Analysis Of Job Scheduler Algorithms To Manage The CPU

Dr. Eng LOIE NASER MAHMOUD NIMRAWI

Assistant Professor of Computer systems & Complex Networks

Sajir College of Science and Arts Shaqra University

Abstract.

The aim of this paper is to analyze the algorithms of job and program scheduling in the course of managing the central processor unit, assuming the ability of memory to absorb one job using the algorithms of First In First Out "FIFO" basis, performing the shortest function first together with the Future Knowledge and scheduling Multiprogramming assuming the ability of the computer to absorb more than one job and study the Round Robin algorithm .

Keywords : Job Scheduler ,First in First out , Shortest Job First , Future Knowledge ,Scheduling Multiprogramming , Round Robin .

1. Job Scheduler

The Job Scheduling Department is interested in the process of organizing the work existed in the supporting executable memory due to the limitation of physical resources in the computer, so it is necessary to schedule jobs.

Job scheduling means putting these jobs in the standby list so that these they are arranged in the tables within specific priorities , while jobs are followed up by The Job Control Block Register.

The definition of work "Job Control block register" in order to implement them while some resources can't be used or exploited for more than one subscriber due to fear of falling into the problem of race and deadlocks , the Job Control Block Register consists of:

1. Job's Title
2. Job's Current Status
3. Job's Priority
4. The time required or remaining for implementation

When the Job becomes executable, it moves from the Pause to the Standby status, where the Job Scheduler surveys the Job Control Block Register to identify the status of each job and choose jobs in the standby status.

1.1 Methods of scheduling

Some computer sources, such as "printers," can't be shared by more than one job or a subscriber, fearing the problem of race and conflict over these sources. Therefore, when scheduling jobs, we pay attention to three principles, each of which aims at reducing the waiting time for job, assuming that the computer can't absorb more than one job at the same time.

1.1.1 First in First out

The functions are arranged according to the priority of their arrival. This algorithm is considered non-preemptive, i.e., the process does not go out of the CPU until it is implemented, and it is characterized to be easily programmed and implemented, while its disadvantages are that One Process Can Monopolize CPU, i.e. when processing any operation, it takes a long time that may take up all the CPU time so that there is no space to perform the lesser time operations and makes it take much time.

In order to solve the problems of this algorithm, you must determine how much time the process may take on the CPU without exchanging with another one and this time is called (time slice) which means the division of the CPU time to the operations. In order to clarify this principle, let's take data provided in the following table "Table 1".

Time: we use decimal fraction of the hour, i.e. 0.1 means: 0.1× 60= 6 minutes.

Table "1"

Implementation Period	Access Time	Job No
0.2	12.0	1
0.3	12.1	2
0.1	12.2	3
0.4	12.3	4

The waiting time is known as **W "Wait"** , Access Time is **A "Access"** and Finish Time **" F "**,

The waiting time is defined as follows :

Waiting time = Finish time - Access time

which shall be Symbolized as follows:

W = F- A

The average waiting time = Total waiting time / Number of works , Symbolized as follows :

$$\overline{W} = W \div N$$

Where N = number of works.

In the case of calculating the waiting time "W" for each work in the previous table "Table 1" and the average waiting time "W", the output shall be as follows "Table 2" based on the "FIFO"

Table 2 "FIFO"

Waiting Time	Finish Time	Execution Time	Implementation Period	Access Time	Job No
0.2	12.2	12.0	0.2	12.0	1
0.4	12.5	12.2	0.3	12.1	2
0.4	12.6	12.5	0.1	12.2	3
0.7	13.0	12.6	0.4	12.3	4

Average waiting time = 0.2 + 0.4 + 0.4 + 0.7 = 1.7 ÷ 4 = 0.425

Where The Finish time = the period of execution + the time of starting execution

For example : Finish Time " 1" = 0.2 + 12.0 = 12.2

For example: Time of Completion of Work" 2 "

0.3 + 12.2 = 12.5

Waiting Time= The Finish Time - The Access Time

For example: The Waiting Time for Work"1"

12.2 -12.0 = 0.2

For example: Waiting Time for Work"2"

12.5 - 12.1 = 0.4

1.1.2 Shortest Job First

This algorithm is based on the principle of prioritizing work implementation according to the implementation time required for each of them where the work accessed first is implemented first and then start selecting the next work according to the shortest implementation time where each operation is attached with the time expected to take on the CPU to be finished, this time is called (CPU Burst) to be scheduled, according to this time in the ready queue, where the shorter time shall be first, but, if the time of two processes is the equal, then arrangement shall be made according to the access time. This algorithm is characterized that is makes the waiting time for the operations less, while the problems of the algorithm are as follows:

