Multiprocessor Scheduling I: Partitioned Scheduling

Prof. Dr. Jian-Jia Chen

LS 12, TU Dortmund

22/23, June, 2015
Outline

Introduction to Multiprocessor Scheduling

Partitioned Scheduling for Implicit-Deadline EDF Scheduling

Partitioned Scheduling for Implicit-Deadline RM Scheduling

Partitioned Scheduling for Constrained-Deadline DM Scheduling

Partitioned Scheduling for Arbitrary-Deadline EDF 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.

- **Semi-Partitioned Scheduling:**
  - Adopt task partitioning first and reserve time slots (bandwidths) for tasks that allow migration.
  - It requires some on-line overhead.
Course Material

Terminologies Used in Scheduling Theory

Graham’s Scheduling Algorithm Classification

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

- Makespan problem:
  - \( M||C_{\text{max}} \)
  - The course material mainly comes from the traditional makespan scheduling in the context of minimizing the maximum utilization. (The paper by Baruah and Fisher in RTSS 2005 is an exception.)
  - Imagine that the goal is to minimize the maximum utilization after task partitioning.
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 to Multiprocessor Scheduling

Partitioned Scheduling for Implicit-Deadline EDF Scheduling

Partitioned Scheduling for Implicit-Deadline RM Scheduling

Partitioned Scheduling for Constrained-Deadline DM Scheduling

Partitioned Scheduling for Arbitrary-Deadline EDF Scheduling
Problem Definition

Partitioned Scheduling

Given a set $T$ of tasks with implicit deadlines, i.e., $\forall \tau_i \in T$, $T_i = D_i$, 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 variable.
Deciding whether there exists a feasible task assignment is $\mathcal{NP}$-complete in the strong sense.

Proof

Reduced from the makespan or the bin packing 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 makespan or the bin packing 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.
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. $T_i \leftarrow \emptyset, U_i \leftarrow 0, \forall m = 1, 2, \ldots, M$;
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 $U_{m^*} \leftarrow U_{m^*} + \frac{C_i}{T_i}, T_i \leftarrow T_i \cup \{\tau_i\}$;
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. $T_i \leftarrow \emptyset$, $U_i \leftarrow 0, \forall m = 1, 2, \ldots, M$;
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 $U_{m^*} \leftarrow U_{m^*} + \frac{C_i}{T_i}$, $T_i \leftarrow T_i \cup \{\tau_i\}$;
   9. 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

(0, 0, 0)

\[\tau_1 \quad 0.5\]

\[\tau_2 \quad 0.45\]

\[\tau_3 \quad 0.37\]

\[\tau_4 \quad 0.3\]

\[\tau_5 \quad 0.2\]

\[\tau_6 \quad 0.2\]

\[\tau_7 \quad 0.15\]

\[\tau_8 \quad 0.1\]
Algorithm LUF

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

\((0.5, 0, 0)\)
Algorithm LUF

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

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

\((0.5, 0.45, 0)\)
Algorithm LUF

(.5, .45, .37)
Algorithm LUF

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

\[
(0.5, 0.45, 0.7)
\]
Algorithm LUF

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

\[
(0.5, 0.65, 0.67)
\]
Algorithm LUF

\[
\begin{align*}
\tau_1 & \quad (0.5, 0.0, 0.0) \\
\tau_2 & \quad (0.45, 0.0, 0.0) \\
\tau_3 & \quad (0.37, 0.0, 0.0) \\
\tau_4 & \quad (0.3, 0.0, 0.2) \\
\tau_5 & \quad (0.2, 0.0, 0.2) \\
\tau_6 & \quad (0.2, 0.15, 0.0) \\
\tau_7 & \quad (0.15, 0.0, 0.1) \\
\tau_8 & \quad (0.0, 0.0, 0.1) \\
\end{align*}
\]

\[
(0.7, 0.65, 0.67)
\]
Algorithm LUF

\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*}

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

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

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

\[ P_1, P_2, P_3 \]

\[ U = 0.7, U = 0.8, U = 0.77 \]
# 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. If you are interested, you can take the proof as an exercise. The proof can be found from Graham’s paper.
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, $1 \geq U_{opt} \geq \sum_{i=1}^{j} \frac{C_i}{T_i} M$.
- $\frac{C_i}{T_i} \leq \frac{1}{3} U_{opt}$: otherwise, there will be at most two tasks on any processors in the optimal solution. ⇒ this contradicts the assumption that Algorithm LUF fails as it is optimal.
- Since $U_{m*} \leq U_m$, we know that $U_{m*} \leq \sum_{m=1}^{M} \frac{U_m}{M} = \sum_{i=1}^{j-1} \frac{C_i/T_i}{M}$.
- Therefore,

