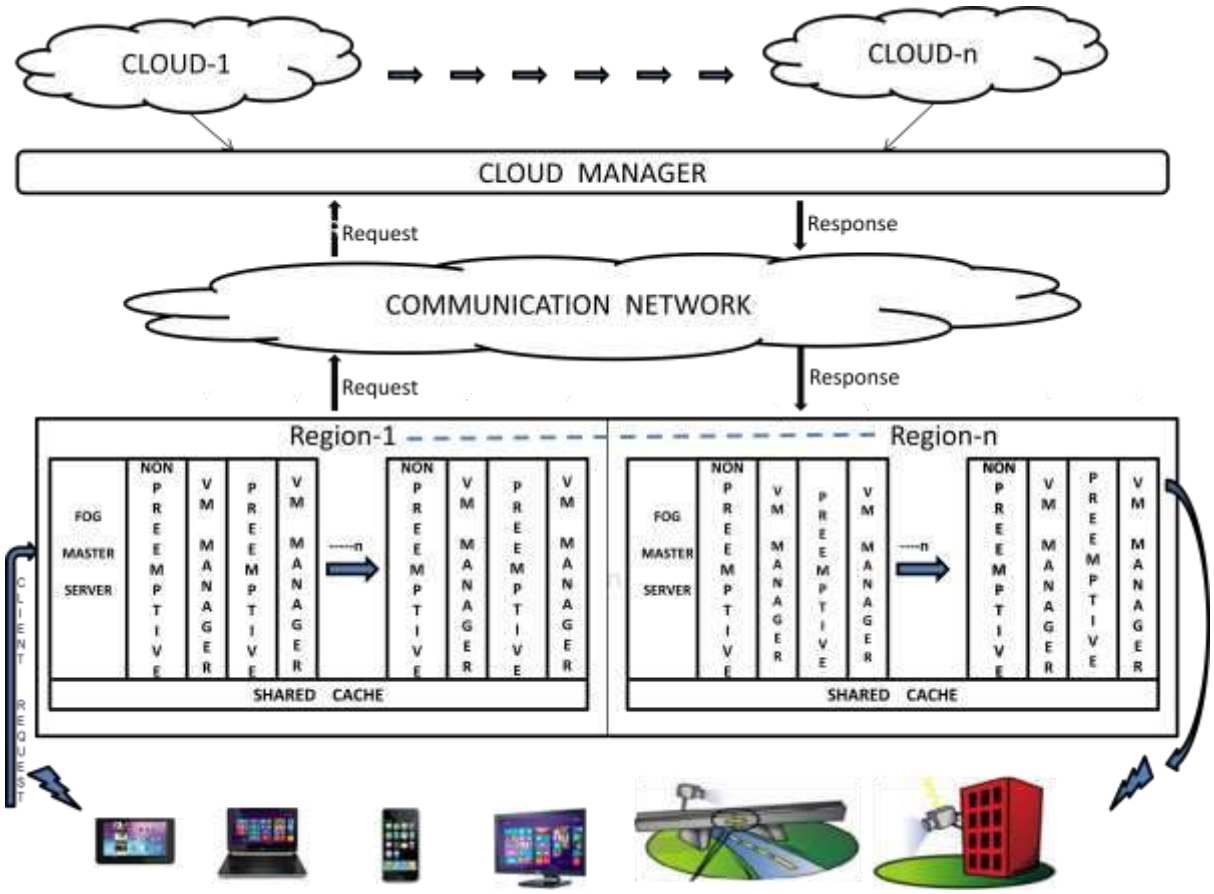

Fig.3. A three Layer architecture for efficient Load balancing in cloud -fog environment.

To solve the issue of load balancing in a fog computing environment, we have proposed a design model.

