International Journal of Research in Advent Technology, Vol.2, No.12, December2014
E-ISSN: 2321-9637

Improvement of Queue Management for Real Time Task in Operating System

Rohan R. Kabugade[1], S. S Dhotre[2]

M.Tech Computer Department, Bharati Vidyapeeth University College of Engineering Pune, India[1]

Associate Professor Computer Department, Bharati Vidyapeeth University College of Engineering Pune, India[2]

rohan.7991@gmail.com[1]

sdhotre@bvucoep.edu.in[2]

Abstract- The main objective of this paper is effective management of Queue. Run queue contain all runnable processes. Each and every runnable process has exactly one run queue and there is only one run queue per-processor. In this paper, for managing queue or improvement of queue management we use time slice and time interval and also assign priority to each process dynamically. In this paper, priority base scheduling are used for fulfil our main objective. The higher priority process run first then runs the lower priority process. Time slots are also assigned to each process for calculating execution time of particular process. The advantage of dynamic priority is that they currently run basic priority process and then enables scheduler to boost or reduce priority dynamically. Due to this schedulers achieve his objective.

Keywords — kernel, real-time OS, run queue, schedule, runnable, priority.

1. INTRODUCTION

One of the most important components of computer resources is CPU. In Operating System CPU Scheduling is one of the fundamental concepts. Scheduling is the technique used for controlling the order of job which is to be performed by a CPU of a computer. Most CPU scheduling algorithms concentrate on increase in CPU utilization and throughput and reducing turnaround time, response time, waiting time, etc. Scheduling is also a fundamental function of operating system. Before use almost all Computer devices and resources are scheduled. Thus its scheduling is central to O.S designs. Scheduling is the strategy by which the system decides which task should be executed at any given time. When CPU become idle then operating system select at least on process for execution.

- ***Priority Scheduling***: - The process with high priority is allocated to CPU first in this scheduling.
- ***Round Robin Scheduling***: - RR scheduling is used in timesharing systems. It is same as FCFS scheduling with pre-emption is added to switch between processes.
- ***Context Switch***: A context switch is process of maintaining state of processes which are a pre-empted, so that execution of process is restarted from the beginning. Context switching is consumption of time and memory that leads to the increase in the overhead of scheduler, to optimize only these switches is main goal of CPU scheduling algorithms [6].

1.1. Scheduling criteria

CPU Utilization – They measure the CPU busy time. The proper use of CPU then the scientific developed technique is used. Utilization of CPU can range from 0 to 100 percent.

Throughput - Throughput is the total time required to execute the number of processes per time period. It is the number of processes completed in how much per time period. Two processes per hour is time rate may be required for long processes. For short transaction throughput might be less than long processes.

Waiting time**:** It is the time spends in waiting by the process in the ready queue. CPU executes only one process at a time. The rest of processes wait for the CPU.

Turnaround time - It is the total time taken by the process to execute. The turnaround time is the time between starting and execution of the process.

Response time - Response time is the time until the first response is produce from the submission of a request.

Context Switch- A Context switch is the time taken by the processes for switching from one process to another, so the resumption of process is done in [3].

A priority-based scheduling is a common type of scheduling algorithm. The thought is to ranking of processes based on their significance and

processor time. First of all run the higher priority processes then run the lower priority processes, the new idea comes and provides dynamic priority-based scheduling. The system is pre-emptive, when a process comes in the task running state, the kernel checks the priority of the currently executing process whether its priority is higher than that then the scheduler is stop the currently executing process and run the newly coming process. The pre-emption process is done, when time slice of a process reaches zero.

2. RELATED WORK

This section gives the overview of the research work carried out related to the Queue Management. This overview mainly focuses on the Improvement of queue management and Improvement of process analysis.

The scheduler is meant to allocate its resources to all applications in designing the Operating System (OS). The author in [2] has introduced the scheduling techniques used by scheduler is O(1) .The goal of design for O(1) is to provide fair CPU resource allotment among executing tasks without debasing the whole performance. To avoid the starvation the good fairness between processes is necessary in distributing CPU resource among tasks. The benchmark is very important concept in operating system for calculating or measuring system performance significantly, also by using this concept they measure the fairness and interactivity performance of processes. The design goals of CFS evaluate scientifically by experimental assessment. By comparing two type of scheduler, they provide a meaningful representation of results. So the experience indicated that the CFS does achieve its design goals.