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

**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. $T_i \leftarrow \emptyset$, $U_i \leftarrow 0$, $\forall m = 1, 2, \ldots, M$;
3. for $i = 1$ to $N$, where $N = |T|$ do
   4. find the processor $m^*$ with the minimum task utilization, i.e.,
      \[ U_{m^*} = \min_m U_m; \]
   5. assign task $\tau_i$ onto processor $m^*$, where
      \[ U_i \leftarrow U_i + \frac{C_i}{T_i}, T_i \leftarrow T_i \cup \{\tau_i\}; \]
4. return task assignment $T_1, T_2, \ldots, T_M$ with a speedup factor
   \[ \max\{1, U_m\}; \]
Properties of $LUF^*$

Theorem
If the input task set can be feasibly scheduled, Algorithm $LUF^*$ derives a feasible task assignment/schedule by running the processors at a speedup factor by at most $\frac{4}{3} - \frac{1}{3M}$.

Theorem
If the input task set cannot be feasibly scheduled by Algorithm $LUF$, the task set is not schedulable by running at a slow-down factor $\frac{1}{\frac{4}{3} - \frac{1}{3M}}$. 
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.

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

\[
\begin{align*}
P_1 & : 0.6 \\
P_2 & : 0.7 \\
P_3 & : 0.5 \\
P_4 & : 0.65
\end{align*}
\]
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.

\[ \begin{array}{cccc}
\text{First Fit} & 0.7 & 0.5 & 0.65 \\
\hline
P_1 & P_2 & P_3 & P_4
\end{array} \]
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.
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
  \[
  \sum_{\tau_i \in T} \frac{C_i}{T_i} \geq U_{m^*} + \sum_{m=1, m \neq m^*}^{\hat{M}} U_m \geq \hat{M} - 1 - (\hat{M} - 2)U_{m^*} = (\hat{M} - 2)(1 - U_{m^*}) + 1 \geq \frac{\hat{M}}{2}.
  \]
- 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.
Algorithm \textit{LUF Revisit: Upper and Lower Bounds}

- Let $m^*$ be the processor with the maximum utilization among $M$ processors.
- Let $\tau_k$ be the last task that is added to $m^*$.
- Therefore, we know that $U_m \geq U_{m^*} - \frac{C_k}{T_k}$ for any other processor $m$.
- The lower bound of the maximum utilization to map $T$ on $M$ processor is

\[
LB \geq \max \left\{ \max_{\tau_i \in T} \frac{C_i}{T_i}, \frac{\sum_{\tau_i \in T} C_i}{M} \right\}
\]

$\Rightarrow U_{m^*} \leq U_m + \frac{C_k}{T_k} \leq \sum_{m=1}^{M} U_m + \frac{C_k}{T_k} = \frac{\sum_{\tau_i \in T} C_i}{M} + \frac{C_k}{T_k} \leq 2LB.$

- Therefore, we reach that

\[
\max \left\{ \max_{\tau_i \in T} \frac{C_i}{T_i}, \frac{\sum_{\tau_i \in T} C_i}{M} \right\} = LB \leq U_{m^*} \leq 2LB.
\]
Outline

Introduction to Multiprocessor Scheduling

Partitioned Scheduling for Implicit-Deadline EDF Scheduling

Partitioned Scheduling for Implicit-Deadline RM Scheduling

Partitioned Scheduling for Constrained-Deadline DM Scheduling

Partitioned Scheduling for Arbitrary-Deadline EDF Scheduling
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 $U_i \leftarrow U_i + \frac{C_i}{T_i}$, $T_i \leftarrow T_i \cup \{\tau_i\}$, $n_{m^*} \leftarrow n_{m^*} + 1$;
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.
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 $U_{m^*} + \frac{C_i}{T_i} \leq (n_{m^*} + 1) \left(2^{\frac{n_{m^*}+1}{n_{m^*}+1}} - 1\right)$;
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 $U_i \leftarrow U_i + \frac{C_i}{T_i}, T_i \leftarrow T_i \cup \{\tau_i\}, n_{m^*} \leftarrow n_{m^*} + 1$
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 \( \frac{2}{0.693} \approx 2.887 \).
A More Precise Analysis

- If $\hat{M}$ is 1, we know that $\sum_{\tau_i \in T} \frac{c_i}{T_i} \leq N(2^{\frac{1}{N}} - 1)$.

