Scheduling

Reading:
Silberschatz
chapter 6

Additional Reading:
Stallings
chapter 9
Outline

- Introduction
- Types of Scheduling
- Scheduling Criteria
- FCFS Scheduling
- Shortest-Job-First Scheduling
- Priority Scheduling
- Round Robin Scheduling
- Multilevel Queue Scheduling
- Multiprocessor Scheduling
  - Load Balancing
  - Symmetric Multithreading
- Algorithm Evaluation
- Real Time Scheduling
- Scheduling Examples
  - Windows XP, 2000
  - Linux
Introduction

➤ Basic Points

- Process Scheduling, Thread Scheduling
- Max CPU Utilization $\rightarrow$ Multiprogramming
- CPU Burst $\leftrightarrow$ I/O Burst
- CPU Burst Distribution
Histogram – CPU Burst Duration
Scheduling Types

- **CPU Scheduling Decisions**
  - Running → Waiting state
    - e.g. I/O Request, wait by Parent
  - Running → Ready state
    - e.g. Interrupt
  - Waiting → Ready state
    - e.g. Completion of I/O
  - Process Termination

- **Nonpreemptive Scheduling**

- **Preemptive Scheduling**
  - Associated Cost
  - Design of OS Kernel
    - Process → Kernel, wait for sys call or I/O completion *before* context switch
Scheduling Criteria

- **CPU Utilization** – How busy is the CPU?

- **Throughput** – Number of processes that are completed per unit time

- **Turnaround Time** – How long to execute a process? Submission ↔ Completion

- **Waiting Time** – Sum of periods spent in ready queue

- **Response Time** – Process Request → First response
Optimization Criteria

- Max CPU utilization
- Max throughput
- Min turnaround time
- Min waiting time
- Min response time

- Conflicting goals! Requires careful balance
- Average, Min/Max, Variance
**FCFS Scheduling**

<table>
<thead>
<tr>
<th>Process</th>
<th>Burst Time</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>

Arrivals in the order: $P_1$, $P_2$, $P_3$

The Gantt Chart for the schedule:

<table>
<thead>
<tr>
<th></th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>24</td>
<td>27</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td>30</td>
</tr>
</tbody>
</table>

- Waiting time $P_1 = 0$; $P_2 = 24$; $P_3 = 27$
- Average waiting time $\frac{(0 + 24 + 27)}{3} = 17$
FCFS Scheduling

Say, if the processes arrive in the order \( P_2, P_3, P_1 \)

- The Gantt chart for the schedule:

```
<table>
<thead>
<tr>
<th>P2</th>
<th>P3</th>
<th>P1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

- Waiting time \( P_1 = 6; P_2 = 0; P_3 = 3 \)
- Average waiting time \( (6 + 0 + 3)/3 = 3 \), Previously 17
- Convoy effect \( \rightarrow \) Short process behind long process
- Nonpreemptive \( \rightarrow \) Problem for time sharing systems
SJF Scheduling

- CPU assigned to process with smallest next CPU burst, Tie $\rightarrow$ FCFS

- Shortest-next-CPU-burst algorithm

- Major difficulty
  - Estimating the processing time of each job, Predicting the Next!
  - Long running jobs may starve, steady supply of short jobs to CPU

- SJF is optimal – minimum average waiting time
### SJF Scheduling

<table>
<thead>
<tr>
<th>Process</th>
<th>Arrival Time</th>
<th>Burst Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>0.0</td>
<td>7</td>
</tr>
<tr>
<td>$P_2$</td>
<td>2.0</td>
<td>4</td>
</tr>
<tr>
<td>$P_3$</td>
<td>4.0</td>
<td>1</td>
</tr>
<tr>
<td>$P_4$</td>
<td>5.0</td>
<td>4</td>
</tr>
</tbody>
</table>

- **SJF (non-preemptive)**

- **Average waiting time** $\rightarrow (0 + 6 + 3 + 7)/4 = 4$
SJF Scheduling

<table>
<thead>
<tr>
<th>Process</th>
<th>Arrival Time</th>
<th>Burst Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>0.0</td>
<td>7</td>
</tr>
<tr>
<td>$P_2$</td>
<td>2.0</td>
<td>4</td>
</tr>
<tr>
<td>$P_3$</td>
<td>4.0</td>
<td>1</td>
</tr>
<tr>
<td>$P_4$</td>
<td>5.0</td>
<td>4</td>
</tr>
</tbody>
</table>

- SJF (preemptive)

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

- A priority number (integer) associated with each process
  - SJF – A Priority scheduling
  - Equal Priority - FCFS

- CPU → Process with the highest priority, High ↔ Low
  - Preemptive
  - Nonpreemptive

- Defining Priorities
  - Internally, Measurable Quantities
    - Memory required, time limits, # open files, ratio of avg I/O to CPU burst, etc.
  - Externally, Outside OS
    - Importance of Process, type/amount of funds, etc.

- Starvation
  - Low priority processes may never execute

- Solution?
  - Aging
Round Robin (RR)

- Each process gets a small unit of CPU time
  - Time Quantum (*time-slice*)
    - usually 10-100 milliseconds
  - Time elapsed $\rightarrow$ Preempted
    - If not completed $\rightarrow$ end of the ready queue

- RR reduces penalty for short jobs in FCFS

- Critical Issue $\rightarrow$ Length of quantum, $q$
  - $q$ large $\rightarrow$ FIFO or FCFS
  - $q$ small $\rightarrow$ Context switch overhead
Round Robin (RR)

<table>
<thead>
<tr>
<th>Process</th>
<th>Burst Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P_1 )</td>
<td>53</td>
</tr>
<tr>
<td>( P_2 )</td>
<td>17</td>
</tr>
<tr>
<td>( P_3 )</td>
<td>68</td>
</tr>
<tr>
<td>( P_4 )</td>
<td>24</td>
</tr>
</tbody>
</table>

➢ The Gantt chart is:

