International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435(Online) Volume 5, Issue 10, Oct 2016)

Load Balancing Over Cloud Computing

Sangeeta Devi
M. Tech Scholar, Department of Computer Science Engineering and Information Technology
Bhagat Phool Singh MahilaVisvidyalya, KhanpurKalan, Sonipat

Abstract: - Nowadays load balancing has become a popular platform for scientific applications. Load balancing intends to share many number of hardware resources as like equipment's for record and calculations, and information and knowledge data for scientific researches over cloud computing. Load balancing algorithm is one of the most challenging theoretical issues in the computing field. How we can utilize computing resources very effectively and increase user satisfaction with load balancing system is one of the calculating service provider's main issues. Some intensive researches have been done in the area of load balancing of computing resources. In this research work we have proposed Hybrid Algorithm in computing. In order to achieve our proposal we will execute this work as the following steps. First of all, we will declare some task which we want to execute that store in cloud database. Then, according to the tasks selection, we will select the exegete branch of the function and calculate the deserved evaluation. A hybrid algorithm is combination of FCFS and Priority concept. Reflection of the hybrid algorithm about to select local optimum is representing best performance. Compare to other methods like FCFS, it is found that there is less time consumption in complete execution of submitted tasks and increases the user satisfaction.

Keyword: - Cloud Computing, FCFS, Priority Queue, Hybrid Approach.

I. INTRODUCTION

Job scheduling is a term in which take some jobs and send them to the scheduler to execute them. The issue is generated how efficiently a work can be performed so that less amount of energy gets occupied. New parallel calculating systems, as like the SUN Microsystems D10000, the SRC-6, and the SGI Origin 2000, provide a pool of homogeneous processors, a big common memory, compatible I/O connectivity, and expandable primary and secondary disk storage support.

Every source in the structure of these systems may be scaled independently based on cost and user need. A site which typically runs processor intensive tasks may option for a configuration which is fully populated with CPUs but has a reduced memory to keep the estimated price of these systems is less. In the other way, if the estimated task mix contains a high percentage of I/O and memory intensive works, a big memory configuration may be occupied with high I/O connectivity to network or storage devices [1]. At last, a combined task set may be best serviced by a leveled system configuration. Therefore, given an expected job mix, a "common-everything" side by side system can be configured with the less number of set of resources needed to occupy the dissevered performance. The issue, then, is how to schedule jobs from the actual job stream onto a given machine to occupy the as hoped performance. This is known as the K-devices scheduling problem.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435(Online) Volume 5, Issue 10, Oct 2016)

Consider extending the FCFS-based schemes to maintain record for multiple (K) resources in a particular physical system configuration. High level FCFS task allocation method would pack jobs from the job queue into the system, in order of their arrival, until some system devices was exhausted. In this case, the job allocation scheme is blocked from scheduling [2] further tasks until suitable resources become available for this big task. This potentially results in big segment of resources being under-utilized. The FCFS with backfill probabilistically performs better by skipping over jobs which block while waiting for big portion of a particular device and getting smaller tasks which can develop use of the remaining devices. Still, a single resource becomes exhausted while others remain under-utilized.

The FCFS-based algorithms are restricted in selecting jobs based on their general arrival order. In order for a job allocation scheme to efficiently utilize the independently locatable resources of the K-resource system, it must be free to select any job based on matching all of the tasks resource need with the available system devices. As an example [3], consider the JMS state depicted. The job allocation scheme must map the six jobs in the job queue to a two-resource system with 16 CPUs and 32 Bytes of memory. The CPU and memory requirements of each job are specified [4]. Assume that the order in the job queue displays the sequence of arrival and that each job requires the same amount of execution time t.

II. PREEMPTIVE VS. NON-PREEMPTIVE SCHEDULING

The Scheduling algorithms can be divided into two categories with respect to how they deal with clock interrupts.

1. Non-preemptive Scheduling

A scheduling discipline is non-preemptive if, when a process has been provided to the CPU; the CPU cannot be put away from that process [5].

There are some characteristics of non-preemptive balancing technique

1. In non-preemptive system, short jobs are made to wait by longer jobs but the overall treatment of all processes is fair.
2. In non-preemptive system, response times are more predictable because incoming high priority jobs cannot displace waiting jobs.
3. In non-preemptive scheduling, a scheduler executes jobs in the following two situations.

a) When a process switches from running state to the waiting state.

b) When a process terminates.