- Suppose that the processor used by Algorithm $LUF^+$ is $\hat{M} \geq 2$.

- Let $k$ be the index of the task, at which processor $\hat{M}$ is allocated when running $LUF^+$. We only look at the iteration when $i$ is $k$. Therefore,

$$U_k + \sum_{\tau_i \in T_m} U_i > (n_m + 1) \left(2^{\frac{1}{nm+1}} - 1\right), \quad \forall m = 1, \ldots, \hat{M} - 1.$$ 

- By the sorting of the tasks, we also know that $U_i \geq U_k$ for any $i \leq k$. This also implies that $\sum_{\tau_i \in T_m} U_i > n_m \left(2^{\frac{1}{nm+1}} - 1\right)$.

- $x \left(2^{\frac{1}{x+1}} - 1\right)$ is an increasing function of $x$ when $x \geq 1$.

- Let $q$ be the minimum number of tasks assigned on a processor before task $\tau_k$, i.e., $1 \leq q \leq n_m, \forall m = 1, \ldots, \hat{M} - 1$. The approximation factor is $\sqrt{2} + 1$ since

$$U_k + \sum_{i=1}^{k-1} U_i > (1 + (\hat{M} - 1)q) \left(2^{\frac{1}{q+1}} - 1\right) \geq \hat{M}(\sqrt{2} - 1) \approx 0.414\hat{M}.$$
Remarks (Augmenting the Number of Processors)

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

Table 3: Approximation Ratios.

<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>
Outline

Introduction to Multiprocessor Scheduling

Partitioned Scheduling for Implicit-Deadline EDF Scheduling

Partitioned Scheduling for Implicit-Deadline RM Scheduling

Partitioned Scheduling for Constrained-Deadline DM Scheduling

Partitioned Scheduling for Arbitrary-Deadline EDF Scheduling
Definition and Theorem Recalled

For a task set, we say that the task set is with

- *constrained deadline* when the relative deadline $D_i$ is no more than the period $T_i$, i.e., $D_i \leq T_i$, for every task $\tau_i$, or
- *arbitrary deadline* when the relative deadline $D_i$ could be larger than the period $T_i$ for some task $\tau_i$.

**Theorem**

A task set $T$ of independent, preemptable, periodic tasks can be feasibly scheduled (under EDF) on one processor if and only if

$$\forall t \geq 0, \sum_{i=1}^{n} \max \left\{ 0, \left\lfloor \frac{t + T_i - D_i}{T_i} \right\rfloor \right\} C_i \leq t.$$
Problem Definition

Partitioned Scheduling

Given a set $T$ of tasks with arbitrary deadlines, the objective is to decide a feasible task assignment onto $M$ processors such that all the tasks meet their timing constraints, where $C_i$ is the execution time of task $τ_i$ on any processor $m$.

I will only focus on speed-up (resource augmentation) factors for the rest of the slides.
Shortest-Relative-Deadline First (SDF) with DM (SDF+DM)

**Input:** $T, M$
1. re-index (sort) tasks such that $D_i \leq D_j$ for $i < j$;
2. $T_i \leftarrow \emptyset$, $U_i \leftarrow 0$, $\forall m = 1, 2, \ldots, M$;
3. for $i = 1$ to $N$, where $N = |T|$ do
4. for $m = 1$ to $M$ do
5. if task $\tau_i$ can be feasibly scheduled with $T_m$ under DM scheduling then
6. assign task $\tau_i$ onto processor $m$ and $T_m \leftarrow T_m \cup \{\tau_i\}$;
7. break;
8. if $\tau_i$ is not assigned then
9. return "The task assignment fails";
10. return feasible task assignment $T_1, T_2, \ldots, T_M$;
Shortest-Relative-Deadline First (SDF) with DM (SDF+DM)