A. Design Model

In this model how the servers in cloud environment are interacting with the servers in fog environment is presented. It consists of three layers named as the client layer, fog layer and the cloud layer. During load balancing first we balance the load between client and the fog layer so as to fulfill clients' requirement of resources with minimum delay. In Second part, if clients do not get the required resource then the request is forwarded to the cloud layer. Given below various intermediate steps of First and Second portion.

Step 1: In Fog and cloud computing layer we have data centers which comprise of hosts and hosts in turn consists of VMs and VM manager.

Step 2: Fog computing layer consists of Fog Server Masters (F_S_M) and Coprocessor Fog Servers in different regions. Fog Servers on even indexes on the Fog Server Master Table are non preemptive in nature and dedicated to Real tasks while on odd indexes are preemptive dedicated to Soft tasks.

Step 3: Initially, when a client sends its request, it will be forwarded to fog server master (F_S_M) which is nearest to its region. The threshold capacity of F_S_M used is 90 %, and rest 10 % is reserved for ideal use, for instance suppose if all the system present in this region get overloaded and unexpectedly real tasks arrived at F_S_M so on during that instant of time real time tasks will be executed on F_S_M while soft tasks are kept waiting on the queue.

Step 4: If F_S_M fails in a worst case, then Mirror Server of F_S_M will process Request.

Step 5: If the demand for VM requirement increases during executing of real tasks on even coprocessor then F_S_M by seeing its table will allocate nearby low loaded preemptive Coprocessor so that execution of real tasks become feasible or request be will transferred to Ideal Coprocessors. The total threshold capacity of a system comprises of 90 percent capacity of fog servers

and the rest 10 percent belongs to ideal servers, which will be used when during execution of real tasks if even index coprocessor fails.

Step 6: If during execution of Step 3 and Step 5 data needed is not available within the same region, then F_S_M will send an acknowledgement for Data copy to cloud server, and also during the same time F_S_M also checks by seeing its table which fog servers F_S are low overloaded.

Step 7: The received data copy after used will not be forwarded to other Fog servers or Cloud servers till commit acknowledgement is granted by the F_S_M to cloud servers thus maintaining consistency.

Step 8: During the worst case if all the fog servers fail, then request will be forwarded directly to the cloud servers.

B. Functional components

Role of F_S_M: To ensure the availability of all Coprocessors to the client.

Role of VMs: To process the requests, which are sent to the Fog Servers and Cloud Servers, and after processing the results are sent back to the clients.

Role of F_S: Each region consists of Fog Servers which in turn comprises of virtual machines that execute the Real tasks or Soft tasks by using the Server virtualization technique.

IV. Proposed Work

The proposed algorithm, RTES pseudo code for Load balancing in the Fog computing environment is given below. The main objective of RTES is to efficiently balance the load by using the available bandwidth, executing the real tasks before they met their deadline, and to give response to the clients in less time by using the intermediate fog layer.

- ceven- A variable which is used to count the number of real tasks executing on VM.
- codd- A variable which is used to count the number of soft tasks executing on VM.
- cs - Cloud server process the request.
- c_d_m - Cloud data manager.
- copro- Coprocessor which can be preemptive or non preemptive in nature.
- data_ava- Data available.
- data_c - Data copy.
- f- Fails.
- $Ideal_c$ - Ideal coprocessor.
- $index_{even}$- Non preemptive processor dedicated to real tasks.
- $index_{odd}$- Preemptive processor dedicated to soft tasks.
- Maxcap - Capacity of fog server master.
- MasterS - Master server.
- Net_T_{Exe}- Time remains left after subtracting total spent time from total execution time.
- $Net\_T_{Evenexe}$- It is the net total time remains left after execution of even tasks.
- Net_T_{Oddexe}- It is the net Total time remains left after execution of odd tasks.
- Request i_f - The request from users to fog master server.
- rem_t- Remaining time.
- T_{com}- Time spent by client to connect with fog server master.
- $T_{Remeven}$- The time still remaining of a task for proper execution on even processor.
- T_{Remodd}- The time still remaining of a task for proper execution on odd processor.
- $T_{Spenteven}$- The time already used by a real task after getting executed on coprocessor.
- $T_{Spentodd}$- The time already used by soft task after getting executed on coprocessor.
- T_{ser}- Time spent by fog server master in finding data and load of coprocessor in same region.
- T_{trans}- Transferring tasks.
- T_{Tot} - Total execution time of a task.
- $T_{TotRemeven}$- It is the total current sum of remaining execution time of even tasks.
- $T_{TotRemLow}$- It is the total current sum of load on Coprocessor.
- T_n- Tasks set which is a combination of T_{nreal} and $T_{nsoft.}$.
- σ - Threshold capacity of the fog server master.

