ISSN:0975-9646

Anil Kumar Gupta / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 6 (2) , 2015, 1569-1572

Hybrid CPU Scheduling Algorithm

Anil Kumar Gupta
Department of Computer Science &Engineering
Galgotia's College of Engineering and Technology Greater Noida U.P. (India)

***Abstract*— The purpose of multiprogramming is to maximize the CPU utilization through some process running at all times in the CPU. A process change its state during the time of its execution that may be in one of the following states queue (new, ready, waiting, running, terminated) in the operating system. The procedure of selecting processes among these state queues is carried out by a scheduler. Scheduling of CPU is one of the critical factors that affect the efficiency and the efficiency of the system is maximized when we allocate processes to processor in a precise manner in process scheduling. The goal of CPU scheduler is to allocate processes to be executed by the processor. In this paper we proposed a new simple algorithm that is both pre-emptive and non-preemptive in nature to find a solution for CPU Scheduling. This algorithm is based on the NOVEL Algorithm [1]. Our aim is to enhance the novel algorithm and minimize average waiting time & average turnaround time for the given number of processes and result of this algorithm is then compared with the FCFS, SJF, SRJF, RR, Priority scheduling algorithms that are already discussed [1].**

***Keywords*— Scheduler, State Diagrams, CPU-Scheduling, Performance.**

I. INTRODUCTION

In a single-processor system, only one process can run at a time; any others must wait until the CPU is free. The aim of multiprogramming is to have some process running at all times, to maximize CPU utilization [2]. Scheduling is a fundamental operating-system function. Approximately all computer resources are scheduled before use. The CPU is, of course, one of the most important computer resources. Thus, its scheduling is essential to operating system design. CPU scheduling determines which processes run when there are multiple run-able processes. CPU scheduling is important because it can have a big effect on resource utilization and the overall performance of the system [3].

Generally, there are three types of schedulers, which may co-exist in a complex operating system:

- Long Term Scheduler
- Medium term scheduler
- Short term scheduler.

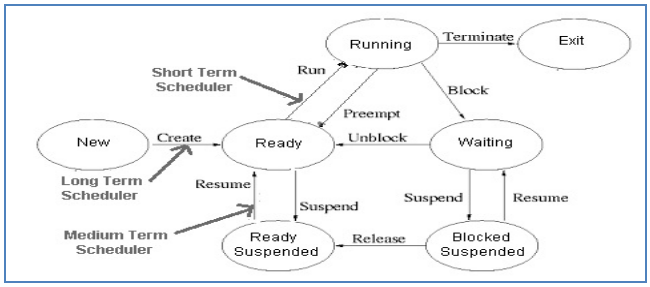

Figure-1: Process Life Cycle

A. LONG TERM SCHEDULER

Long term scheduling is performed when a new process is created. It is shown in the figure above. It decides which processes are to be admitted from NEW state queue to READY state queue and control the number of processes in the READY state queue because if the number of ready processes are high in the ready queue then it goes overhead in the operating system (i.e., processor) for maintaining long lists, context switching and dispatching increases[4]. So it allows only limited number of processes into the ready queue.

B. MEDIUM TERM SCHEDULER

Medium-term scheduling is an element of the swapping function. When the main memory gets freed, the OS looks at the list of suspend ready processes, decides which one is to be swapped in (depending on priority, memory and other resources required, etc)[4]. It will do the swapping-in function among the swapped-out processes.

C. SHORT TERM SCHEDULER

Short-term scheduler is also known as CPU scheduler. Short-term scheduler is invoked whenever an event raised, that may lead to the interruption of the current running process [4]. It selects one process from among the processes that are ready to execute in ready state and allocates the CPU.

The rest of the paper maintains as follows: **Section-2** Presents CPU scheduling objectives and performance criteria. **Section 3** Introduces existing scheduling algorithms. **Section-4** Explains the proposed Algorithm. **Section-5** Result **Section-6** will provide conclusion and future work.