**Input:** \( T, M; \)

1. re-index (sort) tasks such that \( D_i \leq D_j \) for \( i < j \);
2. \( T_i \leftarrow \emptyset, U_i \leftarrow 0, \forall m = 1, 2, \ldots, M; \)
3. **for** \( i = 1 \) to \( N \), where \( N = |T| \) **do**
   4. **for** \( m = 1 \) to \( M \) **do**
      5. if task \( \tau_i \) can be feasibly scheduled with \( T_m \) under DM scheduling **then**
      6. assign task \( \tau_i \) onto processor \( m \) and \( T_m \leftarrow T_m \cup \{\tau_i\}; \)
      7. break;

10. return feasible task assignment \( T_1, T_2, \ldots, T_M; \)
Shortest-Relative-Deadline First (SDF) with DM (SDF+DM)

**Input:** $T, M$;

1. re-index (sort) tasks such that $D_i \leq D_j$ for $i < j$;
2. $T_i \leftarrow \emptyset, U_i \leftarrow 0, \forall m = 1, 2, \ldots, M$;
3. for $i = 1$ to $N$, where $N = |T|$ do
4. for $m = 1$ to $M$ do
5. if task $\tau_i$ can be feasibly scheduled with $T_m$ under DM scheduling then
6. assign task $\tau_i$ onto processor $m$ and $T_m \leftarrow T_m \cup \{\tau_i\}$;
7. break;
8. if $\tau_i$ is not assigned then
9. return "The task assignment fails";
10. return feasible task assignment $T_1, T_2, \ldots, T_M$;
Shortest-Relative-Deadline First (SDF) with DM (SDF+DM)

**Input:** \( T, M; \)
1. re-index (sort) tasks such that \( D_i \leq D_j \) for \( i < j \);
2. \( T_i \leftarrow \emptyset, U_i \leftarrow 0, \forall m = 1, 2, \ldots, M; \)
3. **for** \( i = 1 \) to \( N \), where \( N = |T| \) **do**
4. **for** \( m = 1 \) to \( M \) **do**
5. **if** task \( \tau_i \) can be feasibly scheduled with \( T_m \) under DM scheduling **then**
6. assign task \( \tau_i \) onto processor \( m \) and \( T_m \leftarrow T_m \cup \{\tau_i\}; \)
7. **break**;
8. **if** \( \tau_i \) is not assigned **then**
9. return ”The task assignment fails”;
10. return feasible task assignment \( T_1, T_2, \ldots, T_M; \)

The used test can be an exact test (by using TDA) or a utilization-based analysis.
Constrained-Deadline: Failure when Using TDA

- Suppose that task $\tau_k$ is the first task that fails to be assigned on any of the $M$ processors.
- Let $T^*$ be the set $\{\tau_1, \tau_2, \ldots, \tau_{k-1}\}$. Therefore,

$$\forall m, \forall t, \text{ with } 0 < t \leq D_k, \quad C_k + \sum_{\tau_i \in T_m} \left\lceil \frac{t}{T_i} \right\rceil C_i > t.$$ 

By taking a summation of all the $m = 1, 2, \ldots, M$ inequalities with respect to any $t$, we have

$$\forall t \text{ with } 0 < t \leq D_k, \quad MC_k + \sum_{\tau_i \in T^*} \left\lceil \frac{t}{T_i} \right\rceil C_i > Mt.$$ 

By taking the negation, we know that if

$$\exists t \text{ with } 0 < t \leq D_k, \text{ and } C_k + \sum_{\tau_i \in T^*} \left\lceil \frac{t}{T_i} \right\rceil \frac{C_i}{M} \leq t, \quad (1)$$

then the SDF+DM by using TDA should succeed to assign task $\tau_k$ on one of the $M$ processors.
Constrained-Deadline: Schedulability Test for TDA

This is basically very similar to TDA with a minor difference by dividing the higher-priority workload by $M$. Testing the schedulability condition of task $\tau_k$ can be done by using the same strategy used in the $k^2U$ framework. We classify the $k - 1$ tasks in $T^*$ into two subsets.

- $T^{*1}$ consists of the tasks in $T^*$ with period smaller than $D_k$.
- $T^{*2}$ consists of the tasks in $T^*$ with period larger than or equal to $D_k$.

