Chapter 7: Scheduling

Process Scheduler
- Why do we even need a process scheduler?
  - In simplest form, CPU must be shared by
    - OS
    - Application
  - In reality, [multiprogramming]
    - OS: many separate pieces (processes)
    - Many Applications
- Scheduling [Policy] addresses...
  - When to remove a process from CPU?
  - Which ready process to allocate the CPU to?

Model of Process Execution

Recall Resource Manager

Scheduler as CPU Res Mgr

Scheduler Components
**Context Switch**
- Processes are switched out using Context Switching
- Context Switch:
  - **Save** pertinent info for current process
    - PC, Register, Status, etc.
  - **Update** PC, Register, Status, etc.
    - with info for process selected to run
- Switching User Process
  - 2 Context switches (CTX)
    - Process 1 running CTX
    - Dispatcher selects next process
    - Process 2 running

**Process Context**
- Right Operand
- Left Operand
- Functional Unit
- Status Registers
- ALU
- PC
- IR
- Ctrl Unit

**Invoking the Scheduler**
- Need a mechanism to call the scheduler
- Voluntary call
  - Process blocks itself
  - Calls the scheduler
- Involuntary call
  - External force (interrupt) blocks the process
  - Calls the scheduler

**Contemporary Scheduling**
- Involuntary CPU sharing – timer interrupts
  - *Time quantum* determined by interval timer – usually fixed size for every process using the system
  - Sometimes called the *time slice length*

**Choosing a Process to Run**
- *Mechanism* never changes
- Strategy = policy the dispatcher uses to select a process from the ready list
- Different policies for different requirements
Policy Considerations
- Policy can control/influence:
  - CPU utilization
  - Average time a process waits for service
  - Average amount of time to complete a job
- Could strive for any of:
  - Equitability
  - Favor very short or long jobs
  - Meet priority requirements
  - Meet deadlines

Optimal Scheduling
- Suppose the scheduler knows each process $p_i$’s service time, $\tau(p_i)$ -- or it can estimate each $\tau(p_i)$:
- Policy can optimize on any criteria, e.g.,
  - CPU utilization
  - Waiting time
  - Deadline
- To find an optimal schedule:
  - Have a finite, fixed # of $p_i$
  - Know $\tau(p_i)$ for each $p_i$
  - Enumerate all schedules, then choose the best

However ...
- The $\tau(p_i)$ are almost certainly just estimates
- General algorithm to choose optimal schedule is $O(n^2)$
- Other processes may arrive while these processes are being serviced
- Usually, optimal schedule is only a theoretical benchmark -- scheduling policies try to approximate an optimal schedule

Model of Process Execution

Selection Strategies
- Motivation
  - To “optimize” some aspect of system behavior
- Considerations
  - Priority of process
    - External : assigned
    - Internal : aging
  - Fairness : no starvation
  - Overall Resource Utilization
  - ...

Selection Strategies...
- Considerations...
  - Turnaround time
    - Average time / job
  - Throughput
    - Jobs / time unit
  - Response time
  - System availability
  - Deadlines
Talking About Scheduling ...

- Let $P = \{p_i \mid 0 \leq i < n\} =$ set of processes
- Let $S(p_i) \in \{\text{running, ready, blocked}\}$
- Let $t(p_i) =$ Time process needs to be in running state (the service time)
- Let $W(p_i) =$ Time $p_i$ is in ready state before first transition to running (wait time)
- Let $T_{\text{Trnd}}(p_i) =$ Time from $p_i$ first enter ready to last exit ready (turnaround time)
- Batch Throughput rate = inverse of avg $T_{\text{Trnd}}$
- Timesharing response time = $W(p_i)$

Definition & Terms

- Time Quantum
  - Amount of time between timer interrupts
  - Also called Time Slice

- Service Time $T(p_i)$
  - Amount of time process needs to be in Running state (acquired CPU) before it is completed

- Wait Time $W(p_i)$
  - Time a process spends waiting in the Ready state before its first transition to the Running state

Definition & Terms...

