Shared-Resource-Centric Limited Preemptive Scheduling: A Comprehensive Study of Suspension-based Partitioning Approaches

Zheng Dong  Cong Liu  Soroush Bateni
The University of Texas at Dallas, USA

Kuan-Hsun Chen  Jian-Jia Chen  Georg von der Bruggen  Junjie Shi
Technical University of Dortmund, Germany
Shared-Resource-Centric Limited Preemptive Scheduling

**Multi-Core Processor**

- Simpler and lower power than a single large core
- Smaller inter-core communication latency

- **AMD Barcelona** (4 cores)
- **Intel Core i7** (8 cores)
- **IBM Cell BE** (8+1 cores)
- **IBM POWER7** (8 cores)
- **Sun Niagara 2** (8 cores)
- **Nvidia Fermi** (448 cores)
- **Intel SCC** (48 cores)
- **Tilera TILE Gx** (100 cores)

Zheng Dong @ RTAS'18, Porto
An example of multi-phase task

Autonomous Driving Vehicle
An example of multi-phase task

Autonomous Driving Vehicle

100 cores on chip
An example of multi-phase task

Autonomous Driving Vehicle

GPU (shared resource)

100 cores on chip
An example of multi-phase task

Autonomous Driving Vehicle

- CPUs and GPU **communicate** with each other constantly through the PCI **bus**.

100 cores on chip

GPU (shared resource)
An example of multi-phase task

Autonomous Driving Vehicle

• CPUs and GPU **communicate** with each other constantly through the **PCI bus**.

Multi-phase task: CPU-phase  GPU-phase  CPU-phase  GPU-phase  CPU-phase
Traditional methods encounter multi-core system

Multi-phase task:

- **CPU-phase**
- **GPU-phase**
- **CPU-phase**
- **GPU-phase**
- **CPU-phase**

100 cores on chip

GPU (accelerator)
Traditional methods encounter multi-core system

Multi-phase task:

<table>
<thead>
<tr>
<th>CPU-phase</th>
<th>GPU-phase</th>
<th>CPU-phase</th>
<th>GPU-phase</th>
<th>CPU-phase</th>
</tr>
</thead>
</table>

100 cores on chip

GPU (accelerator)
Traditional methods encounter multi-core system

**Multi-phase task:**

1. CPU-phase
2. GPU-phase
3. CPU-phase
4. GPU-phase
5. CPU-phase

**Core-centric:**

1. CPU-phase
2. **Suspension**
3. CPU-phase
4. **Suspension**
5. CPU-phase

100 cores on chip

GPU (accelerator)

Zheng Dong @ RTAS’18, Porto
Traditional methods encounter multi-core system

Multi-phase task: CPU-phase | GPU-phase | CPU-phase | GPU-phase | CPU-phase

Core-centric: CPU-phase | **Suspension** | CPU-phase | **Suspension** | CPU-phase

The idea is to focus on **judiciously allocating CPU resources** while viewing shared resources as I/O (suspensions) and bounding the worst-case latency a task may experience on such resources.
Traditional methods encounter multi-core system

Multi-phase task:
- CPU-phase
- GPU-phase
- CPU-phase
- bounded

Core-centric:
- CPU-phase
- Suspension
- CPU-phase
- Suspension
- CPU-phase

The idea is to focus on **judiciously allocating CPU resources while viewing shared resources as I/O (suspensions)** and **bounding the worst-case latency a task may experience on such resources.**
Traditional methods encounter multi-core system

Multi-phase task:

- CPU-phase
- GPU-phase
- CPU-phase

Bounded

Core-centric:

- CPU-phase
- Suspension
- CPU-phase
- Suspension
- CPU-phase

The idea is to focus on judiciously allocating CPU resources while viewing shared resources as I/O (suspsensions) and bounding the worst-case latency a task may experience on such resources.


Traditional methods encounter multi-core system

Multi-phase task:

- CPU-phase
- GPU-phase
- CPU-phase
- GPU-phase
- CPU-phase

Core-centric:

- CPU-phase
- Suspension
- CPU-phase
- Suspension
- CPU-phase

Shared resource may become the actual scheduling bottleneck, causing the worst-case latency bound on shared-resource rather pessimistic or impossible to derive.


Shared-Resource-Centric Limited Preemptive Scheduling

Traditional methods encounter multi-core system

Multi-phase task:

- CPU-phase
- [unbounded] GPU-phase
- CPU-phase

Core-centric:

- CPU-phase
- Suspension
- CPU-phase
- Suspension
- CPU-phase

Shared resource may become the actual scheduling bottleneck, causing the **worst-case latency bound** on shared-resource rather pessimistic or **impossible to derive**.


100 cores on chip

Zheng Dong @ RTAS’18, Porto
Traditional methods encounter multi-core system

Multi-phase task:  
- CPU-phase
- GPU-phase
- CPU-phase

Core-centric:  
- CPU-phase  
- Suspension  
- CPU-phase  
- Suspension  
- CPU-phase

Shared resource may become the actual scheduling bottleneck, causing the worst-case latency bound on shared-resource rather pessimistic or impossible to derive.

---


---

Zheng Dong @ RTAS’18, Porto
A counter-intuitive shared-resource-centric perspective

Multi-phase task:

<table>
<thead>
<tr>
<th>CPU-phase</th>
<th>GPU-phase</th>
<th>CPU-phase</th>
<th>GPU-phase</th>
<th>CPU-phase</th>
</tr>
</thead>
</table>

100 cores on chip

GPU (accelerator)
A counter-intuitive shared-resource-centric perspective

Multi-phase task:

| CPU-phase | GPU-phase | CPU-phase | GPU-phase | CPU-phase |

100 cores on chip

GPU (accelerator)

100 cores on chip
A counter-intuitive shared-resource-centric perspective

Multi-phase task:

Shared-resource-centric:

100 cores on chip
A counter-intuitive shared-resource-centric perspective

Multi-phase task:  

<table>
<thead>
<tr>
<th>CPU-phase</th>
<th>GPU-phase</th>
<th>CPU-phase</th>
<th>GPU-phase</th>
<th>CPU-phase</th>
</tr>
</thead>
</table>

Shared-resource-centric:  

Suspension  

The idea is to judiciously scheduling shared-resource requests (treating shared-resource as the first class unit) and bounding the worst case latency a task may experience on the computing cores (treating cores as “I/O”).