Let $C'_k$ be defined as follows:

$$C'_k = C_k + \sum_{\tau_i \in T^{*2}} \frac{C_i}{M}.$$ 

Now, we can rewrite the condition as follows: if

$$\exists t \text{ with } 0 < t \leq D_k \text{ and } C'_k + \sum_{\tau_i \in T^{*1}} \left\lfloor \frac{t}{T_i} \right\rfloor \frac{C_i}{M} \leq t,$$

then SDF+DM by using TDA should succeed to assign task $\tau_k$ on one of the $M$ processors.
Recall k-Point Effective Schedulability Test: $k^2U$

Suppose that $\{t_1, t_2, \ldots, t_k\}$ are given.

**Definition**

A $k$-point effective schedulability test is a sufficient test by verifying the existence of $t_j \in \{t_1, t_2, \ldots, t_k\}$ with $t_1 \leq t_2 \leq \cdots \leq t_k$ such that

$$C_k + \sum_{i=1}^{k-1} \alpha_i t_i U_i + \sum_{i=1}^{j-1} \beta_i t_i U_i \leq t_j,$$

where $C_k > 0$, $\alpha_i > 0$, $U_i > 0$, and $\beta_i > 0$ are dependent upon the setting of the task models and task $\tau_i$.

**Lemma**

[Lemma 1] For a given $k$-point effective schedulability test of a scheduling algorithm, in which $0 < \alpha_i \leq \alpha \neq \infty$, and $0 < \beta_i \leq \beta \neq \infty$ for any $i = 1, 2, \ldots, k-1$, $0 < t_k \neq \infty$, task $\tau_k$ is schedulable by the scheduling algorithm if the following condition holds

$$\frac{C_k}{t_k} \leq \frac{\alpha}{\beta} + 1 - \frac{\alpha}{\beta}.$$  

(3)
Constrained-Deadline: Utilization-Base Test for TDA

Theorem

If

\[ \prod_{\tau_i \in T^*1} (1 + \frac{U_i}{M}) \leq \frac{2}{1 + \frac{c'_k}{d_k}}, \]

then a constrained-deadline task \( \tau_k \) is schedulable under SDF+DM by using TDA.

Proof

- Let \( t_i \) be \( \left\lfloor \frac{T_k}{T_i} \right\rfloor T_i \) for \( \tau_i \in T^*1 \).
- When using the \( k^2U \) framework, we can reach the conclusion that
  - \( \alpha_i = \frac{1}{M}, \) and \( 0 < \beta_i \leq \frac{1}{M} \)
  - \( t_k \) is \( D_k \) and \( C_k \) (in the previous lemma) is \( C'_k \).
- By adopting the above lemma, we reach the conclusion.
Using Utilization-Based Test in SDF+DM

Instead of using the exact test, we can also use the utilization-based test (in the hyperbolic form) for deadline-monotonic scheduling (see slide 30 in 11-k2u.pdf). Surprisingly, the result is the same as the case by using TDA analysis.

- One of your exercise assignments is for the special case by using the hyperbolic form in the schedulability test for RM.
### Summary of Existing Results

<table>
<thead>
<tr>
<th></th>
<th>implicit deadlines</th>
<th>constrained deadlines</th>
<th>arbitrary deadlines</th>
</tr>
</thead>
</table>
| partitioned with EDF | $\frac{4}{3} - \frac{1}{3M}$ (Graham 1969)
               | $(1 + \epsilon)$ (Hochbaum/Shmoys 1987) |  $3 - \frac{1}{M}$ (Baruah/Fisher 2006)
               |  $4 - \frac{2}{M}$ (Baruah/Fisher 2005) |
|                |                                    | 2.6322 $- \frac{1}{M}$ (Chen/Chakraborty 2011) |  $3 - \frac{1}{M}$ (Chen/Chakraborty 2011) |
| partitioned with DM | (bin-packing) $\frac{7}{4}$ (Burachard et al. 1995)  
               | (bin-packing) 1.5 (Rothvoß2009)            |  $3 - \frac{1}{M}$ (Baker/Fisher/Baruah 2009) |
|                |                                    | 2.84306 (Chen 2015)                      |  $4 - \frac{2}{M}$ (Baker/Fisher/Baruah 2009) |
|                |                                    |                                        |  $3 - \frac{1}{M}$ (Chen 2015) |

