Journal of Telematics and Informatics (JTI)
Vol.1, No.1, March 2013, pp. 14~19
ISSN: 2303-3703

Design of Modified HRRN Scheduling Algorithm for priority systems Using Hybrid Priority scheme

P. Surendra Varma
Department of Computer Science and Engineering,
NRI Institute of Technology, Agiripalli(M), Krishna District, India.
e-mail: surendravarma008@gmail.com

Abstract

The basic aim of this paper is to design a scheduling algorithm which is suitable for priority systems and it should not suffer with starvation or indefinite postponement. Highest Response Ratio Next (HRRN) scheduling is a non-preemptive discipline, in which the priority of each job is dependent on its estimated run time, and also the amount of time it has spent waiting. Jobs gain higher priority the longer they wait, which prevents indefinite postponement (process starvation). In fact, the jobs that have spent a long time waiting compete against those estimated to have short run times. HRRN prevents indefinite postponements but does not suitable for priority systems. So, modifying HRRN in such a way that it will be suitable for priority based systems.

Keywords: HRRN, Starvation, Priority

1. Introduction

A **process** is an instance of a computer program that is being executed. The processes waiting to be assigned to a processor are put in a queue called *ready queue.* The time for which a process holds the CPU is known as *burst time*. *Arrival Time* is the time at which a process arrives at the ready queue. The interval from the time of submission of a process to the time of completion is the turnaround time.. *Waiting time* is the amount of time a process has been waiting in the ready queue. A context switch is the computing process of storing and restoring the state (context) of a CPU so that execution can be resumed from the same point at a later time. This enables multiple processes to share a single CPU. Optimal scheduling algorithm will have minimum waiting time, minimum turnaround time and minimum number of context switches.

The **process scheduler** is the component of the operating system that is responsible for deciding whether the currently running process should continue running and, if not, which process should run next. There are four events that may occur where the scheduler needs to step in and make this decision:

The current process goes from the *running* to the *waiting* state because it issues an I/O request or some operating system request that cannot be satisfied immediately.

The current process terminates.

A timer interrupt causes the scheduler to run and decide that a process has run for its allotted interval of time and it is time to move it from the *running* to the *ready* state.

An I/O operation is complete for a process that requested it and the process now moves from the *waiting* to the *ready* state. The scheduler may then decide to move this *ready* process into the *running* state.

A scheduler is a **preemptive** scheduler if it has the ability to get invoked by an interrupt and move a process out of a *running* state and let another process run. The last two events above may cause this to happen. If a scheduler cannot take the CPU away from a process then it is a **cooperative** or **non-preemptive** scheduler.

The objective of scheduling algorithms is to assign the CPU to the next ready process based on some predetermined policy. We study the following scheduling algorithms:

First-Come First-Served (FCFS) is a non-preemptive algorithm that assigns the CPU to the process in the ready queue that has been waiting for the longest time. This is a simple algorithm and it is not used very often in modern operating systems. A long process could cause a delay for all other processes that arrive after that process.

Received October 15, 2012; Revised December 24, 2012; Accepted January 16, 2013

Shortest Process (Job) Next (SJN) is another non-preemptive algorithm that attempts to decrease the average waiting time (and response time) of the system. This algorithm performs better than FCFS, however, it is not fair to long processes. In general preemptive scheduling algorithms are preferred due to their abilities to switch the CPU to another process even when the current running process is not completed.

In *Round Robin* (RR) scheduling a time slice is defined and the CPU is assigned to a process for a maximum of one time slice or until the process releases the CPU (whichever comes first). This algorithm requires more overheads but it is fair to all processes and performs better than non-preemptive scheduling algorithms.

Shortest Remaining Time Next (SRTN) is a preemptive version of SJN algorithm where the remaining processing time is considered for assigning CPU to the next process.

Highest Response Ratio Next (HRRN) selects a process with the largest ratio of waiting time over service time. This guarantees that a process does not starve due to its requirements.

Feedback Queue (FQ) scheduling algorithm partitions the ready processes into several separate queues and the processes are assigned to one queue and they are allowed to move between queues. Each queue has its own scheduling algorithm.

In priority systems, each process is assigned a priority and the scheduler will always select the highest priority ready process. Priority queues replace the ready queue, and processes are dispatched, starting with the head of the highest priority queue. There are three possible ways of assigning priorities to processes

1. **Statically or externally**

Priority is assigned by some external system manager before process is scheduled.

2. **Dynamically or internally**

Priority is assigned according to an algorithm.

3. **Hybrid**

Priority is assigned by some combination of external and internal schemes.

A problem with such a scenario is low-priority process starvation, in that if there is a steady stream of high-priority ready processes, the low-priority processes may not get any time on the processor.