The process scheduler is an important part of the kernel because running processes is the point of using the computer in the first place. The new process scheduler, however, comes very close to comforting all parties and providing an most advantageous solution for all case with perfect scalability. A completely new scheduler that's commonly referred to as the O(1) scheduler introduced in Linux kernel 2.6. In constant time the scheduler can perform the scheduling of task. How a task is executed on single CPU described by M A Wei-feng and WANG jai-hai in [5, 8]. Data structures run queues, priority array and process are also mentioned in this article.

Wang Chi Zhou Huaibei, Ma Chao Chen Nian. in [1] has proposed an approach to Modify O(1) Algorithm in Scheduling for Real-Time Tasks. In this author presents a modified algorithm named MOFRT based on the Linux kernel scheduler. The real-time system group of people have proposed and studied many new and powerful scheduling algorithms. On the other hand, most of these new algorithms very hard to implement, so supporting to these new algorithm is very difficult and not suitable for most of operating systems. A flexible scheduling framework is good solution for this type of problem. In this they improve the system performance. The number of high powered and faired scheduling algorithms is used to solve this type of problem. For making a perfect balance between quick response and fairness by using fast prototyping scheduling algorithms is the main goal of this architecture they present is that. The over all system performance like steadiness, method of calculation, time slice, time interval and priority.

Dr.Pardeep Kumar Mittal and Raman in [6] has proposed an approach that the performance of CPU decreases due to static quantum taken in RR algorithm. In this author discuss selection of time quantum and propose a new CPU scheduling algorithm for timeshared systems and is called as Efficient Dynamic Round Robin algorithm. The objective of author in this article is to make a change in round robin CPU scheduling algorithm so that the performance of CPU can be improved. Efficient Dynamic Round Robin algorithm also includes advantages of round robin CPU scheduling algorithm of less chance of undernourishment. Round robin CPU scheduling algorithm has high context switch rates large response time, large waiting time, large turnaround time and less throughput, high context switch rates these drawback can be improved with new proposed CPU scheduling algorithm. Author analyze the scheduling criteria according to their number in round robin CPU scheduling algorithm, SRBRR (Shortest Remaining Burst Round Robin), ISRBRR (Improved Shortest Remaining Burst Round Robin) and new proposed EDRR CPU scheduling algorithm has been done.

The author in [7] has discussed about Scheduling techniques and algorithms for multiprogramming in a real-time surroundings. In this authors explain that the problem of multi-program scheduling on a single processor is studied from the viewpoint of the individuality peculiar to the program functions that need guaranteed subordinate. It is shown that an most favourable fixed priority scheduler possesses an upper bound to processor action which may be as low for large task sets. It is also shown that complete processor use can be achieved by dynamically assigning priorities on the basis of their current deadlines. They also discussed about the combination of these two scheduling techniques.

Radhe Shyam and Sunil Kumar Nandal [3] use some of the popular CPU scheduling algorithms are FCFS, SJF, Priority based Scheduling and Round Robin scheduling (RR). The main goal of CPU scheduling algorithms are increasing CPU consumption and throughput and reducing rotate time, response time, waiting time, and number of context switching, etc. In this article author propose or design

a new Round Robin Scheduling. This Scheduling gives better result compare to Round Robin (RR), Improving Round Robin (IRR), Enhanced Round Robin (ERR), Self Adjustment Round Robin (SARR), FCFS and some other scheduling algorithm.

A very important subsystem in operating system is CPU scheduler and which affects on fairness and interactivity. Development of Linux kernel is comparatively fast-paced. In many CPU schedulers have been designed by kernel hackers and researchers. It is necessary to accurately analyse and evaluate different characteristics among these schedulers, so as to design and appreciate better CPU schedulers for various applications. However, a straight-forward method is used to compare and analyse these CPU schedulers precisely, by researchers. All authors [4] have systematically analysed and determine interactivity, fairness and multi-processors performance by micro combination of algorithms used in this article. In Linux kernel-2.6.29, all these schedulers have been ported in a single scheduler framework. Investigational results show that there are small differences in production and real applications and remarkable differences in fairness and interactivity. The author in this article also analysed the impact of implementations of schedulers on fairness and interactivity of applications. They also discussed about the challenging estimation of application resource requirements in different environments. They also present some new challenging ideas for developing future CPU schedulers.