II. OBJECTIVE AND PERFORMANCE CRITERIA

A. OBJECTIVES OF SCHEDULING

There are many objectives that must be considered in the design of scheduling discipline such as:

- ***Fairness:*** Avoid the process from starvation. All the processes must be given equal chance to execute [5].
- ***Throughput:*** Throughput is the rate at which processes are completed per unit of time.
- ***Predictable:*** A given job should run in about the same amount of time and at the same cost irrespective of the load on the system.
- ***Overhead:*** A certain portion of system resources invested as overhead can greatly Improve overall performance of the system.
- ***Resources:*** Scheduling mechanism [6] should keep the resources of the system busy.

- ***Indefinite postponement:*** Avoiding indefinite postponement of any process so that each process is executed in a certain amount of the time.
- ***Priority:*** Give preferential dealing to processes with higher priorities [5].

B. SCHEDULING PERFORMANCE CRITERIA

Scheduling criteria is also called as scheduling methodology. In multiprogramming system different CPU scheduling algorithms have different properties. The criteria used for comparing these algorithms include the following:

- ***CPU Utilization:*** Keep the CPU as busy as possible. It ranges from 0 to 100%. In practice, it ranges from 40 to 90%.
- ***Throughput:*** Throughput is the rate at which processes are completed per unit of time.
- ***Turnaround time (TAT):*** This is the how long a process takes to execute a process. It is calculated as the interval between the submission of a process and its completion.
- ***Waiting time (AT):*** Waiting time is the sum of the time periods spent in waiting in the ready queue.
- ***Response time (RT):*** Response time is the time it takes to start responding from submission time.

III. EXISTING ALGORITHM &LITERATURE REVIEW

A. TYPES OF SCHEDULING

i. *Non-Preemptive scheduling* is also known as "voluntary" or "co-operative" scheduling. In this case the scheduler is unable to forcibly removing processes from a CPU.

ii. *Preemptive Scheduling* is able to forcibly removing processes from a CPU when it decides to allocate that another process.

B. SCHEDULING ALGORITHM

i. *First-Come-First-Served (FCFS)* is the simplest scheduling algorithm. It simply placed the processes in running state, in the order that they arrive in the ready queue [1, 5].

ii. *Shortest Job First (SJF)* is the strategy of arranging processes with the least estimated processing time remaining to be next in the queue. It works under the two schemes (preemptive and non-preemptive). It's optimal since it minimizes the average turnaround time and the average waiting time. The main problem with this algorithm is the necessity of the previous knowledge about the time required for a process to complete. Also, it is not free from starvation issue especially in a busy system with many small processes being run [2, 7].

iii. *Round Robin (RR)* which is the main concern of this research is one of the oldest, easiest and fairest and most widely used scheduling algorithms, designed mainly for time-sharing systems. RR is similar to FCFS except that preemption is added to processes [2, 7].

iv. *Multi-Level Feedback Queue (MLFQ)* This algorithm is very popular in interactive systems. It resolves both efficiency and response time problems. It is also known as an "adaptive" algorithm, in that processes are always adapting to their previous execution history. This policy is mainly used if the remaining time of a process cannot be calculated for some reason, and thus turning its attention to the time spent executing. It's essential operation follows:

- A single queue is maintained for each priority level [8].
- A new process is added at the end of the highest priority level [8].
- It is allotted a single time quantum when it reaches the front [8].
- If the process uses up the time slice without blocking, then decrease its priority by one, and double its time slice for its next CPU burst

V. *Highest Response Ratio Next (HRRN)* This algorithm implements the "aging priority" scheme, in that as a process waits, its priority is increased until it finally gets to run. The priority is calculated as follows:

Priority = (w + s) / s

Where:

w = time spent waiting for the processor

s = expected service time

This policy is quite helpful in that long processes will age, and thus will eventually be assigned a higher-priority than the shorter jobs (which already have a high-priority because of the small denominator value).

