Multiprocessor Real-Time Scheduling: A Summary

Prof. Dr. Jian-Jia Chen

LS 12, TU Dortmund

15 Jan. 2020
Outline

Introduction

Partitioned Scheduling for Implicit-Deadline EDF Scheduling

Partitioned Scheduling for Implicit-Deadline RM Scheduling

Global Multiprocessor Scheduling
Multiprocessor Models

• Identical (Homogeneous): All the processors have the same characteristics, i.e., the execution time of a job is independent on the processor it is executed.

• Uniform: Each processor has its own speed, i.e., the execution time of a job on a processor is proportional to the speed of the processor.

  • A faster processor always executes a job faster than slow processors do.
  • For example, multiprocessors with the same instruction set but with different supply voltages/frequencies.

• Unrelated (Heterogeneous): Each job has its own execution time on a specified processor
  • A job might be executed faster on a processor, but other jobs might be slower on that processor.
  • For example, multiprocessors with different instruction sets.
Scheduling Models

- **Partitioned Scheduling:**
  - Each task is assigned on a dedicated processor.
  - Schedulability is done individually on each processor.
  - It requires no additional on-line overhead.

- **Global Scheduling:**
  - A job may execute on any processor.
  - The system maintains a global ready queue.
  - Execute the $M$ highest-priority jobs in the ready queue, where $M$ is the number of processors.
  - It requires high on-line overhead.
Problem Definition: Partitioned Scheduling

Partitioned Scheduling

Given a set $\mathbf{T}$ of tasks with implicit deadlines, i.e., $\forall \tau_i \in \mathbf{T}$, $T_i = D_i$, the objective is to decide a feasible task assignment onto $M$ processors such that all the tasks meet their timing constraints, where $C_{im}$ is the execution time of task $\tau_i$ on processor $m$.

- For identical multiprocessors: $C_i = C_{i1} = C_{i2} = \cdots = C_{iM}$.
- For uniform multiprocessors: each processor $m$ has a speed $s_m$, in which $C_{im}s_m$ is a constant.
- For unrelated multiprocessors: $C_{im}$ is an independent parameter.
\[ \mathcal{NP} \text{-complete} \]

Deciding whether there exists a feasible task assignment is \( \mathcal{NP} \)-complete in the strong sense.

**Proof**

Reduced from the 3-Partition problem.
Hardness and Approximation of Partitioned Scheduling

**NP-complete**

Deciding whether there exists a feasible task assignment is NP-complete in the strong sense.

**Proof**

Reduced from the 3-Partition problem.

- Approximations are possible, but what do we approximate when only binary decisions (Yes or No) have to be made?
  - Deadline relaxation: requires modifications of task specification
  - Period relaxation: requires modifications of task specification
  - Resource augmentation by **speeding up**: requires a faster platform
  - Resource augmentation by **allocating more processors**: requires a better platform
Approximation Algorithms

An algorithm $\mathcal{A}$ is called an $\eta$-approximation algorithm (for a minimization problem) if it guarantees to derive a feasible solution for any input instance $I$ with at most $\eta$ times of the objective function of an optimal solution. That is,

$$\mathcal{A}(I) \leq \eta \text{OPT}(I),$$

where $\text{OPT}(I)$ is the objective function of an optimal solution.
Terminologies Used in Scheduling Theory

Graham’s Scheduling Algorithm Classification

- Classification: $a|b|c$
  - $a$: machine environment
    (e.g., uniprocessor, multiprocessor, distributed, ...)
  - $b$: task and resource characteristics
    (e.g., preemptive, independent, synchronous, ...)
  - $c$: performance metric and objectives
    (e.g., $L_{\text{max}}$, sum of finish times, ...)

