CPU SCHEDULING

Outline

• Scheduling Objectives
• Levels of Scheduling
• Scheduling Criteria
• Scheduling Algorithms
  • FCFS, Shortest Job First, Priority, Round Robin
• Multiple Processor Scheduling
• Real-time Scheduling
• Algorithm Evaluation

Scheduling Objectives

• Enforcement of fairness
  – in allocating resources to processes
• Enforcement of priorities
• Make best use of available system resources
• Give preference to processes holding key resources.
• Give preference to processes exhibiting good behavior.
• Degrade gracefully under heavy loads.

Basic Concepts

• Maximum CPU utilization obtained with multiprogramming.
• CPU-I/O Burst Cycle
  • Process execution consists of a cycle of CPU execution and I/O wait.
Levels of Scheduling

• High Level Scheduling or Job Scheduling
  • Selects jobs allowed to compete for CPU and other system resources.

• Intermediate Level Scheduling or Medium Term Scheduling
  • Selects which jobs to temporarily suspend/resume to smooth fluctuations in system load.

• Low Level (CPU) Scheduling or Dispatching
  • Selects the ready process that will be assigned the CPU.
  • Ready Queue contains PCBs of processes.

CPU Scheduler

• Selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them.
  – Non-preemptive Scheduling
    • Once CPU has been allocated to a process, the process keeps the CPU until
      – Process exits OR
      – Process switches to waiting state
  – Preemptive Scheduling
    • Process can be interrupted and must release the CPU.
      – Need to coordinate access to shared data

Scheduling Criteria

• CPU Utilization
  • Keep the CPU and other resources as busy as possible

• Throughput
  • Number of processes that complete their execution per time unit.

• Turnaround time
  • Amount of time to execute a particular process from its entry time.
Scheduling Criteria (cont.)

- Waiting time
  - amount of time a process has been waiting in the ready queue.
- Response Time (in a time-sharing environment)
  - amount of time it takes from when a request was submitted until the first response is produced, NOT output.

Optimization Criteria

- Max CPU Utilization
- Max Throughput
- Min Turnaround time
- Min Waiting time
- Min response time

First Come First Serve (FCFS) Scheduling

- Policy: Process that requests the CPU FIRST is allocated the CPU FIRST.
  - FCFS is a non-preemptive algorithm.
- Implementation - using FIFO queues
  - incoming process is added to the tail of the queue.
  - Process selected for execution is taken from head of queue.
- Performance metric - Average waiting time in queue.
- Gantt Charts are used to visualize schedules.

First-Come, First-Served (FCFS) Scheduling

- Example
  - Process Burst Time
    - P1 24
    - P2 3
    - P3 3
  - Gantt Chart for Schedule
    - Suppose the arrival order for the processes is P1, P2, P3
    - Waiting time
      - P1 = 0;
      - P2 = 24;
      - P3 = 27;
    - Average waiting time
      - (0 + 24 + 27) / 3 = 17
FCFS Scheduling (cont.)

- Example

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

<table>
<thead>
<tr>
<th>Gantt Chart for Schedule</th>
</tr>
</thead>
<tbody>
<tr>
<td>P2</td>
</tr>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

- Suppose the arrival order for the processes is
  - P2, P3, P1
- Waiting time
  - P1 = 6; P2 = 0; P3 = 3;
- Average waiting time
  - \((6+0+3)/3 = 3\), better.
- Convoy Effect:
  - short process behind long process, e.g. 1 CPU bound process, many I/O bound processes.

Shortest-Job-First (SJF) Scheduling

- Associate with each process the length of its next CPU burst. Use these lengths to schedule the process with the shortest time.
- Two Schemes:
  - Scheme 1: Non-preemptive
    - Once CPU is given to the process it cannot be preempted until it completes its CPU burst.
  - Scheme 2: Preemptive
    - If a new CPU process arrives with CPU burst length less than remaining time of current executing process, preempt. Also called Shortest-Remaining-Time-First (SRTF).
- SJF is optimal - gives minimum average waiting time for a given set of processes.

