Process Scheduling

Ellysa N. Kosinaya
Preview

- Process scheduler
- CPU scheduler
- Scheduling goals
- Performance measures
- Gantt chart calculations
- Scheduling algorithms
Process scheduler

• When CPU is idle, OS must select one of the processes in the ready queue to be executed

• Long-term scheduler is responsible for:
  • Selecting processes from pool of jobs
  • Loads into memory for execution

• Short-term scheduler is responsible for:
  • Selecting processes from ready queue
  • Allocates to CPU

• Memory scheduler
  • Decides whether processes should be kept in memory or disk
Three level scheduling

Arriving job -> Input queue -> Main Memory

Long-Term Scheduler (Admission Scheduler) -> Memory scheduler 

Main Memory -> Short-Term Scheduler (CPU Scheduler) -> CPU

Disk
CPU scheduler

• Scheduling decisions – 4 circumstances:
  1. A process switches from the running state to the block state
  2. A process switches from the running state to the ready state
  3. A process switches from the block state to the ready state
  4. A process terminates

• nonpreemptive – circumstances 1 and 4
  • No choice in terms of scheduling

• preemptive – circumstances 2 and 3
  • There is a choice; with a cost
Scheduling decisions

(Short Term) Scheduling

Create → Ready

Run → Running

Blocked → Running

Preempt → Running

Run → Waiting

Unblock → Ready

Terminate → Ready
Scheduling goals

- **Batch Systems**
  - CPU utilization – keep the CPU as busy as possible
  - Throughput – maximize the number of processes completed per time unit
  - Turnaround time – minimize time between submission and termination

- **Interactive Systems**
  - Response time – minimize response time to request

- **Real Time System**
  - Meeting deadlines – avoid losing data
Performance measures

• **Turnaround time** – the important criterion is how long it takes to execute that process. The interval from the time of submission of a process to the time of completion. It is the sum of the periods spent waiting to get into memory, waiting in the ready queue, execution on the CPU, and doing I/O.

• **Waiting time** – the sum of the periods spent waiting in the ready queue.

• **Response time** – the time from submission of a request until the first response is produced. In other words, it’s the amount of time it takes to start responding, but not the time that it takes to output that response.
Gantt chart calculations

**Process A**
- Turnaround time
- Wait time
- Response time

**Process B**
- Turnaround time
- Wait time
- Response time

**Process C**
- Turnaround time
- Wait time
- Response time
Gantt chart calculations

Process A

- Turnaround time
- Wait time
- Response time

Time

Process B

- Turnaround time
- Wait time
- Response time

Time

Process C

- Turnaround time
- Wait time
- Response time

Time
Gantt chart calculations

Process A:
- Turnaround time
- Wait time
- Response time

Process B:
- Turnaround time
- Wait time
- Response time

Process C:
- Turnaround time
- Wait time
- Response time
Gantt chart calculations

**Process A**
- Turnaround time
- Wait time
- Response time

**Process B**
- Turnaround time
- Wait time
- Response time

**Process C**
- Turnaround time
- Wait time
- Response time
Scheduling algorithms

- First-Come, First-Served \([\text{nonpreemptive}]\)
- Shortest Job First \([\text{nonpreemptive}]\)
- Shortest Remaining Time \([\text{preemptive}]\)
- Round-Robin \([\text{preemptive}]\)
- Priority Queue \([\text{preemptive}]\)
First-Come, First-Served (FCFS)

- Nonpreemptive

- Implementation is easily managed with FIFO queue.

- When a process enters the ready queue, it is put on the end of the queue.

- When the CPU is free, scheduler chooses a process from the head of ready queue.

- Adv. – Simplest of all CPU-scheduling algorithms

- Disadv. – The average turnaround time might be long with FCFS
First-Come, First-Served: nonpreemption

<table>
<thead>
<tr>
<th>Process</th>
<th>Burst Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>24 ms</td>
</tr>
<tr>
<td>$P_2$</td>
<td>3 ms</td>
</tr>
<tr>
<td>$P_3$</td>
<td>3 ms</td>
</tr>
</tbody>
</table>

- Arrival time = 0 ms
  - in order of $P_1 > P_2 > P_3$

- Response time:
  - $P_1 = 0$ ms
  - $P_2 = 24$ ms
  - $P_3 = 27$ ms

- Average waiting time:
  - $(0 + 24 + 27) / 3 = 51/3 = 17$ ms

- Average turnaround time:
  - $(24 + 27 + 30) / 3 = 81/3 = 27$ ms
Shortest Job First (SJF)