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

Introduction to Multiprocessor Scheduling

Partitioned Scheduling for Implicit-Deadline EDF Scheduling

Partitioned Scheduling for Implicit-Deadline RM Scheduling

Partitioned Scheduling for Constrained-Deadline DM Scheduling

Partitioned Scheduling for Arbitrary-Deadline EDF Scheduling
Define demand bound function \( dbf(\tau_i, t) \) as

\[
dbf(\tau_i, t) = \max \left\{ 0, \left\lfloor \frac{t + T_i - D_i}{T_i} \right\rfloor \right\} \quad C_i = \max \left\{ 0, \left\lfloor \frac{t - D_i}{T_i} \right\rfloor + 1 \right\} C_i.
\]

We need approximation to enforce \textit{polynomial-time} schedulability test.

\[
dbf^*(\tau_i, t) = \begin{cases} 
0 & \text{if } t < D_i \\
\left( \frac{t - D_i}{T_i} + 1 \right) C_i & \text{otherwise.}
\end{cases}
\]
Define demand bound function $dbf(\tau_i, t)$ as

$$dbf(\tau_i, t) = \max \left\{ 0, \left\lfloor \frac{t + T_i - D_i}{T_i} \right\rfloor \right\} \quad C_i = \max \left\{ 0, \left\lfloor \frac{t - D_i}{T_i} \right\rfloor + 1 \right\} C_i.$$ 

We need approximation to enforce *polynomial-time* schedulability test.

$$dbf^*(\tau_i, t) = \begin{cases} 0 & \text{if } t < D_i \\ \left( \frac{t-D_i}{T_i} + 1 \right) C_i & \text{otherwise.} \end{cases}$$
Define demand bound function $\text{dbf}(\tau_i, t)$ as

$$
\text{dbf}(\tau_i, t) = \max \left\{ 0, \left\lfloor \frac{t + T_i - D_i}{T_i} \right\rfloor \right\}
$$

$$
C_i = \max \left\{ 0, \left\lfloor \frac{t - D_i}{T_i} \right\rfloor + 1 \right\}
$$

We need approximation to enforce polynomial-time schedulability test.

$$
\text{dbf}^*(\tau_i, t) = \begin{cases} 
0 & \text{if } t < D_i \\
\left( \frac{t - D_i}{T_i} + 1 \right) C_i & \text{otherwise}
\end{cases}
$$

$$
\text{dbf}(\tau_i, t) \leq \text{dbf}^*(\tau_i, t) \leq 2 \text{dbf}(\tau_i, t)
$$
Shortest Relative-Deadline First (SDF + EDF)

**Input:** \( T, M \);

1. re-index (sort) tasks such that \( D_i \leq D_j \) for \( i < j \);
2. \( T_i \leftarrow \emptyset, U_i \leftarrow 0, \forall m = 1, 2, \ldots, M; \)
3. for \( i = 1 \) to \( N \), where \( N = |T| \) do

4. for \( m = 1 \) to \( M \) do

5. if \( C_i \sum_{\tau_j \in T_m} C_j \tau_j \leq 1 \) and \( C_i \sum_{\tau_j \in T_m} \text{dbf}^\ast(\tau_j, D_i) \leq D_i \) then

6. assign task \( \tau_i \) onto processor \( m \) and \( T_m \leftarrow T_m \cup \{\tau_i\} \);

7. \textbf{break};

8. if \( \tau_i \) is not assigned then

9. return "The task assignment fails"

10. return feasible task assignment \( T_1, T_2, \ldots, T_M \);
Shortest Relative-Deadline First (SDF+EDF)

**Input:** $T, M$;

1: re-index (sort) tasks such that $D_i \leq D_j$ for $i < j$;
2: $T_i \leftarrow \emptyset, U_i \leftarrow 0, \forall m = 1, 2, \ldots, M$;
3: for $i = 1$ to $N$, where $N = |T|$ do
4: for $m = 1$ to $M$ do
5: if $\frac{C_i}{T_i} + \sum_{\tau_j \in T_m} \frac{C_j}{T_j} \leq 1$ and $C_i + \sum_{\tau_j \in T_m} df^*(\tau_j, D_i) \leq D_i$ then
6: assign task $\tau_i$ onto processor $m$ and $T_m \leftarrow T_m \cup \{\tau_i\}$;
7: break;