Non-Preemptive SJF Scheduling

- Example

<table>
<thead>
<tr>
<th>Process</th>
<th>Arrival Time</th>
<th>Burst Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>P2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>P3</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>P4</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Gantt Chart for Schedule</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
</tr>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

Average waiting time = \((0+6+3+7)/4 = 4\)

Preemptive SJF Scheduling

- Example

<table>
<thead>
<tr>
<th>Process</th>
<th>Arrival Time</th>
<th>Burst Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>P2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>P3</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>P4</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Gantt Chart for Schedule</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
</tr>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

Average waiting time = \((9+1+0+2)/4 = 3\)
Priority Scheduling

- A priority value (integer) is associated with each process. Can be based on
  - Cost to user
  - Importance to user
  - Aging
  - %CPU time used in last X hours.
- CPU is allocated to process with the highest priority.
  - Preemptive
  - Nonpreemptive

Priority Scheduling (cont.)

- SJN is a priority scheme where the priority is the predicted next CPU burst time.
- Problem
  - Starvation!! - Low priority processes may never execute.
- Solution
  - Aging - as time progresses increase the priority of the process.

Round Robin (RR)

- Each process gets a small unit of CPU time
  - Time quantum usually 10-100 milliseconds.
  - After this time has elapsed, the process is preempted and added to the end of the ready queue.
- \( n \) processes, time quantum = \( q \)
  - Each process gets \( 1/n \) CPU time in chunks of at most \( q \) time units at a time.
  - No process waits more than \((n-1)q\) time units.
- Performance
  - Time slice \( q \) too large - FIFO behavior
  - Time slice \( q \) too small - Overhead of context switch is too expensive.
  - Heuristic - 70-80% of jobs block within timeslice

Round Robin Example

- Time Quantum = 20

<table>
<thead>
<tr>
<th>Process</th>
<th>Burst Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>53</td>
</tr>
<tr>
<td>P2</td>
<td>17</td>
</tr>
<tr>
<td>P3</td>
<td>68</td>
</tr>
<tr>
<td>P4</td>
<td>24</td>
</tr>
</tbody>
</table>

Gantt Chart for Schedule

<table>
<thead>
<tr>
<th>Time</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P1</th>
<th>P3</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>P1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>P2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>57</td>
<td></td>
<td>P3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>77</td>
<td></td>
<td></td>
<td>P4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>97</td>
<td></td>
<td></td>
<td></td>
<td>P1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>117</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>121</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P3</td>
<td>P3</td>
</tr>
</tbody>
</table>
Multiple-Processor Scheduling

- CPU scheduling becomes more complex when multiple CPUs are available.
  - Have one ready queue accessed by each CPU.
    - Self scheduled - each CPU dispatches a job from ready Q
    - Master-Slave - one CPU schedules the other CPUs
- Homogeneous processors within multiprocessor.
  - Permits Load Sharing
- Asymmetric multiprocessing
  - only 1 CPU runs kernel, others run user programs
  - alleviates need for data sharing

Real-Time Scheduling

- Hard Real-time Computing -
  - required to complete a critical task within a guaranteed amount of time.
- Soft Real-time Computing -
  - requires that critical processes receive priority over less fortunate ones.
- Types of real-time Schedulers
  - Periodic Schedulers - Fixed Arrival Rate
  - Demand-Driven Schedulers - Variable Arrival Rate
  - Deadline Schedulers - Priority determined by deadline
  - .....
Algorithm Evaluation

- **Deterministic Modeling**
  - Takes a particular predetermined workload and defines the performance of each algorithm for that workload. Too specific, requires exact knowledge to be useful.

- **Queuing Models and Queuing Theory**
  - Use distributions of CPU and I/O bursts. Knowing arrival and service rates - can compute utilization, average queue length, average wait time etc...
  - Little's formula - $n = \lambda \times W$ where $n$ is the average queue length, $\lambda$ is the avg. arrival rate and $W$ is the avg. waiting time in queue.

- **Other techniques: Simulations, Implementation**