In [8] author explains that the process scheduler decides which process runs, when, and for how long. The finite resource of processor time is divides between the runnable processes on a system by the process scheduler. In Linux, the scheduler is the basic concept for executing a multiple task of operating system. For multiple processes are executing concurrently, the scheduler is responsible for best utilizing the system performance and giving users the feeling by deciding which process runs next. A processes should always be running, in this assume that there are runnable process for best utilize processor time. If number of processor is less than the runnable processes in system, at a given moment some processes will not be running, so that these processes are in waiting state. Scheduler must make a fundamental decision that deciding which process run next from a given set of runnable processes.

3. PROPOSED SYSTEM

Many operating systems (older versions of Linux) use loop over each task to recalculate each task's time slice when they have all reached zero. The incipient Linux schedule transmutes this to alleviate the desideratum for a recalculate loop. An active array and an expired array are two priority arrays maintains for each processor. All the tasks of active array contains in the associated run queue that have time slice left. The all the tasks contain in expired array are associated runqueue that have exhausted their time slice. Time slice is recalculated before it is moved to the expired array when each task's time slice reaches zero. Recalculating all the time slices is then as simple as just switching the active and expired arrays.

1) **Processes**- It is a small task in execution. Processes are given as an input to the system.
2) **Run Queue**: Run queue contains all runnable process. This run queue contains the two arrays:
   - **Active Array**: All the tasks in the associated run queue that have time slice left are contained in the active array
   - **Expire Array**: The expired array contains all the tasks in the associated run queue that have exhausted their time slice.
3) **Check Priority**: Each priority array contains one queue of runnable processors per priority level. For discovering the highest priority runnable task in the system, we use a priority bitmap, which contained in priority array. This will improve the performance of the system by different scheduling techniques.
4) **Wait**: It consists of the processes in waiting state.
5) **Execution**: It consists of the processes in execution state.

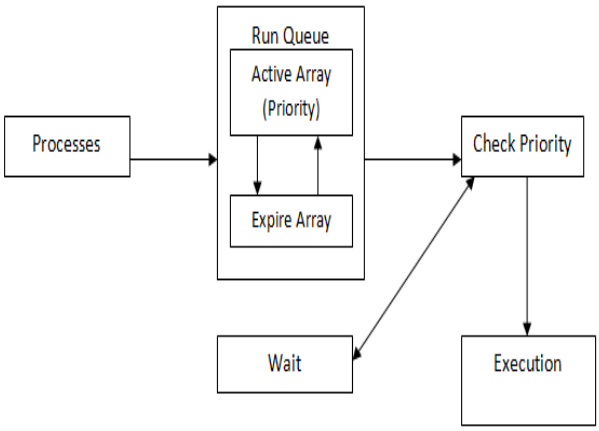

Fig. 1. System Architecture

.

4. RESULTS AND DISCUSSION

For checking result of system we take array of some processes like {0, 1, 2, ……..,1000}. First of all for clear and better result we use process array p=5, p=25 and p=50 i.e. {0,1,2,3,4 } , {0,1,2,……,24} and {0,1,2,…….. ,49}. The Results are shown in Fig 2, Fig 3 and Fig 4. In all graphs, We can see the execution time of each Task. In graph, X-axis represent number of processes and Y-axis represent time in millisecond.

We calculate average time of execution of processes is as follows:

$$\text{Average time} = \frac{\sum \text{Time of Interval}}{\text{Number of Processes}}$$

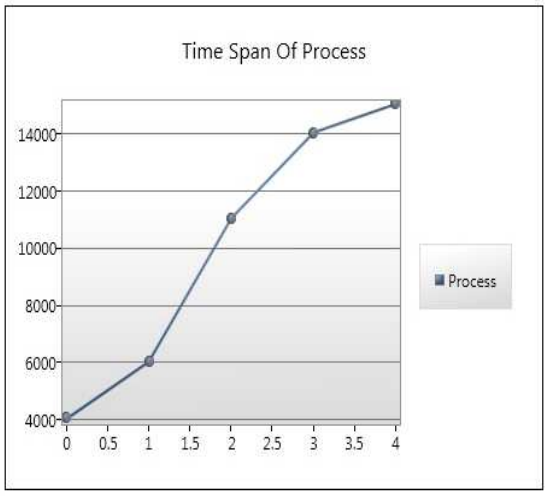

Fig.2. Time for processes p=5.

