Chapter 10: Multiprocessor & Real-time Scheduling
Multiprocessor classes

- Loosely coupled (clusters)
- Special-purpose processors (CPU + GPU)
- Tightly coupled (SMP)
Synchronization granularity

- Independent
- Very coarse-grained
- Coarse-grained
- Medium-grained
- Fine-grained
Design issues
Design issues

- Assignment of process to CPU
  - Dynamic vs. static
  - Master/slave vs. peer
- Multiprogramming
- Process dispatching
(a) Comparison of RR and FCFS

(b) Comparison of SRT and FCFS
Lower number \[ \approx \]
identical process service times
B, C, D, ..., J arrive at time 1
B, C, D, ..., J arrive at time 1
<table>
<thead>
<tr>
<th>Process</th>
<th>High CoV</th>
<th>Low CoV</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$T_s$</td>
<td>FCFS</td>
</tr>
<tr>
<td>A</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>103</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>105</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>107</td>
</tr>
<tr>
<td>F</td>
<td>2</td>
<td>109</td>
</tr>
<tr>
<td>G</td>
<td>2</td>
<td>111</td>
</tr>
<tr>
<td>H</td>
<td>2</td>
<td>113</td>
</tr>
<tr>
<td>I</td>
<td>2</td>
<td>115</td>
</tr>
<tr>
<td>J</td>
<td>2</td>
<td>117</td>
</tr>
<tr>
<td>A</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>F</td>
<td>2</td>
<td>11</td>
</tr>
<tr>
<td>G</td>
<td>2</td>
<td>13</td>
</tr>
<tr>
<td>H</td>
<td>2</td>
<td>15</td>
</tr>
<tr>
<td>I</td>
<td>2</td>
<td>17</td>
</tr>
<tr>
<td>J</td>
<td>2</td>
<td>19</td>
</tr>
</tbody>
</table>

B, C, D, ..., J arrive at time 1
B, C, D, ..., J arrive at time 1
B, C, D, ..., J arrive at time 1
B, C, D, ..., J arrive at time 1
### High CoV

<table>
<thead>
<tr>
<th>Process</th>
<th>$T_s$</th>
<th>FCFS</th>
<th>RR</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>100</td>
<td>100</td>
<td>108</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>F</td>
<td>2</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>G</td>
<td>2</td>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>H</td>
<td>2</td>
<td>14</td>
<td>9</td>
</tr>
<tr>
<td>I</td>
<td>2</td>
<td>16</td>
<td>9</td>
</tr>
<tr>
<td>J</td>
<td>2</td>
<td>18</td>
<td>10</td>
</tr>
</tbody>
</table>

### Low CoV

<table>
<thead>
<tr>
<th>Process</th>
<th>$T_s$</th>
<th>FCFS</th>
<th>RR</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>F</td>
<td>2</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>G</td>
<td>2</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>H</td>
<td>2</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>I</td>
<td>2</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>J</td>
<td>2</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

B, C, D, ..., J arrive at time 1
(a) Comparison of RR and FCFS

(b) Comparison of SRT and FCFS
Thread scheduling

- Load sharing
- Gang sharing
- Dedicated processor assignment
- Dynamic scheduling
Load sharing
Load sharing

CPU

Core
Thread
Core

Memory

Thread
Thread
Thread
Thread
Thread

Monday, October 10, 2011
Load sharing
Gang scheduling

Figure 10.3 Example of Scheduling Groups with Four and One Threads [FEIT90b]
Gang scheduling
Dedicated processor assignment

- Extreme version of gang scheduling
- No multiprogramming (blocked thread)
- Utilization not emphasized
- No context switch gives speedup
Dedicated processor assignment

- Extreme version of gang scheduling
- No multiprogramming (blocked thread)
- Utilization not emphasized
- **No context switch** gives speedup
16 CPUs simultaneous execution

Figure 10.4 Application Speedup as a Function of Number of Threads
16 CPUs simultaneous execution

Figure 10.4 Application Speedup as a Function of Number of Threads
Real-time systems
Real-time scheduling

- Correctness
  - Logical result of computation
  - Time results produced
- Hard vs. soft real-time
- Periodic vs. aperiodic
Characteristics

- Determinism
- Responsiveness
- User control
- Reliability
- Fail-soft operation
Characteristics

- Determinism
  - start of processing time
- Responsiveness
  - service time
- User control
  - policy changes
- Reliability
  - failure == bad
- Fail-soft operation
  - online recovery
RTOS features
RTOS features

- Fast context switch
- Small size
- Quick interrupt response
- IPC/synchronization
- Sequential files
- Priority scheduling
- Minimal interrupt disabling
- Primitives to delay tasks
- Alarms/timeouts
RTOS features

- Fast context switch
- Small size
- Quick interrupt response
- IPC/synchronization
- Sequential files
- Priority scheduling
- Minimal interrupt disabling
- Primitives to delay tasks
- Alarms/timeouts

watchdog timer
(a) Round-robin Preemptive Scheduler

Request from a real-time process
Real-time process added to run queue to await its next slice

