Scheduling Algorithms

Gursharan Singh Tatla
professorgstatla@gmail.com

1. Scheduling Algorithms

- CPU Scheduling algorithms deal with the problem of deciding which process in ready queue should be allocated to CPU.
- Following are the commonly used scheduling algorithms:

  - First-Come-First-Served (FCFS)
  - Shortest Job First (SJF)
  - Priority Scheduling
  - Round-Robin Scheduling (RR)
  - Multi-Level Queue Scheduling (MLQ)
  - Multi-Level Feedback Queue Scheduling (MFQ)

2. First-Come-First-Served (FCFS)

- In this scheduling, the process that requests the CPU first, is allocated the CPU first.
- Thus, the name First-Come-First-Served.
- The implementation of FCFS is easily managed with a FIFO queue.

   ![Ready Queue Diagram]

3. Example of FCFS Scheduling

- Consider the following set of processes that arrive at time 0 with the length of the CPU burst time in milliseconds:

<table>
<thead>
<tr>
<th>Process</th>
<th>Burst Time (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_1</td>
<td>24</td>
</tr>
<tr>
<td>P_2</td>
<td>3</td>
</tr>
<tr>
<td>P_3</td>
<td>3</td>
</tr>
</tbody>
</table>
Example of FCFS Scheduling

- Suppose that the processes arrive in the order: P₁, P₂, P₃.
- The Gantt Chart for the schedule is:
  
```
  0  24  27  30
  P₁
  P₂
  P₃
```
- Waiting Time for P₁ = 0 milliseconds
- Waiting Time for P₂ = 24 milliseconds
- Waiting Time for P₃ = 27 milliseconds

Example of FCFS Scheduling

- Average Waiting Time = (Total Waiting Time) / No. of Processes
- \[ \frac{(0 + 24 + 27)}{3} \]
- \[ = \frac{51}{3} \]
- \[ = 17 \text{ milliseconds} \]

Example of FCFS Scheduling

- Suppose that the processes arrive in the order: P₂, P₃, P₁.
- The Gantt chart for the schedule is:
  
```
  0  30
  P₂
  P₃
  P₁
```
- Waiting Time for P₂ = 0 milliseconds
- Waiting Time for P₃ = 3 milliseconds
- Waiting Time for P₁ = 6 milliseconds

Example of FCFS Scheduling

- Average Waiting Time = (Total Waiting Time) / No. of Processes
- \[ \frac{(0 + 3 + 6)}{3} \]
- \[ = \frac{9}{3} \]
- \[ = 3 \text{ milliseconds} \]
- Thus, the average waiting time depends on the order in which the processes arrive.

Shortest Job First Scheduling (SJF)

- In SJF, the process with the least estimated execution time is selected from the ready queue for execution.
- It associates with each process, the length of its next CPU burst.
- When the CPU is available, it is assigned to the process that has the smallest next CPU burst.
- If two processes have the same length of next CPU burst, FCFS scheduling is used.
- SJF algorithm can be preemptive or non-preemptive.

Non-Preemptive SJF

- In non-preemptive scheduling, CPU is assigned to the process with least CPU burst time.
- The process keeps the CPU until it terminates.
  
**Advantage:**
- It gives minimum average waiting time for a given set of processes.
  
**Disadvantage:**
- It requires knowledge of how long a process will run and this information is usually not available.
Preemptive SJF

- In preemptive SJF, the process with the smallest estimated run-time is executed first.
- Any time a new process enters into ready queue, the scheduler compares the expected run-time of this process with the currently running process.
- If the new process’s time is less, then the currently running process is preempted and the CPU is allocated to the new process.

Example of Non-Preemptive SJF

- Consider the following set of processes that arrive at time 0 with the length of the CPU burst time in milliseconds:

<table>
<thead>
<tr>
<th>Process</th>
<th>Burst Time (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P₁</td>
<td>6</td>
</tr>
<tr>
<td>P₂</td>
<td>8</td>
</tr>
<tr>
<td>P₃</td>
<td>7</td>
</tr>
<tr>
<td>P₄</td>
<td>3</td>
</tr>
</tbody>
</table>

Example of Non-Preemptive SJF

- The Gantt Chart for the schedule is:

<table>
<thead>
<tr>
<th>Process</th>
<th>AT</th>
<th>BT</th>
</tr>
</thead>
<tbody>
<tr>
<td>P₁</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>P₂</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>P₃</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>P₄</td>
<td>3</td>
<td>5</td>
</tr>
</tbody>
</table>

- Waiting Time for P₄ = 0 milliseconds
- Waiting Time for P₁ = 3 milliseconds
- Waiting Time for P₃ = 9 milliseconds
- Waiting Time for P₂ = 16 milliseconds

