IN4343 Real-Time Systems

Non-Preemptive Scheduling
Preemptive scheduling

Most of work on scheduling has been focused on fully preemptive systems, because they allow higher responsiveness:

- **Preemptive**

- **Non Preemptive**
Disadvantages of preemptions

However, each preemption has a cost:

- **Context switch cost**: time taken by the scheduler to suspend the running task, switch the context, and dispatch the new incoming task.
Preemption indirect costs

- **Cache-related cost**: time to reload the cache lines evicted by the preempting task.

- **Pipeline cost**: time to flush the pipeline when a task is interrupted and to refill it when task is resumed.

- **Bus cost**: time spent waiting for the bus due to additional conflicts with I/O devices, caused by extra accesses to the RAM for the extra cache misses.
Preemption indirect costs

- **Additional preemptions:** the extra execution time also increases the number of preemptions:
Preemption cost

- WCETs may increase up to 35% in the presence of preemptions (less efficiency):

- WCETs become also more variable (less predictability):
Influence on WCETs

- As a consequence, WCETs estimations for preemptive tasks are
  - higher
  - less predictable (highly variable)
Advantages of NP scheduling

- It reduces context-switch overhead:
  - making WCETs smaller and more predictable.

- It simplifies the access to shared resources:
  - No semaphores are needed for critical sections

- It reduces stack size:
  - Task can share the same stack, since no more than one task can be in execution

- It allows achieving zero I/O Jitter:
  - finishing_time – start_time = \( C_i \) (constant)
Advantages of NP scheduling

In fixed priority systems can improve schedulability:

\[
U = \frac{2}{5} + \frac{4}{7} \approx 0.97
\]
Disadvantages of NP scheduling

- In general, NP scheduling reduces schedulability introducing blocking delays in high priority tasks:
Disadvantages of NP scheduling

- The utilization bound under non-preemptive scheduling drops to zero:

\[ U = \frac{\varepsilon}{T_1} + \frac{C_2}{\infty} \rightarrow 0 \]
Non preemptive scheduling anomalies

double speed
Non-preemptive analysis

Analysis of non-preemptive systems is more complex, because the largest response time may not occur in the first job, after the critical instant.

Self-pushing phenomenon

High priority jobs activated during non-preemptive execution of lower priority tasks are pushed ahead and introduce higher delays in subsequent jobs of the same task.

errata for book that starts drawing at -1, should be -ε
Non-preemptive analysis

Hence, the analysis of $\tau_i$ must be carried out for multiple jobs, until all tasks with priority $\geq P_i$ are completed.

Can we derive a bound $N_i$ on the number of instances to consider?
• yes
• and quite often 1 is enough! (extra conditions)
Non-preemptive analysis

Level-i busy period

It is the interval in which the processor is busy executing tasks with priority higher than or equal to $P_i$, including blocking times.
Non-preemptive analysis

Level-i busy period

It can be computed as the shortest interval that satisfies:

\[
L_i = B_i + \sum_{h: P_h \geq P_i} \left[ \frac{L_i}{T_h} \right] C_h
\]

Initial value can be:

\[
L_i^{(0)} = B_i + \sum_{h: P_h \geq P_i} C_h
\]
Non-preemptive analysis

Hence, the analysis of $\tau_i$ must be carried out for multiple jobs, until all tasks with priority $\geq P_i$ are completed.

NOTE
Analysis can reduce to the first job of each task if and only if
1. the task set is feasible under preemptive scheduling;
2. All deadlines are less than or equal to periods.
Response time analysis
for preemptively feasible task sets with $D \leq T$

$$WO_i = s_i - r_i$$

$$B_i = \max_{P_j < P_i} \{C_j\}$$

$$\begin{cases} hp(i) & \text{set of tasks with priority higher than } P_i \\ lp(i) & \text{set of tasks with priority lower than } P_i \end{cases}$$

$WO_i$ = Worst-case Occupied time: due to blocking $B_i$ from $lp(i)$ tasks and interference $I_i$ from $hp(i)$ tasks.
Response time analysis
for preemptively feasible task sets with $D \leq T$

\[ WO_i = s_i - r_i \]

**NOTE:** the end of $I_i$ cannot coincide with the activation of a higher priority task, because it would increase $I_i$.

