k²U and k²Q: A General Framework from k-Point Effective Schedulability Analysis to Utilization-Based Tests

Prof. Dr. Jian-Jia Chen

TU Dortmund

30/31,05,2016
Outline

Introduction

Utilization-Based Analytical Framework

Selected Applications

Conclusions
Outline

Introduction

Utilization-Based Analytical Framework

Selected Applications

Conclusions
Schedulability Test of Fixed-Priority (FP) Scheduling

Time Demand Analysis (TDA): Task $\tau_k$ (with $D_i = T_i$) can be feasibly scheduled by a fixed-priority scheduling algorithm if

$$\exists t \text{ with } 0 < t \leq T_k \text{ and } C_k + \sum_{j=1}^{k-1} \left\lceil \frac{t}{T_j} \right\rceil C_j \leq t.$$  

(I will implicitly assume $k - 1$ higher-priority tasks.)
Schedulability Test of Fixed-Priority (FP) Scheduling

Time Demand Analysis (TDA): Task \(\tau_k\) (with \(D_i = T_i\)) can be feasibly scheduled by a fixed-priority scheduling algorithm if

\[
\exists t \text{ with } 0 < t \leq T_k \text{ and } C_k + \sum_{j=1}^{k-1} \left\lceil \frac{t}{T_j} \right\rceil C_j \leq t.
\]

(I will implicitly assume \(k - 1\) higher-priority tasks.)

- This test takes pseudo-polynomial time
- If \(C_k\) is small enough, it always answers “schedulable”.
- Can we derive such a bound of \(C_k\) efficiently?
The non-schedulability of task $\tau_k$ in rate-monotonic scheduling (RM) implies the following structure if $2T_1 \geq T_k$:

$$\tau_1 \uparrow$$
$$\tau_2 \uparrow$$
$$\tau_3 \uparrow$$
$$\vdots$$
$$\tau_{k-1} \uparrow$$
$$\tau_k \uparrow$$

Minimize $\sum_{i=1}^{k} U_i$

$$C_k + \sum_{i=1}^{k-1} C_i + \sum_{i=0}^{j-1} C_i > T_j, \forall j = 1, 2, \ldots, k - 1,$$

$$C_k + 2 \sum_{i=1}^{k-1} C_i > T_k.$$
The non-schedulability of task $\tau_k$ in rate-monotonic scheduling (RM) implies the following structure if $2T_1 \geq T_k$:

Minimize $\sum_{i=1}^{k} U_i$

$$C_k + \sum_{i=1}^{k-1} T_i U_i + \sum_{i=0}^{j-1} T_i U_i > T_j, \forall j = 1, 2, \ldots, k - 1,$$

$$C_k + 2 \sum_{i=1}^{k-1} T_i U_i > T_k.$$
Key Questions

Does there exist a unified utilization-based analysis, regardless of the platform model or the task model?
Key Questions

Does there exist a unified utilization-based analysis, regardless of the platform model or the task model?

How can we find the bound of $C_k$ efficiently?

Can we build utilization-based analysis almost automatically by referring to a schedulability test?
Outline

Introduction

Utilization-Based Analytical Framework

Selected Applications

Conclusions
Revisit (Liu and Layland-Bound) Structure

- \( C_i = T_i U_i \)
- The non-schedulability of \( \tau_k \) implies such a structure if \( 2T_1 \geq T_k \):

\[
\tau_1 \quad \tau_2 \quad \tau_3 \quad \tau_{k-1} \quad \tau_k
\]

Objective is to find the minimum \( C_k \) such that

\[
C_k + \sum_{i=1}^{k-1} C_i + \sum_{i=0}^{j-1} C_i > T_j, \forall j = 1, 2, \ldots, k - 1,
\]

\[
C_k + 2 \sum_{i=1}^{k-1} C_i > T_k.
\]
Revisit (Liu and Layland-Bound) Structure

- $C_i = T_i U_i$
- The non-schedulability of $\tau_k$ implies such a structure if $2T_1 \geq T_k$:

\[
\tau_1 \uparrow \quad \tau_2 \uparrow \quad \tau_3 \uparrow \quad \tau_{k-1} \uparrow \quad \tau_k \uparrow
\]