Example of Preemptive SJF

- Consider the following set of processes. These processes arrived in the ready queue at the times given in the table:

<table>
<thead>
<tr>
<th>Process</th>
<th>Arrival Time</th>
<th>Burst Time (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P₁</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>P₂</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>P₃</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>P₄</td>
<td>3</td>
<td>5</td>
</tr>
</tbody>
</table>

Example of Preemptive SJF

- The Gantt Chart for the schedule is:

<table>
<thead>
<tr>
<th>Process</th>
<th>AT</th>
<th>BT</th>
</tr>
</thead>
<tbody>
<tr>
<td>P₁</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>P₂</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>P₃</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>P₄</td>
<td>3</td>
<td>5</td>
</tr>
</tbody>
</table>

- Waiting Time for P₁ = 10 – 1 – 0 = 9
- Waiting Time for P₂ = 1 – 1 = 0
- Waiting Time for P₃ = 17 – 2 = 15
- Waiting Time for P₄ = 5 – 3 = 2

Average Waiting Time = (Total Waiting Time) / No. of Processes
= (0 + 3 + 9 + 16) / 4
= 28 / 4
= 7 milliseconds
Example of Preemptive SJF

- Average Waiting Time = (Total Waiting Time) / No. of Processes
  = (9 + 0 + 15 + 2) / 4
  = 26 / 4
  = 6.5 milliseconds

Explanation of the Example

- Process P1 is started at time 0, as it is the only process in the queue.
- Process P2 arrives at the time 1 and its burst time is 4 milliseconds.
- This burst time is less than the remaining time of process P1 (7 milliseconds).
- So, process P1 is preempted and P2 is scheduled.

Explanation of the Example

- Process P3 arrives at time 2. Its burst time is 9 which is larger than remaining time of P2 (3 milliseconds).
- So, P2 is not preempted.
- Process P4 arrives at time 3. Its burst time is 5.
  Again it is larger than the remaining time of P2 (2 milliseconds).
- So, P2 is not preempted.

Explanation of the Example

- After the termination of P2, the process with shortest next CPU burst i.e. P4 is scheduled.
- After P4, processes P1 (7 milliseconds) and then P3 (9 milliseconds) are scheduled.

Priority Scheduling

- In priority scheduling, a priority is associated with all processes.
- Processes are executed in sequence according to their priority.
- CPU is allocated to the process with highest priority.
- If priority of two or more processes are equal than FCFS is used to break the tie.

Priority Scheduling

- Priority scheduling can be preemptive or non-preemptive.
  - Preemptive Priority Scheduling:
    - In this, scheduler allocates the CPU to the new process if the priority of new process is higher than the priority of the running process.
  - Non-Preemptive Priority Scheduling:
    - The running process is not interrupted even if the new process has a higher priority.
    - In this case, the new process will be placed at the head of the ready queue.
**Priority Scheduling**

- **Problem:**
  - In certain situations, a low priority process can be blocked infinitely if high priority processes arrive in the ready queue frequently.
  - This situation is known as **Starvation**.

**Solution:**

- **Aging** is a technique which gradually increases the priority of processes that are victims of starvation.
  - For e.g.: Priority of process X is 10.
  - There are several processes with higher priority in the ready queue.
  - Processes with higher priority are inserted into ready queue frequently.
  - In this situation, process X will face starvation.

---

**Priority Scheduling**

(Cont.):

- The operating system increases priority of a process by 1 in every 5 minutes.
- Thus, the process X becomes a high priority process after some time.
- And it is selected for execution by the scheduler.

---

**Example of Priority Scheduling**

- Consider the following set of processes that arrive at time 0 with the length of the CPU burst time in milliseconds. The priority of these processes is also given:

<table>
<thead>
<tr>
<th>Process</th>
<th>Burst Time</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_1</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>P_2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>P_3</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>P_4</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>P_5</td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>

- The Gantt Chart for the schedule is:

- Waiting Time for P_2 = 0
- Waiting Time for P_5 = 1
- Waiting Time for P_1 = 6
- Waiting Time for P_3 = 16
- Waiting Time for P_4 = 18

**Average Waiting Time**

\[
\text{Average Waiting Time} = \frac{\text{Total Waiting Time}}{\text{No. of Processes}}
\]

\[
= \frac{0 + 1 + 6 + 16 + 18}{5}
\]

\[
= \frac{41}{5}
\]

\[
= 8.2 \text{ milliseconds}
\]
Another Example of Priority Scheduling

- Processes $P_1$, $P_2$, $P_3$ are the processes with their arrival time, burst time and priorities listed in table below:

