Introduction

An important aspect of multiprogramming is scheduling. The resources that are scheduled are IO and processors.

The goal is to achieve

- High processor utilization
- High throughput
  - number of processes completed per unit time
- Low response time
  - time elapse from the submission of a request to the beginning of the response
Topics for discussion

- Motivation
- Types of scheduling
- Short-term scheduling
- Various scheduling criteria
- Various algorithms
  - Priority queues
  - First-come, first-served
  - Round-robin
  - Shortest process first
  - Shortest remaining time and others
- Queuing Model and Performance Analysis

The CPU-I/O Cycle

- We observe that processes require alternate use of processor and I/O in a repetitive fashion
- Each cycle consist of a CPU burst (typically of 5 ms) followed by a (usually longer) I/O burst
- A process terminates on a CPU burst
- CPU-bound processes have longer CPU bursts than I/O-bound processes
CPU/IO Bursts

- Bursts of CPU usage alternate with periods of I/O wait
  - a CPU-bound process
  - an I/O bound process

Motivation

- Consider these programs with processing-component and IO-component indicated by upper-case and lower-case letters respectively.
  - JOB A: A1 a1 A2 a2 A3
    - CPU: 0 30 50 80 120 130
    - I/O: 0 20 40 60
  - JOB B: B1 b1 B2
    - CPU: 0 20 40 60
    - I/O: 0 10 20 60 80 100 110 130 140 150
  - JOB C:
Motivation

- The starting and ending time of each component are indicated beneath the symbolic references (A1, b1 etc.)
- Now lets consider three different ways for scheduling: no overlap, round-robin, simple overlap.
- Compare utilization \( U = \frac{\text{time CPU busy}}{\text{total run time}} \)

Scheduling Criteria

- CPU utilization – keep the CPU as busy as possible
- Throughput – # of processes that complete their execution per time unit
- Turnaround time – amount of time to execute a particular process
- Waiting time – amount of time a process has been waiting in the ready queue
- Response time – amount of time it takes from when a request was submitted until the first response is produced, not output (for time-sharing environment)
Optimization Criteria

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

Types of scheduling

- Long-term: To add to the pool of processes to be executed.
- Medium-term: To add to the number of processes that are in the main memory.
- Short-term: Which of the available processes will be executed by a processor?
- IO scheduling: To decide which process’s pending IO request shall be handled by an available IO device.
Classification of Scheduling Activity

- Long-term: which process to admit
- Medium-term: which process to swap in or out
- Short-term: which ready process to execute next

First-Come, First-Served (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>

Suppose that the processes arrive in the order: $P_1, P_2, P_3$

The Gantt Chart for the schedule is:

```
  0 24 27 30
P_1
P_2
P_3
```

Waiting time for $P_1 = 0$; $P_2 = 24$; $P_3 = 27$

Average waiting time: $(0 + 24 + 27)/3 = 17$
FCFS Scheduling (Cont.)

Suppose that the processes arrive in the order $P_2$, $P_3$, $P_1$.

The Gantt chart for the schedule is:

<table>
<thead>
<tr>
<th></th>
<th>$P_2$</th>
<th>$P_3$</th>
<th>$P_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>6</td>
<td>30</td>
</tr>
</tbody>
</table>

- Waiting time for $P_1 = 6$; $P_2 = 0$; $P_3 = 3$
- Average waiting time: $(6 + 0 + 3)/3 = 3$
- Much better than previous case.
- Convoy effect: short process behind long process.

Shortest-Job-First (SJR) 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:
- nonpreemptive – once CPU given to the process it cannot be preempted until completes its CPU burst.
- preemptive – if a new process arrives with CPU burst length less than remaining time of current executing process, preempt. This scheme is known as the Shortest-Remaining-Time-First (SRTF).

- SJF is optimal – gives minimum average waiting time for a given set of processes.
Example of Non-Preemptive SJF

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

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

Example of Preemptive SJF

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

Average waiting time = \((9 + 1 + 0 + 2)/4 = 3\)
Determining Length of Next CPU Burst

- Can only estimate the length.
- Can be done by using the length of previous CPU bursts, using exponential averaging.

