# Load Balancing in Cloud Computing using Shortest Job First and Round Robin Approach

**Shreya Saini[1], Amninder Kaur[2]**

[1]Student, Punjabi University Regional Center

[2]Assistant Professor, Punjabi University Regional Center

**Abstract**: ***Cloud computing depends on sharing of assets to accomplish intelligibility and economies of scale, like a utility (like the power matrix) over a network In cloud computing various users sends request for the transmission of data for different demands. The access to different no. of user increases load on the cloud servers. Due to these cloud server does not provides best efficiency. To provide best efficiency load has to be balanced main problem in the paper is that different jobs can be divides in tasks. This paper the job dependency checking is done on the basis of directed a cyclic graph. The dependency checking the make span has to be created on the basis of shortest job first and pound robin approach. The minimization can be done on the basis of using min-min algorithm.***

**Keywords:** Cloud computing, round robin, shortest job first scheduling

## 1. Introduction

**Cloud computing** is a model for enabling ubiquitous network access to a shared pool of configurable computing resources.[1]
Cloud computing and storage solutions provide users and enterprises with various capabilities to store and process their data in third-party data centers.[2] It relies on sharing of resources to achieve coherence and economies of scale, similar to a utility (like the electricity grid) over a network.[3] At the foundation of cloud computing is the broader concept of converged infrastructure and shared services. The line "moving to cloud" likewise refers to an association moving far from a traditional CAPEX model (purchase the committed equipment and devalue it over a time of time) to the OPEX model (utilize an imparted cloud base and pay as one uses it).

Defenders guarantee that cloud computing permits organizations to avoid foundation expenses, and concentrate on tasks that separate their organizations rather than on framework. Proponents also guarantee that distributed computing permits enterprises to get their applications up and running quicker, with enhanced sensibility and less support, and empowers IT to more quickly change resources to meet fluctuating and capricious business demands. Cloud suppliers normally utilize a "pay as you go" model. This can prompt startlingly high charges if directors don't adjust to the cloud pricing model. The present accessibility of high-capacity network, low capacity PCs , gadgets and additionally the far reaching reception of equipment virtualization, administration situated structural planning, and autonomic and utility processing have prompted a development in cloud computing. Organizations can scale up as processing needs increment and after that scale down again as requests decreasing.

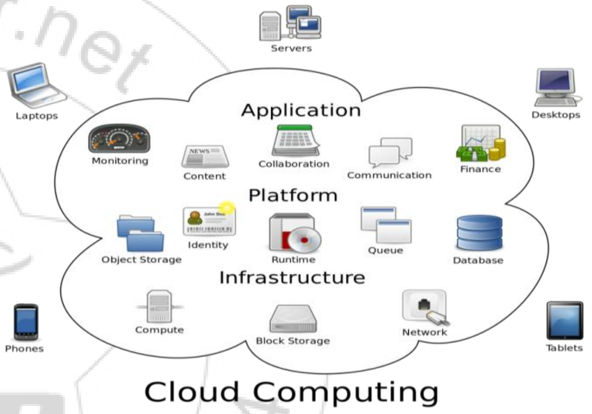


**Figure 1:** Cloud computing

## 2. Algorithms Used

### 2.1 Round Robin

Round-robin (RR) is one of the algorithms employed by process and network schedulers in computing. As the term is generally used, time slices are assigned to each process in equal portions and in circular order, handling all processes without priority (also known as cyclic executive). Round-robin scheduling is simple, easy to implement, and starvation-free. Round-robin scheduling can also be applied to other scheduling problems, such as data packet scheduling in computer networks. It is an Operating System concept. In order to schedule processes fairly, a round-robin scheduler generally employs time-sharing, giving each job a time slot or *quantum* (its allowance of CPU time), and interrupting the job if it is not completed by then. The job is resumed next time a time slot is assigned to that process. If the process terminates or changes its state to waiting during its attributed time quantum, the scheduler selects the first process in the ready queue to execute. In the absence of time-sharing, or if the quanta were large relative to the sizes of the jobs, a process that produced large jobs would be favoured over other processes. Round Robin algorithm is a pre-emptive algorithm as the scheduler forces the process out of the CPU once the time quota expires.