Process 1  Process 2

Process n  Real-time process

Clock tick

Scheduling time

(b) Priority-Driven Nonpreemptive Scheduler

Request from a real-time process
Real-time process added to head of run queue

Current process  Real-time process

Scheduling time

Current process blocked or completed

(c) Priority-Driven Preemptive Scheduler on Preemption Points

Request from a real-time process
Wait for next preemption point

Current process  Real-time process

Scheduling time
Request from a real-time process

Real-time process added to run queue to await its next slice

(a) Round-robin Preemptive Scheduler

BAD

Request from a real-time process

Real-time process added to head of run queue

Current process

Real-time process

Scheduling time

(b) Priority-Driven Nonpreemptive Scheduler

Request from a real-time process

Wait for next preemption point

Current process

Real-time process

Preemption point

Scheduling time

(c) Priority-Driven Preemptive Scheduler on Preemption Points
(a) Round-robin Preemptive Scheduler

BAD

(b) Priority-Driven Nonpreemptive Scheduler

CRITICAL

(c) Priority-Driven Preemptive Scheduler on Preemption Points
Real-time scheduling

• Static table-driven
• Static priority-driven
• Dynamic planning-based
• Dynamic best effort
Real-time scheduling

- Static table-driven: set schedule for processes
- Static priority-driven: set priorities, not schedule
- Dynamic planning-based: can deny processes
- Dynamic best effort: abort if deadline missed
Deadline scheduling
Deadline scheduling

• Ready time
• Starting deadline
• Completion deadline
• Processing time
• Resource requirements
• Priority
• Subtask structure
<table>
<thead>
<tr>
<th>Process</th>
<th>Arrival Time</th>
<th>Execution Time</th>
<th>Ending Deadline</th>
</tr>
</thead>
<tbody>
<tr>
<td>A(1)</td>
<td>0</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>A(2)</td>
<td>20</td>
<td>10</td>
<td>40</td>
</tr>
<tr>
<td>A(3)</td>
<td>40</td>
<td>10</td>
<td>60</td>
</tr>
<tr>
<td>A(4)</td>
<td>60</td>
<td>10</td>
<td>80</td>
</tr>
<tr>
<td>A(5)</td>
<td>80</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>B(1)</td>
<td>0</td>
<td>25</td>
<td>50</td>
</tr>
<tr>
<td>B(2)</td>
<td>50</td>
<td>25</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Figure 10.6 Scheduling of Periodic Real-time Tasks with Completion Deadlines (based on Table 10.2)
Figure 10.6 Scheduling of Periodic Real-time Tasks with Completion Deadlines (based on Table 10.2)
Figure 10.6 Scheduling of Periodic Real-time Tasks with Completion Deadlines (based on Table 10.2)
Figure 10.6 Scheduling of Periodic Real-time Tasks with Completion Deadlines (based on Table 10.2)
Figure 10.7 Scheduling of Aperiodic Real-time Tasks with Starting Deadlines
**Figure 10.7** Scheduling of Aperiodic Real-time Tasks with Starting Deadlines
<table>
<thead>
<tr>
<th>Process</th>
<th>Arrival Time</th>
<th>Execution Time</th>
<th>Starting Deadline</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>10</td>
<td>20</td>
<td>110</td>
</tr>
<tr>
<td>B</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>C</td>
<td>40</td>
<td>20</td>
<td>50</td>
</tr>
<tr>
<td>D</td>
<td>50</td>
<td>20</td>
<td>90</td>
</tr>
<tr>
<td>E</td>
<td>60</td>
<td>20</td>
<td>70</td>
</tr>
</tbody>
</table>

Figure 10.7 Scheduling of Aperiodic Real-time Tasks with Starting Deadlines
### Table 10.1: Scheduling of Aperiodic Real-time Tasks with Starting Deadlines

<table>
<thead>
<tr>
<th>Process</th>
<th>Arrival Time</th>
<th>Execution Time</th>
<th>Starting Deadline</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>10</td>
<td>20</td>
<td>110</td>
</tr>
<tr>
<td>B</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>C</td>
<td>40</td>
<td>20</td>
<td>50</td>
</tr>
<tr>
<td>D</td>
<td>50</td>
<td>20</td>
<td>90</td>
</tr>
<tr>
<td>E</td>
<td>60</td>
<td>20</td>
<td>70</td>
</tr>
</tbody>
</table>

![Figure 10.7 Scheduling of Aperiodic Real-time Tasks with Starting Deadlines](image)

**Figure 10.7** Scheduling of Aperiodic Real-time Tasks with Starting Deadlines
Rate monotonic scheduling

Rate = \frac{1}{T}

Higher rate ==> higher priority
Figure 10.8 A Task Set with RMS [WARR91]
Rate monotonic theorem

- Rate monotonic scheduling is **optimal**
- If any static priority scheduler meets all deadlines, so does RMS
- Does not guarantee best utilization