Objective is to find the minimum $C_k$ such that

\[
C_k + \sum_{i=1}^{k-1} T_i U_i + \sum_{i=0}^{j-1} T_i U_i > T_j, \quad \forall j = 1, 2, \ldots, k - 1,
\]

\[
C_k + 2 \sum_{i=1}^{k-1} T_i U_i > T_k.
\]
Let us replace $T_i > 0$ with $t_i > 0$ (as a variable)

**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 $0 \leq t_1 \leq t_2 \leq \cdots \leq t_{k-1} \leq t_k$
- such that
  \[
  C_k + \sum_{i=1}^{k-1} t_i U_i + \sum_{i=1}^{j-1} t_i U_i \leq t_j.
  \]
k-Point Effective Schedulability Test

Let us replace $T_i > 0$ with $t_i > 0$ (as a variable)

**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 $0 \leq t_1 \leq t_2 \leq \cdots \leq t_{k-1} \leq t_k$
- such that

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

In the above Liu&Layland task model as an example:

- $\alpha_i = 1, \beta_i = 1, \forall i = 1, 2, \ldots, k - 1$
- $t_i = T_i, \forall i = 1, 2, \ldots, k$
Key Result

Suppose 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 $i = 1, 2, \ldots, k - 1$, $0 < t_k$.
Key Result

Suppose 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 $i = 1, 2, \ldots, k-1$, $0 < t_k$.

Lemma

Lemma 1 Task $\tau_k$ is schedulable by the scheduling algorithm if the following condition holds

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

Suppose 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 $i = 1, 2, \ldots, k - 1$, $0 < t_k$.

Lemma

Lemma 1 Task $\tau_k$ is schedulable by the scheduling algorithm if the following condition holds

$$
\left( \frac{C_k}{t_k} + \frac{\alpha}{\beta} \right)^{k-1} \prod_{j=1}^{k-1} (\beta U_j + 1) \leq \frac{\alpha}{\beta} + 1
$$
Key Result

Suppose 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 $i = 1, 2, \ldots, k - 1$, $0 < t_k$.

Lemma

Lemma 1 Task $\tau_k$ is schedulable by the scheduling algorithm if the following condition holds

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

In the above Liu&Layland task model as an example:

- $\alpha = 1$, $\beta = 1$
- $t_k = T_k$
- Hyperbolic Bound: $\prod_{j=1}^{k} (U_j + 1) \leq 2$.
A Sketched Proof

The unschedulability implies that $C_k > C_k^*$, where $C_k^*$ is defined in the following optimization problem:

$$\inf C_k^*$$

such that

$$C_k^* + \sum_{i=1}^{k-1} \alpha t_i^* U_i + \sum_{i=1}^{j-1} \beta t_i^* U_i > t_j^*, \quad \forall j = 1, 2, \ldots, k$$

$$t_j^* \geq 0, \quad \forall j = 1, 2, \ldots, k,$$

where $t_1^*, t_2^*, \ldots, t_{k-1}^*$ and $C_k^*$ are variables, $\alpha, \beta$ are constants, and $t_k^*$ is defined as $t_k$. 
A Sketched Proof

The unschedulability implies that $C_k > C^*_k$, where $C^*_k$ is defined in the following optimization problem:

\[
\inf \ C^*_k \quad \text{such that} \quad C^*_k + \sum_{i=1}^{k-1} \alpha t_i^* U_i + \sum_{i=1}^{j-1} \beta t_i^* U_i > t_j^*, \quad \forall j = 1, 2, \ldots, k
\]

\[
t_j^* \geq 0, \quad \forall j = 1, 2, \ldots, k,
\]

where $t_1^*, t_2^*, \ldots, t_{k-1}^*$ and $C^*_k$ are variables, $\alpha, \beta$ are constants, and $t_k^*$ is defined as $t_k$.