- Makespan problem:
  - $M||C_{\text{max}}$
  - Input: $M$ identical processors and $N$ jobs with given execution times arriving at time 0
  - Output: Assign a job to a processor and execute the jobs to minimize the maximum completion time
Bin Packing Problem

- Given a bin size $b$, and a set of items with individual sizes, the objective is to assign each item to a bin without violating the bin size constraint such that the number of allocated bins is minimized.
Outline

Introduction

Partitioned Scheduling for Implicit-Deadline EDF Scheduling

Partitioned Scheduling for Implicit-Deadline RM Scheduling

Global Multiprocessor Scheduling
Largest-Utilization-First (LUF) - for EDF Scheduling

**Input:** $T, M$

1. re-index (sort) tasks such that $\frac{C_i}{T_i} \geq \frac{C_j}{T_j}$ for $i < j$;
2. $T_m \leftarrow \emptyset$, $U_m \leftarrow 0$, $\forall m = 1, 2, \ldots, M$;
3. for $i = 1$ to $N$, where $N = |T|$ do
4. find $m^*$ with the minimum utilization, i.e., $U_{m^*} = \min_{m \leq M} U_m$;
5. if $U_{m^*} + \frac{C_i}{T_i} > 1$ then
6. return ”The task assignment fails”;
7. else
8. assign task $\tau_i$ onto processor $m^*$, where $U_{m^*} \leftarrow U_{m^*} + \frac{C_i}{T_i}$, $T_{m^*} \leftarrow T_{m^*} \cup \{\tau_i\}$;
9. return feasible task assignment $T_1, T_2, \ldots, T_M$;
Largest-Utilization-First (LUF) - for EDF Scheduling

Input: \( T, M; \)
1: re-index (sort) tasks such that \( \frac{C_i}{T_i} \geq \frac{C_j}{T_j} \) for \( i < j \);
2: \( T_m \leftarrow \emptyset, U_m \leftarrow 0, \forall m = 1, 2, \ldots, M; \)
3: \( \text{for } i = 1 \text{ to } N, \text{ where } N = |T| \text{ do} \)
4: \( \text{find } m^* \text{ with the minimum utilization, i.e., } U_{m^*} = \min_{m \leq M} U_m; \)
5: \( \text{if } U_{m^*} + \frac{C_i}{T_i} > 1 \text{ then} \)
6: \( \text{return "The task assignment fails"; } \)
7: \( \text{else} \)
8: \( \text{assign task } \tau_i \text{ onto processor } m^*, \text{ where} \)
\( U_{m^*} \leftarrow U_{m^*} + \frac{C_i}{T_i}, T_{m^*} \leftarrow T_{m^*} \cup \{\tau_i\}; \)
9: \( \text{return feasible task assignment } T_1, T_2, \ldots, T_M; \)

Properties

- The time complexity is \( O((N + M) \log(N + M)) \)
- If a solution is derived, the task assignment is feasible by using EDF.
Algorithm LUF

\[ \tau_1 = 0.5 \]
\[ \tau_2 = 0.45 \]
\[ \tau_3 = 0.37 \]
\[ \tau_4 = 0.3 \]
\[ \tau_5 = 0.2 \]
\[ \tau_6 = 0.2 \]
\[ \tau_7 = 0.15 \]
\[ \tau_8 = 0.1 \]
Algorithm LUF

$\tau_1 \approx 0.5$
$\tau_2 \approx 0.45$
$\tau_3 \approx 0.37$
$\tau_4 \approx 0.3$
$\tau_5 \approx 0.2$
$\tau_6 \approx 0.2$
$\tau_7 \approx 0.15$
$\tau_8 \approx 0.1$

$(0, 0, 0) \rightarrow P_1, P_2, P_3$
Algorithm LUF

\[
\begin{align*}
\tau_1 & : (0.5, 0, 0) \\
\tau_2 & : (0.45, 0, 0) \\
\tau_3 & : (0.37, 0, 0) \\
\tau_4 & : (0.3, 0, 0) \\
\tau_5 & : (0.2, 0, 0) \\
\tau_6 & : (0.2, 0, 0) \\
\tau_7 & : (0.15, 0, 0) \\
\tau_8 & : (0.1, 0, 0)
\end{align*}
\]
Algorithm LUF

\((.5, .45, 0)\)