1. For each Request i_f.
2. Each Request i_f is sent to MasterS nearest to its Region
3. if Maxcap > σ then
   MasterS check if task is T_{real} or T_{soft} through arguments if T_{real} then $T_{Release}$, $T_{Deadline}$
   else $T_{soft} = T_{Release}$
   if T_{real} = yes then
   Req. will be forward to $index_{even}$ copro. &&
   $Net\_T_{Evenexe} = T_{Tot} - T_{Spent}$ (T_{com}+ T_{ser})
   while T_{nreal} execute on $index_{even}$ copro rem_t
   for i= 1 to T_{nreal}
   $T_{Remeven}[i] = Net\_T_{Evenexe}[i] - T_{Spenteven}[i]$
   If copro $index_{even}$=f && $T_{Remeven}[i]$ =0
   then
   $Update_{data}$ = successful
   ceven ++
   else
   if $index_{even}$= f &&$T_{Remeven}[i]$>0
   then
   T_{nreal} – ceven will T_{trans} to $Ideal_c$
   end if
   end if
   end for
   end if
   else
   if T_{real} != yes then
   Req. is forwarded to $index_{odd}$
   for i = 1 to T_{nodd}
   $T_{Remodd}[i] = Net\_T_{Oddexe}[i] - T_{Spentodd}[i]$
   if copro $index_{odd}$ =f &&$T_{Remodd}[i]$=0 then
   $Update_{data}$ = successful
   codd++

```
            else
               if index_odd = f && T_Remodd[i]>0 then
                T_nodd − codd will be kept waited
                if Ideal_c! =available
                 and T_Remodd >=T_TotRemLow then
                    T_soft will release Res.
                 end if
                 else
                   if Ideal_c! =available
                   and T_Remodd <=T_TotRemLow then
                     execute T_soft tasks
                   end if
               else
                 if Ideal_c =available
                 and T_TotRemeven = 0 then
                      execute T_soft task
                 end if
               end if
            end for
            while executing of T_nreal tasks for data
            consistency call Update_data function
 4. if Maxcap < σ then
     MasterS will forward Req to nearby server
     in the same Region
     if T_real = yes and
     ((Net_T_Exe=T_Tot - T_Spent(T_com+T_ser)) <=T_TotRemLow
    then
       execute T_real task
     end if
    else
      if T_real = no then
        Req. will sent to F_S_M of other Region
       end if
       Here also for data consistency call Update_data
5. else
    if F_S_M fails then
      its Mirror Server will process T_n tasks
    end if
6. if all F_S fails then
       all Req. will be forwarded directly to CS
   end if
```

Algorithm for maintaining consistency of Data

```
Update_data
1. while client sent Req. to MasterS
2.  if data_ava = yes && copro are LOL then
      client Req. is processed.
    else
      if data_ava =no && copro are LOL then
        MasterS sent ack. to c_d_m for data_c.
        and at same time MasterS checks which
        copro are LOL.
      end if
    end if
  else
    if data_c processed = yes then
      commit ack. is sent to other copro. to fetch
      same data_c
    end if
end while
```

Our algorithm objective is to complete the Real tasks before or on the QoS parameter deadline, which in itself a very critical factor for execution of real tasks. Resource allocation to VM's are done in such a way that no VM has to remain in the idle state. It also maintains data consistency during the communication between VM's in the cloud layer and fog layer.