For example, if the time slot is 100 milliseconds, and *job1* takes a total time of 250 ms to complete, the round-robin scheduler will suspend the job after 100 ms and give other jobs their time on the CPU. Once the other jobs have had their equal share (100 ms each), *job1* will get another allocation of CPU time and the cycle will repeat. This process continues until the job finishes and needs no more time on the CPU.

**Pseudo Code:**

```
a) CPU scheduler picks the process from the circular/ready
   queue , set a timer to interrupt it after 1 time slice /
   quantum and dispatches it .
b) If process has burst time less than 1 time slice/quantum
    • Process will leave the CPU after the completion
    • CPU will proceed with the next process in the ready
      queue / circular queue .
      else If process has burst time longer than 1 time
      slice/quantum
    • Timer will be stopped . It cause interruption to the OS .
    • Executed process is then placed at the tail of the
      circular / ready querue by applying the context switch
    • CPU scheduler then proceeds by selecting the next
      process in the ready queue .
```

### 2.2 Shortest-Job-First Scheduling

A different approach to CPU scheduling is the shortest-job-first (SJF) scheduling algorithm. This algorithm associates with each process the length of the process's next CPU burst. When the CPU is available, it is assigned to the process that has the smallest next CPU burst. If the next CPU bursts of two processes are the same, FCFS scheduling is used. As an example of SJF scheduling, consider the following set of processes, with the length of the CPU burst given in milliseconds:

## 3. Parameters in Cloud

The factors that always be considered in various load balancing techniques in cloud computing are as follows detailed description of the load balancing factor is as Follows:

Response **Time-** It is the amount of time taken to provide the response by some load balancing algorithm in a distributed environment. This parameter should be minimized. It is represented as R (t). Formula to calculate the Response Time is:
R (t) = Finish Time - Start Time. = T (f) – T(s) (1) Where T(f ) is finish time and T(s) is start time.

**Communication Time-** It is defined as time taken by number of hops to travel in the communication channel. It is represented by C (t). Formula to calculate the Communication Time is: C (t) =2(Number of hops*Time to traverse between hops) (2)

Processing **Time-** It is defined as the difference between Communication Time and Response Time. It is represented by P (t). Formula to calculate the Processing Time is: P (t) = Response Time- Communication Time
= R (t) – C (t)

**Throughput-** It is used to calculate, number of tasks per unit time, whose execution has been completed. It is highly important factor for reliability and performance of the system. It is represented as T h (VI). T h (Vi) = (Cloudlet length*Number of cloudlets) / Response
Time= [Length (C i) – Ni] / R (t) (4)
Where Length (C i) is cloudlet length and Ni is number of cloudlets for specific virtual machine.

**Network Delay-** Delay in sending request and receiving response. It is the time taken by the network to send the number of cloudlets to particular VM and time taken by the VM to receive the cloud lets. D (t) = No. of cloudlets / Rate of transmission=N/r (5) Where “r” is the rate of transmission. So, in regression testing if the test cases that reveal the faults of output module execute first and test cases reveals faults of input module executes later then it will be delayed and in many cases will take long time to detect the original cause of output faults. If the dependencies can be detected earlier in regression testing then debugging can be started earlier and fault removal time will improve. In this paper I present a metric APFDD which measures fault dependency detection rate and I also present an algorithm to improve APFDD. A comparison between prioritized and non-prioritized test cases is also shown with the help of APFDD.