The above linear programming gives the minimum $C^*_k$ to be unschedulable. Therefore, if $C_k \leq C^*_k$, task $\tau_k$ is guaranteed to be schedulable.
Framework (First Part)

Demonstrated Applications:
- Sec. 5.1: Constrained-deadline sporadic tasks
- Sec. 5.2: Arbitrary-deadline sporadic tasks
- App. C: Multiframe tasks
- Sec. 6.1: Multiprocessor DAG
- Sec. 6.2: Multiprocessor self-suspension

Derive $k$-point schedulability parameters

$U_i, \forall i < k$
$\alpha_i, \forall i < k$
$\beta_i, \forall i < k$
$C_k, t_k$
Framework (First Part)

**Demonstrated Applications:**
- Sec. 5.1: Constrained-deadline sporadic tasks
- Sec. 5.2: Arbitrary-deadline sporadic tasks
- App. C: Multiframe tasks
- Sec. 6.1: Multiprocessor DAG
- Sec. 6.2: Multiprocessor self-suspension

Define $t_i, \forall i < k$ and order $k-1$ tasks

Derive k-point schedulability parameters

$U_i, \forall i < k$
$\alpha_i, \forall i < k$
$\beta_i, \forall i < k$
$C_k, t_k$

$k^2U$ framework

Hyperbolic bound

Lemma 1
Framework (First Part)

**Demonstrated Applications:**
- Sec. 5.1: Constrained-deadline sporadic tasks
- Sec. 5.2: Arbitrary-deadline sporadic tasks
- App. C: Multiframe tasks
- Sec. 6.1: Multiprocessor DAG
- Sec. 6.2: Multiprocessor self-suspension

Derive $t_i, \forall i < k$ and order $k-1$ tasks

$U_i, \forall i < k$
$\alpha_i, \forall i < k$
$\beta_i, \forall i < k$
$C_k, t_k$

$k^2U$ framework

Hyperbolic bound

Lemma 1

Other utilization bounds

Lemmas 2&3

Extreme points test

Lemma 4
Let $t_i$ be $\left\lfloor \frac{T_k}{T_i} \right\rfloor T_i$. Therefore, we have $\alpha_i = 1$ and $\beta_i \leq 1$.

**Theorem**

[Bini and Buttazzo, ECRTS 2001] Task $\tau_k$ is schedulable by RM on a uniprocessor system if

$$\prod_{i=1}^{k} (U_i + 1) \leq 2.$$
Direct Implications

From the schedulability condition \( \left( \frac{C_k}{t_k} + \frac{\alpha}{\beta} \right) \prod_{j=1}^{k-1} (\beta U_j + 1) \leq \frac{\alpha}{\beta} + 1 \)

Lemma

(Lemma 2) Task \( \tau_k \) is schedulable by the scheduling algorithm if

\[
\frac{C_k}{t_k} + \sum_{i=1}^{k-1} U_i \leq \begin{cases} 
1, & (\alpha + \beta) \frac{1}{k} < 1 \\
(k - 1) \left( \frac{\left(1 + \frac{\beta}{\alpha}\right) \frac{1}{k-1} - 1}{\beta} \right), & (\alpha + \beta) \frac{1}{k} < \alpha \\
(k-1)((\alpha+\beta) \frac{1}{k} - 1) + ((\alpha+\beta) \frac{1}{k} - \alpha) & \text{otherwise.}
\end{cases}
\]

Lemma

(Lemma 3) Task \( \tau_k \) is schedulable by the scheduling algorithm if

\[
\beta \sum_{i=1}^{k-1} U_i \leq \ln \left( \frac{\alpha}{\beta} + 1 \right). 
\]
k-Point Effective Schedulability Test: $k^2Q$

**Definition**

[Last Release Time Ordering] Let $\pi$ be the last release time ordering assignment as a bijective function $\pi : hp(\tau_k) \rightarrow \{1, 2, \ldots, k - 1\}$ to define the last release time ordering of task $\tau_j \in hp(\tau_k)$ in the window of interest.

**Definition**

A $k$-last-release effective schedulability test under a given ordering $\pi$ of the $k - 1$ higher priority tasks is a sufficient schedulability test of a fixed-priority scheduling policy by verifying the existence of $t_1 \leq t_2 \leq \cdots \leq t_k - 1 \leq t_k$ such that

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

(1)

where $C_k > 0$, for $i = 1, 2, \ldots, k - 1$, $\alpha_i > 0$, $U_i > 0$, $C_i \geq 0$, and $\beta_i > 0$ are dependent upon the setting of the task models and task $\tau_i$. 
Key Lemma