C. RELATED WORK

The closest work in this area is A novel algorithm [1] who gave the nature of the algorithm both preemptive and non preemptive based on the arrival time. But the nature of the proposed algorithm is based on the burst time and later, we show that proposed algorithm gives better average waiting time and average turnaround time through example which is already discussed in novel algorithm and compare with other algorithm. To our knowledge, no earlier works have demonstrated like this however; this is what we show here.

IV. PROPOSED WORK: HYBRID CPU SCHEDULING ALGORITHM

The proposed algorithm *Hybrid CPU Scheduling Algorithm* is both preemptive and non-preemptive in nature. In this algorithm we find a factor known as *Total Elapsed Time (TET)* is calculated by the summation of burst time (B.T.) and arrival time(A.T.) i.e., *TET = B.T. + A.T.*. *TET* is assigned to each process and on the basis of *TET* process are sort in ascending order. Process having shortest *TET* is executed first and process with next shortest *TET* value is executed next. By considering the Burst Time (B.T.) the new algorithms acts as preemptive or non-preemptive. Proposed CPU scheduling algorithm decreases turnaround time, waiting time & response time and also increases CPU utilization and throughput. The working procedure of *Hybrid CPU Scheduling of Non-Preemptive and Preemptive* algorithm is as given below:

- ✓ Obtain the list of processes, their arrival time (A.T.) and burst time (B.T.).
- ✓ Find the *Total Elapsed Time (TET)* by summation of arrival time and burst time of processes.

- ✓ Arrange the processes in ascending order based on *TET*.
- ✓ Take the processes for execution as follows

Initially we assume that CPU arrival time is having some value (ZERO).

1. Pick lowest TET.
2. Compare Process arrival time with CPU arrival time is either equal or less.
3. If step 2 is not satisfied then, take next lowest TET and repeat step 2 until burst time of all processes become zero.
4. If *Total Elapsed Time (TET)* of any two processes is equal and satisfied step 2 then execute process based on lowest process ID.

A. CALCULATION

(i) Turnaround Time (TAT) is difference of Completion Time (C.T.) and Arrival Time (A.T.)

(ii) Waiting Time (W.T.) is difference of Turnaround Time (T.A.T.) and Burst Time (B.T.)

(iii) Average waiting time is calculated by dividing total waiting time with total number of processes.

(iv) Average turnaround time is calculated by dividing total turnaround time by total number of processes.

i. Calculation of T.A.T. And W.T.

These examples have been taken from the Novel Algorithm [1] to show that proposed algorithm give better performance compare to Novel Algorithm.

EXAMPLE-1

Process ID	Arrival Time	Burst Time
0	**04**	**02**
1	**01**	**04**
2	**02**	**06**
3	**03**	**01**

Solution:

Step-1 Calculation of Total Elapsed Time (TET) =AT+BT

Process ID	AT	BT	Total Elapsed Time(TET)
0	04	02	6
1	01	04	5
2	02	06	8
3	03	01	4

Step-2 Sort the processes based on TET

Process ID	AT	BT	TET
3	03	01	4
1	01	04	5
0	04	02	6
2	02	06	8

Step-3 Calculation of Completion Time (CT), Turnaround Time (TAT) and Waiting Time (WT).

