Uniprocessor Scheduling

Chapter 9
Operating Systems: Internals and Design Principles, 6/E
William Stallings
CPU- and I/O-bound processes

Bursts of CPU usage alternate with periods of I/O wait
Goals of Scheduling

All systems
- Fairness - giving each process a fair share of the CPU
- Policy enforcement - seeing that stated policy is carried out
- Balance - keeping all parts of the system busy

Batch systems
- Throughput - maximize jobs per hour
- Turnaround time - minimize time between submission and termination
- CPU utilization - keep the CPU busy all the time

Interactive systems
- Response time - respond to requests quickly
- Proportionality - meet users’ expectations

Real-time systems
- Meeting deadlines - avoid losing data
- Predictability - avoid quality degradation in multimedia systems
Types of Scheduling

- Long-term: admission into the system
- Medium-term: between main memory and hard disk
- Short-term/Dispatcher: between ready and running
- I/O
Scheduling and Process State Transitions

New

Ready/Suspend

Ready

Running

Exit

Long-term scheduling

Ready/Suspend

Ready

Running

Exit

Blocked/Suspend

Blocked

Exit

Medium-term scheduling

Short-term scheduling

Medium-term scheduling
Levels of Scheduling
Long-Term Scheduling
• Determines which programs are admitted to the system for processing
• Controls the degree of multiprogramming
• More processes, smaller percentage of time each process is executed

Medium-term Scheduling
• Part of the swapping function
• Based on the need to manage the degree of multiprogramming
Short-Term Scheduling

• Known as the dispatcher
• Executes most frequently
• Invoked when an event occurs that triggers process switch
  – Clock interrupts
  – I/O interrupts
  – Operating system calls
  – Signals
Priorities

• Scheduler chooses a process of higher priority over one of lower priority
• Have multiple ready queues to represent each level of priority
• Lower-priority may suffer starvation
  – Change process priority based on its age or execution history – dynamic allocation of priorities
Priority Queuing

Diagram showing the flow of processes through various queues and the dispatch to the processor for execution or release.
Decision Mode

• Non-preemptive
  – Once a process is in the running state, it will continue until it terminates or blocks for I/O

• Preemptive
  – Currently running process may be interrupted and moved to the Ready state by the OS
  – Allows for better service, since no process can monopolize the processor for very long
<table>
<thead>
<tr>
<th>Process</th>
<th>Arrival Time</th>
<th>Service Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>C</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>D</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>E</td>
<td>8</td>
<td>2</td>
</tr>
</tbody>
</table>
First-Come-First-Served (non-preemptive)

Each process joins the Ready queue
When the current process ceases to execute, the oldest process in the Ready queue is selected
First-Come-First-Served (2)

- A short process may have to wait a very long time before it can execute
- Favours CPU-bound processes
  - I/O processes have to wait until CPU-bound process completes
Round Robin

• Uses preemption based on a clock
• Importance of the length of the quantum!!!
Round Robin (2)

- Clock interrupt is generated at periodic intervals (dozens of msec)
- When an interrupt occurs, the currently running process is placed in the ready queue
- Next ready job is selected
- Known as time slicing
- Importance of quantum – responsiveness
- Bad service for I/O-bound processes
Virtual Round-Robin
Shortest Process Next

- Non-preemptive policy
- Process with shortest expected processing time is selected next
- Short process jumps ahead of longer processes
Shortest Process Next (2)

• OS has to estimate remaining running time or length of next CPU burst – simple or weighted average: for some $0<\alpha<1$
  
  \[ S(n+1) = \alpha T(n) + (1-\alpha)S(n) \]
  
  \[ = \alpha T(n) + (1-\alpha)\alpha T(n-1) + \ldots + (1-\alpha)^n T(1) \]

• Possibility of starvation for longer processes
Exponential Smoothing Coefficients

The graph above illustrates the distribution of exponential smoothing coefficients for different values of $\alpha$. The coefficients are shown for $\alpha = 0.2$, $\alpha = 0.5$, and $\alpha = 0.8$. The $x$-axis represents the age of observation, while the $y$-axis shows the coefficient value.