In this first the request is sent by the client to fog layer which is far more nearer to them as compared to the cloud layer. So this algorithm is efficient enough to allocate the resources, minimizing the response time and to give maximize the throughput.

V. Simulation Setup and Expected Results

A. Simulation Tool (CloudSim)

Our proposed RTES algorithm, have been implemented on simulator CloudSim [20, 21] 3.0.2 to execute tasks along with Window 7 OS, core i3 2.10 GHz processor and NetBeans IDE 7.2.1. The results obtained are then compared on the basis of the parameter turnaround time with the existing task scheduling algorithm like FCFS, Priority and Multi Objective Task Scheduling. In this task having varying size are executed on different number of VM's. The VM's created have the processing power ranging from 1000 to 5000 MIPS while executing task size on VM varies from 1000 to 8000. During execution of tasks, VM gives priority to real tasks over to soft tasks.

B. Result Comparison

Table 1. Simulation results using Turnaround time in seconds.

WL	VM	TASK	Turn Around Time Using			
			RTES	MOTS	FCFS	PRIORITY
WL 1	3	20	4.08	10	12	12
WL 2	3	50	7.46	18	22	20
WL 3	3	100	18.57	30	42	44
WL 4	3	200	37.6	62	85	80
WL 5	5	50	1.43	10	12	11
WL 6	10	100	0.88	6	7	8

The results obtained after implementing our proposed architecture and algorithm on CloudSim by using various workloads (WL) is shown in the above table. The detailed result obtained by Real time effective scheduling algorithm is compared on the basis of parameter Turnaround time with another algorithm like Multi Objective Task Scheduling, Priority Scheduling, First Come First Serve of the research paper [13]. It clearly shows that our algorithm gives the minimum amount of Turnaround time in seconds. In this we have shown comparison of six workloads having various numbers of

virtual machine executing different types of task. In workload first when we execute twenty tasks on three VM's then total time taken by all the tasks in retrieving data individually from cloud and fog layer is 4.08 seconds. It is quite a less time as compared to the time taken by MOTS, FCFS and PRIORITY i.e. 10, 12, 12 seconds respectively to retrieve the data from the fog and cloud layer for 20 tasks over three VM's. In workload second when fifty tasks are executed on three VM then again the time taken by RTES i.e. 7.46 is very less as compared to the time taken by MOTS, FCFS and PRIORITY which is 18, 22 and 20 respectively. Following this when we take workload third, fourth, fifth and sixth respectively our proposed algorithm does not affect the scalability of the system it still gives the less amount of turnaround time. The consequently increasing the number of VM's and executing tasks on them does not affect the efficiency. In the above table, we can clearly see that in the workload third and fourth taking the same number of VM's and executing 100 and 200 tasks still gives the less amount of time, i.e. 18.57 and 37.6 respectively , increasing workload does not affect the functioning of the system. In workload five when we execute 50 tasks on 5 VM's, by increasing the data size up to 5000 in size fog server master still balance the load and give total results in 1.43 seconds, it is quite less as compared to 10, 12, 11 taken in whole number from research paper [13] of MOTS, FCFS and PRIORITY. The last result obtained from workload six by executing 100 tasks on 5 VM's consisting of data size up to 3000 gives 0.88 seconds, which is still great as compared to 6, 7, 8 seconds of MOTS, FCFS and PRIORITY. RTES algorithm implemented in our architecture does data updating in fog layer and cloud layer in lesser amount of time. The fog servers in fog layer give quick response to the client requests because they are nearer to the clients as compared to the cloud when they demand for processed data. The increase in the intensity of client requests does not affect the efficiency and system throughput. Our proposed algorithm ensures proper utilization of resources as no virtual machine has remained in idle state during execution of tasks because the fog server master keep track of everything like continuously checking load on servers and allocating proper resources to them. RTES algorithm ensures that when the arrival intensity of real tasks increased on the fog server master, it does not crash it and must complete real tasks before they met their deadline. When the system is overloaded beyond its limit, then fog server master dispatches real tasks for execution over ideal server and within the same time it searches in its table which servers are low loaded so that it can allocate resources to the future upcoming real tasks. Fog server master categorizes the tasks into real tasks or soft tasks by looking into its table contents and then assign them even or odd coprocessor which are non preemptive or preemptive in nature. It also ensures that soft tasks do not have to wait for a long time in a queue as they too are also important for getting maximum throughput of the system. If the execution of real tasks becomes vital then it can be assigned to odd coprocessor by preempting already executing soft tasks from the odd coprocessor. The final output of our proposed algorithm on the basis of Turnaround time, clearly shows that our proposed algorithm proves to be the best as compared to others as it gives the minimum amount of execution time and lowers it to a greater extent. The results obtained here will also remain dependent on the system state, i.e. how many tasks are executing on your computer, how much capacity of ram you are using and whether you are using an Intel 7 or Dual core processor or anyone else. RTES algorithm does not get affected by the increase in the transactions of data, it balance the load effectively to give maximum throughput.