HRRN scheduling algorithm

Highest Response Ratio Next (HRRN) scheduling is a non-preemptive discipline, in which the priority of each job is dependent on its estimated run time, and also the amount of time it has spent waiting. Jobs gain higher priority the longer they wait, which prevents indefinite postponement (process starvation). In fact, the jobs that have spent a long time waiting compete against those estimated to have short run times.

Priority = waiting time + estimated runtime / estimated runtime

(Or)

Ratio = waiting time + service time / service time

Advantages

Improves upon SPF scheduling
Still non-preemptive
Considers how long process has been waiting
Prevents indefinite postponement

Disadvantages

Does not support external priority system. Processes are scheduled by using internal priority system.

2. RESEARCH METHOD

Design of Modified HRRN algorithm

The major problem with HRRN is it will not consider the external priorities of the process.

The Modified HRRN uses the following formula to derive internal priority.

Ratio = waiting time + service time / service time ----- (1)

Here we Consider Hybrid Priority Systems,In Hybrid priority systems the priority is assigned by some combination of external and internal schemes.The Hybrid priority of the

processes is obtained by giving equal wieghtage for both external priority and internal priority which is calculated as

$$H_p = 0.5* E_p + 0.5 * R$$

Where, H_p represents Hybrid priority
E_p represents External priority
R is the ratio obtained from equation (1)

Advantages

- Improves upon SPF scheduling
- Still nonpreemptive
- Considers how long process has been waiting
- Prevents indefinite postponement
- Supports external priority system also.

Disadvantage

- No disadvantage.

Procedure for Algorithm

Step-1: start
Step-2 : Processes with Arrival time,Burst Time and priority are considered.
Step-3 : Ready Queue filled according to arrival times.
Step-4: Hybrid priority is computed for each process using the formula
$H_p = 0.5 * E_p + 0.5 * R$
Step-5: Process with highest hybrid priority is executed first.
Step-6: Repeat steps 4 and 5 until queue becomes empty.
Step-7: Now, Calculate average waiting time, average turnaround time, number of context switches.
Step-8: stop

3. RESULTS AND ANALYSIS

A. Assumptions

All experiments are assumed to be performed in uniprocessor environment and all the processes are independent from each other. Attributes like burst time and priority are known prior to submission of process. All processes are CPU bound. No process is I/O bound. Processes with same arrival time are scheduled.

B. Illustration and Results

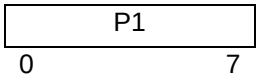
Case I :

Consider the Processes with following Arrival time, Burst Time and priorities

Process	Arrival Time	Burst Time	Priority
P1	0	7	3(high)
P2	2	4	1(low)
P3	3	4	2

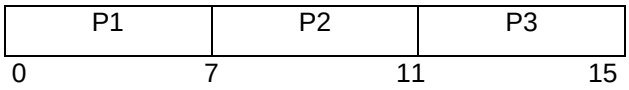
HRRN

At time 0 only process p1 is available, so p1 is considered for execution



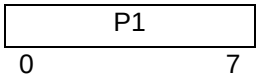
Since it is Non-preemptive, it executes process p1 completely. It takes 7 ms to complete process p1 execution.
Now, among p2 and p3 the process with highest response ratio is chosen for execution.
Ratio for p2 = 5+4 / 4 = 2.25

Ratio for p3 = 4+4 / 4 = 2
As process p2 is having highest response ratio than that of p3. Process p2 will be considered for execution and then followed by p3.

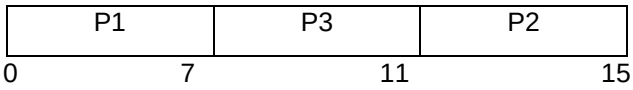


Average waiting time = 0+ (7-2) +(11-3) / 3 = 4.33
Average Turnaround time = 7+(11-2)+(15-3) / 3 = 9.33

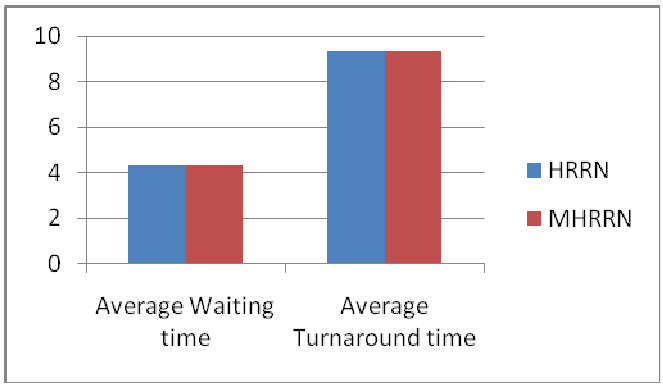
MHRRN
At time 0 only process p1 is available, so p1 is considered for execution



