TIME SCHEDULING ALGORITHMS FOR LOAD BALANCING THROUGH OPTIMIZED HYBRID APPROACH

[1]DEVANSHU DHAR SRIVASTAVA, [2]Rekha

[1]M. Tech. Scholar, [2]Assistant Professor
Computer Science Engineering
CBS Group of Institutions

***Abstract*: It is expected that if the FCFS is used in conjunction with the priority queue concept, the job's execution time would be reduced. The performance of the system while utilising hybrid production techniques such as FCFS and priority queues is improved. Despite the fact that the system configuration has already been determined, in this research we are building three systems to meet our specific requirements and carrying out operations on those systems. Independently of one another, FCFS and the priority concept both fail to provide good performance when used together. A hybrid strategy, on the other hand, enables us to execute fast while simultaneously improving performance. Any concept that could be implemented on the basis of priority in the future will be unavailable since there is no function accessible to allow the system to work fast if the top priority is a job but it consumes more execution time when compared to the other tasks in the priority queue. The most serious flaw in this project is that it produces a tiny system in which the addition of a new system has a negative impact on the performance of all other systems. Priority must be specified at the time of task selection, but it automatically assigns priority without the need for an explicit command based on the system settings that are currently in effect. Anyone will be able to enhance the performance of systems in the future by implementing the priority queue concept in accordance with their setup.**

***Keywords*: Scheduling Algorithms, Load Balancing, Optimized Hybrid Approach**

I. INTRODUCTION

Job scheduling is a term that refers to the process of taking specific duties and sending them to the planner. The question today is how efficiently a job can be completed in order to minimize energy use. the required support for growing main and secondary disc storage, share of, and a connection, among other features. Depending on the cost and user needs, each resource in these systems may be scaled independently of the other resources. A site that performs CPU-intensive activities on a regular basis may choose that integrated capacity in order down. you expect a substantial percentage of I/O and storage activities in your job mix, you may want to consider purchasing a connection to your instead., it is desirable to retain a diverse task set via the use of a well-balanced system architecture. As a result, given the, parallel "shared-everything" may built smallest amount necessary specified. It becomes a matter of determining allocate workflow to certain in order required problem with the planning method.

If you have a large number of (K) resources, you should consider extending the FCFS-based schemes in the physical system configuration. The pure FCFS work allocation technique would load jobs into the system in the order in which they were received from the work queue until specific system resources (such as CPUs, memory, and disc space) were exhausted. In such a situation, the system for the allocation of work will be suspended until sufficient resources are allocated to this massive undertaking. This has the potential to result in the underutilization of vast amounts of resources. The backfilling material By overwriting activities that are stalled projects that take, FCFS is likely to be more successful. Some resources, on the other hand, have been exhausted while others have been underutilised.

Where Scheduling Fit in Process

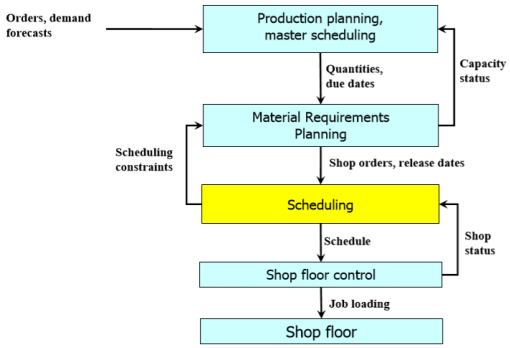

Fig. 1: Where Scheduling Fits in Process

General Goals

1. Fairness: Fairness in all situations is essential. A scheduler ensures that every process has its own fair portion of the CPU and that no task may be postponed indefinitely. Note that it is not fair to provide comparable or equal time. Consider safety control and payroll for a nuclear power station.
2. Policy Enforcement: The scheduler must ensure that the policies of the system are implemented. For example, if the local safety policy is, the security control procedures must be able to operate whenever they want even while payroll operations are delayed.
3. Efficiency: scheduler should keep the system (or CPU in particular), if feasible, occupied 100% of the time. If the CPU and all I/O devices are continuously operating, more work is done per second than when certain components are idle.
4. Response Time: A scheduler should reduce the interactive user response time.
5. Setting: A scheduler should reduce the time chunk that consumers have to wait for an output.
6. Performance: the scheduler should maximize the number of tasks per unit time handled.

Some of these objectives are conflicting, a little thinking will reveal. Method supports tasks harms (KLEINROCK), may be demonstrated. After all, the quantity of CPU time available is limited

II. EXPERIMENTAL SETUP

PREEMPTIVE VS. NON-PREEMPTIVE SCHEDULING
may split groups according how clock breaks are handled.

