Scheduling

Uwe R. Zimmer - The Australian National University
References for this chapter

[Ben2006]
Ben-Ari, M
Principles of Concurrent and Distributed Programming
second edition, Prentice-Hall 2006

[AdaRM2012]
ISO/IEC 8652:201x (E)

[Stallings2001]
Stallings, William
Operating Systems
Prentice Hall, 2001
Motivation and definition of terms

Purpose of scheduling
Motivation and definition of terms

Purpose of scheduling

Two scenarios for scheduling algorithms:

1. Ordering resource assignments (CPU time, network access, …).
   - live, on-line application of scheduling algorithms.

2. Predicting system behaviours under anticipated loads.
   - simulated, off-line application of scheduling algorithms.

Predictions are used:

- at compile time: to confirm the feasibility of the system, or to predict resource needs, …
- at run time: to permit admittance of new requests or for load-balancing, …
## Motivation and definition of terms

### Criteria

<table>
<thead>
<tr>
<th>Performance criteria:</th>
<th>Predictability criteria:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Process / user perspective:</strong></td>
<td>minimize deviation from given ...</td>
</tr>
<tr>
<td><strong>Waiting time</strong></td>
<td>value / minima / maxima value / minima / maxima / deadlines value / minima / maxima / deadlines</td>
</tr>
<tr>
<td><strong>Response time</strong></td>
<td>maximize the ...</td>
</tr>
<tr>
<td><strong>Turnaround time</strong></td>
<td>minima / maxima / average / variance</td>
</tr>
<tr>
<td><strong>System perspective:</strong></td>
<td>minima / maxima / average</td>
</tr>
<tr>
<td><strong>Throughput</strong></td>
<td>CPU busy time</td>
</tr>
<tr>
<td><strong>Utilization</strong></td>
<td></td>
</tr>
</tbody>
</table>

© 2018 Uwe R. Zimmer, The Australian National University
Definition of terms

Time scales of scheduling
**Definition of terms**

**Time scales of scheduling**

---

- **CPU**
- **pre-emption or cycle done**
- **dispatch**
- **ready**
- **ready, suspended**
- **swap-in**
- **block or synchronize**
- **unblock**
- **suspend (swap-out)**
- **blocked**
- **blocked, suspended**
- **swap-out**
- **Medium-term**

---

© 2018 Uwe R. Zimmer, The Australian National University
Definition of terms

Time scales of scheduling
Tasks have an average time between instantiations of $T_i$ and a constant computation time of $C_i$. 
Performance scheduling

First come, first served (FCFS)

As tasks apply concurrently for resources, the actual sequence of arrival is non-deterministic. Hence even a deterministic scheduling schema like FCFS can lead to different outcomes.

Waiting time: 0..11, average: 5.9 – Turnaround time: 3..12, average: 8.4
Performance scheduling

First come, first served (FCFS)

Waiting time: 0.11, average: 5.4 – Turnaround time: 3.12, average: 8.0

In this example:
- the average waiting times vary between 5.4 and 5.9
- the average turnaround times vary between 8.0 and 8.4

Shortest possible maximal turnaround time!
Performance scheduling

**Round Robin (RR)**

- **Waiting time:** 0.5, average: 1.2 – **Turnaround time:** 1.20, average: 5.8

- Optimized for swift initial responses.
- “Stretches out” long tasks.
- **Bound maximal waiting time!** (depended only on the number of tasks)
Feedback with $2^i$ pre-emption intervals

- Implement multiple hierarchical ready-queues.
- Fetch processes from the highest filled ready queue.
- Dispatch more CPU time for lower priorities ($2^i$ units).

Processes on lower ranks may suffer starvation.

New and short tasks will be preferred.
Performance scheduling

Feedback with $2^i$ pre-emption intervals - sequential

Waiting time: 0.5, average: 1.5 – Turnaround time: 1.21, average: 5.7

- Optimized for swift initial responses.
- Prefers short tasks and long tasks can suffer starvation.
- Very short initial response times! and good average turnaround times.
Performance scheduling

Feedback with $2^i$ pre-emption intervals - overlapping

Waiting time: 0..3, average: 0.9 – Turnaround time: 1..45, average: 7.7

- Optimized for swift initial responses.
- Prefers short tasks and long tasks can suffer starvation.
- Long tasks are delayed until all queues run empty!
Performance scheduling

Shortest job first

Waiting time: 0..11, average: 3.7 – Turnaround time: 1..14, average: 6.3

- Optimized for good average performance with minimal task-switches.
- Prefers short tasks but all tasks will be handled.
- Good choice if computation times are known and task switches are expensive!
Performance scheduling

Shortest job first

- Waiting time: 0.10, average: 3.4 – Turnaround time: 1.14, average: 6.0

Can be sensitive to non-deterministic arrival sequences.
Performance scheduling

**Highest Response Ration** \( \frac{W_i + C_i}{C_i} \) First (HRRF)

- Blend between Shortest-Job-First and First-Come-First-Served.
- Prefers short tasks but long tasks gain preference over time.
- More task switches and worse averages than SJF but better upper bounds!
Performance scheduling