Since it is Non-preemptive, it executes process p1 completely. It takes 7 ms to complete process p1 execution.
Now, among p2 and p3 the process with highest Hybrid priority is chosen for execution.
Hybrid priority for p2 will be
$H_p = 0.5 * E_p + 0.5 * R = 0.5 * 1 + 0.5 * (5+4 /4) = 1.625$
Hybrid priority for p3 will be
$H_p = 0.5 * E_p + 0.5 * R = 0.5 * 2 + 0.5 * 2 = 2$
As process p3 is having highest hybrid priority than that of p2.So, Process p3 will be considered for execution and then followed by p2.



Average waiting time = 0+ (7-3) +(11-2) / 3 = 4.33
Average Turnaround time = 7+(11-3)+(15-2) / 3 =9.33

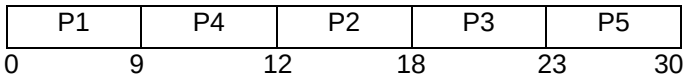


Comparing parameters of HRRN and Modified HRRN for case I

Case II:
Consider the Processes with following Arrival time, Burst Time and priorities

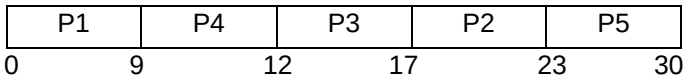
Process	Arrival time	Burst time	Priority
P1	0	9	1(LOW)
P2	2	6	3
P3	4	5	4
P4	5	3	5(HIGH)
P5	8	7	2

HRRN

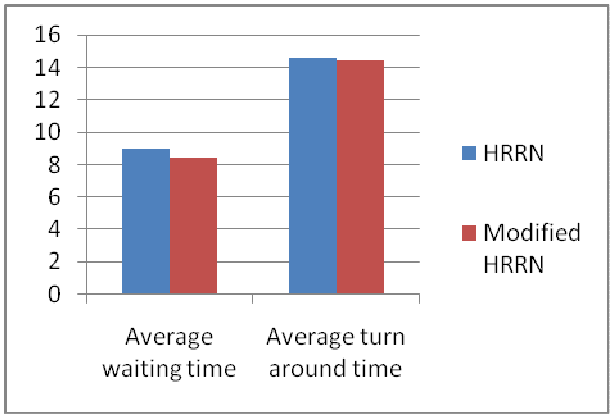


Average waiting time = (0+10+16+4+15) / 5 = 9
Average Turnaround time = (9+16+19+7+22) / 5 =14.6

Modified HRRN



Average waiting time = (0+15+8+4+15)/ 5 = 8.4
Average Turnaround time = (9+21+13+7+22) / 5=14.4



Comparing parameters of HRRN and Modified HRRN for case II

From the above cases, the Modified HRRN produces the better or at least same results as HRRN although it is considering the external priority.

	Waiting Time	Service Time	Internal Priority	External Priority
HRRN	Considers	Considers	Considers	Never Considers
Modified HRRN	Considers	Considers	Considers	Considers

4. Conclusion

Modified HRRN is suitable for priority systems and it is not suffering with starvation or indefinite postponement. Also it is producing good results. Therefore, Modified HRRN is suitable for priority systems. In future, the same algorithm can be extended such that it is also applicable to time shared systems.

References

[1] Silberschatz A, PB Galvin and G Gagne. "*Operating Systems Concepts*". 7th Edn., John Wiley and Sons, USA. ISBN: 13: 978-0471694663, 2004: 944.
[2] Tanebaun AS. "*Modern Operating Systems*". 3rd Edn. Prentice Hall, ISBN: 13: 9780136006633, 2008:1104.
[3] William Stallings, "*Operating Systems: Internals and Design Principles*". 6th edition, Prentice Hall. ISBN-13:978-0136006329.
[4] Himanshi Saxena, Prashant Agarwal. "Design and Performance Evaluation of Precedence Scheduling Algorithm with Intelligent Service Time (PSIST)". 2012 4th *International Conference on Computer Modeling and Simulation* (ICCMS 2012).
[5] P Surendra Varma. "Design and Performance Evaluation of Precedence Scheduling algorithm with Mean Average as Time Quantum (PSMTQ)". ISSN: 2277 – 9043 *International Journal of Advanced Research in computer Science and Electronics Engineering (IJARCSEE)*. 2012; 1(7).
[6] http://en.wikipedia.org/wiki/Highest_response_ratio_next
[7] "dhamdhere" operating systems, second edition.