## 4. Methodology

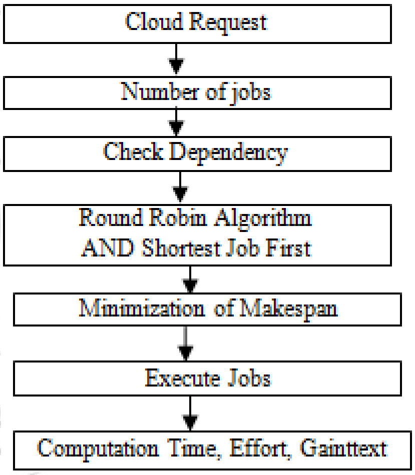


**Figure 2:** Flow of work

In the Purposed work various phases has to use for the development of the load balancing system in the cloud computing environment. These different phases have to be done for the completion of purposed work. Load balancing has been done by using dividing different tasks into no of jobs so that they can be allocate to different resource for processing to completes in less computation time. In cloud computing scenario no. of tasks has to be assigned on various processes to handle load on the cloud. These tasks have been divided into sets and the dependency checking is done for prevention of dead lock state or to prevent demand of various extra resource allocations. Make span has been developed on the basis of the allocation. Tasks have to be check for dependency by using directed Acyclic Graph. The round robin and shortest job first approach for the allocation of different tasks on the resources available in the cloud computing environment.

## 5. Results & Discussion

Cloud computing, or in more straightforward shorthand simply "the cloud", likewise concentrates on expanding the viability of the imparted resources. Cloud resources are normally imparted by different clients as well as reallocated by every interest. This can work for allocating resources to clients. load balancing distributes appropriates workloads over various processing resources, for example, PCs, a PC group, system joins, focal transforming units or circle drives. Load balancing aims to streamline resource utilization, maximize throughput, minimize reaction time, and keep away from over-load of any single resource. Utilizing multiple segments with burden adjusting rather than a solitary segment may build unwavering quality through repetition. Burden adjusting more often than not includes committed software or hardware, for example, a multilayer switch or a Domain Name System server process. The access to different no. of user increases load on the cloud servers. Due to these cloud server does not provides best efficiency. To provide best efficiency load has to be balanced main problem in the paper is that different jobs can be divides in tasks. The job dependency checking is done on the basis of directed a cyclic graph. The dependency checking the make span has to created on the basis of shortest job first and round robin approach. The minimization can be done on the basis of using min-min algorithm.

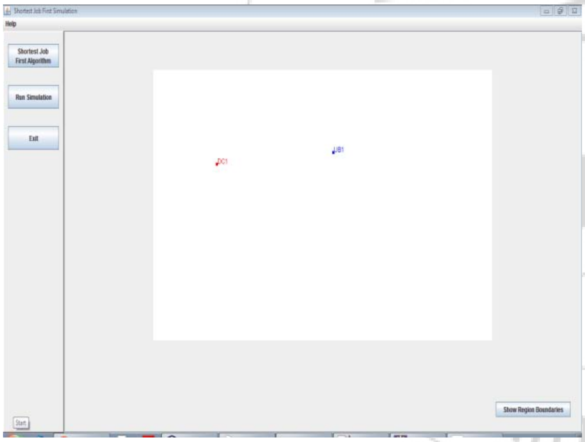


**Figure 3:** GUI

This figure is use to represent the Graphical User Interface. Red text represents the Datacenter, which can be server & Blue text represent the User base. SJF is shortest job first algorithm.

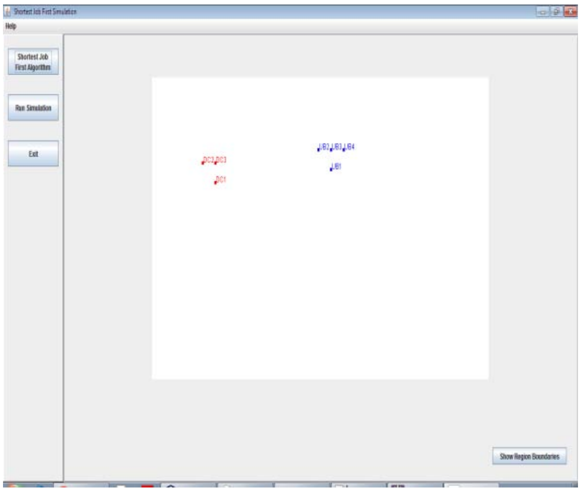
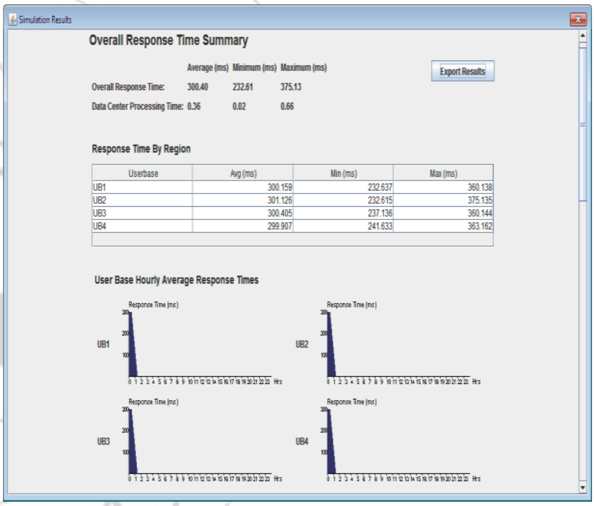