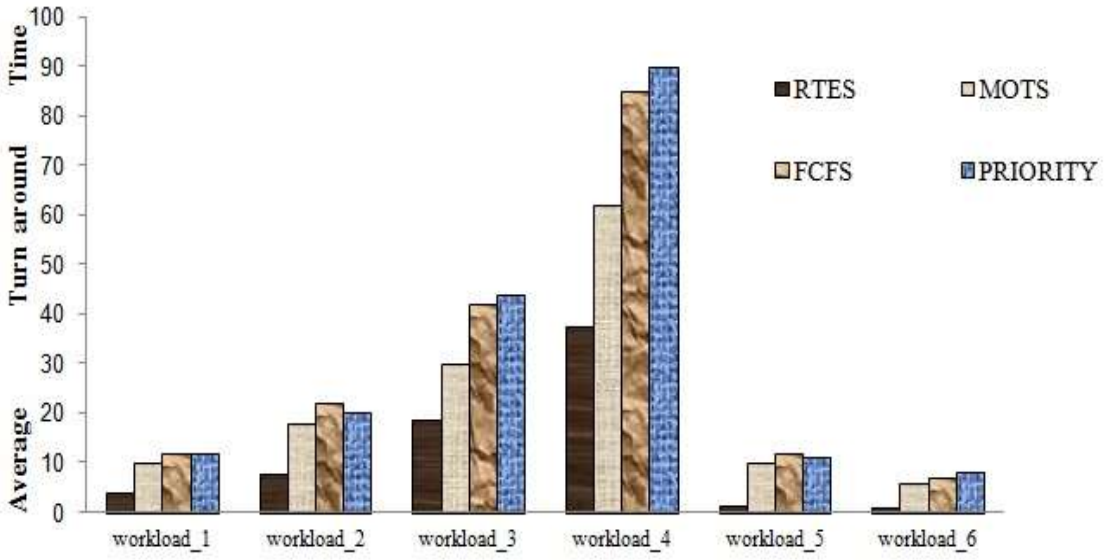

Fig.4. Perfrmance evaluation of proposed scheduling algorithm with FCFS , Priority and Multi Objective Task Scheduling.

In the above graph we can clearly see that our proposed technique is better among existing algorithm on the basis of Turnaround time.