2. Preemptive Scheduling

A scheduling discipline is preemptive if, once a process has been given the CPU can take away. The strategy of allowing processes that are logically runnable to be temporarily suspended is called Preemptive Scheduling and it is contrast to the "run to completion" method.

III. FCFS (FIRST COME FIRST SERVE)

FCFS stands for “First Come First Serve”. In this algorithm the first data which reaches to the queue first gets executed first. This algorithm is time consuming and does not perform quite efficiently when there is a case of priority in the segmentation. Other names of this algorithm are:

- First-In-First-Out (FIFO)
- Run-to-Completion
- Run-Until-Done

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435(Online) Volume 5, Issue 10, Oct 2016)

Perhaps, First-Come-First-Served algorithm is the simplest scheduling algorithm is the simplest scheduling algorithm. Processes are dispatched according to their arrival time on the ready queue. Being a non-preemptive discipline, once a process has a CPU, it runs to completion. The FCFS scheduling is fair in the formal sense or human sense of fairness but it is unfair in the sense that long jobs make short jobs wait and unimportant jobs make important jobs wait [6].

FCFS is more predictable than most of other schemes since it offers time. FCFS scheme is not useful in scheduling interactive users because it cannot guarantee good response time. The code for FCFS scheduling is simple to write and understand. One of the major drawbacks of this scheme is that the average time is often quite long.

IV. PRIORITY SCHEDULING ALGORITHM

The shortest-Job-First (SJF) algorithm is a special case of general priority scheduling algorithm. The basic idea is straightforward: each process is assigned a priority, and priority is allowed to run. Equal-Priority processes are scheduled in FCFS order.

An SJF algorithm is simply a priority algorithm where the priority is the inverse of the (predicted) next CPU burst. That is, the longer the CPU burst, the lower the priority and vice versa. Priority can be defined either internally or externally. Internally defined priorities use some measurable quantities or qualities to compute priority of a process.

Examples of Internal priorities are:

- Time limits.
- Memory requirements.
- file requirements,
- For example, number of open files.
- CPU Vs. I/O requirements.

Externally defined priorities are set by criteria that are external to operating system such as

- The importance of process.
- Type or amount of funds being paid for computer use.
- The department sponsoring the work.
- Politics.

Priority scheduling can be either preemptive or non-preemptive

- A preemptive priority algorithm will preemptive the CPU if the priority of the newly arrival process is higher than the priority of the currently running process.
- A non-preemptive priority algorithm will simply put the new process at the head of the ready queue.

A major problem with priority scheduling is indefinite blocking or starvation [6]. A solution to the problem of indefinite blockage of the low-priority process is aging. Aging is a technique of gradually increasing the priority of processes that wait in the system for a long period of time.

V. OBJECTIVE OF RESEARCH WORK

Our objective includes the following:

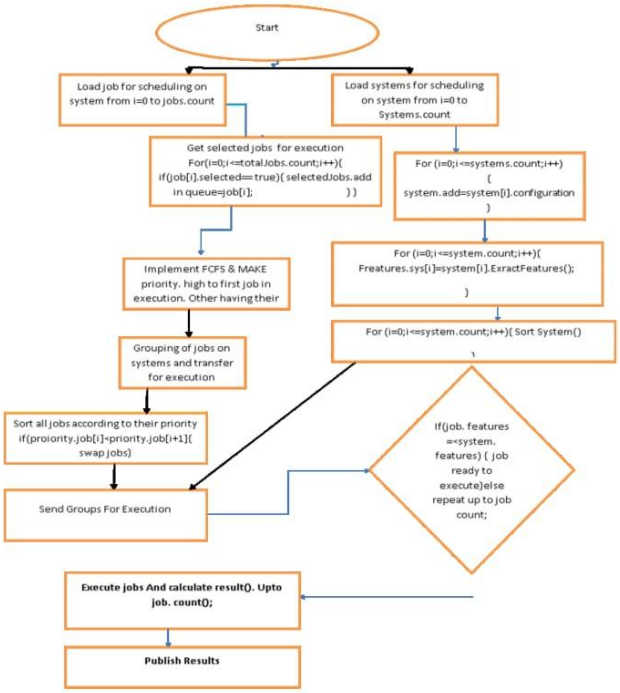
- Our first objective is to design a task and scheduler system
- Our second objective is to design the increasing time algorithm for scheduling
- Our third objective is to implement a hybrid structure of FCFS and priority algorithm.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435(Online) Volume 5, Issue 10, Oct 2016)