- For $\alpha = 0.2$, the coefficients are relatively high and decrease sharply as the age of observation increases.
- For $\alpha = 0.5$, the coefficients decrease more gradually compared to $\alpha = 0.2$.
- For $\alpha = 0.8$, the coefficients are even lower and decrease even more gradually.

This graph can be used to understand how different smoothing parameters affect the smoothing of time series data over time.
Use Of Exponential Averaging
Use Of Exponential Averaging (2)

The graph illustrates the use of exponential averaging with different values of \( \alpha \). The observed value is represented by squares, and the simple average is shown by diamonds. The lines with circles and dots represent the exponential average for \( \alpha = 0.8 \) and \( \alpha = 0.5 \), respectively. The x-axis represents time, and the y-axis shows the observed or average value.
Shortest Remaining Time

- Preemptive version of shortest process next policy
- Must estimate processing time
Highest Response Ratio Next

• Choose next process with the greatest

\[ \text{Ratio} = \frac{\text{time spent waiting} + \text{expected service time}}{\text{expected service time}} \]
Feedback Scheduling

Figure 9.10  Feedback Scheduling
Feedback

- Penalize jobs that have been running longer
- Don’t know remaining time process needs to execute
<table>
<thead>
<tr>
<th></th>
<th>FCFS</th>
<th>Round robin</th>
<th>SPN</th>
<th>SRT</th>
<th>HRRN</th>
<th>Feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Selection function</strong></td>
<td>max[w]</td>
<td>constant</td>
<td>min[s]</td>
<td>min[s – e]</td>
<td>max (\frac{w + s}{s})</td>
<td>(see text)</td>
</tr>
<tr>
<td><strong>Decision mode</strong></td>
<td>Non-preemptive</td>
<td>Preemptive (at time quantum)</td>
<td>Non-preemptive</td>
<td>Preemptive (at arrival)</td>
<td>Non-preemptive</td>
<td>Preemptive (at time quantum)</td>
</tr>
<tr>
<td><strong>Through-Put</strong></td>
<td>Not emphasized</td>
<td>May be low if quantum is too small</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td><strong>Response time</strong></td>
<td>May be high, especially if there is a large variance in process execution times</td>
<td>Provides good response time for short processes</td>
<td>Provides good response time for short processes</td>
<td>Provides good response time</td>
<td>Provides good response time</td>
<td>Not emphasized</td>
</tr>
<tr>
<td><strong>Overhead</strong></td>
<td>Minimum</td>
<td>Minimum</td>
<td>Can be high</td>
<td>Can be high</td>
<td>Can be high</td>
<td>Can be high</td>
</tr>
<tr>
<td><strong>Effect on processes</strong></td>
<td>Penalizes short processes; penalizes I/O bound processes</td>
<td>Fair treatment</td>
<td>Penalizes long processes</td>
<td>Penalizes long processes</td>
<td>Good balance</td>
<td>May favor I/O bound processes</td>
</tr>
<tr>
<td><strong>Starvation</strong></td>
<td>No</td>
<td>No</td>
<td>Possible</td>
<td>Possible</td>
<td>No</td>
<td>Possible</td>
</tr>
<tr>
<td>Process</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>Mean</td>
</tr>
<tr>
<td>------------------------</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>------</td>
</tr>
<tr>
<td>Arrival Time</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Service Time ($T_s$)</td>
<td>3</td>
<td>6</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

**FCFS**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finish Time</td>
<td>3</td>
<td>9</td>
<td>13</td>
<td>18</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Turnaround Time ($T_r$)</td>
<td>3</td>
<td>7</td>
<td>9</td>
<td>12</td>
<td>12</td>
<td>8.60</td>
</tr>
<tr>
<td>$T_r/T_s$</td>
<td>1.00</td>
<td>1.17</td>
<td>2.25</td>
<td>2.40</td>
<td>6.00</td>
<td>2.56</td>
</tr>
</tbody>
</table>