- Turnaround Time $T(p_i)$
  - Amount of time between moment process first enters Ready state and the moment the process exits Running state for the last time (completed)

- Service time, Wait time & Turnaround time are measurable metrics used to compare scheduling algorithms

Simplified Model

- Simplified, but still provide analysis result
- Easy to analyze performance

Classes of Scheduling Algorithms

- 2 major classes
  - Non-preemptive
    - Run to completion
  - Preemptive
    - Process with highest priority always gets CPU

Recall: Several ways to establish priority

Nonpreemptive Schedulers

- Try to use the simplified scheduling model
- Only consider running and ready states
- Ignores time in blocked state:
  - "New process created when it enters ready state"
  - "Process is destroyed when it enters blocked state"
  - Really just looking at "small phases" of a process
First-Come-First-Served

\[ T_{\text{max}}(p_0) = T(p_0) = 350 \quad W(p_0) = 0 \]

First-Come-First-Served

\[ T_{\text{max}}(p_0) = T(p_0) = 350 \quad W(p_0) = 0 \]

\[ T_{\text{max}}(p_1) = (T(p_1) + T_{\text{max}}(p_0)) = 125 + 350 = 475 \]
\[ W(p_1) = T_{\text{max}}(p_0) = 350 \]

\[ T_{\text{max}}(p_2) = (T(p_2) + T_{\text{max}}(p_1)) = 475 + 475 = 950 \]
\[ W(p_2) = T_{\text{max}}(p_1) = 475 \]

\[ T_{\text{max}}(p_3) = (T(p_3) + T_{\text{max}}(p_2)) = 250 + 950 = 1200 \]
\[ W(p_3) = T_{\text{max}}(p_2) = 475 \]

\[ T_{\text{max}}(p_4) = (T(p_4) + T_{\text{max}}(p_3)) = 475 + 1200 = 1675 \]
\[ W(p_4) = T_{\text{max}}(p_3) = 950 \]

\[ \text{avg} = \frac{(0 + 350 + 475 + 950 + 1200)}{5} = \frac{3275}{5} = 655 \]

**FCFS Average Wait Time**

\[ \text{avg} = \frac{(0 + 350 + 475 + 950 + 1200)}{5} = \frac{3275}{5} = 655 \]

• Easy to implement
• Ignores service time, etc
• Not a great performer

**W_{avg}** = \( \frac{(0 + 350 + 475 + 950 + 1200)}{5} \times 0.5 = 297.5 \times 0.5 = 148.75 \)

\[ W_{avg} = 148.75 \]

\[ \text{avg} = \frac{(0 + 350 + 475 + 950 + 1200)}{5} = \frac{3275}{5} = 655 \]

**W_{avg}** = \( \frac{(0 + 350 + 475 + 950 + 1200)}{5} \times 0.5 = 297.5 \times 0.5 = 148.75 \)

\[ W_{avg} = 148.75 \]
Shortest Job Next

- Minimizes wait time
- May starve large jobs
- Must know service times
### Priority Scheduling

<table>
<thead>
<tr>
<th>i</th>
<th>( t(p_i) )</th>
<th>( \text{Pi} )</th>
<th>( d_i )</th>
<th>( w_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>350</td>
<td>5</td>
<td>850</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>125</td>
<td>2</td>
<td>200</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>475</td>
<td>3</td>
<td>125</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>250</td>
<td>1</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>75</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\( T_{\text{TRnd}}(p_i) = t(p_i) + t(p_{i+1}) + t(p_{i+2}) + t(p_{i+3}) \)

\( W(p_0) = 0 \)

\( W(p_1) = 50 \)

\( W(p_2) = 100 \)

\( W(p_3) = 0 \)

\( W(p_4) = 50 \)

### Deadline Scheduling

<table>
<thead>
<tr>
<th>i</th>
<th>( t(p_i) )</th>
<th>( \text{Deadline} )</th>
<th>( w_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>350</td>
<td>575</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>125</td>
<td>550</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>475</td>
<td>1050</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>250</td>
<td>(none)</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>75</td>
<td>200</td>
<td>0</td>
</tr>
</tbody>
</table>

### Preemptive Schedulers

- Highest priority process is guaranteed to be running at all times
- Or at least at the beginning of a time slice
- Dominant form of contemporary scheduling
- But complex to build & analyze