IV. RESEARCH METHODOLOGY

Step 1-Data is loaded on Microsoft Excel Sheet that works backend and accessed by our project.
Step 2- Use frontend .net accesses that data for implementation of algorithm.
Step 3- FCFS concept applies on data known as increasing algorithm and get results.
Step 4-For better performance of new algorithm, FCFS concept is used with priority scheduling.
Step 5- First task execute using concept of increasing method and then concept of priority applied.
Step 5- All task exclude first task will implement on basis of priority queue.
Step 6- FCFS and Priority implemented and got results.
Step 7- The result should be better than existing FCFS algorithm.

V. FLOWCHART[7]



VI. RESULT

When execute the code various results are provided. These results are as following:

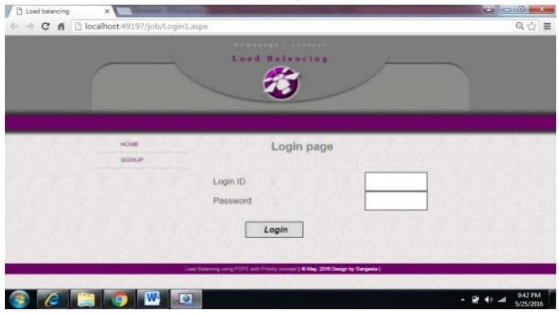


Figure 1: Shows a login page when login on this page that will open the next page this is main page of project

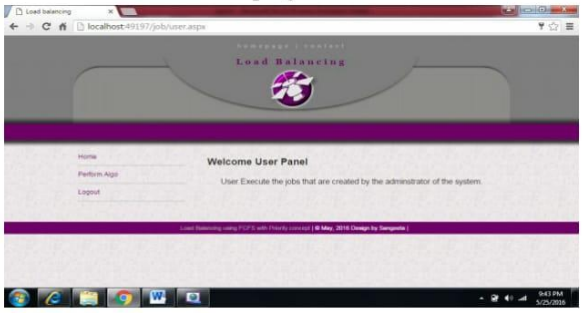


Figure2: shows the main page of the system. At this stage project welcome user for task execution and provide menu to choose existing options.

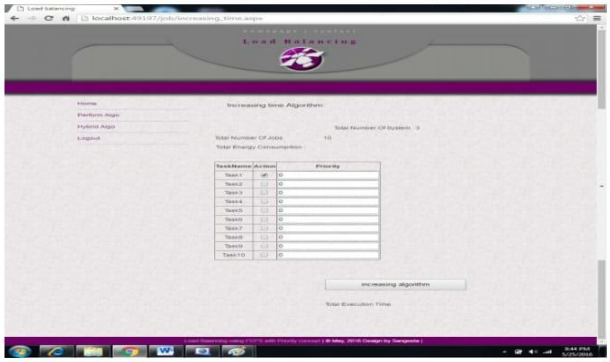


Figure 3: Shows window where task will be selected and priority assigned.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435(Online) Volume 5, Issue 10, Oct 2016)

When click on the 'Perform Algo' button on previous page then this page open and in this page tasks are selected for the execution and for FCFS implementation where provide button with name 'Increasing Algorithm'. In this page task can easily select and assign priority to each task. There is need to assign priority to each task otherwise it takes by default value of priority that is zero.

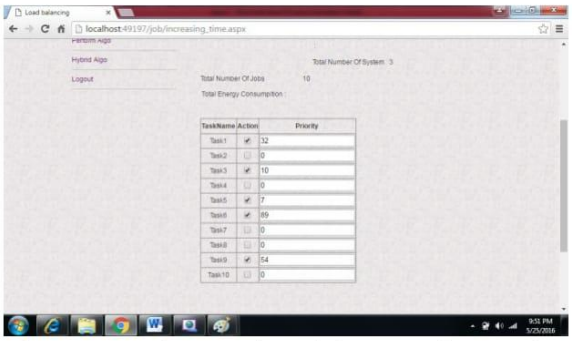


Figure 4: Tasks are selected for execution. Assign priority to each task.

In this page tasks are selected for the execution on system. Now choose Task1, Task3, Task5, Task6, and Task9 with priority 32, 10, 7, 89, and 54 respectively. Now proceed for FCFS execution.