8: if $\tau_i$ is not assigned then
9: return "The task assignment fails";
10: return feasible task assignment $T_1, T_2, \ldots, T_M$;
Shortest Relative-Deadline First (SDF + EDF)

**Input:** $T, M$

1. re-index (sort) tasks such that $D_i \leq D_j$ for $i < j$;
2. $T_i \leftarrow \emptyset, U_i \leftarrow 0, \forall m = 1, 2, \ldots, M$;
3. **for** $i = 1$ to $N$, where $N = |T|$ **do**
   4. **for** $m = 1$ to $M$ **do**
   5. if $\frac{C_i}{T_i} + \sum_{\tau_j \in T_m} \frac{C_j}{T_j} \leq 1$ and $C_i + \sum_{\tau_j \in T_m} dbf^*(\tau_j, D_i) \leq D_i$ **then**
   6. assign task $\tau_i$ onto processor $m$ and $T_m \leftarrow T_m \cup \{\tau_i\}$;
   7. break;
   8. if $\tau_i$ is not assigned **then**
   9. return "The task assignment fails";
10. return feasible task assignment $T_1, T_2, \ldots, T_M$;
Feasibility

Theorem

If the tasks are partitioned successfully by using SDF+EDF Partition, EDF can feasibly schedule the tasks.

This is due to the feasibility analysis of the linear approximation of the demand bound functions.
Define that

- $\delta_{\text{max}}$ is the maximum density, i.e., $\max_{\tau_i \in \mathcal{T}} \frac{C_i}{D_i}$.
- $\mu_{\text{max}}$ is the maximum utilization, i.e., $\max_{\tau_i \in \mathcal{T}} \frac{C_i}{T_i}$.
- $\delta_{\text{max}}$ and $\mu_{\text{max}}$ are both less than or equal to 1.
- $\delta_{\text{sum}}$ is the maximum total dbf density, i.e.,
  $$\max_{t>0} \sum_{\tau_i \in \mathcal{T}} \frac{\text{dbf}(\tau_i, t)}{t}.$$
- $\mu_{\text{sum}}$ is the total utilization, i.e., $\sum_{\tau_i \in \mathcal{T}} \frac{C_i}{T_i}$.

**Theorem**

Any sporadic task set is successfully scheduled by SDF+EFD on $M$ identical processors, when

$$M \geq \frac{2\delta_{\text{sum}} - \delta_{\text{max}}}{1 - \delta_{\text{max}}} + \frac{\mu_{\text{sum}} - \mu_{\text{max}}}{1 - \mu_{\text{max}}}.$$
Proof of Schedulability

Suppose that $\tau_k$ is not able to be assigned on any processor by SDF+EDF.

- Either $\frac{C_k}{T_k} + \sum_{\tau_j \in T'_m} \frac{C_j}{T_j} > 1$ or $C_k + \sum_{\tau_j \in T'_m} dbf^*(\tau_j, D_k) > D_k$.

- Suppose that among $M$ processors, there are $M_1$ processors, denoted by set $M_1$, with $C_k + \sum_{\tau_j \in T'_m} dbf^*(\tau_j, D_k) > D_k$.

- Therefore, by summing the inequalities together, we have

$$M_1 C_k + \sum_{\tau_j \in T'_m \text{ and } m \in M_1} dbf^*(\tau_j, D_k) > M_1 D_k.$$  

$$\Rightarrow \sum_{\tau_j \in T \setminus \{\tau_k\}} 2dbf(\tau_j, D_k) > M_1(D_k - C_k)$$

$$\Rightarrow \sum_{\tau_j \in T} 2dbf(\tau_j, D_k) > M_1(D_k - C_k) + C_k$$

Divided by $D_k$

$$\Rightarrow \frac{\sum_{\tau_j \in T} 2dbf(\tau_j, D_k)}{D_k} > M_1 \frac{D_k - C_k}{D_k} + \frac{C_k}{D_k}$$

$$\Rightarrow 2\delta_{\text{sum}} > M_1 \frac{D_k - C_k}{D_k} + \frac{C_k}{D_k}$$

- By above, we know that $M_1 < \frac{2\delta_{\text{sum}} - \frac{C_k}{D_k}}{1 - \frac{C_k}{D_k}}$. 
Proof of Schedulability

- $M_2 = M - M_1$ processors are purely with $\frac{C_k}{T_k} + \sum_{T_j \in T_m} \frac{C_j}{T_j} > 1$.