VI. Conclusion

In our proposed work, the load balancing scheduling algorithm has been developed in a Fog Computing environment. Fog layer is basically an intermediate layer between client layer and cloud layer and has been introduced to improve the efficiency of cloud computing environment by proper utilization of bandwidth, as data transmitted or exchanged between cloud and fog for processing get reduced. In fog computing environment, the enormous amount of data of wireless objects such as sensors and Internet of things in distributed environment has been placed the at the edge of the cloud, so that it allows faster accessing, give maximum throughput and met other computing requirement of real time applications. It's become feasible as it has not to be hosted and worked from a centralized cloud thereby fog computing is also called as an edge computing. In our proposed work, a real time efficient scheduling (RTES) load balancing algorithm has been proposed and implemented in the CloudSim tool in the fog computing environment. The result obtained after implementing our proposed architecture and algorithm are tremendously good, it has given minimum execution time, fast response to the client request, completing real tasks before they met their deadline, meanwhile maintaining data consistency along with proper resource and bandwidth utilization as compared to the existing algorithms like FCFS, Priority and Multi Objective Tasks scheduling algorithm in fog computing environment. Our proposed algorithm is 90 percent efficient and in future it can be further improved by including other QoS factors such as security etc.

References

[1] Rajkumar Buyya, James Broberg and Andrzej Goscinski CLOUD COMPUTING Principles and Paradigm, Jhon Wiley & Sons,2011.

[2] M.D. Dikaiakos, G. Pallis, D Katsa and P.Mehra "Cloud Computing: Distributed Internet Computing for IT and Scientific Research", in Proc. of IEEE Journal of Internet Computing, Vol. 13, No. 5, pp.10-13, 2009.

[3] Ivan Stojmenovic, sheng Wen, "The Fog Computing Paradigm: Scenarios and security issues" Proceedings of the IEEE International Fedrerated Conference on Computer Science and Information Systems, 2014, pp.1-8.

[4] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, Sateesh Addepalli "Fog Computing and its Role in the internet of things",http://conferences.sigcomm.org/sigcomm/2012/paper/mcc/p13.pdf.

[5] Zhen Xiao, Senior Member, IEEE, Weijia Song, and Qi Chen, "Dynamic Resource Allocation Using Virtual Machines for Cloud Computing Environment", ",IEEE Transactions on Parallel and Distributed Systems, Vol.24, No. 6, June 2013, pp. 1107-1117.

[6] LoadBalancing,https://en.wikipedia.org/wiki/Load_balancing_%28computing%29.

[7] Aditya Marphatia, Aditi Muhnot, Tanveer Sachdeva, Esha Shukla, Prof. Lakshmi Kurup," Optimization of FCFS Based Resource Provisioning Algorithm for Cloud Computing" , IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661, p- ISSN: 2278-8727 Vol. 10, Issue 5 (Mar. - Apr. 2013), PP 01-05.

[8] Chandrasekhar S. Pawar, Rajnikant B. Wagh," Priority Based Dynamic resource allocation in Cloud Computing with modified Waiting Queue", Proceeding of the IEEE 2013 International Conference on Intelligent System and Signal Processing(ISSP) Pages 311-316.

[9] El-Sayed et al." Extended Max-Min Scheduling Using Petri Net and Load Balancing",International Journal of Soft Computing and Engineering(IJSCE) ISSN: 2231-2307, Vol. 2, Issue 4, September 2012.

[10] Antony Thomas, Krishnalal G, Jagathy Raj V P,"Credit Based Scheduling Algorithm in Cloud Computing Environment", International Conference on Information and Communication Technologies, Procedia Computer Science 46(2014) 913-920.

[11] Dhinesh Babu L.D, P. Venkata Krishna,"Honey bee behavior inspired load balancing", Elsevier, Applied Soft Computing 13(2013) 2292-2303.

[12] Brototi Mondala, Kousik Dasguptaa, Paramartha Duttab"Load Balancing in Cloud Computing using Stochastic Hill Climbing-A Soft Computing Approach", Elsevier, Procedia Technology 4(2012) pp. 783 – 789.