```
0  20  37  57  77  97  117  121  134  154  162
P_1 P_2 P_3 P_4 P_1 P_3 P_4 P_1 P_3 P_3
```

➢ Typically, higher average turnaround than SJF, but better *response*. 
Round Robin (RR)

![Diagram of Round Robin scheduling](image)

Process time = 10

<table>
<thead>
<tr>
<th>quantum</th>
<th>context switches</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
</tr>
</tbody>
</table>
Round Robin (RR)

![Graph showing average turnaround time vs. time quantum. The graph includes a table with processes and their corresponding times: $P_1$ with 6, $P_2$ with 3, $P_3$ with 1, and $P_4$ with 7.]
Multilevel Queue

- Ready queue → separate queues
  - foreground (interactive)
  - background (batch)
- Each queue → own scheduling algorithm, e.g.
  - foreground – RR
  - background – FCFS
- Scheduling must be done between the queues
  - Fixed priority scheduling; (i.e., serve all from foreground then from background), Starvation
  - Time slice – each queue gets a certain amount of CPU time which it can schedule amongst its processes; i.e., 80% to foreground in RR
  - 20% to background in FCFS
Multilevel Queue Scheduling

highest priority

- system processes

- interactive processes

- interactive editing processes

- batch processes

- student processes

lowest priority
Multilevel Feedback Queue

- Separate processes → CPU burst characteristics
- Process moves *up* ↔ *down* in queues
  - Too much time ↓
  - Aging ↑
- Key points
  - number of queues
  - scheduling algorithms for each queue
  - method used to determine when to upgrade a process
  - method used to determine when to demote a process
  - method used to determine which queue a process will enter when that process needs service
Example of Multilevel Feedback Queue

quantum = 8

quantum = 16

FCFS

Primary CPU Scheduling

CPU
Multiple-Processor Scheduling

- Multiple CPUs → High scheduling complexity
- Homogeneous Processors
  - Asymmetric Multiprocessing
    - No data sharing, System data structures → one processor
  - Symmetric Multiprocessing
    - Self Scheduling, Ready queue
- Processor Affinity
  - Soft Vs Hard affinity
- Load Balancing
  - Push Migration
  - Pull Migration
Algorithm Evaluation

- Deterministic modeling
  - Takes a particular predetermined workload and defines the performance of each algorithm for that workload

- Queueing models
  - Queue of network servers
  - Little’s formula, $l = \lambda \times w$
    - $\lambda$ - Avg arrival rate, $w$ - Avg waiting time, $l$ - Avg queue length

- Simulation
  - Model, clock
  - Simulation $\rightarrow$ modifies system with clock $\uparrow$
  - Distribution driven simulation
  - Only # instances of an event, order?
Evaluation of CPU Schedulers by Simulation
Real-Time Scheduling

- **Hard real-time systems**
  - Complete a *critical task* within a guaranteed time
  - Admit or Reject
  - Impossible with SS, VM
  - Resource Reservation

- **Soft real-time computing**
  - Critical processes receive priority over less fortunate ones
  - General-purpose systems → Multimedia, Graphics
  - *Priority Inversion*
  - *Priority-inheritance protocol*
Example: Windows XP, 2000

- Scheduling
  - Priority-based, preemptive scheduling
  - Thread runs → preempted by higher priority thread, terminates, Qu
  - Does not guarantee execution of a real-time thread within time-limit

- Thread Priorities
  - 32 level priority scheme
  - Real time class → 16-32
  - Variable class → 1-15
  - Memory Management → Thread at 0 priority
  - Six Classes (Win32 API) – 1 + 5
  - Within each 6 classes – 7 relative priorities
  - Currently selected foreground process → Scheduling Quantum ↑ 3
## Windows XP, 2000 Priorities

<table>
<thead>
<tr>
<th>Priority Classes</th>
<th>real-time</th>
<th>high</th>
<th>above normal</th>
<th>normal</th>
<th>below normal</th>
<th>idle priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>time-critical</td>
<td>31</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>highest</td>
<td>26</td>
<td>15</td>
<td>12</td>
<td>10</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>above normal</td>
<td>25</td>
<td>14</td>
<td>11</td>
<td>9</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>normal</td>
<td>24</td>
<td>13</td>
<td>10</td>
<td>8</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>below normal</td>
<td>23</td>
<td>12</td>
<td>9</td>
<td>7</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>lowest</td>
<td>22</td>
<td>11</td>
<td>8</td>
<td>6</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>idle</td>
<td>16</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Relative Priority ↓

Priority Classes →

Base Priority
Example: Linux

- Scheduling
  - Increased support for SMP, Scaling with # tasks
  - Processor affinity, load balancing
  - High priority tasks → longer quanta, vice-versa
  - Real time tasks – static priorities
  - Rest dynamic → nice values ± 5 (interactivity)

- Numeric Priorities
  - 0-140 level priority scheme
  - Real time → 0-99
  - Nice values → 100-140