Speedup and Scheduling

CS 241
Oct. 4, 2013
One of the greatest advantages of parallelism is to speed up computation. We formally define this speedup as: 

\[ S_P = \frac{T_1}{T_P} \]

- \( P \): The number of processors
- \( S_P \): The speedup for a given number of processors
- \( T_1 \): The execution time of a sequential algorithm
- \( T_P \): The execution time of a parallel algorithm on \( P \) processors

**Ideal Speedup:**
Scheduling

• When do we need scheduling?
Scheduling

• Basic scheduling decision:
High-Level Scheduling Example

• Processes:

• Schedule:
Scheduling Goals

• Throughput:

• Latency:

• Fairness:
Quantitative Metrics

• Waiting Time:

• Turnaround Time:

• Response Time:
FCFS: First Come First Serve

- FCFS: Schedule tasks in the order they arrived at the scheduler.

<table>
<thead>
<tr>
<th>Process</th>
<th>Duration</th>
<th>Priority</th>
<th>Arrival Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>24</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>P2</td>
<td>3</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>P3</td>
<td>4</td>
<td>3</td>
<td>7</td>
</tr>
</tbody>
</table>

Schedule:

Time: 0

<table>
<thead>
<tr>
<th>Process</th>
<th>Waiting Time</th>
<th>Turnaround</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Average:
SJF: Shortest Job First

- SJF: Scheduling the job with the shortest running time first (non-preemptive).

<table>
<thead>
<tr>
<th>Process</th>
<th>Duration</th>
<th>Priority</th>
<th>Arrival</th>
<th>Waiting Time</th>
<th>Turnaround Time</th>
<th>Response Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>6</td>
<td>2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>8</td>
<td>6</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td>7</td>
<td>3</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P4</td>
<td>3</td>
<td>5</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Schedule:

Time: 0
PRI: Fixed Priority Scheduling

- PRI: Schedule the job with the highest priority first. *(Low number ➔ Higher priority)*

<table>
<thead>
<tr>
<th>Process</th>
<th>Duration</th>
<th>Priority</th>
<th>Arrival</th>
<th>Waiting Time</th>
<th>Turnaround Time</th>
<th>Response Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>6</td>
<td>2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>8</td>
<td>6</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td>7</td>
<td>3</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P4</td>
<td>3</td>
<td>5</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Schedule:

Time: 0
Scheduling Example

• Initially, two processes in the scheduler:
  – P1 (duration = 4, priority = 1, arrival = 0 )
  – P2 (duration = 100, priority = 100, arrival = 0 )

• Every three seconds, a new process arrives:
  – Px (duration = 4, priority = 1, arrival = 0 )

• FCFS:
• SJF:
• PRI:
Starvation

• A scheduling algorithm may cause **starvation** if, for any job, it cannot guarantee that the job will run within a finite amount of time.
  
  – This is not **deadlock**!
Advantages / Disadvantages

• FCFS:

• SJF:

• PRI: