Multiprocessor Scheduling I: Partitioned Scheduling

Prof. Dr. Jian-Jia Chen

LS 12, TU Dortmund

11 July 2017
17 July 2017

Outline

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

- Ronald L. Graham: Bounds for certain multiprocessing anomalies. in Bell System Technical Journal (1966).
- Ronald L. Graham: Bounds on Multiprocessing Timing Anomalies. SIAM Journal of Applied Mathematics 17(2): 416-429 (1969)
- Dorit S. Hochbaum, David B. Shmoys: Using dual approximation algorithms for scheduling problems theoretical and practical results. J. ACM 34(1): 144-162 (1987) (in textbook Approximation Algorithms by Vijay Vazirani, Chapter 10, not covered)
- Sanjoy K. Baruah, Nathan Fisher: The Partitioned Multiprocessor Scheduling of Sporadic Task Systems. RTSS 2005: 321-329
- Jian-Jia Chen: Partitioned Multiprocessor Fixed-Priority Scheduling of Sporadic Real-Time Tasks. ECRTS 2016.

Problem Definition

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 τ_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.

Hardness and Approximation of Partitioned Scheduling

$\mathcal{NP}$-complete

Deciding whether there exists a feasible task assignment is $\mathcal{NP}$-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 η-approximation algorithm (for a minimization problem) if it guarantees to derive a feasible solution for any input instance I with at most η times of the objective function of an optimal solution. That is,

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

where $OPT(I)$ 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., $L_{\max}$, sum of finish times, ...)
- Makespan problem:
  - $M||C_{\max}$
  - 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

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

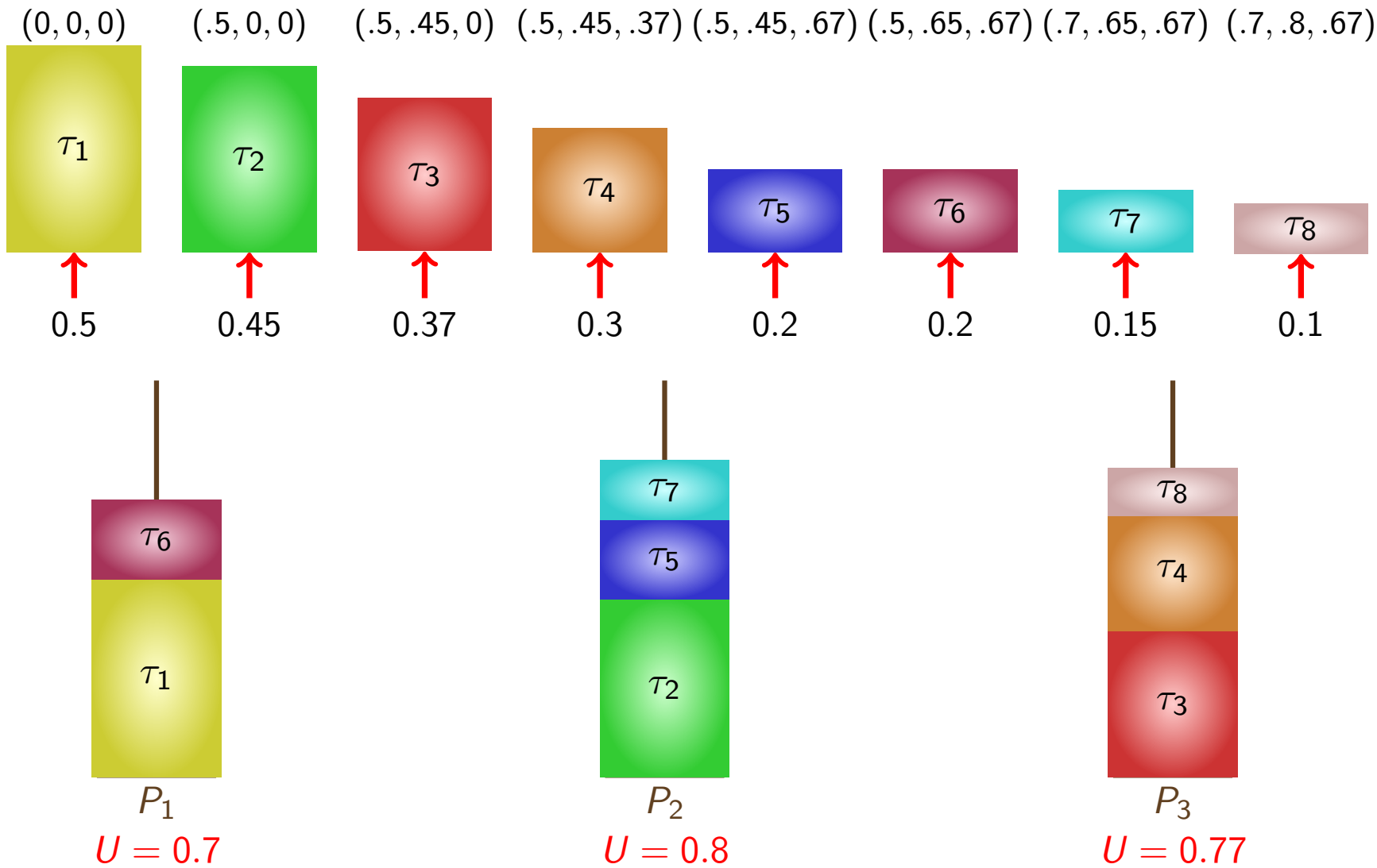
Input: $\mathbf{T}, M$;

1: re-index (sort) tasks such that $\frac{C_i}{T_i} \geq \frac{C_j}{T_j}$ for $i < j$;
2: $\mathbf{T}_m \leftarrow \emptyset, U_m \leftarrow 0, \forall m = 1, 2, \ldots, M$;
3: **for** $i = 1$ to N, where $N = |\mathbf{T}|$ **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 τ_i onto processor m^*, where $U_{m^*} \leftarrow U_{m^*} + \frac{C_i}{T_i}, \mathbf{T}_{m^*} \leftarrow \mathbf{T}_{m^*} \cup \{\tau_i\}$;
9: return feasible task assignment $\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_M$;

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



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 τ_j and U_m for $m = 1, 2, \ldots, M$ is the utilization of processor m before assigning τ_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, $1 \geq U_{opt} \geq \sum_{i=1}^{j} \frac{C_i/T_i}{M}$.
- $\frac{C_j}{T_j} \leq \frac{1}{3} U_{opt}$: 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 $U_{m^*} \leq \sum_{m=1}^{M} \frac{U_m}{M} = \sum_{i=1}^{j-1} \frac{C_i/T_i}{M}$.
- 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: $\mathbf{T}, M$;

1: re-index (sort) tasks such that $\frac{C_i}{T_i} \geq \frac{C_j}{T_j}$ for $i < j$;
2: $\mathbf{T}_m \leftarrow \emptyset, U_m \leftarrow 0, \forall m = 1, 2, \ldots, M$;
3: **for** $i = 1$ to N, where $N = |\mathbf{T}|$ **do**
4: find the processor m^* with the minimum task utilization, i.e., $U_{m^*} = \min_m U_m$;
5: assign task τ_i onto processor m^*, where $U_{m^*} \leftarrow U_{m^*} + \frac{C_i}{T_i}, \mathbf{T}_{m^*} \leftarrow \mathbf{T}_{m^*} \cup \{\tau_i\}$;
6: return task assignment $\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_M$ 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: $\mathbf{T}$;

1: re-index (sort) tasks such that $\frac{C_i}{T_i} \geq \frac{C_j}{T_j}$ for $i < j$;
2: $\mathbf{T}_1 \leftarrow \emptyset, U_1 \leftarrow 0, \hat{M} \leftarrow 1$;
3: **for** $i = 1$ to N, where $N = |\mathbf{T}|$ **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: $\hat{M} \leftarrow \hat{M} + 1, \mathbf{T}_{\hat{M}} \leftarrow \emptyset, U_{\hat{M}} \leftarrow 0$;
7: $m^* \leftarrow \hat{M}$;
8: assign task τ_i onto processor m^*, where $U_{m^*} \leftarrow U_{m^*} + \frac{C_i}{T_i}, \mathbf{T}_{m^*} \leftarrow \mathbf{T}_{m^*} \cup \{\tau_i\}$;
9: return task assignment $\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_{\hat{M}}$;

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.

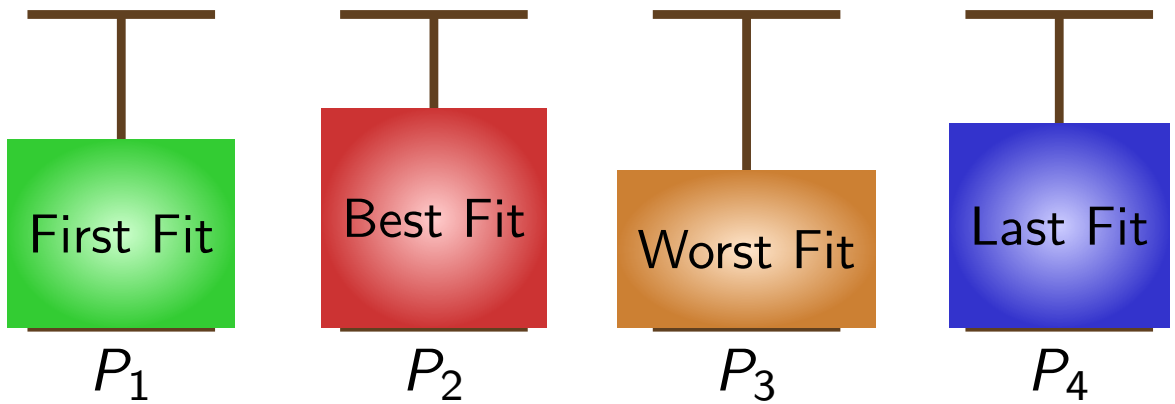
Different Fitting Approaches

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.



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 ms.
- If $U_{m^*} \leq 0.5$, by $U_m > 1 - U_{m^*}$, we know that

$$\sum_{\tau_i \in \mathbf{T}} \frac{C_i}{T_i} \geq U_{m^*} + \sum_{m=1, m \neq m^*}^{\hat{M}} U_m \geq \hat{M} - 1 - (\hat{M} - 2) U_{m^*} \leq (\hat{M} - 2)(1 - U_{m^*}) + 1 \geq \frac{\hat{M}}{2}.$$

- If $U_{m^*} > 0.5$, by $U_m \geq U_{m^*}$, we know that

$$\sum_{\tau_i \in \mathbf{T}} \frac{C_i}{T_i} \geq U_{m^*} + \sum_{m=1, m \neq m^*}^{\hat{M}} U_m \geq \frac{\hat{M}}{2}.$$

Theorem

Algorithm LUF^+ is a 2-approximation algorithm.

Algorithm *LUF* Revisit: Upper and Lower Bounds

- Let m^* be the processor with the maximum utilization among M processors.
- Let τ_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 $\mathbf{T}$ on M processor is

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

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

- Therefore, we reach that

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

Outline

Largest-Utilization-First (LUF^+) - for RM Scheduling

Input: $\mathbf{T}$;

1: re-index (sort) tasks such that $\frac{C_i}{T_i} \geq \frac{C_j}{T_j}$ for $i < j$;
2: $\mathbf{T}_1 \leftarrow \emptyset, U_1 \leftarrow 0, n_1 \leftarrow 0; \hat{M} \leftarrow 1$;
3: **for** $i = 1$ to N, where $N = |\mathbf{T}|$ **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: $\hat{M} \leftarrow \hat{M} + 1, \mathbf{T}_{\hat{M}} \leftarrow \emptyset, U_{\hat{M}} \leftarrow 0, n_{\hat{M}} \leftarrow 0$;
7: $m^* \leftarrow \hat{M}$;
8: assign task τ_i onto processor m^*, where
$U_{m^*} \leftarrow U_{m^*} + \frac{C_i}{T_i}, \mathbf{T}_{m^*} \leftarrow \mathbf{T}_{m^*} \cup \{\tau_i\}, n_{m^*} \leftarrow n_{m^*} + 1$;
9: return task assignment $\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_{\hat{M}}$;

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 $U_{m^*} + \frac{C_i}{T_i} \leq (n_{m^*} + 1)\left(2^{\frac{1}{n_{m^*}+1}} - 1\right)$ 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 $\sum_{\tau_i \in \mathbf{T}} \frac{C_i}{T_i} \leq N(2^{\frac{1}{N}} - 1)$.
- 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,

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

- By the sorting of the tasks, we also know that $U_i \geq U_k$ for any $i \leq k$. This also implies that $\sum_{\tau_i \in \mathbf{T}_m} U_i > n_m\left(2^{\frac{1}{n_m+1}} - 1\right)$.
- $x\left(2^{\frac{1}{x+1}} - 1\right)$ is an increasing function of x when $x \geq 1$.
- Let q be the minimum number of tasks assigned on a processor before task τ_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.

Algorithm	**Approximation Ratio ($\Re_A$)**	**Ref.**
RMNF	2.67	[Dhall and Liu 1978]
RMFF	2.33	[Oh and Son 1993]
RMBF	2.33	[Oh and Son 1993]
RRM-FF	2	[Oh and Son 1995]
FFDUF	2	[Davari and Dhall 1986]
RMST	$1/(1-u_{\max})$	[Burchard et al. 1995]
RMGT	7/4	[Burchard et al. 1995]
RMMatching	3/2	[Rothvoß 2009]
EDF-FF	1.7	[Garey and Johnson 1979]
EDF-BF	1.7	[Garey and Johnson 1979]

Outline

Problem Definition

Partitioned Scheduling

Given a set $\mathbf{T}$ 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: $\mathbf{T}, M$;

re-index (sort) tasks such that $D_i \leq D_j$ for $i < j$;
$\mathbf{T}_m \leftarrow \emptyset, U_m \leftarrow 0, \forall m = 1, 2, \ldots, M$;
for $i = 1$ to N, where $N = |\mathbf{T}|$ **do**
  for $m = 1$ to M **do**
    if task τ_i can be feasibly scheduled with $\mathbf{T}_m$ under DM scheduling **then**
      assign task τ_i onto processor m and $\mathbf{T}_m \leftarrow \mathbf{T}_m \cup \{\tau_i\}$;
      break;
  if τ_i is not assigned **then**
    return "The task assignment fails";
return feasible task assignment $\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_M$;

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

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

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

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

By taking the negation, we know that if

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

then the SDF+DM by using TDA should succeed to assign task τ_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 τ_k can be done by using the same strategy used in the k^2U framework.
We classify the $k-1$ tasks in $\mathbf{T}^*$ into two subsets.

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

Let C'_k be defined as follows:

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

Now, we can rewrite the condition as follows: if

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

then SDF+DM by using TDA should succeed to assign task τ_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

$$C_k + \sum_{i=1}^{k-1} \alpha_i t_i U_i + \sum_{i=1}^{j-1} \beta_i t_i U_i \leq t_j, \tag{2}$$

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 τ_i.

Lemma

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

$$\frac{C_k}{t_k} \leq \frac{\frac{\alpha}{\beta} + 1}{\prod_{j=1}^{k-1}(\beta U_j + 1)} - \frac{\alpha}{\beta}. \tag{3}$$

Constrained-Deadline: Utilization-Base Test for TDA

Theorem

If

$$\prod_{\tau_i \in \mathbf{T}^{*1}} (1 + \frac{U_i}{M}) \leq \frac{2}{1 + \frac{C'_k}{D_k}},$$

then a constrained-deadline task τ_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 $\tau_i \in \mathbf{T}^{*1}$.
- 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. 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.

Outline

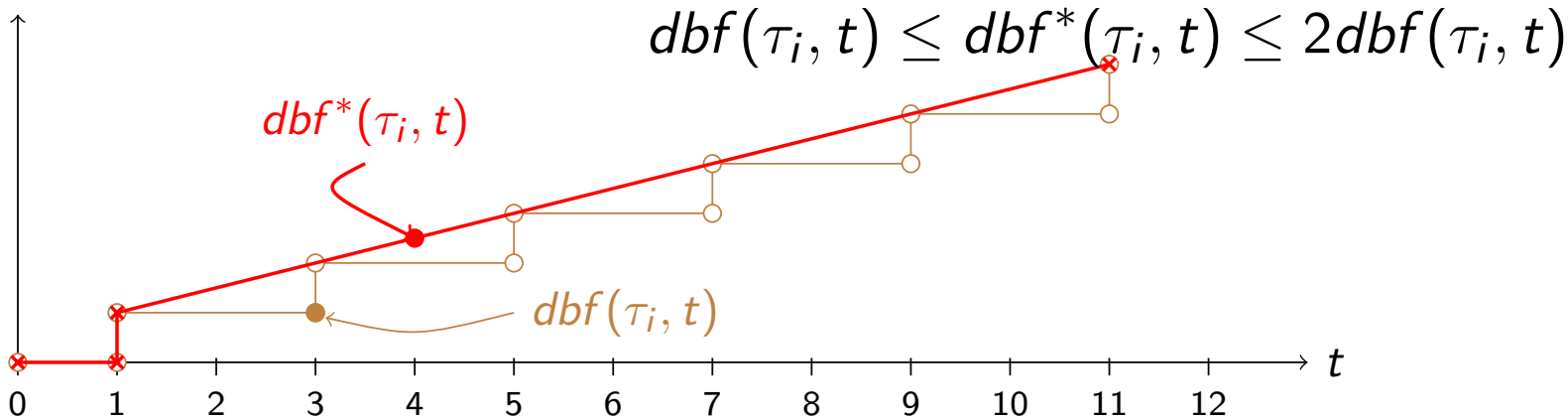
Demand Bound Function Revisit for EDF

Define demand bound function $dbf(\tau_i, t)$ as

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

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

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



Shortest Relative-Deadline First (SDF+EDF)

Input: $\mathbf{T}, M$;

re-index (sort) tasks such that $D_i \leq D_j$ for $i < j$;
$\mathbf{T}_m \leftarrow \emptyset, U_m \leftarrow 0, \forall m = 1, 2, \ldots, M$;
for $i = 1$ to N, where $N = |\mathbf{T}|$ **do**
  for $m = 1$ to M **do**
    if $\frac{C_i}{T_i} + \sum_{\tau_j \in \mathbf{T}_m} \frac{C_j}{T_j} \leq 1$ and
    $C_i + \sum_{\tau_j \in \mathbf{T}_m} dbf^*(\tau_j, D_i) \leq D_i$ **then**
      assign task τ_i onto processor m and $\mathbf{T}_m \leftarrow \mathbf{T}_m \cup \{\tau_i\}$;
      break;
  if τ_i is not assigned **then**
    return "The task assignment fails";
return feasible task assignment $\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_M$;

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.

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

	implicit deadlines	constrained deadlines	arbitrary deadlines
partitioned with EDF	$\frac{4}{3} - \frac{1}{3M}$ (Graham 1969)	$3 - \frac{1}{M}$ (Baruah/Fisher 2006)	$4 - \frac{2}{M}$ (Baruah/Fisher 2005)
	$(1 + \epsilon)$ (Hochbaum/Shmoys 1987)	2.6322 $- \frac{1}{M}$ (Chen/Chakraborty 2011)	$3 - \frac{1}{M}$ (Chen/Chakraborty 2011)
partitioned with DM	(bin-packing) $\frac{7}{4}$ (Burchard et al. 1995)	$3 - \frac{1}{M}$ (Baker/Fisher/Baruah 2009)	$4 - \frac{2}{M}$ (Baker/Fisher/Baruah 2009)
	(bin-packing) 1.5 (Rothvoß2009)	2.84306 (Chen 2016)	$3 - \frac{1}{M}$ (Chen 2016)

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