Lemma

[Lemma 5] For a given $k$-point last-release schedulability test of a scheduling algorithm, in which $0 < \alpha_i$, and $0 < \beta_i$ for any $i = 1, 2, \ldots, k - 1$, $0 < t_k$, $\sum_{i=1}^{k-1} \alpha_i U_i \leq 1$, and $\sum_{i=1}^{k-1} \beta_i C_i \leq t_k$, task $\tau_k$ is schedulable by the fixed-priority scheduling algorithm if the following condition holds

$$
\frac{C_k}{t_k} \leq 1 - \sum_{i=1}^{k-1} \alpha_i U_i - \frac{\sum_{i=1}^{k-1} (\beta_i C_i - \alpha_i U_i (\sum_{\ell=i}^{k-1} \beta_\ell C_\ell))}{t_k}.
$$
[Lemma 5] For a given k-point last-release schedulability test of a scheduling algorithm, in which $0 < \alpha_i$, and $0 < \beta_i$ for any $i = 1, 2, \ldots, k - 1$, $0 < t_k$, $\sum_{i=1}^{k-1} \alpha_i U_i \leq 1$, and $\sum_{i=1}^{k-1} \beta_i C_i \leq t_k$, task $\tau_k$ is schedulable by the fixed-priority scheduling algorithm if the following condition holds

\[
\frac{C_k}{t_k} \leq 1 - \sum_{i=1}^{k-1} \alpha_i U_i - \frac{\sum_{i=1}^{k-1} (\beta_i C_i - \alpha_i U_i (\sum_{\ell=i}^{k-1} \beta_\ell C_\ell))}{t_k}.
\] (2)

The worst-case ordering $\pi$ of the $k - 1$ higher-priority tasks is to order the tasks in a non-increasing order of $\frac{\beta_i C_i}{\alpha_i U_i}$. 
A Sketched Proof

The unschedulability implies that $C_k > C_k^*$, where $C_k^*$ is defined in the following optimization problem:

\[
\inf\limits_{C_k^*} \quad \text{such that} \quad C_k^* + \sum_{i=1}^{k-1} \alpha_i t_i^* U_i + \sum_{i=1}^{j-1} \beta_i C_i > t_j^*, \quad \forall j = 1, 2, \ldots, k
\]
\[
t_j^* \geq 0, \quad \forall j = 1, 2, \ldots, k,
\]

where $t_1^*, t_2^*, \ldots, t_{k-1}^*$ and $C_k^*$ are variables, $\alpha_i, \beta_i$ are constants, and $t_k^*$ is defined as $t_k$. 
A Sketched Proof

The unschedulability implies that $C_k > C_k^*$, where $C_k^*$ is defined in the following optimization problem:

$$\inf C_k^*$$

such that

$$C_k^* + \sum_{i=1}^{k-1} \alpha_i t_i^* U_i + \sum_{i=1}^{j-1} \beta_i C_i > t_j^*, \quad \forall j = 1, 2, \ldots, k$$

$$t_j^* \geq 0, \quad \forall j = 1, 2, \ldots, k,$$

where $t_1^*, t_2^*, \ldots, t_{k-1}^*$ and $C_k^*$ are variables, $\alpha_i, \beta_i$ are constants, and $t_k^*$ is defined as $t_k$.

The above linear programming gives the minimum $C_k^*$ to be unschedulable. Therefore, if $C_k \leq C_k^*$, task $\tau_k$ is guaranteed to be schedulable.
Quadratic Bound for Sporadic Task Systems

Let $t_i$ be $\left\lfloor \frac{T_k}{T_i} \right\rfloor T_i$. Therefore, $\alpha_i = 1$ and $\beta_i \leq 1$.

**Theorem**

Task $\tau_k$ is schedulable by RM on a uniprocessor system if

$$0 \leq 1 - U_k - 2 \sum_{i=1}^{k-1} U_i + 0.5 \left( \sum_{i=1}^{k-1} U_i^2 \right) \right).$$
Quadratic Bound for Sporadic Task Systems

Let $t_i$ be $\left\lfloor \frac{T_k}{T_i} \right\rfloor T_i$. Therefore, $\alpha_i = 1$ and $\beta_i \leq 1$.