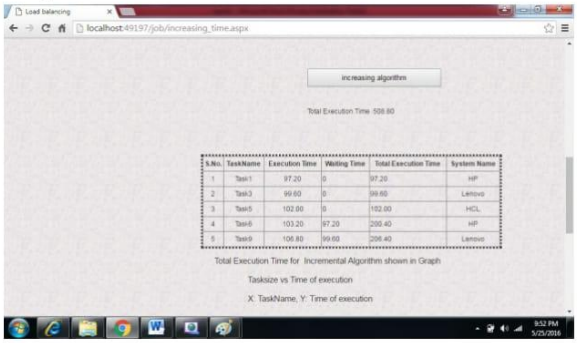


Figure 5 Table shows result of FCFS algorithm.

In this table each task execute in a manner of 'first come first serve'. Column shows time of execution with waiting time for each task on available systems. Waiting time for first three tasks is null because all three systems are free to execute. Total time consume for each task is 508.80 millisecond.

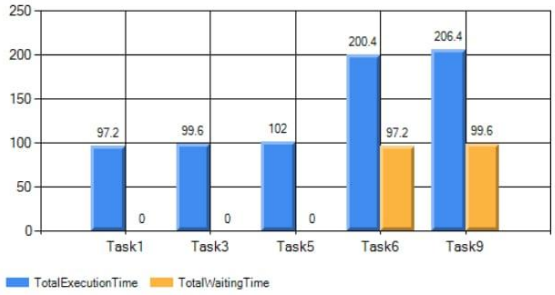


Fig 6 Bar Graph display both execution and waiting time with different colors for each task.

Blue: This color represents execution time of task in a system.

Orange: This color represents waiting time of task in a system.

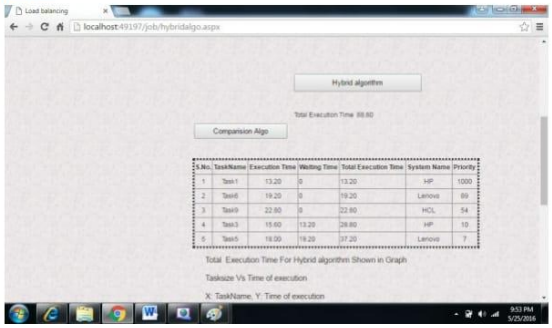


Figure 7: Table generated by execution of Hybrid Algorithm.

First task will execute on FCFS basis but remaining tasks are performed on Priority basis. Total execution time is 88.80 milliseconds. This time is very less as compared to FCFS algorithm.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435(Online) Volume 5, Issue 10, Oct 2016)

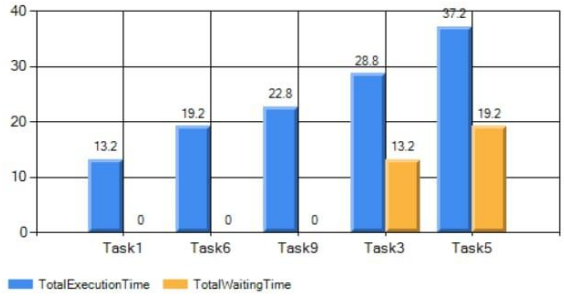


Fig 8 Graph represents execution time and waiting time for each task.

Blue: This color represents execution time of task in a system.

Orange: This color represents waiting time of task in a system.

2 Comparisons between Existing and Proposed Algorithm

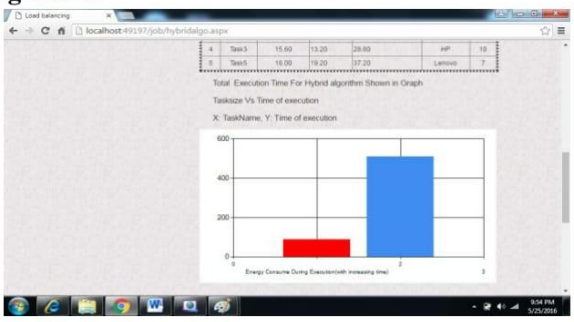


Figure 9: Comparative graph of Hybrid & FCFS algorithm

Red Color: This represents execution time of Hybrid Algorithm

Blue Color: This represents execution time of FCFS Algorithm

Table 1: Comparative Table for 3 Task Executions Time

S. No.	Task Name	Existing Technique (FCFS)			Proposed Technique (FCFS + Priority)		
		Execution Time	Waiting Time	Total Execution Time	Execution Time	Waiting Time	Total Execution Time
1.	Task 1	97.20	0	97.20	13.20	0	13.20
2.	Task 2	98.40	0	98.40	14.40	0	14.40
3.	Task 3	99.60	0	99.60	15.60	0	15.60