\(\tau_1\) \(\tau_2\) \(\tau_3\) \(\tau_4\) \(\tau_5\) \(\tau_6\) \(\tau_7\) \(\tau_8\)

0.5 0.45 0.37 0.3 0.2 0.2 0.15 0.1

(0, 0, 0) (0.5, 0, 0) (0.5, 0.45, 0) (0.5, 0.45, 0.37) (0.5, 0.45, 0.67) (0.5, 0.65, 0.67) (0.7, 0.65, 0.67) (0.7, 0.8, 0.67)
Algorithm LUF

\[(0.5, 0.45, 0.37)\]

\begin{align*}
\tau_1 & : 0.5 \\
\tau_2 & : 0.45 \\
\tau_3 & : 0.37 \\
\tau_4 & : 0.3 \\
\tau_5 & : 0.2 \\
\tau_6 & : 0.2 \\
\tau_7 & : 0.15 \\
\tau_8 & : 0.1
\end{align*}
Algorithm LUF

\( \tau_1 \quad \tau_2 \quad \tau_3 \quad \tau_4 \quad \tau_5 \quad \tau_6 \quad \tau_7 \quad \tau_8 \)

\( 0.5 \quad 0.45 \quad 0.37 \quad 0.3 \quad 0.2 \quad 0.2 \quad 0.15 \quad 0.1 \)

\( (0.5, 0.45, 0.67) \)
Algorithm LUF

\[ (0.5, 0.65, 0.67) \]

\[ \tau_1 \quad \tau_2 \quad \tau_3 \quad \tau_4 \quad \tau_5 \quad \tau_6 \quad \tau_7 \quad \tau_8 \]

\[ 0.5 \quad 0.45 \quad 0.37 \quad 0.3 \quad 0.2 \quad 0.2 \quad 0.15 \quad 0.1 \]

\[ P_1 \quad P_2 \quad P_3 \]
Algorithm LUF

\[ \tau_1 = 0.5, \quad \tau_2 = 0.45, \quad \tau_3 = 0.37, \quad \tau_4 = 0.3, \quad \tau_5 = 0.2, \quad \tau_6 = 0.2, \quad \tau_7 = 0.15, \quad \tau_8 = 0.1 \]

\[ (.7, .65, .67) \]

\[ P_1, P_2, P_3 \]
Algorithm LUF

\[ (0.7, 0.8, 0.67) \]
Algorithm LUF

\[ U = 0.7 \]

\[ U = 0.8 \]

\[ U = 0.77 \]
Optimality of Algorithm LUF

**Theorem**
If an optimal assignment for minimizing the maximal utilization results in at most two tasks on any processor, LUF is optimal.

**Proof**
The proof is omitted.
What Happens if Algorithm LUF Fails?

Assume that there exists a feasible task partition on $M$ processors (for providing the analysis of resource augmentation).

- Suppose that Algorithm LUF fails when assigning task $\tau_j$ and $U_m$ for $m = 1, 2, \ldots, M$ is the utilization of processor $m$ before assigning $\tau_j$.
- Let $U_{opt}$ be the utilization of the optimal assignment for minimizing the maximal utilization for tasks $\{\tau_1, \tau_2, \ldots, \tau_j\}$.
- By definition, $1 \geq U_{opt} \geq \sum_{i=1}^{j} \frac{C_i}{T_i}$.
- $\frac{C_j}{T_j} \leq \frac{1}{3} U_{opt}$: otherwise, there will be at most two tasks on any processors in the optimal solution. $\Rightarrow$ this contradicts the assumption that Algorithm LUF fails as it is optimal.
- Since $U_m^* \leq U_m$, we know that $U_m^* \leq \sum_{m=1}^{M} \frac{U_m}{M} = \sum_{i=1}^{j-1} \frac{C_i}{T_i}$.
- Therefore,

$$\frac{C_j}{T_j} + U_m^* \leq \frac{C_j}{T_j}(1 - \frac{1}{M}) + \sum_{i=1}^{j} \frac{C_i}{T_i} \leq \left(\frac{4}{3} - \frac{1}{3M}\right) U_{opt} \leq \left(\frac{4}{3} - \frac{1}{3M}\right).$$
Algorithm $LUF^+$: Resource Augmentation on Processors