**RR $q = 1$**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finish Time</td>
<td>4</td>
<td>18</td>
<td>17</td>
<td>20</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>Turnaround Time ($T_r$)</td>
<td>4</td>
<td>16</td>
<td>13</td>
<td>14</td>
<td>7</td>
<td>10.80</td>
</tr>
<tr>
<td>$T_r/T_s$</td>
<td>1.33</td>
<td>2.67</td>
<td>3.25</td>
<td>2.80</td>
<td>3.50</td>
<td>2.71</td>
</tr>
</tbody>
</table>

**RR $q = 4$**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finish Time</td>
<td>3</td>
<td>17</td>
<td>11</td>
<td>20</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>Turnaround Time ($T_r$)</td>
<td>3</td>
<td>15</td>
<td>7</td>
<td>14</td>
<td>11</td>
<td>10.00</td>
</tr>
<tr>
<td>$T_r/T_s$</td>
<td>1.00</td>
<td>2.5</td>
<td>1.75</td>
<td>2.80</td>
<td>5.50</td>
<td>2.71</td>
</tr>
</tbody>
</table>

**SPN**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finish Time</td>
<td>3</td>
<td>9</td>
<td>15</td>
<td>20</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>Turnaround Time ($T_r$)</td>
<td>3</td>
<td>7</td>
<td>11</td>
<td>14</td>
<td>3</td>
<td>7.60</td>
</tr>
<tr>
<td>$T_r/T_s$</td>
<td>1.00</td>
<td>1.17</td>
<td>2.75</td>
<td>2.80</td>
<td>1.50</td>
<td>1.84</td>
</tr>
</tbody>
</table>

**SRT**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finish Time</td>
<td>3</td>
<td>15</td>
<td>8</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>Turnaround Time ($T_r$)</td>
<td>3</td>
<td>13</td>
<td>4</td>
<td>14</td>
<td>2</td>
</tr>
<tr>
<td>$T_r/T_s$</td>
<td>1.00</td>
<td>2.17</td>
<td>1.00</td>
<td>2.80</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>HRRN</td>
<td>FB $q = 1$</td>
<td>FB $q = 2^i$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------</td>
<td>---------------------------</td>
<td>-----------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Finish Time</td>
<td>3</td>
<td>9</td>
<td>13</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Turnaround Time ($T_r$)</td>
<td>3</td>
<td>7</td>
<td>9</td>
<td>14</td>
</tr>
<tr>
<td>$T_r/T_s$</td>
<td>1.00</td>
<td>1.17</td>
<td>2.25</td>
<td>2.80</td>
<td>3.5</td>
</tr>
</tbody>
</table>

|                | Finish Time  | 4  | 20 | 16 | 19 | 11 | 10.00 |
|                | Turnaround Time ($T_r$) | 4  | 18 | 12 | 13 | 3  | 2.29  |
| $T_r/T_s$      | 1.33        | 3.00 | 3.00 | 2.60 | 1.5 | 2.29 |

|                | Finish Time  | 4  | 17 | 18 | 20 | 14 | 10.60 |
|                | Turnaround Time ($T_r$) | 4  | 15 | 14 | 14 | 6  | 2.63  |
| $T_r/T_s$      | 1.33        | 2.50 | 3.50 | 2.80 | 3.00 | 2.63 |
Figure 9.14 Simulation Results for Normalized Turnaround Time
Fair-Share Scheduling

• User’s application runs as a collection of processes (threads)
• User is concerned about the performance of the application
• Need to make scheduling decisions based on process sets
Policy versus Mechanism

• Separate what is allowed to be done from how it is done
  – a process knows which of its children threads are important and need priority
• Scheduling algorithm parameterized
  – mechanism in the kernel
• Parameters filled in by user processes
  – policy set by user process for its threads
• Lottery scheduling
Traditional UNIX Scheduling

• Multilevel feedback using round robin within each of the priority queues
• If a running process does not block or complete within the quantum, it is preempted
• Priorities are recomputed once per second
The UNIX scheduler is based on a multilevel queue structure.