Table 2: Comparative Table for 6 Task Executions Time

S. No.	Task Name	Existing Technique (FCFS)			Proposed Technique (FCFS + Priority)		
		Execution Time	Waiting Time	Total Execution Time	Execution Time	Waiting Time	Total Execution Time
1.	Task 1	97.20	0	97.20	13.20	0	13.20
2.	Task 2	98.40	0	98.40	14.40	0	14.40
3.	Task 3	99.60	0	99.60	15.60	18.00	33.60
4.	Task 4	100.80	97.20	198.00	16.80	14.40	31.20
5.	Task 5	102.00	98.40	200.40	18.00	0	18.00
6.	Task 6	103.20	99.60	202.80	19.20	13.20	32.40

Table 3: Comparative Table for 10 Task Executions Time

S. No.	Task Name	Existing Technique (FCFS)			Proposed Technique (FCFS + Priority)		
		Execution Time	Waiting Time	Total Execution Time	Execution Time	Waiting Time	Total Execution Time
1.	Task1	97.20	0	97.20	13.20	0	13.20
2.	Task2	98.40	0	98.40	14.40	13.20	27.60
3.	Task3	99.60	0	99.60	15.60	46.80	62.40
4.	Task4	100.80	97.20	198.00	16.80	48.00	64.80
5.	Task5	102.00	98.40	200.40	18.00	0	18.00
6.	Task6	103.20	99.60	202.80	19.20	18.00	37.20
7.	Task7	104.40	198.00	302.40	20.40	27.60	48.00
8.	Task8	105.60	200.40	306.00	21.60	37.20	58.80
9.	Task9	106.80	202.80	309.60	22.80	0	22.80
10.	Task10	108.0	302.40	410.40	24.00	22.80	46.80

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435(Online) Volume 5, Issue 10, Oct 2016)

VII. CONCLUSION

It is expected that the time for task execution will be reduced if there will be implementing the FCFS along with the priority queue concept. Performance of systems is improved with the hybrid production using FCFS and priority queue techniques. In a common way, Configuration of system about hardware is already allotted but in this research work we develop three systems as per our need and execute the tasks on these systems. FCFS and priority concept is failure to provide good throughput in case of separately execution. But a hybrid method provides us the fast execution and enhances the throughput performance.

References

[1] Masoud Nosrati Ronak Karimi Mehdi Hariri, "Task Scheduling Algorithms Introduction" World Applied Programming, Vol. (2), Issue (6), June 2012

[2] Pinky Rosemarry, Ravinder Singh, Payal Singhal and Dilip Sisodia,"Grouping Based Job Scheduling Algorithm Using Priority Queue and Hybrid Algorithm In Grid Computing" International Journal of Grid Computing & Applications (IJGCA) Vol.3, No.4, December 2012

[3] Neeraj Kumar, Nirvikar,"Performance Improvement Using CPU Scheduling Algorithm-SRT", International Journal of Emerging Trends & Technology in Computer Science, Volume 2, Issue 2, March – April 2013

[4] Arezou Mohammadi and Selim G. Akl,"Scheduling Algorithms for Real-Time Systems", School of Computing, Queen's University, Canada

[5] Sukumar Babu Bandarupalli1, Neelima Priyanka Nutulapati, Prof. Dr. P. Suresh Varma, "A Novel CPU Scheduling Algorithm- Preemptive & Non-Preemptive" International Journal of Modern Engineering Research, Vol.2, Issue.6, Nov-Dec. 2012

[6] Deepali Maste, Leena Ragha and Nilesh Marathe,"Intelligent Dynamic Time Quantum Allocation in MLFQ Scheduling"International Journal of Information and Computation Technology, Volume 3, Number 4 2013

[7] Monika and Neelam "Job Scheduling using FCFC and Priority Queue in System" International Journal of Electrical Electronics & Computer Science Engineering Volume 2, Issue 3 (June, 2015) | E-ISSN : 2348-2273 | P-ISSN : 2454-1222, pp6-9

.