Hence, $\lceil x \rceil + 1$ must be used instead of $\lfloor x \rfloor$.

\[ WO_i = B_i + \sum_{k=1}^{i-1} \left( \left \lfloor \frac{WO_i}{T_k} \right \rfloor + 1 \right) C_k \]

\[ R_i = WO_i + C_i \]
Response time analysis

for preemptively feasible task sets with $D \leq T$

\[
\begin{align*}
WO_i^{(0)} &= B_i + \sum_{k=1}^{i-1} C_k \\
WO_i^{(s)} &= B_i + \sum_{k=1}^{i-1} \left( \left\lfloor \frac{WO_i^{(s-1)}}{T_k} + 1 \right\rfloor \right) C_k
\end{align*}
\]

Stop when $WO_i^{(s)} = WO_i^{(s-1)}$

\[R_i = WO_i + C_i\]
Exercise

Determine the schedulability of the following task set under non-preemptive RM

<table>
<thead>
<tr>
<th></th>
<th>( C_i )</th>
<th>( T_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>t2</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>t3</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>t4</td>
<td>3</td>
<td>19</td>
</tr>
</tbody>
</table>
Taking advantage of NP scheduling

- Preemptive scheduling
- Limited Preemptive scheduling
- Non Preemptive scheduling
Trade-off solutions

Joe is reading an interesting book and doesn't want to be disturbed, but several people try to call him.

What can he do?

- Respond only to important calls (e.g., to his parents)
  ⇒ Preemption Thresholds
- “I'll come in 10 minutes”
  ⇒ Deferred Preemptions
- “I'll come as soon as I finish the chapter”
  ⇒ Fixed Preemption Points
Preemption Thresholds (PT)

Each task has two priorities:

- $P_i$ nominal priority: used to enqueue the task in the ready queue and to preempt
- $\theta_i$ threshold priority: used for task execution ($\theta_i \geq P_i$)

A task $\tau_i$ can be preempted by $\tau_h$ only if $P_h > \theta_i$
Unfeasible task set

Fully preemptive

\( \tau_1 \)

\( \tau_2 \)

\( \tau_3 \)

Fully non-preemptive

\( \tau_1 \)

\( \tau_2 \)

\( \tau_3 \)

deadline miss

deadline miss
But feasible with preemption thresholds

<table>
<thead>
<tr>
<th>$P_i$</th>
<th>$\theta_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

$\tau_1$ can preempt $\tau_3$

$\tau_2$ cannot preempt $\tau_3$

$\tau_1$ cannot preempt $\tau_2$

**NOTE:**
The same feasible schedule is obtained by splitting $\tau_3$ in two non preemptive chunks: $q_{31} = 2$, $q_{32} = 3
Response time analysis (PT)
for preemptively feasible task sets with $D \leq T$

\[ \tau_i \text{ can only be delayed by tasks } \tau_h: P_h > P_i \]
\[ \tau_i \text{ can only be preempted by tasks } \tau_h: P_h > \theta_i \]

\[ B_i = \max_j \{ C_j | P_j < P_i \leq \theta_j \} \]
\[ S_i = B_i + \sum_{h: P_h > P_i} \left( \left\lfloor \frac{S_i}{T_h} \right\rfloor + 1 \right) C_h \]
\[ R_i = S_i + C_i + \sum_{h: P_h > \theta_i} \left( \left\lfloor \frac{R_i}{T_h} \right\rfloor - \left( \left\lfloor \frac{S_i}{T_h} \right\rfloor + 1 \right) \right) C_h \]
Preemption Thresholds

How to compute $\theta_i$?

algorithm assign minimum preemptive thresholds()
begin
  for $i := n$ downto 1 do // from the lowest priority task
    $\theta_i := P_i$
    $R_i :=$ response time analysis ($P_i$, $\theta_i$)
  while ($R_i > D_i$) do // while not schedulable
    $\theta_i := \theta_i + 1$ // increase the preemption threshold
    if ($\theta_i >$ max prio) then
      return UNFEASIBLE
    end
  end
  $R_i :=$ response time analysis ($P_i$, $\theta_i$)
end
end
return FEASIBLE
Deferred Preemption

Each task can defer preemption up to $q_i$

$$B_i = \max_{P_j < P_i} \{q_j\}$$
Using $Q_i$

Once $Q_i$ is computed, it can be used as follows:

- Partition each task into a set of NP regions no larger than $Q_i$ inserting suitable preemption points.

