# MODIFIED OPTIMAL ALGORITHM

FOR LOAD BALANCING IN CLOUD COMPUTING

Mrs. Shruti Tripathi
Assistant Professor
Department of Computer Science & Engineering
Feroze Gandhi Institute of Engineering and Technology
Raebareli ,India
shru_tri@yahoo.com

Shriya Prajapati
B.Tech Student
Department of Computer Science & Engineering
Feroze Gandhi Institute of Engineering and Technology
Raebareli ,India
prjpshriya12@gmail.com

Nazish Ali Ansari
B.Tech Student
Department of Computer Science & Engineering
Feroze Gandhi Institute of Engineering and Technology
Raebareli ,India
ali123nazish@gmail.com

**Abstract-A cloud is a technology which provides services over the network on demand.In cloud computing, the cloud user may get access to the data and applications from anywhere at any time. Users have the advantage of paying only for what they are using satisfying their fluctuating demands. Managing such a large amount of user requests coming from all over the globe is a tedious task, thus the cloud service provider should take efficient steps to handle them. This involves the concept of load balancing. Load Balancing basically helps in even distribution of loads among all the nodes such that no node is overloaded or under loaded. Considering the growing users and importance of cloud, finding new ways which can prove efficient in balancing the load among various processing nodes will always be an area of concern and research. Various algorithms already exist for load balancing, each having its own concept but have the same goal to reduce the response time to client problems. The goal of this paper is to propose an algorithm for Load balancing. In the existing optimized scheduling algorithm, Shortest Job First (SJF) should be used to arrange the tasks and then Round Robin (RR) should be used for processing, without giving any preference to their priorities preventing important tasks from getting the amount of attention they need. The algorithm proposed in this paper aims to give higher importance to requests with higher priorities.**

Keywords—***Broker, Burst time , Cloudlet, Context switch, Data center, Priority queue, Queue, Threshold ,Time quantum, Virtual Machines, Waiting time***.

## I. INTRODUCTION

Cloud computing is an emerging technology, organizations whether in the public sector or private sector is now switching over to the cloud because of features like ubiquitous on-demand computing, pay-per-use, and most importantly virtualization which saves cost. In the IT industry, the main reason behind the development of new technologies is to reduce the response time to client problems and gain more and more client satisfaction. Load balancing in cloud computing does the same work Here, requests after reaching the data center(where information are stored in bulk) are distributed among the virtual machines following certain load balancing algorithms. The way the data centers are selected, virtual machines are created, tasks are allocated and other similar things are done, are dependent on policies of the cloud made by the cloud service providers. Load balancing is a complicated task in cloud computing because in a network we have computers with varying capacities and users with different requirements which make the process of assigning tasks to different nodes complex. Thus various techniques are proposed from time to time in order to make this complex task easier.

## II. KEYTERMS

1. ***Datacenter:***models the infrastructure services provided by the cloud service providers.
2. ***Cloudlet:*** A cloudlet is considered to be a task to be performed. These are actually user requests.
3. ***Context Switch:*** It is a process of storing the context of a preempted process and starting the execution of some other process, the process whose context was saved can be restored later from the same point where it was stopped.
4. ***Queue:***Data structure in which elements are handled in FIFO (First in First out) manner. The element entered first is the first one to be extracted.
5. ***Broker:*** A broker is responsible for creating virtual machines, assigning ID's to those virtual machines and for other similar tasks. It follows certain policies for resource allocation. In case of any kind of interruption, it transfers a task to another broker only when it is having exact replica of the virtual machine, the transferring broker is having.
6. ***Virtual Machine (VM):*** Virtualized machines dynamically created to meet user requirements like actual machines.

7. ***Waiting Time:*** Waiting time is defined as the total time a process has been waiting in ready queue.
8. ***Burst time (BT):***(Execution time)It is the maximum amount of time a process needs for execution.
9. ***Priority Queue:***Queue in which processes are stored priority wise.
10. ***Time Quantum:***It is defined as the small time slice decided in Round Robin.
11. ***Threshold:*** It is the value decided in the beginning with which the difference in execution times of tasks is compared. It helps in choosing the scheduling algorithm for proper load balancing.

:

## III. Basic Scheduling Algorithms[4]

1. ***First Come First Serve (FCFS):***In this algorithm the process that arrives first is the first one to be executed completely.