**Theorem**

Task $\tau_k$ is schedulable by RM on a uniprocessor system if

$$0 \leq 1 - U_k - 2 \sum_{i=1}^{k-1} U_i + 0.5 \left( \left( \sum_{i=1}^{k-1} U_i \right)^2 + \left( \sum_{i=1}^{k-1} U_i^2 \right) \right).$$
How to Use the Framework

- Parameters $\alpha_i$ and $\beta_i$ affect the quality of the schedulability bounds.
- Deriving the *good* settings of $\alpha_i$ and $\beta_i$ is not part of this framework.
  - The framework simply derives the bounds/tests according to $\alpha_i$ and $\beta_i$.
  - The correctness of $\alpha_i$ and $\beta_i$ is not verified by the framework.
- The hyperbolic/quadratic bounds or utilization bounds can be automatically derived.
  - The other approaches seek for the total utilization bounds.
  - They have limited applications and are less flexible.
- After $\alpha_i$ and $\beta_i$ or their safe upper bounds $\alpha$ and $\beta$ are derived, the task model is not further referred.
Comparisons

Adopting different tests from $k^2 U$ and $k^2 Q$ for RM uniprocessor scheduling with $k = 2$ and $U_1 = 0.3$. 

![Diagram]

- Lemma 1 - $k^2 U$
- Lemma 5 - $k^2 Q$
- Lemma 7 - $k^2 Q$
When to Use Which?

- $k^2U$ (Version 1 above)
  - define any valid $k$ points to obtain the corresponding $\alpha_i$ and $\beta_i$
  - more precise if the corresponding exponential-time (pseudo-polynomial-time) test is an exact test
  - may be less precise if the corresponding test requires some pessimism to be constructed, to be shown later

- $k^2Q$ (Version 2 above)
  - define $k$ last release points to obtain the corresponding $\alpha_i$ and $\beta_i$
  - has to typically consider the last release ordering
  - less precise if the corresponding exponential-time (pseudo-polynomial-time) test is an exact test
  - may be more precise by starting from the exponential-time test, to be shown later
  - can be generalized for response-time analysis
Outline

Introduction

Utilization-Based Analytical Framework

Selected Applications

Conclusions
Deadline-Monotonic Scheduling

- let $hp(\tau_k)$ be the set of tasks with higher priority than $\tau_k$.
  - $hp_1(\tau_k)$ consists of the higher-priority tasks $\tau_i$ with $T_i < D_k$.
  - $hp_2(\tau_k)$ consists of the higher-priority tasks $\tau_i$ with $T_i \geq D_k$.
- The schedulability test is equivalent to the verification of

$$\exists 0 < t \leq D_k \quad C_k + \sum_{\tau_i \in hp_2(\tau_k)} C_i + \sum_{\tau_i \in hp_1(\tau_k)} \left\lceil \frac{t}{T_i} \right\rceil C_i \leq t.$$ 

$$\Rightarrow \exists 0 < t \leq D_k \quad C'_k + \sum_{\tau_i \in hp_1(\tau_k)} \left\lceil \frac{t}{T_i} \right\rceil C_i \leq t.$$ 

- Apply $k^2U$ or $k^2Q$ to get the utilization-based schedulability tests, by setting $\alpha_i = 1$ and $0 < \beta_i \leq 1$. 
Deadline-Monotonic Scheduling (cont.)

The schedulability condition of task $\tau_k$ by using $k^2 U$ is

$$\left(\frac{C'_k}{D_k} + 1\right) \prod_{\tau_j \in hP_1(\tau_k)} (U_j + 1) \leq 2.$$ 

The schedulability condition of task $\tau_k$ by using $k^2 Q$ is

$$\frac{C'_k}{D_k} \leq 1 - 2 \sum_{i=1}^{k-1} U_i + 0.5 \left(\sum_{i=1}^{k-1} U_i^2\right) + \left(\sum_{i=1}^{k-1} U_i\right)^2.$$ 

- It can be proved that the speed-up factor of DM is 1.76322, compared to EDF.
Uniprocessor Self-Suspending Task Systems

For all $0 < t \leq T_k$

$$W_k(t) = \sum_{i=1}^{k-1} \left( \left\lfloor \frac{t}{T_i} \right\rfloor - 1 \right) C_i + 2C_i.$$ 

Schedulability test for task $\tau_k$:

$$\exists t \text{ with } 0 < t \leq T_k \text{ and } C_k + S_k + W_k(t) \leq t.$$
Uniprocessor Self-Suspending Task Systems

For all $0 < t \leq T_k$

$$W_k(t) = \sum_{i=1}^{k-1} \left( \left\lceil \frac{t}{T_i} \right\rceil - 1 \right) C_i + 2C_i.$$  

Schedulability test for task $\tau_k$:

$$\exists t \text{ with } 0 < t \leq T_k \text{ and } C_k + S_k + W_k(t) \leq t.$$ 

bursty $= 2C_i$

Prof. Dr. Jian-Jia Chen (TU Dortmund)
Uniprocessor Self-Suspending Task Systems

For all $0 < t \leq T_k$

$$W_k(t) = \sum_{i=1}^{k-1} \left( \left\lfloor \frac{t}{T_i} \right\rfloor - 1 \right) C_i + 2C_i.$$ 

Schedulability test for task $\tau_k$:

$$\exists t \text{ with } 0 < t \leq T_k \text{ and } C_k + S_k + W_k(t) \leq t.$$
Uniprocessor Self-Suspending: k2U Framework

- Let \( t_i \) be \( \left\lfloor \frac{T_k}{T_i} \right\rfloor T_i \)
- When testing task \( \tau_k \),
  - \( \alpha_i \leq \alpha = 2 \) and \( \beta_i \leq \beta = 1 \) for \( i=1,2,\ldots,k-1 \)
- By using \( k^2U \) framework, \( \tau_k \) is schedulable by RM scheduling if

\[
\left( \frac{C_k + S_k}{T_k} + 2 \right) \prod_{j=1}^{k-1} (U_j + 1) \leq 3.
\]

In the key lemma:

\[
\left( \frac{C_k}{t_k} + \frac{\alpha}{\beta} \right) \prod_{j=1}^{k-1} (\beta U_j + 1) \leq \frac{\alpha}{\beta} + 1
\]
Utilization Bounds

Let $t_i = \left\lfloor \frac{T_i}{T_k} \right\rfloor T_i$. Therefore, we have $\alpha_i \leq 2$ and $\beta_i \leq 1$.

Theorem (Liu and Chen in RTSS 2014)

Any implicit-deadline sporadic self-suspending task set is schedulable under RM if the following conditions hold:

$$\forall 1 \leq k \leq n, \quad U_k + \frac{S_k}{T_k} \leq 1 - (2 + 1) \cdot \left(1 - \frac{1}{\prod_{i=1}^{k-1} (U_i + 1)}\right). \quad (3)$$

Theorem (Liu and Chen in RTSS 2014)

Any implicit-deadline sporadic self-suspending task set is schedulable under RM if the following conditions hold:

$$\forall 1 \leq k \leq n, \quad U_k + \frac{S_k}{T_k} + \sum_{i=1}^{k-1} U_i \leq k \left(\left(\frac{2 + 1}{2}\right) \frac{1}{k} - 1\right). \quad (4)$$
Uniprocessor Non-Preemptive (NP) Scheduling

Let $\hat{C}_k = C_k + B_k + \sum_{\tau_i \in hp_2(\tau_k)} C_i$, where $B_k$ is $\{\max_{\tau_i \in lp(\tau_k)} C_i\}$. The schedulability condition of task $\tau_k$ by using $k^2U$ is

$$\left(\frac{\hat{C}_k}{D_k} + 1\right) \prod_{\tau_j \in hp_1(\tau_k)} (U_j + 1) \leq 2 \quad (5)$$

Theorem

[Theorem 4 in von der Brüeggen, Chen, and Huang, 2015]

Suppose that $\gamma = \max_{\tau_k} \left\{\max_{\tau_i \in lp(\tau_k)} \left\{\frac{C_i}{C_k}\right\}\right\}$. A task set can be feasibly scheduled by RM-NP if

$$U_{\text{sum}} \leq \begin{cases} \frac{\gamma}{1+\gamma} + \ln \left(\frac{2}{1+\gamma}\right) & \text{if } \gamma \leq 1 \\ \frac{1}{1+\gamma} & \text{if } \gamma > 1 \end{cases}$$

(leave as an exercise)
Real-Time Systems with Mode Changes

- Real-time tasks run in different modes over time to react to the change of physical environments
  - Avionic systems
  - Automotive systems

<table>
<thead>
<tr>
<th>rotation (rpm)</th>
<th>functions to be executed</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0, 2000]</td>
<td>$f_1(); f_2(); f_3(); f_4();$</td>
</tr>
<tr>
<td>(2000, 4000]</td>
<td>$f_1(); f_2(); f_3();$</td>
</tr>
<tr>
<td>(4000, 6000]</td>
<td>$f_1(); f_2();$</td>
</tr>
<tr>
<td>(6000, 8000]</td>
<td>$f_1();$</td>
</tr>
</tbody>
</table>
Multi-Mode Task Model

- A multi-mode task $\tau_i$ is denoted by a set of triplets:

$$
\tau_i = \{ \tau_i^1 = (C_i^1, T_i^1, D_i^1), \\
\tau_i^2 = (C_i^2, T_i^2, D_i^2), \ldots, \\
\tau_i^{M_i} = (C_i^{M_i}, T_i^{M_i}, D_i^{M_i}) \}
$$

- $C_i^m$ denotes the worst-case execution time (WCET) of task $\tau_i$ under mode $m$
- $T_i^m$ denotes the minimum inter-arrival time of task $\tau_i$ under mode $m$
- $D_i^m$ denotes relative deadline (Constrained-deadline system ($D_i^m \leq T_i^m$))

- Fixed-priority scheduling
  - Fixed-priority task-level (FPT)
  - Fixed-priority mode-level (FPM)
Problems with a Naive Analysis

Deadline miss during mode transition under fixed-priority scheduling

- $\tau_1 = \{\tau_1^1 = (2, 3, 3), \tau_1^2 = (4, 8, 8)\}$
- $\tau_2 = (4, 12, 12)$
A Safe Exponential-Time Test

Theorem

Task mode $\tau^h_k$ is schedulable under an FPM scheduling if

$$\forall y \geq 0, \ \forall \text{combinations of } t^*_i \text{ with } 0 \leq t^*_i \leq t^*_i + 1, \forall i = 1, 2, \ldots, k - 1$$

$$\exists j = 1, 2, \ldots, k, \text{ s.t. } C^h_k + y \cdot U^\text{max}_k + \sum_{i=1}^{k-1} (U^\text{max}_i \cdot t^*_i) + \sum_{i=1}^{j-1} C^\text{max}_i \leq t^*_j.$$  

$C^\text{max}_i$ is the maximum execution time among the modes of task $\tau_i$ with priority higher than or equals to task mode $\tau^h_k$. $U^\text{max}_i$ is the maximum utilization among the modes of task $\tau_i$ with priority higher than or equals to task mode $\tau^h_k$. The constant $t^*_k$ is defined as $T^h_k + y$. 

Hint: the above test also requires to enumerate all possible orderings. Under $k^2Q$, it is possible to safely only test one specific ordering.
A Safe Exponential-Time Test

**Theorem**

Task mode $\tau_k^h$ is schedulable under an FPM scheduling if

$$\forall y \geq 0, \ \forall \text{combinations of } t_i^* \text{ with } 0 \leq t_i^* \leq t_{i+1}^*, \forall i = 1, 2, \ldots, k - 1$$

$$\exists j = 1, 2, \ldots, k, \text{ s.t. } C_k^h + y \cdot U_k^{\text{max}} + \sum_{i=1}^{k-1} (U_i^{\text{max}} \cdot t_i^*) + \sum_{i=1}^{j-1} C_i^{\text{max}} \leq t_j^*.$$  

$C_i^{\text{max}}$ is the maximum execution time among the modes of task $\tau_i$ with priority higher than or equals to task mode $\tau_k^h$. $U_i^{\text{max}}$ is the maximum utilization among the modes of task $\tau_i$ with priority higher than or equals to task mode $\tau_k^h$. The constant $t_i^*$ is defined as $T_k^h + y$.

**Hint:** the above test also requires to enumerate all possible orderings. Under $k^2 Q$, it is possible to safely only test one specific ordering.
Utilization Test under FPM-RM

For a given \( y \), we have \( C^h_k + y \cdot U^\text{max}_k \leq (T_k + y) \cdot U^\text{max}_k \). So, the remaining is a case with \( \alpha_1 = 1 \) and \( \beta_i = 1 \) in the \( k^2Q \) framework.