100 cores on chip
A counter-intuitive shared-resource-centric perspective

Multi-phase task:

```
CPU-phase   GPU-phase   CPU-phase   GPU-phase   CPU-phase
```

Shared-resource-centric:

```
Suspension   GPU-phase   Suspension   GPU-phase   Suspension
```

The idea is to *judiciously scheduling shared-resource requests* (treating shared-resource as the first class unit) and *bounding the worst case latency* a task may experience on the computing cores (treating cores as “I/O”).
A counter-intuitive shared-resource-centric perspective

Multi-phase task:  

<table>
<thead>
<tr>
<th>CPU-phase</th>
<th>GPU-phase</th>
<th>CPU-phase</th>
<th>GPU-phase</th>
<th>CPU-phase</th>
</tr>
</thead>
</table>

Shared-resource-centric:  

- Suspension  
- GPU-phase  
- Suspension  
- GPU-phase  
- Suspension

The idea is to judiciously scheduling shared-resource requests (treating shared-resource as the first class unit) and bounding the worst case latency a task may experience on the computing cores (treating cores as “I/O”).


A counter-intuitive shared-resource-centric perspective

Multi-phase task:

<table>
<thead>
<tr>
<th>Phase</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU-phase</td>
<td></td>
</tr>
<tr>
<td>GPU-phase</td>
<td></td>
</tr>
<tr>
<td>CPU-phase</td>
<td></td>
</tr>
<tr>
<td>GPU-phase</td>
<td></td>
</tr>
<tr>
<td>CPU-phase</td>
<td></td>
</tr>
</tbody>
</table>

Shared-resource-centric: Suspension

The idea is to judiciously scheduling shared-resource requests (treating shared-resource as the first class unit) and bounding the worst case latency a task may experience on the computing cores (treating cores as “I/O”).

100 cores on chip

---


Challenges in shared-resource-centric scheduling

- shared-resource-centric suspending task model VS. Traditional suspending task model
Challenges in shared-resource-centric scheduling

- shared-resource-centric suspending task model VS. Traditional suspending task model
  - Limited preemptive scheduling VS. Fully preemptive scheduling


Challenges in shared-resource-centric scheduling

- **shared-resource-centric suspending task model** VS. Traditional suspending task model
  - Limited preemptive scheduling VS. Fully preemptive scheduling
  - Parallel Scheduling VS. Single thread scheduling

---


Challenges in shared-resource-centric scheduling

- shared-resource-centric suspending task model VS. Traditional suspending task model
  - Limited preemptive scheduling VS. Fully preemptive scheduling
  - Parallel Scheduling VS. Single thread scheduling

GPU with parallel programmable units at their cores.

Not Applicable


Challenges in shared-resource-centric scheduling

- **shared-resource-centric suspending task model** VS. Traditional suspending task model

  - Limited preemptive scheduling VS. Fully preemptive scheduling
  - **Parallel Scheduling** VS. Single thread scheduling

---


Challenges in shared-resource-centric scheduling

- shared-resource-centric suspending task model VS. Traditional suspending task model
  - Limited preemptive scheduling VS. Fully preemptive scheduling
  - Parallel Scheduling VS. Single thread scheduling

GPU with parallel programmable units at their cores.

---


Objectives and solution overview

❖ Our goals is two-fold:

- Schedulability tests for shared-resource-centric limited preemptive suspending task model.
- Minimizing the number of units of required shared resource.
Objectives and solution overview

❖ Our goals is two-fold:

- Schedulability tests for shared-resource-centric limited preemptive suspending task model.
- Minimizing the number of units of required shared resource.

❖ Solution overview (two steps):

- Step1: Developing schedulability tests that can guarantee HRT constraints
- Step2: Partitioning tasks onto a multi-unit shared resource to minimize the amount of required shared resource.
Shared-resource-centric suspending task model

- Implicit-deadline sporadic multi-phase task: $T_i(e_i, s_i, p_i, d_i, \sigma_i, z_i)$
Shared-resource-centric suspending task model

* Implicit-deadline sporadic multi-phase task: $T_i(e_i, s_i, p_i, d_i, \sigma_i, z_i)$
Shared-resource-centric suspending task model

- Implicit-deadline sporadic multi-phase task: $T_i(e_i, s_i, p_i, d_i, \sigma_i, z_i)$

Execution on GPU
Shared-resource-centric suspending task model

- Implicit-deadline sporadic multi-phase task: $T_i(e_i, s_i, p_i, d_i, \sigma_i, z_i)$
Shared-resource-centric suspending task model

- Implicit-deadline sporadic multi-phase task: $T_i(e_i, s_i, p_i, d_i, \sigma_i, z_i)$

Execution on GPU
Shared-resource-centric suspending task model

- Implicit-deadline sporadic multi-phase task: \( T_i(e_i, s_i, p_i, d_i, \sigma_i, z_i) \)

Execution on GPU

Suspensions from shared-resource-centric perspective

Diagram:
- GPU timeline from 0 to 66.6 ms
- Phases labeled with \( z_i \)
- Execution time labeled with 33.3 ms
- Preemption point labeled with \( p_i \)
Shared-resource-centric suspending task model

- Implicit-deadline sporadic multi-phase task: $T_i(e_i, s_i, p_i, d_i, \sigma_i, z_i)$

![Diagram showing execution on GPU and suspensions from shared-resource-centric perspective](image-url)
Shared-resource-centric suspending task model

- Implicit-deadline sporadic multi-phase task: $T_i(e_i, s_i, p_i, d_i, \sigma_i, z_i)$

Each operation on the shared resource is atomic, with a processing time upper-bounded by $B$.
Shared-resource-centric suspending task model

- Implicit-deadline sporadic multi-phase task: \( T_i(e_i, s_i, p_i, d_i, \sigma_i, z_i) \)

Execution on GPU

Suspensions from shared-resource-centric perspective

Each operation on the shared resource is atomic, with a processing time upper-bounded by \( B \)

Rate-monotonic (RM) Scheduling on the shared resource.
Step 1: Developing schedulability tests for Limited preemptive suspending task scheduling
Step 1: Developing schedulability tests for Limited preemptive suspending task scheduling

Non-preemptive blocking from a lower priority job
Step 1: Developing schedulability tests for Limited preemptive suspending task scheduling

Observation

The maximum blocking time experienced by any job released by \( T_i \) due to non-preemptive blocking on shared resource is at most \( \sigma_i \) times \( B \).
Step 1: Developing schedulability tests for Limited preemptive suspending task scheduling

- Schedulability tests for preemptive task scheduling
Step 1: Developing schedulability tests for Limited preemptive suspending task scheduling

❖ Schedulability tests for preemptive task scheduling

Time-demand Analysis (TDA)

Lemma 1 (Suspension as Carry-In).

$$\exists t \mid 0 < t \leq p_k, \quad e_k + s_k + \sum_{\tau_i \in h_p(k)} \left( \left\lfloor \frac{t}{p_i} \right\rfloor + 1 \right) s_i \leq t$$

Lemma 2 (Suspension as Jitter).

$$\exists t \mid 0 < t \leq p_k, \quad e_k + s_k + \sum_{\tau_i \in h_p(k)} \left\lfloor \frac{t + p_i - s_i}{p_i} \right\rfloor s_i \leq t$$