Shortest Remaining Time First (SRTF)

- Waiting time: 0.6, average: 0.7 – Turnaround time: 1.21, average: 4.4

- Optimized for good averages.
- Prefers short tasks and long tasks can suffer starvation.
- Better averages than Feedback scheduling but with longer absolute waiting times!
Performance scheduling

Comparison (in order of appearance)
Performance scheduling

Comparison by shortest maximal waiting time

- FCFS
- RR
- SJF
- SRTF
- HRRF
- FB-seq.
- FB-ovlp

Providing upper bounds to waiting times Swift response systems
Performance scheduling

Comparison by shortest average waiting time

- SRTF
- FB-ovlp
- RR
- FB-seq.
- SJF
- SJF
- HRRF
- FCFS
- FCFS

Providing short average waiting times

Very swift response in most cases
Performance scheduling

Comparison by shortest maximal turnaround

Providing upper bounds to turnaround times  No tasks are left behind
Performance scheduling

Comparison by shortest average turnaround

- SRTF
- FB-seq.
- RR
- SJF
- SJF
- HRRF
- FB-ovlp
- FCFS
- FCFS

Providing good average performance
High throughput systems
### Performance scheduling

#### Comparison overview

<table>
<thead>
<tr>
<th>Selection</th>
<th>Pre-emption</th>
<th>Waiting</th>
<th>Turnaround</th>
<th>Preferred jobs</th>
<th>Starvation possible?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Methods without any knowledge about the processes</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FCFS</td>
<td>max($W_i$)</td>
<td>no</td>
<td>long</td>
<td>long average &amp; short maximum</td>
<td>equal</td>
</tr>
<tr>
<td>RR</td>
<td>equal share</td>
<td>yes</td>
<td>bound</td>
<td>good average &amp; large maximum</td>
<td>short</td>
</tr>
<tr>
<td>FB</td>
<td>priority queues</td>
<td>yes</td>
<td>very short</td>
<td>short average &amp; long maximum</td>
<td>short</td>
</tr>
<tr>
<td><strong>Methods employing computation time $C_i$ and elapsed time $E_i$</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SJF</td>
<td>min($C_i$)</td>
<td>no</td>
<td>medium</td>
<td>medium</td>
<td>short</td>
</tr>
<tr>
<td>HRRF</td>
<td>$\max(\frac{W_i + C_i}{C_i})$</td>
<td>no</td>
<td>controllable compromise</td>
<td>controllable compromise</td>
<td>controllable</td>
</tr>
<tr>
<td>SRTF</td>
<td>min($C_i - E_i$)</td>
<td>yes</td>
<td>very short</td>
<td>wide variance</td>
<td>short</td>
</tr>
</tbody>
</table>
Predictable scheduling

Towards predictable scheduling …

Task requirements (Quality of service):

- Guarantee data flow levels
- Guarantee reaction times
- Guarantee deadlines
- Guarantee delivery times
- Provide bounds for the variations in results

Examples:

- Streaming media broadcasts, playing HD videos, live mixing audio/video, …
- Reacting to users, Reacting to alarm situations, …
- Delivering a signal to the physical world at the required time, …
Predictable scheduling

Temporal scopes

Common attributes:

- Minimal & maximal delay after creation
- Maximal elapsed time
- Maximal execution time
- Absolute deadline
Predictable scheduling

Temporal scopes

Common attributes:

- Minimal & maximal delay after creation
- Maximal elapsed time
- Maximal execution time
- Absolute deadline
Predictable scheduling

Temporal scopes

Common attributes:

- Minimal & maximal delay after creation
- Maximal elapsed time
- Maximal execution time
- Absolute deadline
Predictable scheduling

Temporal scopes

Common attributes:

- Minimal & maximal delay after creation
- Maximal elapsed time
- Maximal execution time
- Absolute deadline
Predictable scheduling

Common temporal scope attributes

Temporal scopes can be:

<table>
<thead>
<tr>
<th>Periodic</th>
<th>controllers, routers, schedulers, streaming processes, …</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aperiodic</td>
<td>periodic ‘on average’ tasks, i.e. regular but not rigidly timed, …</td>
</tr>
<tr>
<td>Sporadic / Transient</td>
<td>user requests, alarms, I/O interaction, …</td>
</tr>
</tbody>
</table>

Deadlines can be:

<table>
<thead>
<tr>
<th>“Hard”</th>
<th>single failure leads to severe malfunction and/or disaster</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Firm”</td>
<td>results are meaningless after the deadline</td>
</tr>
<tr>
<td>“Soft”</td>
<td>results are still useful after the deadline</td>
</tr>
</tbody>
</table>
Summary

Scheduling

• Basic performance scheduling
  • Motivation & Terms
  • Levels of knowledge / assumptions about the task set
  • Evaluation of performance and selection of appropriate methods

• Towards predictable scheduling
  • Motivation & Terms
  • Categories & Examples