**Input:** $T$;
1. re-index (sort) tasks such that $\frac{C_i}{T_i} \geq \frac{C_j}{T_j}$ for $i < j$;
2. $T_1 \leftarrow \emptyset$, $U_1 \leftarrow 0$, $\hat{M} \leftarrow 1$;
3. for $i = 1$ to $N$, where $N = |T|$ do
4. find a processor $m^*$ with $U_{m^*} + \frac{C_i}{T_i} \leq 1$;
5. if no such a processor exists then
6. $\hat{M} \leftarrow \hat{M} + 1$, $T_{\hat{M}} \leftarrow \emptyset$, $U_{\hat{M}} \leftarrow 0$;
7. $m^* \leftarrow \hat{M}$;
8. assign task $\tau_i$ onto processor $m^*$, where $U_i \leftarrow U_i + \frac{C_i}{T_i}$, $T_i \leftarrow T_i \cup \{\tau_i\}$;
9. return task assignment $T_1, T_2, \ldots, T_{\hat{M}}$;
Algorithm $LUF^+$: Resource Augmentation on Processors

**Input:** $T$

1. re-index (sort) tasks such that $\frac{C_i}{T_i} \geq \frac{C_j}{T_j}$ for $i < j$;
2. $T_1 \leftarrow \emptyset$, $U_1 \leftarrow 0$, $\hat{M} \leftarrow 1$;
3. for $i = 1$ to $N$, where $N = |T|$ do
4. find a processor $m^*$ with $U_{m^*} + \frac{C_i}{T_i} \leq 1$;
5. if no such a processor exists then
6. $\hat{M} \leftarrow \hat{M} + 1$, $T_{\hat{M}} \leftarrow \emptyset$, $U_{\hat{M}} \leftarrow 0$;
7. $m^* \leftarrow \hat{M}$;
8. assign task $\tau_i$ onto processor $m^*$, where $U_i \leftarrow U_i + \frac{C_i}{T_i}$, $T_i \leftarrow T_i \cup \{\tau_i\}$;
9. return task assignment $T_1, T_2, \ldots, T_{\hat{M}}$;

**Properties**

- The time complexity is $O(N \log N)$ or $O(N^2)$, depending on the fitting approaches.
- The resulting solution is feasible on $\hat{M}$ processors.
Different Fitting Approaches

4: find a processor $m^*$ with $U_{m^*} + \frac{C_i}{T_i} \leq 1$;

Fitting Strategies

- **First-Fit**: choose the feasible one with the smallest index
- **Last-Fit**: choose the feasible one with the largest index
- **Best-Fit**: choose the feasible one with the maximal utilization
- **Worst-Fit**: choose the feasible one with the minimal utilization

Suppose that we want to assign a task with utilization equal to 0.1.
Different Fitting Approaches

4: find a processor $m^*$ with $U_{m^*} + \frac{C_i}{T_i} \leq 1$;

Fitting Strategies

- **First-Fit**: choose the feasible one with the smallest index
- **Last-Fit**: choose the feasible one with the largest index
- **Best-Fit**: choose the feasible one with the maximal utilization
- **Worst-Fit**: choose the feasible one with the minimal utilization

Suppose that we want to assign a task with utilization equal to 0.1.

<table>
<thead>
<tr>
<th></th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
<th>$P_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Fit</td>
<td></td>
<td>0.7</td>
<td>0.5</td>
<td>0.65</td>
</tr>
</tbody>
</table>

Prof. Dr. Jian-Jia Chen (LS 12, TU Dortmund)
Different Fitting Approaches

4: find a processor $m^*$ with $U_{m^*} + \frac{C_i}{T_i} \leq 1$;

Fitting Strategies

- First-Fit: choose the feasible one with the smallest index
- Last-Fit: choose the feasible one with the largest index
- Best-Fit: choose the feasible one with the maximal utilization
- Worst-Fit: choose the feasible one with the minimal utilization

Suppose that we want to assign a task with utilization equal to 0.1.