Step 1: Developing schedulability tests for Limited preemptive suspending task scheduling

- Schedulability tests for limited-preemptive task scheduling

**Time-demand Analysis (TDA)**

**Lemma 1 (Suspension as Carry-In).**

$$\exists t | 0 < t \leq p_k, \quad c_k + s_k + \sum_{\tau_i \in hp(k)} \left( \left\lceil \frac{t}{p_i} \right\rceil + 1 \right) s_i \leq t$$

**Lemma 2 (Suspension as Jitter).**

$$\exists t | 0 < t \leq p_k, \quad c_k + s_k + \sum_{\tau_i \in hp(k)} \left\lceil \frac{t + p_i - s_i}{p_i} \right\rceil s_i \leq t$$

Step 1: Developing schedulability tests for Limited preemptive suspending task scheduling

- Schedulability tests for limited-preemptive task scheduling

**Time-demand Analysis (TDA)**

**Lemma 1 (Suspension as Carry-In).**

\[ \exists t \mid 0 < t \leq p_k, \quad c_k + s_k + \sum_{\tau_i \in h_p(k)} \left( \left\lfloor \frac{t}{p_i} \right\rfloor + 1 \right) s_i \leq t \]

**Lemma 2 (Suspension as Jitter).**

\[ \exists t \mid 0 < t \leq p_k, \quad c_k + s_k + \sum_{\tau_i \in h_p(k)} \left\lfloor \frac{t + p_i - s_i}{p_i} \right\rfloor s_i \leq t \]

**Observation**

The maximum blocking time experienced by any job released by \( \tau_i \) is due to non-preemptive blocking on shared resource is at most \( \sigma_i \) times \( B \).

Step 1: Developing schedulability tests for Limited preemptive suspending task scheduling

- Schedulability tests for limited-preemptive task scheduling

**Time-demand Analysis (TDA)**

**Lemma 1 (Suspension as Carry-In).**

\[ \exists t \mid 0 < t \leq p_k, \quad e_k + s_k + \sum_{\tau_i \in hp(k)} \left( \left\lfloor \frac{t}{p_i} \right\rfloor + 1 \right) s_i \leq t \]

**Lemma 2 (Suspension as Jitter).**

\[ \exists t \mid 0 < t \leq p_k, \quad e_k + s_k + \sum_{\tau_i \in hp(k)} \left( \frac{t + p_i - s_i}{p_i} \right) s_i \leq t \]

**Time-demand Analysis (TDA)**

**Lemma 3 (Suspension and Blocking as Carry-In).**

\[ \exists t \mid 0 < t \leq p_k, \quad s_k + e_k + \sigma_k \times B + \sum_{\tau_i \in hp(k)} \left( \left\lfloor \frac{t}{p_i} \right\rfloor + 1 \right) s_i \leq t \]

**Lemma 4 (Suspension and Blocking as Jitter).**

\[ \exists t \mid 0 < t \leq p_k, \quad s_k + e_k + \sigma_k \times B + \sum_{\tau_i \in hp(k)} \left( \frac{t + p_i - s_i}{p_i} \right) s_i \leq t \]

**Observation**

The maximum blocking time experienced by any job released by \( \tau_i \) is due to non-preemptive blocking on shared resource is at most \( \sigma_i \) times \( B \).

---

Step 1: Developing schedulability tests for Limited preemptive suspending task scheduling

- Schedulability tests for **limited-preemptive** task scheduling

---

**Time-demand Analysis (TDA)**

**Lemma 1 (Suspension as Carry-In).**

\[ \exists t \mid 0 < t \leq p_k, \quad c_k + s_k + \sum_{\tau_i \in h p (k)} \left( \left\lceil \frac{t}{p_i} \right\rceil + 1 \right) s_i \leq t \]

**Lemma 2 (Suspension as Jitter).**

\[ \exists t \mid 0 < t \leq p_k, \quad c_k + s_k + \sum_{\tau_i \in h p (k)} \left( \frac{t + p_i - s_i}{p_i} \right) s_i \leq t \]

---

**Observation**

The maximum blocking time experienced by any job released by \( T_i \) is due to non-preemptive blocking on shared resource is at most \( \sigma_i \) times \( B \).

---


Zheng Dong @ RTAS’18, Porto
Step 1: Developing schedulability tests for Limited preemptive suspending task scheduling

- Schedulability tests for limited-preemptive task scheduling
Step 1: Developing schedulability tests for Limited preemptive suspending task scheduling

- Schedulability tests for limited-preemptive task scheduling

**Time-demand Analysis (TDA)**

Lemma 3 (Suspension and Blocking as Carry-In).

$$ \exists t \mid 0 < t \leq p_k, \ s_k + e_k + \sigma_k \times B + \sum_{\tau_i \in h.p(k)} \left( \left\lceil \frac{t}{p_i} \right\rceil + 1 \right) s_i \leq t $$

Lemma 4 (Suspension and Blocking as Jitter).

$$ \exists t \mid 0 < t \leq p_k, \ s_k + e_k + \sigma_k \times B + \sum_{\tau_i \in h.p(k)} \left\lceil \frac{t + p_i - s_i}{p_i} \right\rceil s_i \leq t $$
Step 1: Developing schedulability tests for Limited preemptive suspending task scheduling

- Schedulability tests for limited-preemptive task scheduling

**Time-demand Analysis (TDA)**

Lemma 3 (Suspension and Blocking as Carry-In).

\[ \exists t \mid 0 < t \leq p_k, \ s_k + e_k + \sigma_k \times B + \sum_{\tau_i \in \bar{p}(k)} \left( \left\lceil \frac{t}{p_i} \right\rceil + 1 \right) s_i \leq t \]

Lemma 4 (Suspension and Blocking as Jitter).

\[ \exists t \mid 0 < t \leq p_k, \ s_k + e_k + \sigma_k \times B + \sum_{\tau_i \in \bar{p}(k)} \left\lceil \frac{t + p_i - s_i}{p_i} \right\rceil s_i \leq t \]

- Higher complexity
- Higher accuracy
**Step 1: Developing schedulability tests for Limited preemptive suspending task scheduling**

- Schedulability tests for limited-preemptive task scheduling

**Time-demand Analysis (TDA)**

Lemma 3 (Suspension and Blocking as Carry-In).

\[ \forall t \mid 0 < t \leq p_k, \ s_k + e_k + \sigma_k \times B + \sum_{\tau_i \in h_p(k)} \left( \left\lfloor \frac{t}{p_i} \right\rfloor + 1 \right) s_i \leq t \]  

(3)

Lemma 4 (Suspension and Blocking as Jitter).

\[ \forall t \mid 0 < t \leq p_k, \ s_k + e_k + \sigma_k \times B + \sum_{\tau_i \in h_p(k)} \left( \frac{t + p_i - s_i}{p_i} \right) s_i \leq t \]