2. ***Shortest Job First (SJF):*** The processes are executed in the increasing order of their execution time.

3. ***Priority Based Scheduling (PBS):*** Processes are executed on the basis of priorities assigned to them.

4. ***Round Robin Scheduling (RR):***This algorithm uses a time quantum. Each process is executed for that time quantum when its turn comes and kept in the waiting state if it is not complete. The scheduler then switches over to the next process in the queue, executes it for the decided time quantum and then moves to the next process. This continues till all the processes are completed.

## IV. Optimized Scheduling Algorithm [5]:

In the existing optimized scheduling algorithm , the tasks are first arranged in the increasing order of their execution time (According to SJF) and then the tasks are executed according to Round Robin. In this algorithm preference is given to the task with minimal execution time and then moving towards higher than that.

Here priorities of the tasks are not takeninto account which is very important, as when we will have large number of user requests, putting all the effort in completing the tasks with shorter time first, will cause the important tasks though with longer execution time to wait longer for their turn to come. Thus when tasks requiring quicker execution will not be entertained,it will increase the waiting time of other tasks which depend upon them to execute. This cycle of dependency of one task over the other can be bigger which can further increase the complexity at a greater level.This can result in slowing down of the entire system.

**Example:**

| Process | Burst Time |
|---|---|
| P1 | 10 |
| P2 | 6 |
| P3 | 8 |
| P4 | 4 |
| P5 | 5 |

**Using FCFS:**

| | | | | |
|---|---|---|---|---|
| P1 | P2 | P3 | P4 | P5 |

0 10 16 24 28 33

| Process | Waiting time | Turn around time |
|---|---|---|
| P1 | 0 | 10 |
| P2 | 10 | 16 |
| P3 | 16 | 24 |
| P4 | 24 | 28 |
| P5 | 28 | 33 |

Average waiting time= (0+10+16+24+28)/5= 15.6 ms

**Turn around time= waiting time +burst time**

Average turn around time= (10+16+24+28+33)/5 = 22.2ms

**Using SJF:**

| | | | | |
|---|---|---|---|---|
| P4 | P5 | P2 | P3 | P1 |

0 4 9 1523 33

| Process | Waiting time | Turn around time |
|---|---|---|
| P1 | 23 | 33 |
| P2 | 9 | 15 |
| P3 | 15 | 23 |
| P4 | 0 | 4 |
| P5 | 4 | 9 |

Average waiting time= (23+9+15+0+4)/5= 10.2ms
Average turn around time= (33+15+23+4+9)/5 = 16.8ms

**Using Round Robin:**

Time quantum for Round robin=ceiling( (10+6+8+4+5)/5) = 6

| | | | | | | |
|---|---|---|---|---|---|---|
| P1 | P2 | P3 | P4 | P5 | P1 | P3 |

0 6 12 18 22 27 31 33

| Process | Waiting time | Turn around time |
|---|---|---|
| P1 | 21 | 31 |
| P2 | 6 | 12 |
| P3 | 25 | 33 |
| P4 | 18 | 22 |
| P5 | 22 | 27 |

Average waiting time= (21+6+25+18+22)/5= 18.4 ms

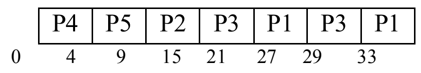
**Turn around time= waiting time +burst time**

Average turn around time= (31+12+33+22+27)/5 = 25 ms

**Using Optimized Scheduling Algorithm:**

Processes will be arranged in the increasing order of their execution time.

| Process | Burst Time |
|---|---|
| P4 | 4 |
| P5 | 5 |
| P2 | 6 |
| P3 | 8 |
| P1 | 10 |



| Process | Waiting time | Turn around time |
|---|---|---|
| P1 | 23 | 33 |
| P2 | 9 | 15 |
| P3 | 21 | 29 |
| P4 | 0 | 4 |
| P5 | 4 | 9 |

Average waiting time= (23+9+21+0+4)/5= 11.4 ms
Average turn around time= (33+15+29+4+9)/5 = 18 ms

## V. PROPOSED WORK

The tasks arrive along with their priority values as well as with their burst time. The tasks are kept into priority queues according to their priority values and a separate time quantum is given to each priority queue.