First Fit

Last Fit

$P_1$

$P_2$

$P_3$

$P_4$
Different Fitting Approaches

4: find a processor $m^*$ with $U_{m^*} + \frac{C_i}{T_i} \leq 1$;

Fitting Strategies

- First-Fit: choose the feasible one with the smallest index
- Last-Fit: choose the feasible one with the largest index
- Best-Fit: choose the feasible one with the maximal utilization
- Worst-Fit: choose the feasible one with the minimal utilization

Suppose that we want to assign a task with utilization equal to 0.1.
Different Fitting Approaches

4: find a processor $m^*$ with $U_{m^*} + \frac{C_i}{T_i} \leq 1$;

Fitting Strategies

- First-Fit: choose the feasible one with the smallest index
- Last-Fit: choose the feasible one with the largest index
- Best-Fit: choose the feasible one with the maximal utilization
- Worst-Fit: choose the feasible one with the minimal utilization

Suppose that we want to assign a task with utilization equal to 0.1.
Algorithm $LUF^+$: How Many Processors?

- Suppose that the processor used by Algorithm $LUF^+$ is $\hat{M} \geq 2$.
- Let $m^*$ be the processor with the minimum utilization.
- By the fitting algorithm, we know that $U_m + U_{m^*} > 1$ and $U_m \geq U_{m^*}$ for all the other processors $m$s.
- If $U_{m^*} \leq 0.5$, by $U_m > 1 - U_{m^*}$, we know that
  \[
  \sum_{\tau_i \in T} \frac{C_i}{T_i} \geq U_{m^*} + \sum_{m=1, m \neq m^*}^{\hat{M}} U_m \geq \hat{M} - 1 - (\hat{M} - 2)U_{m^*} \leq (\hat{M} - 2)(1 - U_{m^*}) + 1 \geq \frac{\hat{M}}{2}.
  \]
- If $U_{m^*} > 0.5$, by $U_m \geq U_{m^*}$, we know that
  \[
  \sum_{\tau_i \in T} \frac{C_i}{T_i} \geq U_{m^*} + \sum_{m=1, m \neq m^*}^{\hat{M}} U_m \geq \frac{\hat{M}}{2}.
  \]

Theorem

Algorithm $LUF^+$ is a 2-approximation algorithm (with respect to allocating more processors).
Outline

Introduction

Partitioned Scheduling for Implicit-Deadline EDF Scheduling

Partitioned Scheduling for Implicit-Deadline RM Scheduling

Global Multiprocessor Scheduling
Largest-Utilization-First (\textit{LUF}+) - for RM Scheduling

\textbf{Input:} \( T \);
\begin{enumerate}
\item re-index (sort) tasks such that \( \frac{C_i}{T_i} \geq \frac{C_j}{T_j} \) for \( i < j \);
\item \( T_1 \leftarrow \emptyset, U_1 \leftarrow 0, n_1 \leftarrow 0; \hat{M} \leftarrow 1 \);
\item \textbf{for} \( i = 1 \) to \( N \), where \( N = |T| \) \textbf{do}\;
\item find a processor \( m^* \) with \( U_{m^*} + \frac{C_i}{T_i} \leq (n_{m^*} + 1) \left( 2^{\frac{1}{n_{m^*}+1}} - 1 \right) \);
\item \textbf{if} no such a processor exists \textbf{then} \;
\item \( \hat{M} \leftarrow \hat{M} + 1, T_{\hat{M}} \leftarrow \emptyset, U_{\hat{M}} \leftarrow 0, n_{\hat{M}} \leftarrow 0 \);
\item \( m^* \leftarrow \hat{M} \);
\item assign task \( \tau_i \) onto processor \( m^* \), where \( U_{m^*} \leftarrow U_{m^*} + \frac{C_i}{T_i}, T_{m^*} \leftarrow T_{m^*} \cup \{\tau_i\}, n_{m^*} \leftarrow n_{m^*} + 1 \);
\item \textbf{return} task assignment \( T_1, T_2, \ldots, T_{\hat{M}} \);
\end{enumerate}
Largest-Utilization-First (LUF$^+$) - for RM Scheduling