### Round Robin (TQ=50)

<table>
<thead>
<tr>
<th>i</th>
<th>( t(p_i) )</th>
<th>( \text{TQ} = 50 )</th>
<th>( w_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>350</td>
<td>50</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>125</td>
<td>50</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>475</td>
<td>50</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>250</td>
<td>50</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>75</td>
<td>50</td>
<td>0</td>
</tr>
</tbody>
</table>

\( W(p_0) = 0 \)

\( W(p_1) = 50 \)

\( W(p_2) = 100 \)

\( W(p_3) = 0 \)

\( W(p_4) = 50 \)
Round Robin (TQ=50)

- $w(p_0) = 0$
- $w(p_1) = 50$
- $w(p_2) = 100$
- $w(p_3) = 150$
- $w(p_4) = 200$

$T_{\text{Round}}(p_1) = 475$

$T_{\text{Round}}(p_2) = 475$

$T_{\text{Round}}(p_4) = 550$

$T_{\text{Round}}(p_3) = 950$

$T_{\text{Round}}(p_0) = 200$

Round Robin (TQ=50)

- $w(p_0) = 0$
- $w(p_1) = 50$
- $w(p_2) = 100$
- $w(p_3) = 150$
- $w(p_4) = 200$

$T_{\text{Round}}(p_0) = 0$

$T_{\text{Round}}(p_1) = 475$

$T_{\text{Round}}(p_4) = 475$

$T_{\text{Round}}(p_3) = 950$

$T_{\text{Round}}(p_2) = 550$

$T_{\text{Round}}(p_0) = 200$
Round Robin (TQ=50)

- Equitable
- Most widely-used
- Fits naturally with interval timer

$T_{RTD}(p_1) = 1100$
$T_{RTD}(p_2) = 550$
$T_{RTD}(p_3) = 950$
$T_{RTD}(p_4) = 475$

$W(p_1) = 0$
$W(p_2) = 50$
$W(p_3) = 100$
$W(p_4) = 200$

$W_{avg} = \frac{0 + 50 + 100 + 150 + 200}{5} = \frac{500}{5} = 100$

RR with Overhead=10 (TQ=50)

- Overhead must be considered

$T_{RTD}(p_1) = 1320$
$T_{RTD}(p_2) = 660$
$T_{RTD}(p_3) = 1535$
$T_{RTD}(p_4) = 1140$
$T_{RTD}(p_5) = 565$

$W(p_1) = 0$
$W(p_2) = 60$
$W(p_3) = 120$
$W(p_4) = 180$
$W(p_5) = 240$

$W_{avg} = \frac{0 + 60 + 120 + 180 + 240 + 300 + 360 + 420 + 480 + 540 + 600 + 660 + 720 + 780 + 840 + 900 + 960 + 1020 + 1080 + 1140 + 1200 + 1260 + 1320}{20} = \frac{20400}{20} = 1020$

Multi-Level Queues

- Preemption or voluntary yield
- All processes at level i run before any process at level j
- At a level, use another policy, e.g. RR

Contemporary Scheduling

- Involuntary CPU sharing -- timer interrupts
  - Time quantum determined by interval timer -- usually fixed for every process using the system
  - Sometimes called the time slice length
- Priority-based process (job) selection
  - Select the highest priority process
  - Priority reflects policy
- With preemption
- Usually a variant of Multi-Level Queues
BSD 4.4 Scheduling
- Involuntary CPU Sharing
- Preemptive algorithms
- 32 Multi-Level Queues
  - Queues 0-7 are reserved for system functions
  - Queues 8-31 are for user space functions
  - nice influences (but does not dictate) queue level

UNIX Scheduler
The UNIX scheduler is based on a multilevel queue structure

Process Life Cycle

Job and Process Scheduler

- Controls when jobs will be allowed to contend the CPU
- Most popular techniques
  - **FIFO** First in, first out
  - **SJF** Shortest job first

- Controls when individual jobs (processes) will actually get the CPU
- Only interesting in multi-programming
- Most popular technique is Round Robin
  - Give each process one time slice in turn until complete

Turnaround and Weighted Turnaround Time

- Let: \( N \) be number of jobs
- \( A_i \) be arrival time of i-th job
- \( F_i \) be finish time of i-th job

Turnaround time for i-th job: \( T_i = F_i - A_i \)