The proposed algorithm uses the concept of following algorithms:

i. RR (Round Robin).
ii. Priority Scheduling.
iii. SJF(Shortest Job First).

Priorities are assigned in the beginning after analyzing the resource requirement ,time or memory requirement or user preference. They will be send to the priority queues with the matching priority.

**Algorithm:**

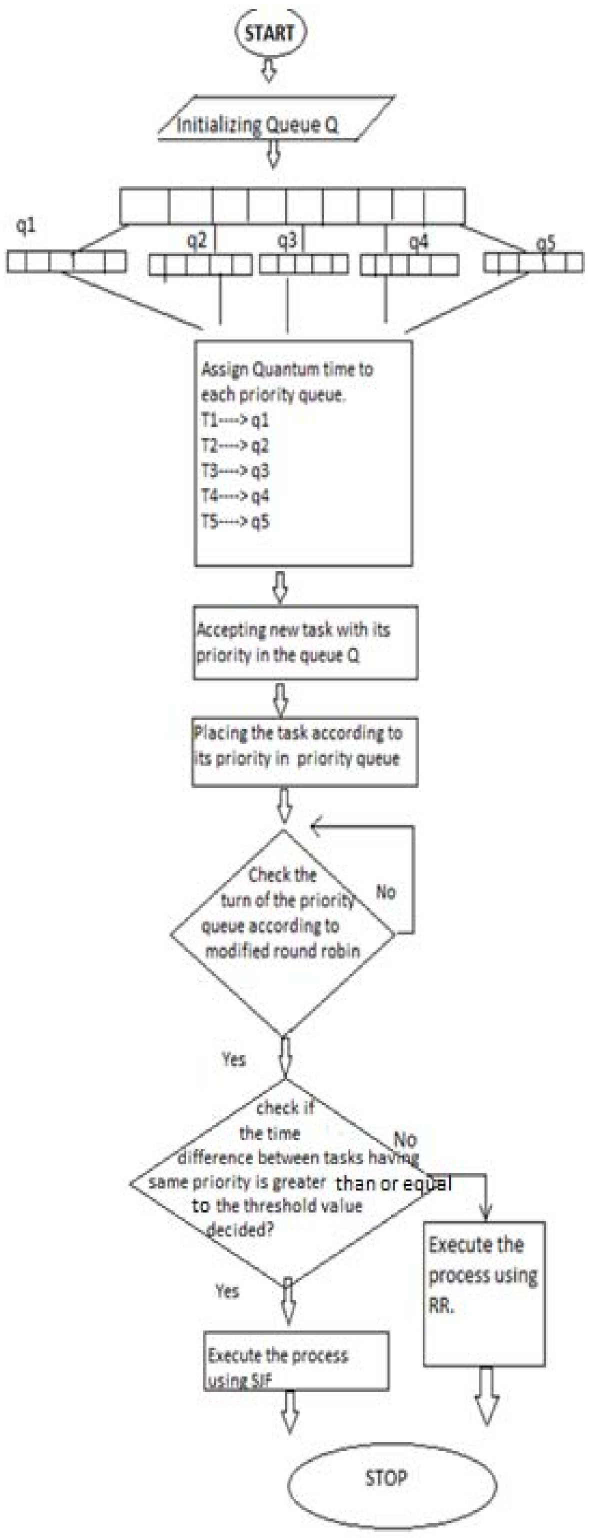
1. **Start the process.**
2. **Initializing the main queue Q. All the incoming processes are stored in the main queue.**
3. **The main queue is divided into five priority queues depending upon their priorities namely (q1, q2, q3, q4, q5).**
   **Highest priority** $\longrightarrow$ **q5**
   **Lowest priority** $\longrightarrow$ **q1**
4. **Each priority queue is assigned a time quantum .**
5. **Any new request for resource (containing its priority value), goes to the main queue Q first.**
6. **The request is assigned to a priority queue with the matching priority.**
7. **Now the task waits in the priority queue until the chance of its priority queue comes (according to modified round robin algorithm).**
8. **A threshold value is decided initially, which is used to compare the time difference ( TimeDiff ) between the tasks within the priority queues.**

```
If (TimeDiff>threshold)
        Then SJF will be used.

Else
        RR will be used.
```

**<u>Note:</u>**
**1.** The time quantum for a priority queue will be equal to the floor function of the average execution time of the processes it is containing.
**2.** The threshold value of the algorithm will be equal to the execution time of the shortest job.