1. Difficulty of implementation and expecting the time taken for the operation (CPU Burst)
2. Succession of the longer-time processes that you cause it long-delayed and may never be implemented, so it suffer From Starvation of Large Processes the problem can be solved by using the "Multi-Level Queue"

1.1.2.1 This algorithm has two methods of work:

1. Preemptive : This method allows the processor to constantly search for the best execution process, even while busy implementing a current process, i.e. when a new process is entered and its CPU Burst time is less than the remaining time for the current process, it is re-entered in the ready queue.
2. Non-Preemptive : When the process is entered into the CPU, it does not go out until it is finished.

How to calculate the time taken (CPU BURST) :

T (n + 1) = W.t (n) + (1-w) .T (n)

Considering that:

T (n + 1): the time of the next operation
W: Waiting time
T (n): The current operation time
T (n): Average time to implement the on the CPU

IF the algorithm of the "Shortest Job First" is applied in calculating the waiting time "W" for each job and the average waiting time for the previous table "Table 1", the output shall be as follows in "Table 3"

Table 3 "SJF"

Waiting Time	Finish Time	Execution time	Impleme - ntation Period	Access Time	Work priority	Job No
0.2	12.2	12.0	0.2	12.0	1	1
0.5	12.6	12.3	**0.3**	12.1	3	2
0.1	12.3	12.2	**0.1**	12.2	2	3
0.4	13.0	12.6	**0.4**	12.3	4	4

Average waiting time = 1.2 ÷ 4 = 0.3

Note that the average waiting time in the second principle is lower than that in the first principle

1.1.3 Future Knowledge

This principle is to make the computer idle without work until the access of all the works and then the work shall be is implemented according to the job of lesser time to be implemented, as shown in Table 4:

Table 4

Waiting Time	Finish Time	Execution Time	Implemen-tation Period	Access Time	Work Priority	Job No
0.6	12.6	12.4	0.2	12.0	2	1
0.8	12.9	12.6	0.3	12.1	3	2
0.2	12.4	12.3	0.1	12.2	1	3
1.0	13.3	12.9	0.4	12.3	4	4

Average waiting time = 2.6 ÷ 4 = 0.65

Where the idle time of the device = **Access time " last work" - Access time " first work "**

12.3 - 12.0 = 0.3

It's also noted that the average waiting time has increased more than that which previously mentioned above.

1.1.4 Round Robin

CPU time is divided to the operation by giving a specific execution time called (time quantum), which is usually ranging between **10-100** part of second. When the process takes its specified time, it shall be turned off, removed from the CPU and returned to the last ready queue.

The longest time that any operation can take cannot exceed the average time specified for operations, where the longest waiting time for any operation is **(n-1) q)** since **"q"** is the time specified for the operation and **"n"** is the number of operations.

1.1.4.1 Round Robin algorithm problems

1. The more the time quantum increases, we get the FIFO algorithm, which is not required, however it gives the operation an opportunity to finish its job on the CPU without interruption.
2. The less the time quantum decreases, the more operation is interrupted after entered into the CPU

These problems can be eliminated by using the Priority Scheduling algorithm, where each process, when created, is assigned an "integer" to indicate its priority, and then the operations are arranged according to the highest priority basis.

This algorithm has two methods of work:

1. Preemptive: Allows the processor to constantly search for the best operation for execution, even while engaging in a current operation.
2. Non-Preemptive

Disadvantage: In this algorithm, we face the starvation problem where the least priority operations are highly unlikely to be implemented.

Solution: To solve the problem of starvation, we resort to (aging)as the longer the operation remains in the ready queue, the more its priority increases.