Input: $T$

1: re-index (sort) tasks such that $\frac{C_i}{T_i} \geq \frac{C_j}{T_j}$ for $i < j$;
2: $T_1 \leftarrow \emptyset$, $U_1 \leftarrow 0$, $n_1 \leftarrow 0$; $\hat{M} \leftarrow 1$;
3: for $i = 1$ to $N$, where $N = |T|$ do
4: find a processor $m^*$ with $U_{m^*} + \frac{C_i}{T_i} \leq (n_{m^*} + 1) \left(2^{\frac{1}{n_{m^*} + 1}} - 1\right)$;
5: if no such a processor exists then
6: $\hat{M} \leftarrow \hat{M} + 1$, $T_{\hat{M}} \leftarrow \emptyset$, $U_{\hat{M}} \leftarrow 0$, $n_{\hat{M}} \leftarrow 0$;
7: $m^* \leftarrow \hat{M}$;
8: assign task $\tau_i$ onto processor $m^*$, where
   $U_{m^*} \leftarrow U_{m^*} + \frac{C_i}{T_i}$, $T_{m^*} \leftarrow T_{m^*} \cup \{\tau_i\}$, $n_{m^*} \leftarrow n_{m^*} + 1$;
9: return task assignment $T_1, T_2, \ldots, T_{\hat{M}}$;

Properties

- The time complexity is $O((N + M) \log(N + M))$
- If a solution is derived, the task assignment is feasible by using RM.
A Simple Analysis

• The schedulability test \( U_{m^*} + \frac{C_i}{T_i} \leq (n_{m^*} + 1) \left( 2^{n_{m^*}+1} - 1 \right) \) is upper bounded by 69.3%.

• According to the above analysis for EDF, we can also conclude that the utilization is at least \( \frac{0.693\hat{M}}{2} \).

• Therefore, the approximation factor of \( LUF^+ \) is \( \frac{2}{0.693} \approx 2.887 \).
Remarks (Augmenting the Number of Processors)

Survey by Davis and Burns (ACM Computing Surveys, 2011):

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Approximation Ratio ($R_A$)</th>
<th>Ref.</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMNF</td>
<td>2.67</td>
<td>[Dhall and Liu 1978]</td>
</tr>
<tr>
<td>RMFF</td>
<td>2.33</td>
<td>[Oh and Son 1993]</td>
</tr>
<tr>
<td>RMBF</td>
<td>2.33</td>
<td>[Oh and Son 1993]</td>
</tr>
<tr>
<td>RRM-FF</td>
<td>2</td>
<td>[Oh and Son 1995]</td>
</tr>
<tr>
<td>FFDUF</td>
<td>2</td>
<td>[Davari and Dhall 1986]</td>
</tr>
<tr>
<td>RMST</td>
<td>$1/(1 - u_{\text{max}})$</td>
<td>[Burchard et al. 1995]</td>
</tr>
<tr>
<td>RMGT</td>
<td>7/4</td>
<td>[Burchard et al. 1995]</td>
</tr>
<tr>
<td>RMMatching</td>
<td>3/2</td>
<td>[Rothvoß 2009]</td>
</tr>
<tr>
<td>EDF-FF</td>
<td>1.7</td>
<td>[Garey and Johnson 1979]</td>
</tr>
<tr>
<td>EDF-BF</td>
<td>1.7</td>
<td>[Garey and Johnson 1979]</td>
</tr>
</tbody>
</table>
## Results for Constrained- and Arbitrary-Deadline Systems