Theorem

Task \( \tau^h_k \) in a multi-mode task system with implicit deadlines is schedulable by the mode-level RM scheduling algorithm on a uniprocessor system if the following condition holds

\[
U^\text{max}_k \leq 1 - 2 \sum_{i=1}^{k-1} U^\text{max}_i + 0.5 \left( \left( \sum_{i=1}^{k-1} U^\text{max}_i \right)^2 + \left( \sum_{i=1}^{k-1} (U^\text{max}_i)^2 \right) \right), \quad (6)
\]

or

\[
\sum_{i=1}^{k-1} U^\text{max}_i \leq \left( \frac{k-1}{k} \right) \left( 2 - \sqrt{2 + 2U^\text{max}_k \frac{k}{k-1}} \right), \quad (7)
\]

or

\[
U^\text{max}_k + \sum_{i=1}^{k-1} U^\text{max}_i \leq \begin{cases} \left( \frac{k-1}{k} \right) \left( 2 - \sqrt{4 - \frac{2k}{k-1}} \right), & \text{if } k > 3 \\ 1 - \frac{(k-1)}{2k} & \text{otherwise.} \end{cases} \quad (8)
\]
Framework

Demonstrated Applications: Task Models

- Derive parameters by Definition 1:
  - $U_i, \forall i < k$
  - $\alpha_i, \forall i < k$
  - $\beta_i, \forall i < k$
  - $C_k, t_k$

- Define $t_i, \forall i < k$ and order $k-1$ tasks

- $k^2U$ framework

- Hyperbolic bound
- Other utilization bounds
- Extreme points test

Proofs:
- Lemma 1
- Lemmas 2&3
- Lemma 4
- Lemmas 7-10
- Lemma 11
- Lemma 12
Framework

Demonstrated Applications: Task Models

Derive parameters by Definition 1

$U_i, \forall i < k$
$\alpha_i, \forall i < k$
$\beta_i, \forall i < k$
$C_k, t_k$

$k^2U$ framework

Derive parameters by Definitions 3 or 4

$U_i, \forall i < k$
$C_i, \forall i < k$
$\alpha_i, \forall i < k$
$\beta_i, \forall i < k$
$C_k, t_k$

$k^2Q$ framework

Hyperbolic bound
Other utilization bounds
Extreme points test
Quadratic bound
Other utilization bounds
Response-time test

Lemma 1
Lemmas 2&3
Lemma 4
Lemma 5
Lemmas 7-10
Lemma 11

Define $t, \forall i < k$ and order $k - 1$ tasks
Define the last release time ordering $\pi$ or use Lemma 6 or 12

Prof. Dr. Jian-Jia Chen (TU Dortmund)
Outline

Introduction

Utilization-Based Analytical Framework

Selected Applications

Conclusions
Polynomial-time schedulability test framework

- Basically handles many problems very well with low time complexity
- The first evidence to translate *exponential-time schedulability tests* to (potentially) *linear-time tests* with test quality guarantees

\[ \exists 0 < t \leq D \text{ s.t. } C_k + k - 1 \sum_{i=1}^{\lfloor t/T_i \rfloor} C_i + bC_i \leq t. \]
Polynomial-time schedulability test framework

- Basically handles many problems very well with low time complexity
- The first evidence to translate *exponential-time schedulability tests* to (potentially) *linear-time tests* with test quality guarantees

- Can we derive the coefficients $\alpha_i$ and $\beta_i$ *automatically*?
  - Yes, for some well-studied forms. See our recent technical report. One commonly used class as an example:

\[
\exists 0 < t \leq D_k \text{ s.t. } C_k + \sum_{i=1}^{k-1} \sigma \left( \left\lceil \frac{t}{T_i} \right\rceil C_i + bC_i \right) \leq t.
\]

- No idea yet for arbitrary forms
Other Applications

- Multi-frame task model
- Digraph task models
- Uniprocessor/Multiprocessor scheduling with self-suspensions
- Multiprocessor global DM scheduling
- Multiprocessor partitioned RM/DM scheduling
- Multiprocessor scheduling with DAG structures
- etc.