- Nonpreemptive
- Pick shortest job as a priority; with least demand on CPU
- When all the jobs are available simultaneously, SJF algorithm is optimal in that it gives the minimal average turnaround time for a given set of processes.
- Adv. – optimal; gives the minimum average waiting time. Great for small jobs.
- Disadv. – works best only in long-term scheduling
Shortest Job First: nonpreemption

- Arrival time = 0 ms
  - in order of \( P_1 > P_2 > P_3 > P_4 \)

- Response time:
  - \( P_1 = 3 \) ms
  - \( P_2 = 16 \) ms
  - \( P_3 = 9 \) ms
  - \( P_4 = 0 \) ms

- Average waiting time:
  - \( (3 + 16 + 9 + 0) / 4 = 28/4 = 7 \) ms

- Average turnaround time:
  - \( (9 + 24 + 16 + 3) / 4 = 52/4 = 13 \) ms
Shortest Remaining Time (SRT)

- *Preemptive* version of SJF

- Occurs when a new process arrives at the ready queue while previous process is executing

- Because the newly arrived process is shorter than what is left of the current executing process... *preemption* occurs!
Shortest Remaining Time: preemption

<table>
<thead>
<tr>
<th>Process</th>
<th>Arrival Time</th>
<th>Burst Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_1</td>
<td>0</td>
<td>8 ms</td>
</tr>
<tr>
<td>P_2</td>
<td>1</td>
<td>4 ms</td>
</tr>
<tr>
<td>P_3</td>
<td>2</td>
<td>9 ms</td>
</tr>
<tr>
<td>P_4</td>
<td>3</td>
<td>5 ms</td>
</tr>
</tbody>
</table>

- **Response time:**
  - $P_1 = 0$ ms
  - $P_2 = 0$ ms
  - $P_3 = 15$ ms
  - $P_4 = 2$ ms

- **Average waiting time:**
  - $((10-1) + (1-1) + (17-2) + (5-3)) / 4 = 26/4 = 6.5$ ms

- **Average turnaround time:**
  - $((17-0) + (5-1) + (26-2) + (10-3)) / 4 = 52/4 = 13$ ms
Round-Robin (RR)

- Designed especially for the timesharing systems
- Similar to FCFS scheduling, but *preemption* is added to switch between processes.
  - *time quantum* (Q) – small unit of time
- Ready queue is treated as a circular queue
- The main issue with round-robin algorithm is the length of the quantum.
  - If quantum is too short, CPU time are wasted for process switching – low CPU utilization.
  - If quantum is too long, slow response time.
Round-Robin: preemption

<table>
<thead>
<tr>
<th>Process</th>
<th>Burst Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>P₁</td>
<td>24 ms</td>
</tr>
<tr>
<td>P₂</td>
<td>3 ms</td>
</tr>
<tr>
<td>P₃</td>
<td>3 ms</td>
</tr>
</tbody>
</table>

- Time quantum (Q) = 4 ms
- Response time:
  - P₁ = 0 ms
  - P₂ = 4 ms
  - P₃ = 7 ms
- Average waiting time:
  - \((10 - 4 + 4 + 7) / 3 = 17/3\)
  - = 5.66 ms
- Average turnaround time:
  - \((30 + 7 + 10) / 3 = 47/3\)
  - = 15.66 ms
Priority Queue (PQ)

- Preemptive

- Priority is associated with each process

- CPU is allocated to the process with highest priority
  - Equal priority scheduled with FCFS or RR

- Disadv. – starvation of lower priority processes
  - A solution to the problem of starvation is *aging*
  - *Aging* – technique of gradually increasing the priorities of processes that wait in the system for a long time
Priority Queue: preemption

<table>
<thead>
<tr>
<th>Process</th>
<th>Arrival Time</th>
<th>Burst Time</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>P₁</td>
<td>0</td>
<td>5 ms</td>
<td>2</td>
</tr>
<tr>
<td>P₂</td>
<td>1</td>
<td>2 ms</td>
<td>3</td>
</tr>
<tr>
<td>P₃</td>
<td>2</td>
<td>2 ms</td>
<td>1</td>
</tr>
</tbody>
</table>

- **Response time:**
  - \( P₁ = 0 \text{ ms} \)
  - \( P₂ = 6 \text{ ms} \)
  - \( P₃ = 0 \text{ ms} \)

- **Average waiting time:**
  - \( \frac{[4-2-0 + (7-1) + (2-2)]}{3} = \frac{8}{3} = 2.67 \text{ ms} \)

- **Average turnaround time:**
  - \( \frac{[7-0 + (9-1) + (4-2)]}{3} = \frac{17}{3} = 5.67 \text{ ms} \)