Non-preemptive Scheduling
A scheduling discipline is not preemptive if the CPU can't remove the CPU from that process after a process is delivered.
Below are some features of non-preventive planning

1. Short tasks are done in a non-preemptive system to wait for larger jobs but the overall treatment of all processes is fair.
2. Response times are more predictable in non-preemptive systems since high priority tasks cannot be relocated.
3. A scheduler does work in non-preemptive scheduling in the following two circumstances.
(a) If a process changes from running to waiting.
b) When the procedure ends.

Preemptive Scheduling
A scheduling discipline is preemptive if the CPU can take away once a process has been provided. Preemptive scheduling is the technique to permit processes that are logically stopped momentarily and contrasts with the "run to completion".

SCHEDULING ALGORITHMS

Addresses issue determining tasks are to be assigned for the CPU in a ready queue.

FCFS (First Come First Serve)

FCFS represents "First Come First Serve." The first data reaching the queue is executed first in this method. This method takes time and does not work very effectively when the segmentation takes precedence. This algorithm has several names:

• First in first out (FIFO)

• Run to finish

• Run-Until-Done

Maybe, the First-Come-First-Served method is the simplest scheduling algorithm. Processes are sent to the ready queue according to their arrival time. As a discipline that is non-preemptive, once a process has a CPU, it is over. The FCFS scheduling is formal or fair, but it is unjust in that lengthy jobs wait for short tasks and essential work is waiting for vital jobs.

FCFS is predictable since it provides time than most other systems. In scheduling interactive users, FCFS is not helpful since it cannot ensure a decent response time. The FCFS programming code is easy to create and comprehend. One of the biggest disadvantages is frequently lengthy.

ALGORITHM STEPS FOR FCFS

FCFS represents "First Come First Serve." The first data that hits the queue are initially accepted in this method. The disadvantage of this method takes time and does not perform very well when priority values are identified in tiny segments. We may call this algorithm various names:

• First in first out • First out (FIFO)

• Run-to-Completion

• Run-Until-Done

Hope, method direct algorithm for A ready queue performs the job when the processes are sent. As an unpreemptive discipline, it runs until its completion once a taskallot to the system. This scheduling method is acceptable in general or humane sense, however it is inappropriate in the sense that lengthy chores wait short and needless activities.

highly compared with methods happens to provide time. For interactive users, the FCFS method is not helpful since there is an alternative to ensure adequate output time. The FCFS programming code is easy to create and comprehend. The most important thing about this method is that the average time is frequently very lengthy. An algorithm is provided to comprehend the idea of FCFS.

Step 1 - Launch

Phase 2 - Choose the tasks from the selection choice portion of the data source in this step.

Step 3 - Access all the work execution tasks from the database using three existing systems called 'HP,' 'Lenovo,' 'HCL.'

Phase 4 - Each job is carried out in this step on the basis of the 'First Come First Come' principle.

Step 5 – The first three tasks are waiting for zero. The fourth job is to wait till the first task of System 1 'HP' is carried out. Therefore, waiting time for the fourth job is the same as running time for the first task.

Step 6 – Step 5 is repeated until all tasks are performed.

Step 7 - You will receive the outcome in this step.

Step 8 – Finish

III. IMPLEMENTAION

HYBRID ALGORITHM

The method proposed is the mix of FCFS (first come first serves) with priority scheduling. To eliminate the issue of FCFS scheduling, introduce the priority scheduling notion. The algorithm illustrates the idea of the algorithm presented.

In this flow chart, no different stages are utilized, which are described here:

Step 1 – Launch

Phase 2 – In this step pick tasks from the data source in the selection option section and prioritize each chosen task.

Step 3 – Access all database tasks to execute jobs using three existing systems called 'HP,' 'Lenovo,' 'HCL.'

Phase 4 - Each job is executed in this step based on 'Priority' and 'First Come First Come' concepts.

Step 5 – Project replaces task priority at position 1 with value 1000 and executes using the FCFS concept. During this job, all priority is sorted in decreasing order. Then perform all tasks by prioritizing the idea exclude task first. At the first location, perform the job each time first.

Step 6- The time to wait for the first three activities is zero. The job number four is to wait for the first task of system 1 'HP.' Therefore, waiting time for the fourth job is the same as running time for the first task.

Step 6 – Repeat Step 5 until all tasks run.

Step 7 – You will receive the outcome in this step.