Average turnaround time for i-th job: \( T = \Sigma T_i / N \)

Weighted turnaround time for i-th job: \( \text{WT}_i = (F_i - A_i) / (\text{Service-time})_i \)

Average Weighted Turnaround time:
\( \text{WT} = \Sigma \text{WT}_i / N \)

Processor Sharing (PS) "Theoretical" Scheduling Algorithm
- Limit of RR as time quantum goes to zero.
- Like giving each CPU cycle to a different process, in round robin fashion.
- \( N \) processes scheduled by PS
  - Each job runs on dedicated \( N \)-fold slower CPU.
  - Thus, READY = RUNNING.
- CPU Time "shared" equally among processes
**Scheduling Example 2**

Assume:

- Multiprogramming
- FIFO Job Scheduling
- Processor Sharing Process Scheduling

<table>
<thead>
<tr>
<th>Job</th>
<th>Arrives</th>
<th>Run Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10.0</td>
<td>0.3</td>
</tr>
<tr>
<td>2</td>
<td>10.2</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>10.4</td>
<td>0.1</td>
</tr>
<tr>
<td>4</td>
<td>10.5</td>
<td>0.4</td>
</tr>
<tr>
<td>5</td>
<td>10.8</td>
<td>0.1</td>
</tr>
</tbody>
</table>

---

**Example 2 Continued**

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th># Jobs</th>
<th>Headway</th>
<th>Time Left</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0</td>
<td>1 A,S</td>
<td>1</td>
<td>0.3</td>
<td>1.00</td>
</tr>
<tr>
<td>10.2</td>
<td>2 A,S</td>
<td>1</td>
<td>0.2</td>
<td>1.00</td>
</tr>
<tr>
<td>10.4</td>
<td>1 F</td>
<td>2</td>
<td>0.1</td>
<td>2.00</td>
</tr>
<tr>
<td>10.5</td>
<td>3 A,S</td>
<td>2</td>
<td>0.05</td>
<td>2.00</td>
</tr>
<tr>
<td>10.65</td>
<td>3 F</td>
<td>2</td>
<td>0.05</td>
<td>2.00</td>
</tr>
</tbody>
</table>

---

**T and W for Example 2**

<table>
<thead>
<tr>
<th>Job</th>
<th>Run</th>
<th>Start</th>
<th>Finish</th>
<th>Ti</th>
<th>WTi</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.3</td>
<td>10.0</td>
<td>10.4</td>
<td>0.4</td>
<td>1.33</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>10.2</td>
<td>11.35</td>
<td>1.15</td>
<td>2.3</td>
</tr>
<tr>
<td>3</td>
<td>0.1</td>
<td>10.4</td>
<td>10.65</td>
<td>0.25</td>
<td>2.5</td>
</tr>
<tr>
<td>4</td>
<td>0.4</td>
<td>10.5</td>
<td>11.4</td>
<td>0.9</td>
<td>2.25</td>
</tr>
<tr>
<td>5</td>
<td>0.1</td>
<td>10.8</td>
<td>11.1</td>
<td>0.3</td>
<td>3.0</td>
</tr>
</tbody>
</table>

Check:

Because CPU was never idle, 1.4 + 10.0 must equal time of last event (11.4)

---

**Definitions**

- Number processes in ready queue before this event:
- Current event: A=Arrival
- S=Schedule
- F=Finish
- H=In Hold

**Example 2 Continued...**

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th># Jobs</th>
<th>Headway</th>
<th>Time Left</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.8</td>
<td>5 A,S</td>
<td>2</td>
<td>0.075</td>
<td>2.00</td>
</tr>
<tr>
<td>11.1</td>
<td>5 F</td>
<td>3</td>
<td>0.1</td>
<td>2.00</td>
</tr>
<tr>
<td>11.35</td>
<td>2 F</td>
<td>2</td>
<td>0.125</td>
<td>4.00</td>
</tr>
<tr>
<td>11.40</td>
<td>4 F</td>
<td>1</td>
<td>0.05</td>
<td>4.00</td>
</tr>
</tbody>
</table>

---

**Scheduling Example 4**

Assume:

- FIFO Job Scheduling
- Processor Sharing Process Scheduling
- 5 Tape Drives
- 100 K Main Memory