<table>
<thead>
<tr>
<th>Process</th>
<th>Arrival Time</th>
<th>Burst Time</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>0</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>$P_2$</td>
<td>1</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>$P_3$</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Another Example of Priority Scheduling

- The Gantt Chart for the schedule is:

```
+---+---+---+---+---+---+---+
| P1| P2| P3| P2| P1| P3| P3|
+---+---+---+---+---+---+---+
| 0 | 1 | 2 | 4 | 8 | 17| 17|
```

- Waiting Time for $P_1 = 0 + (8 - 1) = 7$
- Waiting Time for $P_2 = 1 + (4 - 2) = 3$
- Waiting Time for $P_3 = 2$

Another Example of Priority Scheduling

- Average Waiting Time = \( \frac{(Total \ Waiting \ Time)}{No. \ of \ Processes} \)
  \[
  = \frac{7 + 3 + 2}{3} \]
  \[
  = \frac{12}{3} \]
  \[
  = 4 \text{ milliseconds}
  \]

Round Robin Scheduling (RR)

- In Round Robin scheduling, processes are dispatched in FIFO but are given a small amount of CPU time.
- This small amount of time is known as Time Quantum or Time Slice.
- A time quantum is generally from 10 to 100 milliseconds.

Round Robin Scheduling (RR)

- If a process does not complete before its time slice expires, the CPU is preempted and is given to the next process in the ready queue.
- The preempted process is then placed at the tail of the ready queue.
- If a process is completed before its time slice expires, the process itself releases the CPU.
- The scheduler then proceeds to the next process in the ready queue.

Round Robin Scheduling (RR)

- Round Robin scheduling is always preemptive as no process is allocated the CPU for more than one time quantum.
- If a process's CPU burst time exceeds one time quantum then that process is preempted and is put back at the tail of ready queue.
- The performance of Round Robin scheduling depends on several factors:
  - Size of Time Quantum
  - Context Switching Overhead
Example of Round Robin Scheduling

Consider the following set of processes that arrive at time 0 with the length of the CPU burst time in milliseconds:

<table>
<thead>
<tr>
<th>Process</th>
<th>Burst Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>P₁</td>
<td>10</td>
</tr>
<tr>
<td>P₂</td>
<td>5</td>
</tr>
<tr>
<td>P₃</td>
<td>2</td>
</tr>
</tbody>
</table>

- Time quantum is of 2 milliseconds.

Gantt Chart for the schedule:

```
0 2 6 10 12 13 15 17
```

<table>
<thead>
<tr>
<th>P</th>
<th>BT</th>
</tr>
</thead>
<tbody>
<tr>
<td>P₁</td>
<td>10</td>
</tr>
<tr>
<td>P₂</td>
<td>5</td>
</tr>
<tr>
<td>P₃</td>
<td>2</td>
</tr>
</tbody>
</table>

Waiting Time for P₁:

\[
= 0 + (6 – 2) + (10 – 8) + (13 – 12) = 4 + 2 + 1 = 7
\]

Waiting Time for P₂:

\[
= 2 + (8 – 4) + (12 – 10) = 2 + 4 + 2 = 8
\]

Waiting Time for P₃:

\[
= 4
\]

Average Waiting Time = (Total Waiting Time) / (No. of Processes) = (7 + 8 + 4) / 3 = 19 / 3 = 6.33 milliseconds

Multi-Level Queue Scheduling (MLQ)

- In this scheduling, ready queue is divided into various queues that are called subqueues.
- The processes are assigned to subqueues, based on some properties like memory size, priority or process type.
- Each subqueue has its own scheduling algorithm.
- For e.g.: interactive processes may use round robin algorithm while batch processes may use FCFS.

Multi-Level Queue Scheduling (MLQ)

- Multi-Level Queue scheduling classifies the processes according to their types.
- For e.g.: a MLQ makes common division between the interactive processes (foreground) and the batch processes (background).
- These two processes have different response times, so they have different scheduling requirements.
- Also, interactive processes have higher priority than the batch processes.
Multi-Level Feedback Queue Scheduling (MFQ)

- Multi-Level Feedback Queue scheduling is an enhancement of MLQ.
- In this scheme, processes can move between different queues.
- The various processes are separated in different queues on the basis of their CPU burst times.
- If a process consumes a lot of CPU time, it is placed into a lower priority queue.
- If a process waits too long in a lower priority queue, it is moved into higher priority queue.
- Such an aging prevents starvation.

- The top priority queue is given smallest CPU time quantum.
- If the quantum expires before the process terminates, it is then placed at the back of the next lower queue.
- Again, if it does not complete, it is put to the last priority queue.
- The processes in this queue run on FCFS scheduling.
- If a process becomes a victim of starvation, it is promoted to the next higher priority queue.

Thank You 😊
Have a Nice Day