Step 8 – Finish

FCFS and priority schedule algorithm Flowchart is shown here:

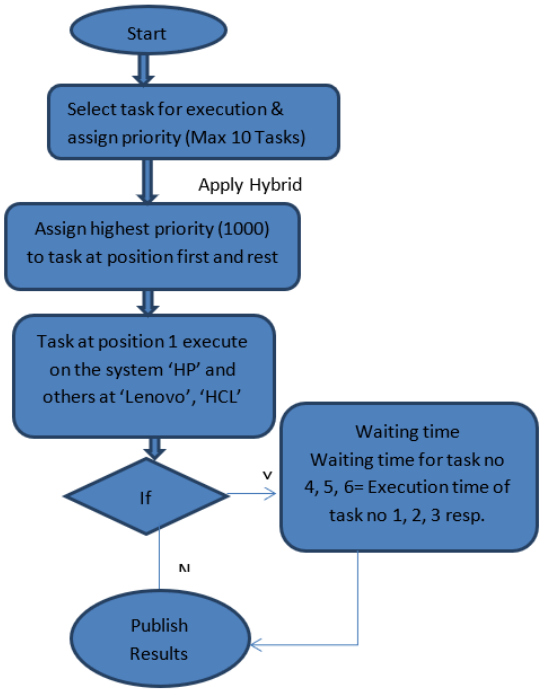

Fig. 2: FCFS + PRIORITY scheduling

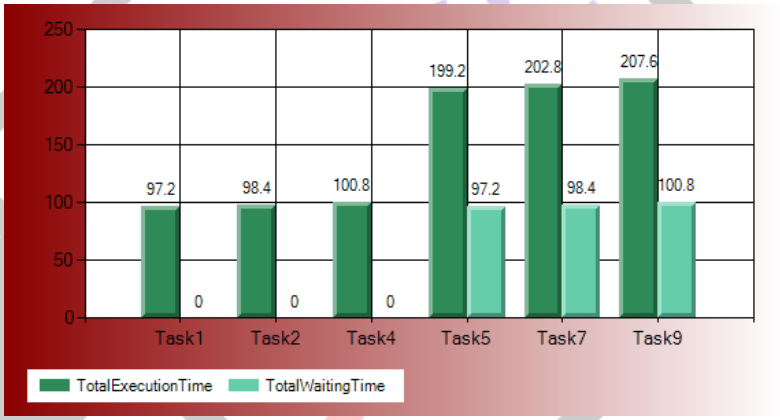

Fig. 3: Bar Graph displayboth execution and waiting time with different colors for each task.

Green: This hue indicates a system execution time.
Light Green: This hue indicates a system waiting time.

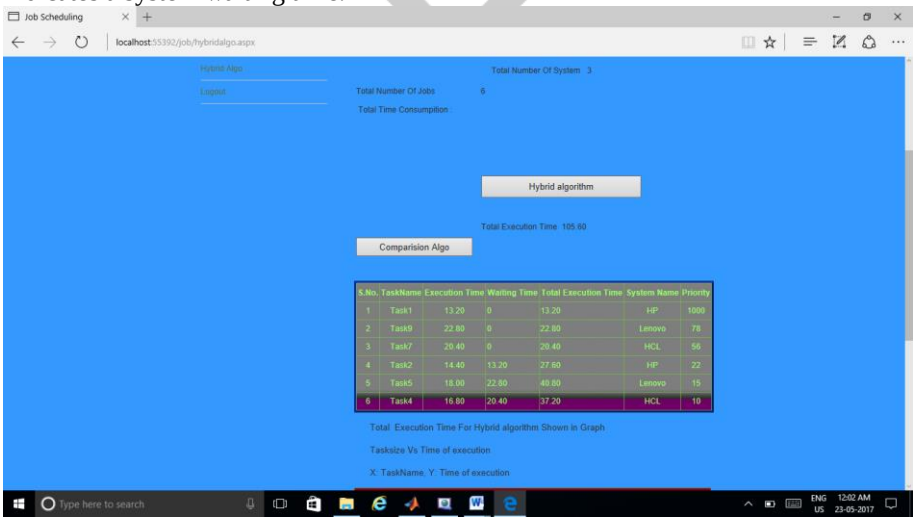

Fig. 4: Table generated by execution of Hybrid Algorithm.

First job will be done on FCFS basis, while further tasks will be performed on the basis of priority. The total time to do is 105.60 milliseconds. This time is far lower than the FCFS method.