<table>
<thead>
<tr>
<th></th>
<th>implicit deadlines</th>
<th>constrained deadlines</th>
<th>arbitrary deadlines</th>
</tr>
</thead>
<tbody>
<tr>
<td>partitioned with EDF</td>
<td>$\frac{4}{3} - \frac{1}{3M}$ (Graham 1969)</td>
<td>$3 - \frac{1}{M}$ (Baruah/Fisher 2006)</td>
<td>$4 - \frac{2}{M}$ (Baruah/Fisher 2005)</td>
</tr>
<tr>
<td></td>
<td>(Hochbaum/Shmoys 1987)</td>
<td>$2.6322 - \frac{1}{M}$ (Chen/Chakraborty 2011)</td>
<td>$3 - \frac{1}{M}$ (Chen/Chakraborty 2011)</td>
</tr>
<tr>
<td>partitioned with DM</td>
<td>(bin-packing) $\frac{7}{4}$ (Burchard et al. 1995)</td>
<td>$3 - \frac{1}{M}$ (Baker/Fisher/Baruah 2009)</td>
<td>$4 - \frac{2}{M}$ (Baker/Fisher/Baruah 2009)</td>
</tr>
<tr>
<td></td>
<td>(bin-packing) 1.5 (Rothvoß 2009)</td>
<td>2.84306 (Chen 2016)</td>
<td>$3 - \frac{1}{M}$ (Chen 2016)</td>
</tr>
</tbody>
</table>

The above factors are for speed-up factors, except the two results in partitioned RM scheduling.

Outline

Introduction

Partitioned Scheduling for Implicit-Deadline EDF Scheduling

Partitioned Scheduling for Implicit-Deadline RM Scheduling

Global Multiprocessor Scheduling
Global Scheduling

- We will only focus on identical multiprocessors in this module.
- The system has a global queue.
- A job can be migrated to any processor.
- Priority-based global scheduling:
  - Among the jobs in the global queue, the $M$ highest priority jobs are chosen to be executed on $M$ processors.
  - Task migration here is assumed no overhead.
  - Global-EDF: When a job finishes or arrives to the global queue, the $M$ jobs in the queue with the shortest absolute deadlines are chosen to be executed on $M$ processors.
  - Global-FP, Global-DM, Global-RM: When a job finishes or arrives to the global queue, the $M$ jobs in the queue with the highest priorities (defined by fixed-priority ordering, deadline-monotonic strategy, or rate-monotonic strategy) are chosen to be executed on $M$ processors.
- Pfair scheduling, and the variances (not discussed in this lecture).
Good News for Global Scheduling

- McNaughton’s wrap-around rule for $P|\text{pmtn}|C_{\text{max}}$ on $M$ processors (historically, task migration is also called task preemption in the literature)

  - Compute $C_{\text{max}}$ as $\max\{\max_{i\in T} C_i, \frac{\sum_{i\in T} C_i}{M}\}$
  
    - Assign the tasks according to any order from time 0 to $C_{\text{max}}$
    - If a task’s processing exceeds $C_{\text{max}}$, the task is migrated to a new processor from time 0
    - Repeat the assignment of tasks until all the tasks are assigned
  
- The resulting schedule minimizes $C_{\text{max}}$

McNaughton’s Algorithm: Example

D

split tasks
unsplitted tasks

D
Weakness of Partitioned Scheduling

- Restricting a task on a processor reduces the schedulability
- Restricting a task on a processor makes the problem $\mathcal{NP}$-hard
- The $\mathcal{NP}$-completeness does no hold any more if the migration has no overhead.
  - Proportionate Fair (pfair) algorithm introduced by Baruah et al. provides an optimal utilization bound for schedulibility
  - A task set with implicit deadlines is schedulable on $M$ identical processors if the total utilization of the task set is no more than $M$.
  - The idea is to divide the time line into quanta, and execute tasks proportionally in each quanta.
  - It has very high overhead.
  - There are several variances to reduce the overhead.

Bad News for Global Scheduling

For Global-EDF or Global-RM, the least upper bound for schedulability analysis is at most 1.