1. $t_n = \text{actual length of } n^{th} \text{CPU burst}$
2. $\tau_{n+1} = \text{predicted value for the next CPU burst}$
3. $\alpha, 0 \leq \alpha \leq 1$
4. Define: $\tau_{n+1} = \alpha t_n + (1 - \alpha) \tau_n$

Examples of Exponential Averaging

- $\alpha = 0$
  - $\tau_{n+1} = \tau_n$
  - Recent history does not count.
- $\alpha = 1$
  - $\tau_{n+1} = t_n$
  - Only the actual last CPU burst counts.
- If we expand the formula, we get:
  
  $\tau_{n+1} = \alpha t_n + (1 - \alpha) \alpha t_{n-1} + \ldots$
  
  $\alpha (1 - \alpha) t_{n-2} + \ldots$
  
  $\alpha (1 - \alpha)^{n-1} t_1$
- Since both $\alpha$ and $(1 - \alpha)$ are less than or equal to 1, each successive term has less weight than its predecessor.
More on Exponential Averaging

- \( S[n+1] \) next burst, \( s[n] \) current burst
- \( S[n+1] = \alpha T[n] + (1-\alpha) S[n] \); \( 0 < \alpha < 1 \)
- more weight is put on recent instances whenever \( \alpha > 1/n \)
- By expanding this eqn, we see that weights of past instances are decreasing exponentially
  - \( S[n+1] = \alpha T[n] + (1-\alpha)\alpha T[n-1] + \ldots + (1-\alpha)^i \alpha T[n-i] \)
  - \ldots + (1-\alpha)^n S[1]
- predicted value of 1st instance \( S[1] \) is not calculated; usually set to 0 to give priority to new processes

Exponentially Decreasing Coefficients

![Graph showing exponentially decreasing coefficients](image)

- \( \alpha = 0.2 \)
- \( \alpha = 0.5 \)
- \( \alpha = 0.8 \)

Age of Observation

Coefficient Value
Shortest Process Next: critique

- Possibility of starvation for longer processes as long as there is a steady supply of shorter processes
- Lack of preemption is not suited in a time sharing environment
  - CPU bound process gets lower priority (as it should) but a process doing no I/O could still monopolize the CPU if he is the first one to enter the system
- SPN implicitly incorporates priorities: shortest jobs are given preferences
- The next (preemptive) algorithm penalizes directly longer jobs

Priority Scheduling

- A priority number (integer) is associated with each process
- The CPU is allocated to the process with the highest priority (smallest integer = highest priority).
  - Preemptive
  - nonpreemptive
- SJF is a priority scheduling where 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.
- If there are *n* processes in the ready queue and the time quantum is *q*, then each process gets 1/*n* of the CPU time in chunks of at most *q* time units at once. No process waits more than (*n-1*)*q* time units.

Performance

- *q* large ⇒ FIFO
- *q* small ⇒ *q* must be large with respect to context switch, otherwise overhead is too high.

Example of RR with Time Quantum = 20

<table>
<thead>
<tr>
<th>Process</th>
<th>Burst Time</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>P</em>₁</td>
<td>53</td>
</tr>
<tr>
<td><em>P</em>₂</td>
<td>17</td>
</tr>
<tr>
<td><em>P</em>₃</td>
<td>68</td>
</tr>
<tr>
<td><em>P</em>₄</td>
<td>24</td>
</tr>
</tbody>
</table>

The Gantt chart is:

Typically, higher average turnaround than SJF, but better *response*. 
Various Metrics

- Turnaround time = Finish time - Arrival time
- Normalized turnaround time = Turnaround time / service time
- Response time = arrival time - start time
- Overall wait time = response time + wait times in the ready queue (ready to run, but CPU not avail)

Scheduling in Real-Time Systems

- Schedulable real-time system
- Rate Monotonic Scheduling:
- Given
  - $m$ periodic events
  - event $i$ occurs within period $P_i$ and requires $C_i$ seconds
- Then the load can only be handled if
  \[ \sum_{i=1}^{m} \frac{C_i}{P_i} \leq 1 \]
Summary

- Scheduling is important for improving the system performance.
- Methods of prediction play an important role in Operating system and network functions.
- Simulation is a way of experimentally evaluating the performance of a technique.