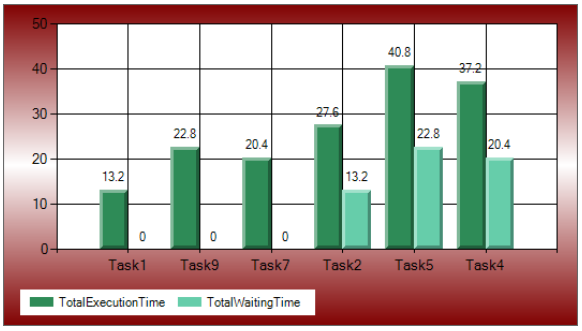

Fig. 5: Graph represents execution time and waiting time for each task.

Green: This hue indicates a system execution time.
Light Green: This hue indicates a system waiting time.

COMPARISON BETWEEN EXISTING AND HYBRID ALGORITHM

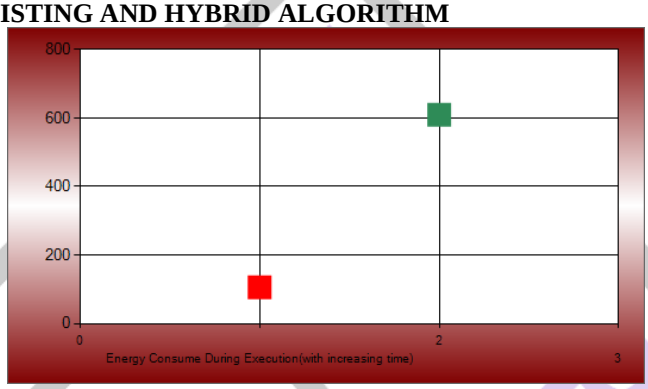

Fig. 6: Comparative graph of Hybrid and FCFS algorithm

Red Color: This indicates hybrid algorithm execution time, Blue Color: This indicates FCFS algorithm execution time.

IV. CONCLUSION AND FUTURE SCOPE

It is expected that if the FCFS is used in conjunction with the priority queue concept, the job's execution time would be reduced. The performance of the system while utilizing hybrid production techniques such as FCFS and priority queues is improved. Despite the fact that the system configuration has already been determined, in this research we are building three systems to meet our specific requirements and carrying out operations on those systems. While used together, FCFS and the priority concept are unable to offer good performance when in use. Any concept that could be implemented on the basis of priority in the future will be unavailable since there is no function accessible to allow the system to work fast if the top priority is a job but it consumes more execution time when compared to the other tasks in the priority queue. The most serious flaw in this project is that it produces a tiny system in which the addition of a new system has a negative impact on the performance of all other systems. Priority must be specified at the time of task selection, but it automatically assigns priority without the need for an explicit command based on the system settings that are currently in effect. Anyone will be able to enhance the performance of systems in the future by implementing the priority queue concept in accordance with their setup.

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] Jia Xu, David Lorge Parnas,"Priority Scheduling Versus Pre-Run-Time Scheduling" The International Journal of Time-Critical Computing Systems, 18, July 2000
[8] Siddharth Tyagi, Sudheer Choudhary, Akshant Poonia, “Enhanced Priority Scheduling Algorithm” International Journal of Emerging Technology and Advanced Engineering, Volume 2, Issue 10, October 2012
[9] Yun Wang and Manas Saksena, “Scheduling Fixed-Priority Tasks with Preemption Threshold" RTCSA’99, Hong Kong December 2005
[10] Prerna Ajmani and Manoj Sethi, “Proposed Fuzzy CPU Scheduling Algorithm (PFCS) for Real Time Operating Systems" International Journal of Information Technology, Vol. 5 No. 2, July-December, 2013
[11] Helen D. Karatza, “Parallel Job Scheduling in Homogeneous Distributed Systems”, Department of Informatics Aristotle University of Thessaloniki 54124 Thessaloniki, Greece
[12] Shalmali Ambike, Dipti Bhansali, Jaee Kshirsagar, Juhi Bansiwal, “An Optimistic Differentiated Job Scheduling System for Cloud Computing”, International Journal of Engineering Research and Applications (IJERA) Vol. 2, Issue 2, Mar-Apr 2012
[13] Pinky Rosemarry, Payal Singhal, Ravinder Singh, “A Study of Various Job & Resource Scheduling Algorithms in Grid Computing”, International Journal of Computer Science and Information Technologies, Vol. 3 (6), 2012
[14] Eitan Frachtenberg, Dror G. Feitelson, “Pitfalls in Parallel Job Scheduling Evaluation”, Modeling, Algorithms, and Informatics Group Los Alamos National Laboratory
[15] Yarong Chen, Zailin Guan, Xinyu Shao, “A comparative analysis of job scheduling algorithm” Management Science and Industrial Engineering (MSIE), 2011 International Conference on IEEE