**Figure 4:** Loading of jobs with SJF

This figure is use to represent the loading of jobs in server by using SJF algorithms. Red text represents the Datacenter, which can be server & Blue text represent the User base. SJF is shortest job first algorithm. In this algorithms sort job load firstly. After loading the shortest job firstly on server we click on the Run Simulation button.


Simulation Results
Export Results
User Base Hourly Average Response Times
Response Time (ms)
UB1
UB2
UB3
UB4
300
200
100
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 Hrs

Overall Response Time Summary

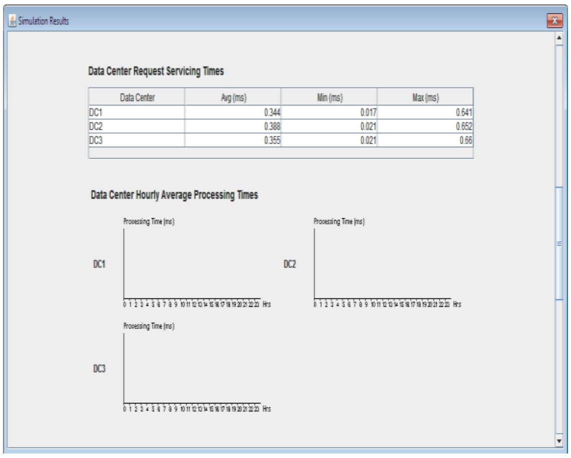
| | Average (ms) | Minimum (ms) | Maximum (ms) |
|---|---|---|---|
| Overall Response Time: | 300.40 | 232.61 | 375.13 |
| Data Center Processing Time: | 0.36 | 0.02 | 0.66 |

Response Time By Region

| Userbase | Avg (ms) | Min (ms) | Max (ms) |
|---|---|---|---|
| UB1 | 300.159 | 232.637 | 360.138 |
| UB2 | 301.126 | 232.615 | 375.135 |
| UB3 | 300.405 | 237.136 | 360.144 |
| UB4 | 299.907 | 241.633 | 363.162 |

**Figure 5:** Represent overall Loading time

This figure is use to represent the Time used to loading the jobs on the server by using SJF algorithms. In this algorithms sort job load firstly. After loading the shortest job firstly on server we click on the Run Simulation button. It shows the overall loading time i.e which job take how much time for loading.

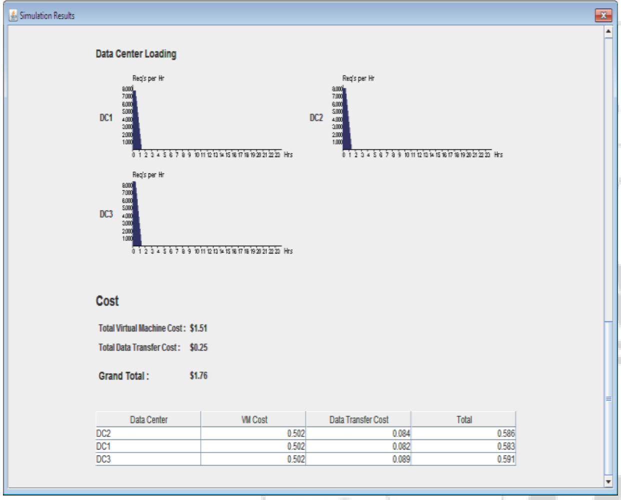

Simulation Results
Data Center Hourly Average Processing Times
Processing Time (ms)
DC1
Processing Time (ms)
DC2
Processing Time (ms)
DC3
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 Hrs