- Similarly, $M_2 < \frac{\mu \text{sum} - \frac{C_k}{T_k}}{1 - \frac{C_k}{T_k}}$.

- As a result, it can be shown that

$$M = M_1 + M_2 < \frac{2\delta \text{sum} - \delta_{\text{max}}}{1 - \delta_{\text{max}}} + \frac{\mu \text{sum} - \mu_{\text{max}}}{1 - \mu_{\text{max}}}.$$
**Theorem**

If a sporadic task set is schedulable on \( M \) identical processors, SDF+EDF will successfully partition this task set upon a platform comprised of \( M \) processors that are each \( (4 - \frac{2}{M}) \) times as fast as the original.

**Proof**

The necessary condition to be schedulable on \( M \) identical processors is

\[
\delta_{\text{max}} \leq 1, \mu_{\text{max}} \leq 1, \mu_{\text{sum}} \leq M, \text{ and } \delta_{\text{sum}} \leq M.
\]

If speeding up by \( \frac{1}{\gamma} \) satisfies

\[
M \geq \frac{2\delta_{\text{sum}} - \delta_{\text{max}}}{1 - \delta_{\text{max}}} + \frac{\mu_{\text{sum}} - \mu_{\text{max}}}{1 - \mu_{\text{max}}},
\]

it is feasible by applying SDF+EDF. Therefore,

\[
\frac{2\delta_{\text{sum}} - \delta_{\text{max}}}{1 - \delta_{\text{max}}} + \frac{\mu_{\text{sum}} - \mu_{\text{max}}}{1 - \mu_{\text{max}}} \leq \frac{2M\gamma - \gamma}{1 - \gamma} + \frac{M\gamma - \gamma}{1 - \gamma} = \frac{3M\gamma - 2\gamma}{1 - \gamma}
\]

If \( M \geq \frac{3M\gamma - 2\gamma}{1 - \gamma} \), the schedulability is enforced. \( \Rightarrow 4 - \frac{2}{M} \leq \frac{1}{\gamma} \).
SDF+EDF for Constrained Deadlines

**Theorem**

Any sporadic task set with constrained deadlines is successfully scheduled by SDF+EDF on $M$ identical processors, when

$$M \geq \frac{2\delta_{\text{sum}} - \delta_{\text{max}}}{1 - \delta_{\text{max}}}.$$  

**Theorem**

If a sporadic task set with constrained deadlines is schedulable on $M$ identical processors, SDF+EDF will successfully partition this task set upon a platform comprised of $M$ processors that are each $(3 - \frac{1}{M})$ times as fast as the original.

Main difference: Condition $\frac{C_i}{T_i} + \sum_{\tau_j \in T'_m} \frac{C_j}{T_j} \leq 1$ in Step 5 is always enforced when $C_i + \sum_{\tau_j \in T_m} \text{dbf}^*(\tau_j, D_i) \leq D_i$ holds. ($\Rightarrow M_2$ is always 0).
Theorem

[Fisher and Baruah, RTSS 2005] The resource augmentation factor for SDF+EDF has

- a $4 - \frac{2}{M}$ resource augmentation factor for tasks with arbitrary deadlines
- a $3 - \frac{1}{M}$ resource augmentation factor for tasks with constrained deadlines
Feasibility

**Theorem**

[Fisher and Baruah, RTSS 2005] The resource augmentation factor for SDF+EDF has

- a $4 - \frac{2}{M}$ resource augmentation factor for tasks with arbitrary deadlines
- a $3 - \frac{1}{M}$ resource augmentation factor for tasks with constrained deadlines

**Theorem**

[Chen and Chakraborty, RTSS 2011] The SDF+EDF has

- a $3 - \frac{1}{M}$ resource augmentation factor for tasks with arbitrary deadlines
- a $\frac{3e-1}{e} - \frac{1}{M} \approx 2.6322 - \frac{1}{M}$ resource augmentation factor for tasks with constrained deadlines
## Summary of Existing Results

<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>((1 + \epsilon)) (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>((\text{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>((\text{bin-packing}) 1.5) (Rothvoß2009)</td>
<td>2.84306 (Chen 2015)</td>
<td>3 - (\frac{1}{M}) (Chen 2015)</td>
</tr>
</tbody>
</table>

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