- **Time-demand Analysis (TDA)**
  - Higher complexity
  - Higher accuracy

**Constant-time Tests (CT)**

Lemma 5 (Suspension and Blocking as Carry-In).

\[ \left( \frac{s_k + e_k + \sigma_k \times B}{p_k} + 2 \right) \prod_{\tau_i \in h_p(k)} \left( \frac{s_i}{p_i} + 1 \right) \leq 3 \]  

(5a)

\[ \sum_{\tau_i \in h_p(k)} \frac{s_i}{p_i} \leq \ln \left( \frac{3}{s_k + e_k + \sigma_k \times B + 2} \right) \]  

(5b)

Lemma 6 (Suspension and Blocking as Jitter).

\[ \frac{e_k + s_k + \sigma_k \times B}{p_k} + \sum_{\tau_i \in h_p(k)} \left( 2s_i - \frac{s_i^2}{p_i} \right) + \sum_{\tau_i \in h_p(k)} \frac{s_i}{p_i} \leq 1 \]  

(6)
Step 1: Developing schedulability tests for Limited preemptive suspending task scheduling

- Schedulability tests for limited-preemptive task scheduling

**Time-demand Analysis (TDA)**

Lemma 3 (Suspension and Blocking as Carry-In).

\[ \exists t \mid 0 < t \leq p_k, \quad s_k + e_k + \sigma_k \times B + \sum_{\tau_i \in hp(k)} \left( \left\lfloor \frac{t}{p_i} \right\rfloor + 1 \right) s_i \leq t \]

Lemma 4 (Suspension and Blocking as Jitter).

\[ \exists t \mid 0 < t \leq p_k, \quad s_k + e_k + \sigma_k \times B + \sum_{\tau_i \in hp(k)} \left( \frac{t + p_i - s_i}{p_i} \right) s_i \leq t \]

**Constant-time Tests (CT)**

Lemma 5 (Suspension and Blocking as Carry-In).

\[ \left( \frac{s_k + e_k + \sigma_k \times B}{p_k} + 2 \right) \prod_{\tau_i \in hp(k)} \left( \frac{s_i}{p_i} + 1 \right) \leq 3 \]  

Lemma 6 (Suspension and Blocking as Jitter).

\[ \frac{e_k + s_k + \sigma_k \times B}{p_k} + \sum_{\tau_i \in hp(k)} \left( \frac{2s_i - \frac{s_i^2}{p_i}}{p_i} \right) + \sum_{\tau_i \in hp(k)} \frac{s_i}{p_i} \leq 1 \]

- Higher complexity
- Higher accuracy

- Lower complexity
- Lower accuracy

Zheng Dong @ RTAS’18, Porto
Step 2: Partitioning tasks onto a multi-unit shared resource
Step 2: Partitioning tasks onto a multi-unit shared resource
Step 2: Partitioning tasks onto a multi-unit shared resource
Step 2: Partitioning tasks onto a multi-unit shared resource
Step 2: Partitioning tasks onto a multi-unit shared resource
Step 2: Partitioning tasks onto a multi-unit shared resource
Step 2: Partitioning tasks onto a multi-unit shared resource
Step 2: Partitioning tasks onto a multi-unit shared resource
Step 2: Partitioning tasks onto a multi-unit shared resource
Step 2: Partitioning tasks onto a multi-unit shared resource
Step 2: Partitioning tasks onto a multi-unit shared resource
Step 2: Partitioning tasks onto a multi-unit shared resource
Step 2: Partitioning tasks onto a multi-unit shared resource
Step 2: Partitioning tasks onto a multi-unit shared resource
Step 2: Partitioning tasks onto a multi-unit shared resource
Step 2: Partitioning tasks onto a multi-unit shared resource

Zheng Dong @ RTAS'18, Porto
Step 2: Partitioning tasks onto a multi-unit shared resource

Observation

If we can schedule tasks with similar parallelism together on one resource partition, the negative impact due to multiple tasks’ parallelism can be “masked” by only one task that has the maximum parallelism.
Step 2: Partitioning tasks onto a multi-unit shared resource
Step 2: Partitioning tasks onto a multi-unit shared resource

- Minimize the amount of required shared resource.
  - Developing a comprehensive set of partitioning algorithms combined with various schedulability tests.
  - Theoretically evaluate the effectivity of the proposed partitioning algorithms through upper bounding the system utilization loss.
  - Further improve the partitioning methods using an ILP-based approach.
Step 2: Partitioning tasks onto a multi-unit shared resource

Sort the tasks according to $z_i$ values ($z_i \geq z_{i+1}$); ties are broken by preferring smaller $p_i$ for a smaller index.

$$\pi_1 = \{\tau_1\}, r = 1 \text{ and } Z_i = z_i$$

If $k \leq n$:

- Whether can $\tau_k$ pass the schedulability test of RM on shared resource $j$ under the interference from $\pi_j$?
  - Yes: Mark $\pi_j$ as a feasible resource.
  - No: Assign $\tau_k$ to an unused resource, $r++$, $Z_r = z_k$ and $k++$.

- If $j = r$:
  - Assign $\tau_k$ to arbitrarily feasible resource and $k++$.
- If $j < r$:
  - $j++$

If $k = n + 1$:

- Return $\sum_{i=1}^{r} z_i$.
Step 2: Partitioning tasks onto a multi-unit shared resource

Sort the tasks according to $z_i$ values ($z_i \geq z_{i+1}$); ties are broken by preferring smaller $p_i$ for a smaller index.

$$\pi_1 = \{\tau_1\}, r = 1 \text{ and } Z_1 = z_1$$

If $k \leq n$

Whether can $\tau_k$ pass the schedulability test of RM on shared resource $j$ under the interference from $\pi_j$?

- Yes
  - Mark $\pi_j$ as a feasible resource
  - If $j = r$
    - Assign $\tau_k$ to arbitrarily feasible resource and $k++$
  - If $j < r$
    - $j++$

- No
  - Assign $\tau_k$ to an unused resource, $r++$, $Z_r = z_k$ and $k++$

If $k = n + 1$

Return $\sum_{i=1}^{r} z_i$
Step 2: Partitioning tasks onto a multi-unit shared resource

Sort the tasks according to $z_i$ values ($z_i \geq z_{i+1}$); ties are broken by preferring smaller $p_i$ for a smaller index.

$$\pi_1 = \{\tau_1\} \text{, } r = 1 \text{ and } Z_1 = z_1$$

If $k \leq n$

---

Whether can $\tau_k$ pass the schedulability test of RM on shared resource $j$ under the interference from $\pi_j$?