Input:

$M + 1$ tasks:

- One heavy task $\tau_k$: $D_k = T_k = C_k$
- $M$ light tasks $\tau_i$s: $C_i = \epsilon$ and $D_i = T_i = C_k - \epsilon$, in which $\epsilon$ is a positive number, very close to 0.

Bad News for Global Scheduling

For Global-EDF or Global-RM, the least upper bound for schedulability analysis is at most 1.

Input:

\(M + 1\) tasks:
- One heavy task \(\tau_k\): \(D_k = T_k = C_k\)
- \(M\) light tasks \(\tau_i\)s: \(C_i = \epsilon\) and \(D_i = T_i = C_k - \epsilon\), in which \(\epsilon\) is a positive number, very close to 0.

Result:

The \(M\) light tasks (with higher priority than the heavy task) will be scheduled on \(M\) processors. The heavy task misses the deadline even when the utilization is \(1 + M\epsilon\).

Gold Approach: Resource Augmentation

• The bad news on the least upper bound was very important in 80’s, since the research in this direction suffered from the so-called “Dhall effect”.
• With resource augmentation, by Phillips et al., the “Dhall effect” disappears

  • For Global-EDF, the resource augmentation factor by “speeding up” is $2 - \frac{1}{M}$.
  • That is, if a feasible schedule exists on $M$ processors, applying Global-EDF is also feasible on $M$ processors by speeding up the execution speed with $2 - \frac{1}{M}$.
  • We will focus on schedulability test here first (for the first two parts) and the resource augmentation at the end.

Critical Instants?

- The analysis for uniprocessor scheduling is based on the gold critical instant theorem.
- Synchronous release of higher-priority tasks and as early as possible for the following jobs do not lead to the critical instant for global multiprocessor scheduling.
  - Suppose that there are two identical processors and 3 tasks: 
    \((C_i, D_i, T_i)\) are \(\tau_1 = (1, 2, 2), \tau_2 = (1, 3, 3), \tau_3 = (5, 6, 6)\)
Identifying Interference

- Problem window (interval) is defined in \([a_k, d_k]\).
- The jobs of task \(\tau_i\) in the problem window can be categorized into three types:
  - Head job (at most one): some computation demand is carried in to the problem window for a job arrival before \(a_k\).
  - Body jobs: the computation demand has to be done in the problem window.
  - Tail job (at most one): some computation demand can be carried out from the problem window.
Necessary Condition for Deadline Misses

- If $\tau_k$ misses the deadline at $d_k$, there must be at least $D_k - C_k$ units of time in which all $M$ processors are executing other higher-priority jobs.
- Definition: demand $W(\Delta)$ in a time interval with length $\Delta$ is the total amount of computation that needs to be completed within the interval.
- If $\tau_k$ misses its deadline at time $d_k$, then

$$W(D_k) > M(D_k - C_k) + C_k$$
Summary of Existing Results

Regarding to speedup factors

<table>
<thead>
<tr>
<th></th>
<th>implicit deadlines</th>
<th>constrained deadlines</th>
<th>arbitrary deadlines</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Global EDF</strong></td>
<td></td>
<td>2 – ( \frac{1}{M} ) (Bonifaci et al. 2008)</td>
<td></td>
</tr>
<tr>
<td><strong>Global DM</strong></td>
<td>3 – ( \frac{1}{M} ) (Bertogna et al. 2005)</td>
<td>3 – ( \frac{1}{M} ) (Baruah et al. 2010)</td>
<td>3 (Chen et al. 2018)</td>
</tr>
<tr>
<td>( \frac{3 + \sqrt{7}}{2} ) ≈ 2.823 (Chen et al. 2015)</td>
<td>3 (Chen et al. 2015)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Biondi and Sun’s Effect?

- The state-of-the-art schedulability analysis have issues for global fixed-priority schedulability and EDF analyses
- For example, if the task set is deemed schedulable under global RM (by using the above schedulability test), there is a partitioned schedule which meets all deadlines

- Youcheng Sun, Marco Di Natale: Assessing the pessimism of current multicore global fixed-priority schedulability analysis. SAC 2018: 575-583