Process ID	AT	BT	TET	CT	TAT	WT
3	03	01	4	6	3	2
1	01	04	5	5	4	0
0	04	02	6	8	4	2
2	02	06	8	14	12	6
Total					23	10

Gantt chart:

Ideal	P1	P3	P0	P2
0	1	5	6	8	14

Avg. Turnaround Time=Total TAT/Total no of process = **23/4**

= 5.75

Avg. Waiting Time=Total WT/Total no of process = **10/4**

= 2.5

EXAMPLE-2

Process ID	Arrival Time	Burst Time
0	**01**	**04**
1	**02**	**06**
2	**03**	**10**
3	**04**	**05**
4	**05**	**20**
5	**06**	**01**

Solution:

Step-1 Calculation of Total Elapsed Time (TET) =AT+BT

Process ID	AT	BT	Total Elapsed Time(TET)
0	01	04	5
1	02	06	8
2	03	10	13
3	04	05	9
4	05	20	25
5	06	01	7

Step-2 Sort the processes based on TET

Process ID	AT	BT	TET
0	01	04	5
5	06	01	7
1	02	06	8
3	04	05	9
2	03	10	13
4	05	20	25

Step-3 Calculation of Completion Time (CT), Turnaround Time (TAT) and Waiting Time(WT).

Process ID	AT	BT	TET	CT	TAT	WT
0	01	04	5	5	4	0
5	06	01	7	12	6	5
1	02	06	8	11	9	3
3	04	05	9	17	13	8
2	03	10	13	27	24	14
4	05	20	25	47	42	22
Total					**98**	**52**

Gantt chart:

Ideal	P0	P1	P5	P3	P2	P4
0	1	5	11	12	17	27	47

Avg. Turnaround Time=Total TAT/Total no of process

= 98/6

=16.33

Avg. Waiting Time=Total WT/Total no of process = **52/6**

= 8.66

V. RESULTS

1)

Algorithms	Example No-1				Example No-2			
	Non-preemtive		Preemptive		Non-preemtive		Preemptive	
	AvgTAT	AvgWT	AvgTAT	AvgWT	AvgTAT	AvgWT	AvgTAT	AvgWT
Proposed Algorithm	*5.75**	*2.5**	5.5	*1.75**	*16.33**	*8.66**	15.66	8.0
Novel Algorithm	6.00	2.7	5.5	2.2	17.3	9.6	15.6	8.0

2) From above Example 1 and 2 it shows that our proposed algorithm give less average turnaround time and average waiting time compare to NOVEL Algorithm as well as other scheduling algorithm which are discussed in [1].

3) It is act as SJF when arrival time of all processes is same.

4) It is act as priority scheduling when TET is consider as priority.

5) This algorithm inherits the nature of both SJF and Priority algorithm therefore it will call Hybrid Algorithm.

VI. CONCLUSION & FUTURE WORK

In this paper we present a new CPU scheduling algorithm called Hybrid CPU Scheduling Algorithm. Paper does not contains any simulation but it based on novel algorithm[1] and example are taken from there to avoid unnecessary work only here we compare the Propose algorithm result against Novel algorithm result to validation of it. From the above results, it is clear that proposed algorithm is more efficient than Novel, FCFS, Pre-emptive Priority and Non Pre-emptive Priority, Round Robin. In Future we can come with new CPU scheduling algorithm that will more efficient to compare to other existing algorithm.

REFERENCES

[1] International Journal of Modern Engineering Research (IJMER) www.ijmer.*com Vol.2, Issue.6, Nov-Dec. 2012 pp-4484-4490 ISSN: 2249-6645*

[2] Silberschatz, A. P.B. Galvin and G. Gagne (2012), Operating System Concepts, 8th edition, Wiley India.

[3] Sabrian, F., C.D. Nguyen, S. Jha, D. Platt and F. Safaei, (2005). Processing resource scheduling in programmable networks. Computer communication, 28:676-687.

[4] www.go4expert.com › Articles › Operating System.

[5] A. Dhore “Opeating Systems”, Technical Publications.

[6] www.sciencehq.com/computing- technology /1308.html.

[7] Lingyun Yang, Jennifer M. Schopf and Ian Foster, “Conservative Scheduling: Using predictive variance to improve scheduling decisions in Dynamic Environments”, Super Computing 2003, November 15-21, Phoenix, AZ, USA.

[8] www.site.uottawa.ca/~rabielmo/scheduling/elg6171_FinalReport.pdf