Yes

Mark $\pi_j$ as a feasible resource

If $j = r$

Assign $\tau_k$ to arbitrarily feasible resource and $k++$

If $j < r$

No

Assign $\tau_k$ to an unused resource, $r++$, $Z_r = z_k$ and $k++$

If $k = n+1$

---

If $k \leq n$

---

Return $\sum_{i=1}^{r} z_i$

If $k = n+1$
Step 2: Partitioning tasks onto a multi-unit shared resource

Sort the tasks according to $z_i$ values ($z_i \geq z_{i+1}$); ties are broken by preferring smaller $p_i$ for a smaller index.

$\pi_1 = \{\tau_1\}, r = 1$ and $Z_1 = z_1$

If $k \leq n$

Whether can $\tau_k$ pass the schedulability test of RM on shared resource $j$ under the interference from $\pi_j$?

Yes

Mark $\pi_j$ as a feasible resource

If $j = r$

Assign $\tau_k$ to an arbitrary feasible resource and $k++$

If $j < r$

Assign $\tau_k$ to an unused resource, $r++$, $Z_r = z_k$, and $k++$

If $k = n+1$

Return $\sum_{i=1}^{r} z_i$

If $k < n$

No

If $k = n+1$

Return $\sum_{i=1}^{r} z_i$

If $k < n$

If $j = r$

Assign $\tau_k$ to an arbitrary feasible resource and $k++$

If $j < r$

Assign $\tau_k$ to an unused resource, $r++$, $Z_r = z_k$, and $k++$
Step 2: Partitioning tasks onto a multi-unit shared resource

Sort the tasks according to $z_i$ values ($z_i \geq z_{i+1}$); ties are broken by preferring smaller $p_i$ for a smaller index.

$$\pi_1 = \{\tau_1\}, r = 1 \text{ and } Z_i = z_i$$

If $k \leq n$:

- Whether can $\tau_k$ pass the schedulability test of RM on shared resource $j$ under the interference from $\pi_j$?
  - Yes: Mark $\pi_j$ as a feasible resource.
  - No:
    - If $j = r$: Assign $\tau_k$ to an unused resource, $r++$, $Z_r = z_k$ and $k++$
    - If $j < r$: Assign $\tau_k$ to arbitrarily feasible resource and $k++$
    - If $k = n+1$: Return $\sum_{i=1}^{r} Z_i$

If $k = n+1$:

- Assign $\tau_k$ to an unused resource, $r++$, $Z_r = z_k$ and $k++$

Zheng Dong @ RTAS’18, Porto
Step 2: Partitioning tasks onto a multi-unit shared resource

Sort the tasks according to $z_i$ values ($z_i \geq z_{i+1}$); ties are broken by preferring smaller $p_i$ for a smaller index.

$$\pi_1 = \{\tau_1\} , r = 1 \text{ and } Z_1 = z_1$$

If $k \leq n$

Whether can $\tau_k$ pass the schedulability test of RM on shared resource $j$ under the interference from $\pi_j$

Yes

Mark $\pi_j$ as a feasible resource

If $j = r$

Assign $\tau_k$ to arbitrarily feasible resource and $k++$

If $j < r$

No

If $k = n$

Assign $\tau_k$ to an unused resource, $r++$, $Z_{r} = z_k$ and $k++$

If $k = n+1$

Partitioning tasks onto a multi-unit shared resource

Time-demand Analysis (TDA)

Lemma 3 (Suspension and Blocking as Carry-In).

$$\exists t \mid 0 < t \leq p_k, s_k + e_k + \sigma_k \times B + \sum_{\tau_i \in h_p(k)} \left( \left\lfloor \frac{t}{p_i} \right\rfloor + 1 \right) s_i \leq t$$

(3)

Lemma 4 (Suspension and Blocking as Jitter).

$$\exists t \mid 0 < t \leq p_k, s_k + e_k + \sigma_k \times B + \sum_{\tau_i \in h_p(k)} \left[ t + \frac{p_i - s_i}{p_i} \right] s_i \leq t$$

Lemma 5 (Suspension and Blocking as Carry-In).

$$\left( \frac{s_k + c_k + \sigma_k \times B}{p_k} + 2 \right) \prod_{\tau_i \in h_p(k)} \left( \frac{s_i}{p_i} + 1 \right) \leq 3$$

(5a)

$$\sum_{\tau_i \in h_p(k)} \frac{s_i}{p_i} \leq \ln \left( \frac{3}{s_k + c_k + \sigma_k \times B + 2} \right)$$

(5b)

Lemma 6 (Suspension and Blocking as Jitter).

$$\frac{e_k + s_k + \sigma_k \times B}{p_k} + \sum_{\tau_i \in h_p(k)} \left( \frac{2s_i - s_i^2}{p_i} \right) + \sum_{\tau_i \in h_p(k)} \frac{s_i}{p_i} \leq 1$$

(6)

Constant-time Tests (CT)

Zheng Dong @ RTAS’18, Porto
Step 2: Partitioning tasks onto a multi-unit shared resource

Upper bounding the system utilization loss
Step 2: Partitioning tasks onto a multi-unit shared resource

Upper bounding the system utilization loss

Constant-time Tests (CT)

Lemma 5 (Suspension and Blocking as Carry-In).

\[
\left( \frac{s_k + e_k + \sigma_k \times B}{p_k} + 2 \right) \prod_{\tau_i \in h_p(k)} \left( \frac{s_i}{p_i} + 1 \right) \leq 3 \tag{5a}
\]

\[
\sum_{\tau_i \in h_p(k)} \frac{s_i}{p_i} \leq \ln \left( \frac{3}{s_k + e_k + \sigma_k \times B + 2} \right) \tag{5b}
\]

Lemma 6 (Suspension and Blocking as Jitter).

\[
\frac{e_k + s_k + \sigma_k \times B}{p_k} + \sum_{\tau_i \in h_p(k)} \left( \frac{2s_i - s_i^2}{p_i} \right) + \sum_{\tau_i \in h_p(k)} \frac{s_i}{p_i} \leq 1 \tag{6}
\]
Step 2: Partitioning tasks onto a multi-unit shared resource  

Upper bounding the system utilization loss

**Constant-time Tests (CT)**

**Lemma 5 (Suspension and Blocking as Carry-In).**

\[
\left( \frac{s_k + e_k + \sigma_k \times B}{p_k} + 2 \right) \prod_{\tau_i \in h_p(k)} \left( \frac{s_i}{p_i} + 1 \right) \leq 3 \quad (5a)
\]

\[
\sum_{\tau_i \in h_p(k)} \frac{s_i}{p_i} \leq \ln \left( \frac{3}{\frac{s_k + e_k + \sigma_k \times B}{p_k} + 2} \right) \quad (5b)
\]

**Lemma 6 (Suspension and Blocking as Jitter).**

\[
\frac{e_k + s_k + \sigma_k \times B}{p_k} + \sum_{\tau_i \in h_p(k)} \left( \frac{2s_i - s_i^2}{p_i} \right) + \sum_{\tau_i \in h_p(k)} \frac{s_i}{p_i} \leq 1 \quad (6)
\]