### *A. FLOW CHART*


START
Initializing Queue Q
q1
q2
q3
q4
q5
Assign Quantum time to each priority queue.
T1----> q1
T2----> q2
T3----> q3
T4----> q4
T5----> q5
Accepting new task with its priority in the queue Q
Placing the task according to its priority in priority queue
Check the turn of the priority queue according to modified round robin
No
Yes
check if the time difference between tasks having same priority is greater than or equal to the threshold value decided?
No
Yes
Execute the process using SJF
Execute the process using RR.
STOP

**Example:**

Processes will arrive along with their priority values.

| Process | Burst Time | Priority |
|---|---|---|
| P1 | 10 | 5 |
| P2 | 6 | 4 |
| P3 | 8 | 3 |
| P4 | 4 | 3 |
| P5 | 5 | 5 |

**q5**

| | |
|---|---|
| P1 | P5 |

Time quantum for q5= floor((10+5)/2)= 8ms

**q4**

| |
|---|
| P2 |

Time quantum for q4=6 ms

**q3**

| | |
|---|---|
| P3 | P4 |

Time quantum for q3=floor((8+4)/2)= 6ms

**Threshold value = 4**

Round 1.

In q5

Difference in execution time of process P1 and P5 is 5 which is greater than 4(threshold value), thus SJF will be used and process P5 will be executed using the time quantum of q5

In q4

Process P2 will be executed using the time quantum of q4.

In q3

Difference in execution time of Process P3 and P4 is 4 which is equal to 4(threshold value), thus SJF will be used and process P4 will be executed.

| | | |
|---|---|---|
| P5 | P2 | P4 |

0 5 11 15

Round 2.

In q5

Process P1 will be executed.

In q3

Process P3 will be executed.

| | | | |
|---|---|---|---|
| P1 | P3 | P1 | P3 |

15 23 29 31 33

The overall gantt chart is :

| | | | | | | |
|---|---|---|---|---|---|---|
| P5 | P2 | P4 | P1 | P3 | P1 | P3 |

0 5 11 15 23 29 31 33

Waiting time and turn around time:

| Process | Waiting time | Turn around time |
|---|---|---|
| P1 | 21 | 31 |
| P2 | 5 | 11 |
| P3 | 25 | 33 |
| P4 | 11 | 15 |
| P5 | 0 | 5 |

Average Waiting time=(21+5+25+11+0)/5= 12.6
Average turn around time=(31+11+33+15+5)/5=19

## B. COMPARISON

MOSA=Modified Optimal scheduling Algorithm
OSA= Optimal Scheduling Algorithm

**TABLE 1**. Comparison of various scheduling algorithms

| Parameters | FCFS | SJF | Priority | RR | OSA | MOSA |
|---|---|---|---|---|---|---|
| Whether Priority is taken into account? | No | No | Yes | No | No | Yes |
| Used in time shared systems | No | No | No | Yes | Yes | Yes |
| CPU Utilization | Low | Medium | Medium | High | High | High |
| Throughput | Low | High | Low | Medium | Medium | Medium |
| Context switching | No | No | No | More | Less than RR | Less than RR |
| Pre-emption | No | No | No | Yes | Yes | Yes |
| Starvation of jobs | Can occur when jobs with larger execution time arrive first | Jobs with larger execution time can suffer from starvation | Jobs with lower priority can face starvation | No | No | No |
| Turn aroundtime | High | Medium | High | Medium | Less than RR | Less than RR |

## C. IMPLEMENTATION FOR LOAD BALANCING IN CLOUD

All the resources are stored in bulk at the data centers. User requests coming from different regions will reach the chosen data center. Here all the requests will be stored in the main queue. These sets of requests coming from different user bases are considered as cloudlets or tasks. They will come along with their priority values.

Now the main queue will move the tasks to next level i.e in the priority queues which will have the same priority value as that of the task. Instead of making all the data centers busy at that the same time, the minimum number of datacenters which can perform the tasks completely are chosen. These datacenters will further create Virtual machines for processing the requests.