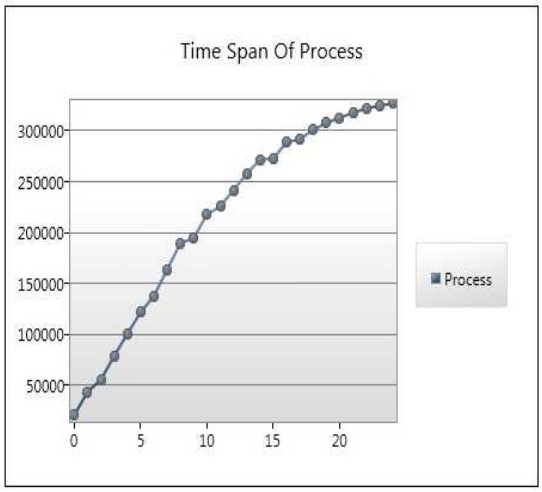

Fig.3. Time for processes p=25

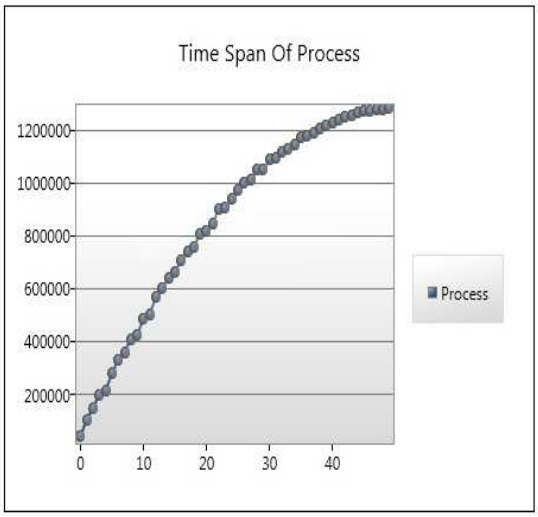

Fig4. Time for processes p=50

5. CONCLUSIONS

This paper presents a scheduling algorithm based on priority. In this algorithm the time interval is assigned to each process in the queue. So the time of execution of each process calculated clearly and also required less time for execution. From above graphs we also conclude that the average time of processes less and due to assigning time interval to each process, this process execute in that time interval. The time slice is also assigned to the processes for decreasing starvation.

REFERENCES

[1] Wang Chi Zhou Huaibei, Ma Chao Chen Nian. "A Modified O(1) Scheduling Algorithm for Real-Time Tasks". In: Proc. of the IEEE, 2006.

[2] Wong C.S., Tan I.K.T., Kumari R.D., Lam J.W., Fun W "Fairness and Interactive Performance of O(1) and CFS Linux Kernel Scheduler". IEEE 2008

[3] Radhe Shyam , Sunil Kumar Nandal "Improved Mean Round Robin with Shortest Job First Scheduling". IJARCSSE, © 2014.

[4] Shen Wang, Yu Chen, Wei Jiang, Peng Li, Ting Dai and Yan Cui "Fairness and Interactivity of Three CPU Schedulers in Linux".2009 15th IEEE international Conference on Embedded and Real-Time Computing Systems and Applications

[5] MA Wei-feng, WANG Jai-hai "Analysis of the Linux 2.6 kernel scheduler". 2010 International Conference on Computer Design and Application (ICCDA 2010).

[6] Dr.Pardeep Kumar Mittal, Raman "An Efficient Dynamic Round Robin CPU Scheduling Algorithm". IJARCSSE, © 2014.

[7] Liu CL, Layland JW. Scheduling algorithms for multiprogramming in a hard real-time environment. Journal of the ACM, 1973.

[8] Robert Love Linux Kernel Development Third Edition Jan 2006.

(A.1)