**ST-FF-CT(Carry)**

\[
U_{sum} \leq m \times \ln \left( \frac{3}{2 + u_{max}} \right)
\]

where \( u_{max} = \max_{i=m+1}^n \frac{e_i + s_i + \sigma_i \times B}{p_i} \)
Step 2: Partitioning tasks onto a multi-unit shared resource

Upper bounding the system utilization loss

**Constant-time Tests (CT)**

Lemma 5 (Suspension and Blocking as Carry-In).

\[
\left( \frac{s_k + c_k + \sigma_k \times B}{p_k} + 2 \right) \prod_{\tau_i \in h_p(k)} \left( \frac{s_i}{p_i} + 1 \right) \leq 3 \tag{5a}
\]

\[
\sum_{\tau_i \in h_p(k)} \frac{s_i}{p_i} \leq \ln \left( \frac{3}{\frac{s_k + c_k + \sigma_k \times B}{p_k} + 2} \right) \tag{5b}
\]

Lemma 6 (Suspension and Blocking as Jitter).

\[
\frac{e_k + s_k + \sigma_k \times B}{p_k} + \frac{\sum_{\tau_i \in h_p(k)} \left( 2s_i - s_i^2 / p_i \right)}{p_k} + \sum_{\tau_i \in h_p(k)} \frac{s_i}{p_i} \leq 1 \tag{6}
\]

**ST-FF-CT (Carry)**

\[
U_{sum} \leq m \times \ln \left( \frac{3}{2 + \bar{u}_{max}} \right)
\]

where \(\bar{u}_{max} = \max_{i=m+1}^{n} \frac{e_i + s_i + \sigma_i \times B}{p_i}\)

**ST-FF-CT (Jitter)**

\[
U_{sum} \leq \frac{m}{3} \left( 1 - \bar{u}_{max} \right)
\]

where \(\bar{u}_{max} = \max_{i=m+1}^{n} \frac{e_i + s_i + \sigma_i \times B}{p_i}\)
Step 2: Partitioning tasks onto a multi-unit shared resource

Upper bounding the system utilization loss

ST-FF-CT (Carry)

$$U_{sum} \leq \frac{m}{3} \left(1 - \frac{u_{max}}{3}\right)$$

where $$u_{max} = \max_{i=m+1}^n \left(\frac{e_i + s_i + a_i \times B}{p_i}\right)$$

Intuitively, task partitioning methods always suffer from both scheduling algorithms and bin-packing induced utilization loss.
Step 2: Partitioning tasks onto a multi-unit shared resource

An Integer linear programming based approach.
Step 2: Partitioning tasks onto a multi-unit shared resource

An Integer linear programming based approach.

\[ \text{min. the number of units of the shared resource} \]
Step 2: Partitioning tasks onto a multi-unit shared resource

An Integer linear programming based approach.

\[
\begin{align*}
\text{min.} & \quad \text{the number of units of the shared resource} \\
\text{s.t.} & \quad \text{All Possible Partitions}
\end{align*}
\]
Step 2: Partitioning tasks onto a multi-unit shared resource

An Integer linear programming based approach.

\[
\text{min. the number of units of the shared resource}
\]

\[
\text{s.t. All Possible Partitions}
\]

**Time-demand Analysis (TDA)**

**Lemma 3 (Suspension and Blocking as Carry-In).**
\[
\exists t \mid 0 < t \leq p_k, \quad s_k + e_k + \sigma_k \times B + \sum_{\tau_i \in h_p(k)} \left( \left\lfloor \frac{t}{p_i} \right\rfloor + 1 \right) \leq t
\]

**Lemma 4 (Suspension and Blocking as Jitter).**
\[
\exists t \mid 0 < t \leq p_k, \quad s_k + e_k + \sigma_k \times B + \sum_{\tau_i \in h_p(k)} \left( \frac{t + p_i - s_i}{p_i} \right) \leq t
\]

**Constant-time Tests (CT)**

**Lemma 5 (Suspension and Blocking as Carry-In).**
\[
\left( \frac{s_k + e_k + \sigma_k \times B}{p_k} + 2 \right) \prod_{\tau_i \in h_p(k)} \left( \frac{s_i}{p_i} + 1 \right) \leq 3 \quad (5a)
\]
\[
\sum_{\tau_i \in h_p(k)} \frac{s_i}{p_i} \leq \ln \left( \frac{3}{s_k + e_k + \sigma_k \times B + 2} \right) \quad (5b)
\]

**Lemma 6 (Suspension and Blocking as Jitter).**
\[
\frac{e_k + s_k + \sigma_k \times B}{p_k} + \sum_{\tau_i \in h_p(k)} \left( \frac{2s_i - s_i^2}{p_i} \right) + \sum_{\tau_i \in h_p(k)} \frac{s_i}{p_i} \leq 1 \quad (6)
\]
Case Study

❖ Our shared-resource-centric partitioning framework is fully implemented on a GPU-enabled platform.

- Ubuntu Linux system with a 4-core Intel i7-4770 CPU
- NVIDIA “Fermi” GTX 480 GPU (shared resource)

❖ Objectives: answer the following 3 questions

- Whether it is practically feasible to resolve the problem from shared-resource-centric angle.
- Whether solving this scheduling issue from the shared-resource-centric view is more effective than traditional CPU-centric view.
- Whether our proposed schedulability tests can provide reliable predictability for given task systems.
## Traditional CPU-centric Scheduling

<table>
<thead>
<tr>
<th>Task</th>
<th>Para.</th>
<th>Period</th>
<th>GPU</th>
<th>CPU</th>
<th>Res. Tim</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>12</td>
<td>6.40</td>
<td>0.80</td>
<td>1.00</td>
<td>N/A</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>12</td>
<td>25.58</td>
<td>1.60</td>
<td>3.00</td>
<td>20.5</td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>2</td>
<td>12.79</td>
<td>1.00</td>
<td>2.00</td>
<td>N/A</td>
</tr>
<tr>
<td>$\tau_4$</td>
<td>2</td>
<td>12.79</td>
<td>0.90</td>
<td>2.50</td>
<td>N/A</td>
</tr>
<tr>
<td>$\tau_5$</td>
<td>4</td>
<td>19.19</td>
<td>1.80</td>
<td>2.00</td>
<td>17.70</td>
</tr>
<tr>
<td>$\tau_6$</td>
<td>6</td>
<td>7.99</td>
<td>1.00</td>
<td>1.00</td>
<td>N/A</td>
</tr>
<tr>
<td>$\tau_7$</td>
<td>6</td>
<td>8.99</td>
<td>0.80</td>
<td>2.00</td>
<td>N/A</td>
</tr>
<tr>
<td>$\tau_8$</td>
<td>1</td>
<td>11.19</td>
<td>0.50</td>
<td>2.50</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**TABLE I: CPU-centric scheduling.**
**Traditional CPU-centric Scheduling**