[13] Atul Vikas Luthra and Dharmendra Kumar Yadav,"Multi-Objective Tasks Scheduling Algorithm for Cloud Computing Throughput Optimization", International Conference on Intelligent, Communication & Convergence, Procedia Computer Science 48(2015) 107-113.

[14] V. Suresh Kumar," Trust Based Resource Selection in Cloud Computing Using Hybrid Algorithm" I.J. Intelligent Systems and Applications, 2015,08, 59-64.

[15] Dr. M. Dakshayini and Dr. H.S. GuruPrasad," An Optimal Model for priority based service Scheduling Policy for Cloud Computing Environment", International Journal of Computer Applications(0975-8887) Vol. 32- No.9, October 2011.

[16] Po-Huei Liang and Jiann-Min Yang,"Evaluation of two level global load balancing framework in Cloud Environment", International Journal of Computer Science and Information Technology(IJCSIT), Vol. 7 No 2, April 2015.

[17] Shimpy, Jagandeep Sidhu," Different Scheduling Algorithms In Different Cloud Environment", International Journal of Advanced Research in Computer and Communication Engineering Vol. 3, Issue 9, September 2014.

[18] Himani and Kamaljit Kaur," Deadline Scheduling in Cloud Computing: A Review", International Journal of Computer Applications(0975-8887),Vol. 96-No.24,\june 2014.

[19] Manisha Verma, Neelam Bhardwaj Arun Kumar Yadav," An architecture for load balancing techniques for Fog computing environment", International Journal of Computer Science and Communication, Vol. 8 • Number 2 Jan - Jun 2015 pp. 43-49.

[20] Rahul Malhotra, Prince Jain," Study and Comparison of Various Cloud Simulators Available in the Cloud Computing",International Journal of Advanced Research in Computer science and Software Engineering ISSN: 2277 128X Vol. 3, Issue 9, Sept 2013.

[21] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling And Simulation Of Scalable Cloud Computing Environments And The CloudSim Toolkit: Challenges And

Opportunities," Proc. Of The 7th High Performance Computing and Simulation Conference (HPCS 09), IEEE Computer Society, June 2009.

Authors' Profiles

Manisha Verma has done B. Tech in Computer Science & Engineering from F.E.T. Engineering College, Agra. She is now pursuing M.Tech in Computer Science & Engineering from Hindustan Institute Technology & Management, Agra.

Her research interest includes Cryptography and Network Security, Distributed System, Object Oriented System, Cloud Computing and Fog Computing.

Neelam Bhardwaj has done M.Sc from Banasthali Vidyapith, Rajasthan in Computer science & Engineering, M.Tech in Computer Science & Engineering from Banasthali Vidyapith, Rajasthan and pursuing Ph.D. in Computer Science and Engineering from MNNIT, Allahabad. Presently she is working as Associate Professor in the Department of Computer Science & Engineering at H.I.T.M Agra, Uttar Pradesh, India.

Her research interest includes Digital image processing, Pattern recognition, Cloud Computing and Fog Computing.

Arun Kumar Yadav has done B.E. (Computer Science & Engineering) from G.B. Pant Engineering College, Pauri Garhwal, M.Tech (Information Technology) from Sam Higginbotom Institute of Agriculture, Technology & Sciences, Allahabad and pursuing Ph.D. in Computer Science and Engineering from Uttarakhand Technical University, Dehradun. Presently he is working as Associate Professor in the Department of Computer Science & Engineering at ITM University Gwalior, Madhya Pradesh, India.

His research interest includes Distributed Database Security, Cloud Computing and Fog Computing. He is a senior member of IACSIT and IAENG Technical Societies.

How to cite this paper: Manisha Verma, Neelam Bhardwaj, Arun Kumar Yadav,"Real Time Efficient Scheduling Algorithm for Load Balancing in Fog Computing Environment", International Journal of Information Technology and Computer Science(IJITCS), Vol.8, No.4, pp.1-10, 2016. DOI: 10.5815/ijitcs.2016.04.01