### PSEUDO CODE

Variables:

- main_queue: stores all the processes in the order they arrive.
- VM_State: stores the status of the virtual machine.
- count: stores the number of processes in the priority queue.
- time_diff: stores the difference in execution time between requests.
- pri_queue: used for priority queues.
- Ex_time: stores the execution time of requests.

```
Modified_Optimal_Algo
    {
    Initialize the main_queue with the processes
    Initialize VM_State, count, time_diff
    Arrange the processes according to their priority in the main_queue.
     Put them in the pri_queue with the matching priority.
   While main_queue != NULL
       {
For each pri_queue starting from the highest priority
    {
    Set the value of count.
If count = 1

        Allocate VM to request and setVM_State to BUSY.
    Execute it according to the timestamp of the respectivepri_queue.
    Deallocate VM and set VM_State to FREE.

Else

        Calculate the time_diff between the request with
    highest and lowest execution time.
    If time_diff> threshold
```

```
        Use SJF
        Allocate VM to request and  setVM_State to
        BUSY
Execute the shortest job for the timestamp
Decided for the priority queue.

        (If the shortest job gets completely executed and
        there is still some time remaining keep executing
        the next higher job until the timestamp of the
        priority queue finishes.)

        .Deallocate VM and set VM_State to FREE.
        Else
        Use RR
        Allocate VM to request and  setVM_State to
         BUSY.
        Execute using Round Robin according to the
        timestamp of therespective pri_queue.
        Deallocate VM and set VM_State to FREE.
        Update the main_queue and the pri_queue.
                    }
                }
        }
```

## VI. CONCLUSION

Thus the given algorithm pays greater attention to tasks with higher priority and attempts to execute them faster and in case of tasks with lower priority it attempts to minimize the context switching by comparing the difference in their execution time with the threshold value and then choosing between round robin and SJF. It aims to utilize the resources efficiently and provides a fair chance to each request for getting executed depending upon their importance.

## VII. REFERENCES

[1] Resource Management and Scheduling in Cloud Environment - Vignesh V, Sendhil Kumar KS, Jaisankar N .School of Computing Science and Engineering, VIT University Vellore, Tamil, Nadu, India – 632 014

[2] Cloud Computing Bible by Barrie Sosinsky

[3] "An Implementation of Load Balancing Policy for Virtual Machines Associated With a Data Center-"B.SANTHOSH KUMAR Assistant Professor, Department Of Computer Science, G.Pulla Reddy Engineering College. Kurnool-518007, Andhra Pradesh India. DR.LATHA PARTHIBAN Department of Computer Science, Community College, Pondicherry University.

[4] Abraham Silberschatz, Peter Baer Galvin, Greg Gangne;" **Operating System Concepts**", edition-7, 2005, John Wiley and Sons, INC..

[5] **"Optimized Scheduling Algorithm "** LalitKishor Dinesh Goyal ,Rajendra Singh ,Praveen Sharma Suresh GyanVihar University, Jaipur, Rajasthan, India. Proceedings published by International Journal of Computer Applications® (IJCA) , International Conference on Computer Communication and Networks CSI-COMNET-2011

[6] Ajit Singh, PriyankaGoyal, SahilBatra," An Optimized Round Robin Scheduling Algorithm for CPU Scheduling", (IJCSE) International Journal on Computer Science and Engineering Vol. 02, No. 07, 2383-2385, 2010.

[7] Mr. Sindhu M, Mr. Rajkamal R and Mr. Vigneshwaran P, "An Optimum Multilevel CPU Scheduling Algorithm". 978-0-7695-4058-0/10 $26.00 © 2010 IEEE

[8] GhalemBelalem, SamahBouamama and LarbiSekhri, "An Effective Economic Management of Resources in Cloud Computing", March 2011, JOURNAL OF COMPUTERS, Vol. 6, No. 3, page no: 404-411.

[9] Anton Beloglazov and RajkumarBuyya," Energy Efficient Resource Management in Virtualized Cloud Data Centers", 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, 978-0- 7695-4039-9/10,IEEE, DOI 10.1109/CCGRID.2010.46, page no: 826-831.

[10] Jasmin James, and Dr. BhupendraVerma," EFFICIENT VM LOAD BALANCING ALGORITHM FOR A CLOUD COMPUTING ENVIRONMENT ", Sep 2012, IJCSE, ISSN: 0975-3397 Vol. 4, No. 09, page no: 1658 – 1663.