<table>
<thead>
<tr>
<th>Task</th>
<th>Para.</th>
<th>Period</th>
<th>GPU</th>
<th>CPU</th>
<th>Res. Tim</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>12</td>
<td>6.40</td>
<td>0.80</td>
<td>1.00</td>
<td>N/A</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>12</td>
<td>25.58</td>
<td>1.60</td>
<td>3.00</td>
<td>20.5</td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>2</td>
<td>12.79</td>
<td>1.00</td>
<td>2.00</td>
<td>N/A</td>
</tr>
<tr>
<td>$\tau_4$</td>
<td>2</td>
<td>12.79</td>
<td>0.90</td>
<td>2.50</td>
<td>N/A</td>
</tr>
<tr>
<td>$\tau_5$</td>
<td>4</td>
<td>19.19</td>
<td>1.80</td>
<td>2.00</td>
<td>17.70</td>
</tr>
<tr>
<td>$\tau_6$</td>
<td>6</td>
<td>7.99</td>
<td>1.00</td>
<td>1.00</td>
<td>N/A</td>
</tr>
<tr>
<td>$\tau_7$</td>
<td>6</td>
<td>8.99</td>
<td>0.80</td>
<td>2.00</td>
<td>N/A</td>
</tr>
<tr>
<td>$\tau_8$</td>
<td>1</td>
<td>11.19</td>
<td>0.50</td>
<td>2.50</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**TABLE I: CPU-centric scheduling.**
### Traditional CPU-centric Scheduling

<table>
<thead>
<tr>
<th>Task</th>
<th>Para.</th>
<th>Period</th>
<th>GPU</th>
<th>CPU</th>
<th>Res. Tim</th>
</tr>
</thead>
<tbody>
<tr>
<td>τ₁</td>
<td>12</td>
<td>6.40</td>
<td>0.80</td>
<td>1.00</td>
<td>N/A</td>
</tr>
<tr>
<td>τ₂</td>
<td>12</td>
<td>25.58</td>
<td>1.60</td>
<td>3.00</td>
<td>20.5</td>
</tr>
<tr>
<td>τ₃</td>
<td>2</td>
<td>12.79</td>
<td>1.00</td>
<td>2.00</td>
<td>N/A</td>
</tr>
<tr>
<td>τ₄</td>
<td>2</td>
<td>12.79</td>
<td>0.90</td>
<td>2.50</td>
<td>N/A</td>
</tr>
<tr>
<td>τ₅</td>
<td>4</td>
<td>19.19</td>
<td>1.80</td>
<td>2.00</td>
<td>17.70</td>
</tr>
<tr>
<td>τ₆</td>
<td>6</td>
<td>7.99</td>
<td>1.00</td>
<td>1.00</td>
<td>N/A</td>
</tr>
<tr>
<td>τ₇</td>
<td>6</td>
<td>8.99</td>
<td>0.80</td>
<td>2.00</td>
<td>N/A</td>
</tr>
<tr>
<td>τ₈</td>
<td>1</td>
<td>11.19</td>
<td>0.50</td>
<td>2.50</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**TABLE I: CPU-centric scheduling.**
GPU-centric Scheduling (First-fit partitioning)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(\tau_1)</td>
<td>1</td>
<td>12</td>
<td>6.40</td>
<td>0.80</td>
<td>1.00</td>
<td>5.8</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>12</td>
</tr>
<tr>
<td>(\tau_6)</td>
<td>6</td>
<td></td>
<td>7.99</td>
<td>1.00</td>
<td>1.00</td>
<td>6.7</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>(\tau_7)</td>
<td>6</td>
<td></td>
<td>8.99</td>
<td>0.80</td>
<td>2.00</td>
<td>8.0</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>(\tau_8)</td>
<td>1</td>
<td>11.19</td>
<td>2.50</td>
<td></td>
<td></td>
<td>10.1</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>(\tau_4)</td>
<td>2</td>
<td>12.79</td>
<td>0.90</td>
<td></td>
<td></td>
<td>11.2</td>
<td>Fail</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>(\tau_3)</td>
<td>2</td>
<td>12.79</td>
<td>1.00</td>
<td></td>
<td></td>
<td>11.9</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>(\tau_5)</td>
<td>4</td>
<td>19.19</td>
<td>1.80</td>
<td></td>
<td></td>
<td>14</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>(\tau_2)</td>
<td>12</td>
<td>25.58</td>
<td>1.60</td>
<td></td>
<td></td>
<td>18.3</td>
<td>Pass</td>
<td>Fail</td>
<td>Pass</td>
<td></td>
</tr>
</tbody>
</table>

TABLE II: Shared-resource-centric scheduling with first-fit partitioning using constant-time (CT) schedulability tests.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(\tau_1)</td>
<td>1</td>
<td>12</td>
<td>6.40</td>
<td>0.80</td>
<td>1.00</td>
<td>6.13</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>12</td>
</tr>
<tr>
<td>(\tau_6)</td>
<td>6</td>
<td></td>
<td>7.99</td>
<td>1.00</td>
<td>1.00</td>
<td>7.5</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>(\tau_7)</td>
<td>6</td>
<td></td>
<td>8.99</td>
<td>0.80</td>
<td>2.00</td>
<td>8.64</td>
<td>Pass</td>
<td>Fail</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>(\tau_8)</td>
<td>1</td>
<td>11.19</td>
<td>0.80</td>
<td></td>
<td></td>
<td>10.5</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>(\tau_4)</td>
<td>2</td>
<td>12.79</td>
<td>0.90</td>
<td></td>
<td></td>
<td>12.2</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>(\tau_3)</td>
<td>2</td>
<td>12.79</td>
<td>1.00</td>
<td></td>
<td></td>
<td>12.6</td>
<td>Fail</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>(\tau_5)</td>
<td>4</td>
<td>19.19</td>
<td>1.80</td>
<td></td>
<td></td>
<td>10.1</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>(\tau_2)</td>
<td>12</td>
<td>25.58</td>
<td>1.60</td>
<td></td>
<td></td>
<td>11.8</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
</tbody>
</table>

TABLE III: Shared-resource-centric scheduling with first-fit partitioning using TDA-based schedulability tests.
Shared-Resource-Centric Limited Preemptive Scheduling