1.1.5 Scheduling Multiprogramming

Multiprogramming means the ability of the computer "basic memory" to absorb more than one program or mob at the same time, as the cost of executing jobs in multiprogramming increases due to increasing of the execution time and the waiting time, therefore, the jobs must be scheduled to achieve a balance between the arithmetic processing operations and the input and output operations to determine the waiting time, if we want to schedule the jobs given in the previous table "Table 1" assuming the ability of the computer to absorb multiprogramming, it will be as follows:

Table of using multiprogramming

Time	Job status	Number of jobs in the Memory	Percentage of the processing time	Time Taken	Time taken for Each Job	Jobs	Remaining Time
12	Job access 1	-	-	-	-	1	0, 2
12,1	Job access 2	1	100%	0,1	0,1	1 2	0, 1 0,3
12,2	Job access 3	2	0.5%	0,1	0,05	1 2 3	0,05 0,25 0, 1
12,3	Job access 4	3	33.3 %	0,1	0,033	1 2 3 4	0,017 0,217 0,067 0,4
12,368	Job Finish 1	4	25%	0,068	0,017	2 3 4	0.2 0,05 0,333
12,518	Job Finish 2	3	33.3%	0,15	0,05	2 4	0,15 0,183
12,818	Job Finish 3	2	50%	0,3	0,15	4	0,033
12,848	Job Finish 4	1	100%	0,033	0,033	-	-

In order to calculate the average waiting time upon execution, it shall be as follows:

- **Processor time ratio** =1 ÷ Number of jobs in the memory
- **Number of jobs in the memory** = Number of jobs entered into the memory before the job status

- **Time taken** = the difference between two consecutive times of job status
- **Time taken per each job** = Processing time ratio multiplied by the time taken
- **The remaining time** = the remaining time of the job status - the time taken for each job.

Time of the waiting time using multiprogramming

Job No	Access time	Execution time	Finish Time	Waiting Time
1	12,0	12,0	12,368	0,368
2	12,1	12,1	12,518	0,418
3	12,2	12,2	12,818	0,618
4	12,3	12,3	12,848	0,548

Average waiting time = (0.548+0618+0418+0.368) ÷ 4

W= 0.488

We find that the priorities of the implementation of the jobs and programs in scheduling multiprogramming is of a variable nature that depends on the number of jobs implemented in the memory and times for implementation of these jobs. In the previous example, we used the "FIFO" algorithm.

1.2 Summary

- The FIFO algorithm is considered to be non-preemptive and characterized to be easily programmed and implemented, while its disadvantages are that: One Process Can Monopolize CPU.In order to solve the problems of this algorithm, it is necessary to determine the amount of time during which the process can be on the CPU without exchanging it with another process, this time called "Time Slice".
- Some disadvantages of the algorithm of "Shortest Job First" are the difficulty of implementation, predicting the CPU Burst and the execution time of the "Starvation OF Large Process". This problem can be solved using the "multi-level queue" algorithm.
- The operation waiting time when applying the SJF algorithm is applied is less than the waiting time of operations when applying the FIFO and Future Knowledge algorithms
- The operation waiting time for the FIFO algorithm is less than the waiting time when applying the "Future Knowledge" algorithm due to the computer idleness to execute jobs.
- The Round Robin algorithm is distinguished of dividing the CPU time on the operations, giving a specific execution time called "Time Quantum"
- Round Robin algorithm's defects are that, the more the time quantum increases, we get the FIFO algorithm and, the less the time quantum decreases, the more operation is interrupted after entered into the CPU. These problems can be eliminated by using the Priority Scheduling algorithm, where each process, when created, is assigned an "integer" to indicate its priority, and then the operations are arranged according to the highest priority basis.
- In the Theory of Multiprogramming, the cost of job execution increases due to increasing of the execution time and the waiting time, therefore, the jobs must be scheduled to achieve a balance between the arithmetic processing operations and the input and output operations to determine the waiting time.
- Priorities of the implementation of the jobs and programs in scheduling multiprogramming is of a variable nature that depends on the number of jobs implemented in the memory and times for implementation of these jobs, it is characterized that all the jobs are in the course of partial implementation due to dividing the processor time knowing that the waiting time is convergent when the "FIFO" algorithm is applied.

References :

1. A. Dusseau, R. H. dan A. C., Operating Systems: Three Easy Pieces, Arpaci - Dusseau Books, 2014.
2. Tanenbaum, Modern Operating Systems, Pearson Education, Inc., 2008.
3. Madnick /Donovan "Operating Systems" Int-ST.Edn Mcgraw Hill International Book Company.
4. Berziss A.T "Data Structures Theory and Practice". Academic
5. Abraham Silberschatz,Peter Baer Galvin,Greg Gagne Operating system Concepts (Seventh a Edition)
6. Ziad Al Qadi, "Operating Systems" Dar Al Mustaqbal, Amman.
7. Silberschatz, A., Operating System Concepts, 2005.