<table>
<thead>
<tr>
<th>Job</th>
<th>Arrives</th>
<th>Run Time</th>
<th>Memory</th>
<th>Tapes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0</td>
<td>0.5</td>
<td>30</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1.2</td>
<td>1.0</td>
<td>50</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1.3</td>
<td>1.5</td>
<td>50</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1.4</td>
<td>2.0</td>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>1.7</td>
<td>0.5</td>
<td>30</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>2.1</td>
<td>1.0</td>
<td>30</td>
<td>2</td>
</tr>
</tbody>
</table>
Example 4 Continued

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th># Jobs</th>
<th>HWay</th>
<th>MM</th>
<th>Tapes</th>
<th>Time Left</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>1 A,S</td>
<td>1</td>
<td>0.2</td>
<td>20</td>
<td>3</td>
<td>1.0</td>
</tr>
<tr>
<td>1.2</td>
<td>2 A,S</td>
<td>1</td>
<td>0.2</td>
<td>20</td>
<td>2</td>
<td>1.0</td>
</tr>
<tr>
<td>1.3</td>
<td>3 A,H</td>
<td>2</td>
<td>0.05</td>
<td>20</td>
<td>2</td>
<td>0.25</td>
</tr>
<tr>
<td>1.4</td>
<td>4 A,S</td>
<td>2</td>
<td>0.05</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>1.7</td>
<td>5 A,H</td>
<td>3</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>0.1</td>
</tr>
<tr>
<td>2.0</td>
<td>1 F</td>
<td>3</td>
<td>0.1</td>
<td>30</td>
<td>2</td>
<td>0.7</td>
</tr>
</tbody>
</table>

**T and W for Example 4**

<table>
<thead>
<tr>
<th>Job</th>
<th>Run</th>
<th>Arrives</th>
<th>Finish</th>
<th>Ti</th>
<th>WTi</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5</td>
<td>1.0</td>
<td>2.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
<td>1.2</td>
<td>4.05</td>
<td>2.85</td>
<td>2.85</td>
</tr>
<tr>
<td>3</td>
<td>1.5</td>
<td>1.3</td>
<td>7.4</td>
<td>6.1</td>
<td>4.06</td>
</tr>
<tr>
<td>4</td>
<td>2.0</td>
<td>1.4</td>
<td>6.6</td>
<td>5.2</td>
<td>2.6</td>
</tr>
<tr>
<td>5</td>
<td>0.5</td>
<td>1.7</td>
<td>7.5</td>
<td>5.8</td>
<td>11.6</td>
</tr>
<tr>
<td>6</td>
<td>2.1</td>
<td>2.1</td>
<td>5.1</td>
<td>3.0</td>
<td>3.0</td>
</tr>
</tbody>
</table>

T = 3.99  WT = 4.35

---

Example 4 Continued ...

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th># Jobs</th>
<th>HWay</th>
<th>MM</th>
<th>Tapes</th>
<th>Time Left</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>6 A,S</td>
<td>2</td>
<td>0.05</td>
<td>0</td>
<td>0</td>
<td>2.065</td>
</tr>
<tr>
<td>4.05</td>
<td>2 F</td>
<td>3</td>
<td>0.65</td>
<td>50</td>
<td>1</td>
<td>4.115</td>
</tr>
<tr>
<td>4.1</td>
<td>3 S</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6.035</td>
</tr>
<tr>
<td>5.1</td>
<td>6 F</td>
<td>3</td>
<td>0.35</td>
<td>30</td>
<td>2</td>
<td>4.75</td>
</tr>
<tr>
<td>6.6</td>
<td>4 F</td>
<td>2</td>
<td>0.75</td>
<td>50</td>
<td>4</td>
<td>3.04</td>
</tr>
<tr>
<td>5.5</td>
<td>5 S</td>
<td>20</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>5.05</td>
</tr>
<tr>
<td>7.4</td>
<td>3 F</td>
<td>2</td>
<td>0.4</td>
<td>70</td>
<td>2</td>
<td>5.1</td>
</tr>
<tr>
<td>7.5</td>
<td>5 F</td>
<td>1</td>
<td>0.1</td>
<td>100</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

T = 3.99  WT = 4.35