- **Incapsulate critical sections** into NP regions, avoiding complex concurrency control protocols.
Experimental Results

Avg. # of preemptions

\[ n = 16 \]

\[ x \times 10^4 \]

\[ 0.1 \quad 0.2 \quad 0.3 \quad 0.4 \quad 0.5 \quad 0.6 \quad 0.7 \quad 0.8 \quad 0.9 \]

Fully preemptive

\(-70\%\)

Using \( Q_i \)
Fixed Preemption Points (FPP)

- Each task $\tau_i$ is divided in $m_i$ chunks: $q_{i,1} \ldots q_{i,m_i}$
- It can only be preempted between chunks

$$B_i = \max_{P_j < P_i} \{q_j^{\text{max}}\}$$
Example

Let: \( \tau_1 \) be fully non preemptive: \( q_{11} = C_1 = 3 \)
\( \tau_2 \) consisting of 2 NP chunks: \( q_{21} = 1, q_{22} = 3, C_2 = 4 \)
\( \tau_3 \) be fully non preemptive: \( q_{31} = C_3 = 1 \)

\[ \text{Note that:} \]

- The worst case response time of \( \tau_2 \) does not occur in the first instance.
- The interference on \( \tau_2 \) is larger than \( B_2 + C_1 \).
Response Time Analysis (FPP)

\[ s_{ik} = B_i + (k-1)C_i + \left( C_i - q_i^{last} \right) + \sum_{h: P_h > P_i} \left( \left\lfloor \frac{s_{ik}}{T_h} \right\rfloor + 1 \right) C_h \]

\[ f_{ik} = s_{ik} + q_i^{last} \]

\[ R_{ik} = f_{ik} - (k-1)T_i \]

\[ R_i = \max_{k \in [1,N_i]} \{ R_{ik} \} \]

NOTE:

\[ s_{ik}^{(0)} = (k-1)T_i + \left( C_i - q_i^{last} \right) \]

\[ N_i = \left\lceil \frac{L_i}{T_i} \right\rceil \]
Response Time Analysis (FPP)

for \((i = 1\) to \(n)\) \{ 
\[
N_i = \left\lfloor \frac{L_i}{T_i} \right\rfloor
\]
\[
s_{ik}^{(0)} = (k-1)T_i + \left(C_i - q_i^{last}\right)
\]
\[
k = 1
\]
do \{ 
\[
s_{ik} = B_i + (k-1)C_i + \left(C_i - q_i^{last}\right) + \sum_{h: P_h > P_i} \left(\left\lfloor \frac{s_{ik}}{T_h} \right\rfloor + 1\right) C_h
\]
\[
f_{ik} = s_{ik} + q_i^{last}
\]
\[
R_{ik} = f_{ik} - (k-1)T_i
\]
if \((R_{ik} > R_i)\) then \(R_i = R_{ik}\)
if \((R_i > D_i)\) then return(UNFEASIBLE)
\[
k++
\}
while \((k \leq N_i)\)
\}
return(FEASIBLE)
Special cases

- **Fully non preemptive scheduling**

\[
\begin{align*}
q_i^{last} &= C_i \\
B_i &= \max_{P_j < P_i} \{ C_j \}
\end{align*}
\]

- **Deferred Preemption**

\[
\begin{align*}
q_i^{last} &= 0 \\
B_i &= \max_{P_j < P_i} \{ Q_j \}
\end{align*}
\]
Final remarks

- Preemption Thresholds are easy to specify, but it is difficult to predict the number of preemptions and where they occur ⇒ large preemption overhead.

- Deferred Preemption allows bounding the number of preemptions but it is difficult to predict where they occur. Note that the analysis assumes $q_i^{last} = 0$.

- Fixed Preemption Points allow more control on preemptions and can be selected on purpose (e.g., to minimize overhead, stack size, and reduce WCETs).
  - A large final chunk in $\tau_i$ reduces the interference from hp-tasks (hence $R_i$), but creates more blocking to hp-tasks.