Data Center Request Servicing Times

| Data Center | Avg (ms) | Min (ms) | Max (ms) |
|---|---|---|---|
| DC1 | 0.344 | 0.017 | 0.641 |
| DC2 | 0.388 | 0.021 | 0.652 |
| DC3 | 0.355 | 0.021 | 0.66 |

**Figure 6:** Represent data center request servicing time

This figure is use to represent the data center request servicing time.



Cost

Total Virtual Machine Cost : $1.51

Total Data Transfer Cost : $0.25

Grand Total : $1.76

| Data Center | VM Cost | Data Transfer Cost | Total |
|---|---|---|---|
| DC2 | 0.502 | 0.084 | 0.586 |
| DC1 | 0.502 | 0.082 | 0.583 |
| DC3 | 0.502 | 0.089 | 0.591 |

**Figure 7:** Represent data center loading

This figure is use to represent the loading of data center.

## 6. Conclusion & Future Scope

Cloud computing depends on sharing of assets to accomplish intelligibility and economies of scale, like a utility (like the power matrix) over a network. At the establishment of cloud computing is the more extensive idea of met foundation and imparted administrations. load balancing distributes appropriates workloads over various processing resources, for example, PCs, a PC group, system joins, focal transforming units or circle drives. Load balancing aims to streamline resource utilization, maximize throughput, minimize reaction time, and keep away from over-load of any single resource. From results various parameters are analyzed & On the basis of these parameters we conclude our system gives us better results.

## References

[1] Hong Tao "A dynamic data allocation method with improved load-balancing for cloud storage system", ISSN 978-1-84919-707-6, PP 220 – 225, IEEE, 2013

[2] Yuqi Zhang "Dynamic load-balanced multicast based on the Eucalyptus open-source cloud-computing system",ISSN 978-1-61284-158-8, pp. 456 – 460, IEEE,2011.

[3] Magade, Krishnanjali A. "Techniques for load balancing in Wireless LAN's**", ISSN** 978-1-4799-3357-0, PP 1831 – 1836, IEEE, 2014.

[4] Yean-Fu Wen "Load balancing job assignment for cluster-based cloud computing", ISSN 14517061, PP 199 – 204, IEEE, 2014.

[5] De Mello, M.O.M.C "Load balancing routing for path length and overhead controlling in Wireless Mesh Networks", ISSN 14630778, PP 1-6, IEEE, 2014.

[6] R. Angel Preethima, Margret Johnson, *"Survey on Optimization Techniques for Task Scheduling in Cloud Environment"*, IJARCSSE, Volume 3, Issue 12, December 2013.

[7] Ahmed DheyaaBasha, Irfan Naufal Umar, and Merza Abbas, Member, IACSIT "Mobile Applications as Cloud Computing: Implementation and Challenge", 7865-7564, IEEE, 2013.

[8] Alabbadi, M.M" Cloud computing for education and learning: Education and learning as a service (ELaaS)", ISSN 978-1-4577-1748-2, PP 589 – 594, IEEE,2011.

[9] Cong Wang, Qian Wang, Kui Ren and Wenjing Lou "Ensuring Data Storage Security in Cloud Computing." IEEE 2009.

[10] Farzad Sabahi, "Cloud Computing Security Threats and Responses," IEEE Trans. on Cloud Computing., vol. 11, no. 6, pp. 670 { 684, 2002.}

[11] Gaurav Raj, Dheerendra Singh, Abhay Bansal, "Using Batch Mode Heuristic Priority in Round Robin (PBRR) Scheduling", IEEE, 2012.

[12] Jianfeng Yang, Zhibin Chen "Cloud Computing Research and Security Issues" Vol. 978-1-4244-5392-4/10/$26.00 ©2010 IEEE

[13] Jaber, A.N. "Use of cryptography in cloud computing", ISSN978-1-4799-1506-4,PP 179 – 184,IEEE,2013.