GPU-centric Scheduling (First-fit partitioning)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>1</td>
<td>12</td>
<td>6.40</td>
<td>0.80</td>
<td>1.00</td>
<td>5.8</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>12</td>
</tr>
<tr>
<td>$\tau_6$</td>
<td>6</td>
<td>7.99</td>
<td>1.00</td>
<td>1.00</td>
<td>6.7</td>
<td></td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>$\tau_7$</td>
<td>6</td>
<td>8.99</td>
<td>0.80</td>
<td>2.00</td>
<td>8.0</td>
<td></td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>$\tau_8$</td>
<td>1</td>
<td>11.19</td>
<td>0.50</td>
<td>2.50</td>
<td>10.1</td>
<td></td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>$\tau_4$</td>
<td>2</td>
<td>12.79</td>
<td>0.90</td>
<td>2.50</td>
<td>11.2</td>
<td></td>
<td>Fail</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>2</td>
<td>12.79</td>
<td>1.00</td>
<td>2.00</td>
<td>11.9</td>
<td></td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>$\tau_5$</td>
<td>3</td>
<td>19.19</td>
<td>1.80</td>
<td>2.00</td>
<td>14</td>
<td></td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>12</td>
<td>25.58</td>
<td>1.60</td>
<td>3.00</td>
<td>18.3</td>
<td></td>
<td>Pass</td>
<td>Fail</td>
<td>Fail</td>
<td></td>
</tr>
</tbody>
</table>

TABLE II: Shared-resource-centric scheduling with first-fit partitioning using constant-time (CT) schedulability tests.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>1</td>
<td>12</td>
<td>6.40</td>
<td>0.80</td>
<td>1.00</td>
<td>6.13</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>12</td>
</tr>
<tr>
<td>$\tau_6$</td>
<td>6</td>
<td>7.99</td>
<td>1.00</td>
<td>1.00</td>
<td>7.5</td>
<td></td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>$\tau_7$</td>
<td>6</td>
<td>8.99</td>
<td>0.80</td>
<td>2.00</td>
<td>8.64</td>
<td></td>
<td>Pass</td>
<td>Fail</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>$\tau_8$</td>
<td>1</td>
<td>11.19</td>
<td>0.50</td>
<td>2.50</td>
<td>10.5</td>
<td></td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>$\tau_4$</td>
<td>2</td>
<td>12.79</td>
<td>0.90</td>
<td>2.50</td>
<td>12.2</td>
<td></td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>2</td>
<td>12.79</td>
<td>1.00</td>
<td>2.00</td>
<td>12.6</td>
<td></td>
<td>Fail</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>$\tau_5$</td>
<td>3</td>
<td>19.19</td>
<td>1.80</td>
<td>2.00</td>
<td>10.1</td>
<td></td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>12</td>
<td>25.58</td>
<td>1.60</td>
<td>3.00</td>
<td>11.8</td>
<td></td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
</tbody>
</table>

TABLE III: Shared-resource-centric scheduling with first-fit partitioning using TDA-based schedulability tests.
### GPU-centric Scheduling (First-fit partitioning)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>1</td>
<td>12</td>
<td>6.40</td>
<td>0.80</td>
<td>1.00</td>
<td>5.8</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>12</td>
</tr>
<tr>
<td>$\tau_6$</td>
<td>6</td>
<td>7.99</td>
<td>1.00</td>
<td>1.00</td>
<td>6.7</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_7$</td>
<td>6</td>
<td>8.99</td>
<td>0.80</td>
<td>2.00</td>
<td>8.0</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_8$</td>
<td>1</td>
<td>11.19</td>
<td>0.50</td>
<td>2.50</td>
<td>10.1</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_4$</td>
<td>2</td>
<td>12.79</td>
<td>0.90</td>
<td>2.50</td>
<td>11.2</td>
<td>Pass</td>
<td>Fail</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>2</td>
<td>12.79</td>
<td>1.00</td>
<td>2.00</td>
<td>11.9</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_5$</td>
<td>4</td>
<td>19.19</td>
<td>1.80</td>
<td>2.00</td>
<td>14</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>12</td>
<td>25.58</td>
<td>1.60</td>
<td>3.00</td>
<td>18.3</td>
<td>Pass</td>
<td>Fail</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TABLE II:** Shared-resource-centric scheduling with first-fit partitioning using constant-time (CT) schedulability tests.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>12</td>
<td>6.40</td>
<td>0.80</td>
<td>1.00</td>
<td>6.13</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>$\tau_6$</td>
<td>6</td>
<td>7.99</td>
<td>1.00</td>
<td>1.00</td>
<td>7.5</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_7$</td>
<td>6</td>
<td>8.99</td>
<td>0.80</td>
<td>2.00</td>
<td>8.64</td>
<td>Pass</td>
<td>Fail</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_8$</td>
<td>1</td>
<td>11.19</td>
<td>0.50</td>
<td>2.50</td>
<td>10.5</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_4$</td>
<td>2</td>
<td>12.79</td>
<td>0.90</td>
<td>2.50</td>
<td>12.2</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>2</td>
<td>12.79</td>
<td>1.00</td>
<td>2.00</td>
<td>12.6</td>
<td>Fail</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_5$</td>
<td>4</td>
<td>19.19</td>
<td>1.80</td>
<td>2.00</td>
<td>10.1</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>12</td>
<td>25.58</td>
<td>1.60</td>
<td>3.00</td>
<td>11.8</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TABLE III:** Shared-resource-centric scheduling with first-fit partitioning using TDA-based schedulability tests.
GPU-centric Scheduling (First-fit partitioning)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>τ₁</td>
<td>1</td>
<td>12</td>
<td>6.40</td>
<td>0.80</td>
<td>1.00</td>
<td>5.8</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>12</td>
</tr>
<tr>
<td>τ₆</td>
<td>6</td>
<td></td>
<td>7.99</td>
<td>1.00</td>
<td>1.00</td>
<td>6.7</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>τ₇</td>
<td>6</td>
<td></td>
<td>8.99</td>
<td>0.80</td>
<td>2.00</td>
<td>8.0</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>τ₈</td>
<td>1</td>
<td>11.19</td>
<td>0.50</td>
<td>2.50</td>
<td>10.1</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>τ₄</td>
<td>2</td>
<td>12.79</td>
<td>0.90</td>
<td>2.50</td>
<td>11.2</td>
<td>Fail</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>τ₃</td>
<td>2</td>
<td>12.79</td>
<td>1.00</td>
<td>2.00</td>
<td>11.9</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>6</td>
</tr>
<tr>
<td>τ₅</td>
<td>4</td>
<td>19.19</td>
<td>1.80</td>
<td>2.00</td>
<td>14</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>12</td>
</tr>
<tr>
<td>τ₂</td>
<td>12</td>
<td></td>
<td>25.58</td>
<td>1.60</td>
<td>3.00</td>
<td>18.3</td>
<td>Pass</td>
<td>Fail</td>
<td>Pass</td>
<td></td>
</tr>
</tbody>
</table>

TABLE II: Shared-resource-centric scheduling with first-fit partitioning using constant-time (CT) schedulability test.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>τ₁</td>
<td>1</td>
<td>12</td>
<td>6.40</td>
<td>0.80</td>
<td>1.00</td>
<td>6.13</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>12</td>
</tr>
<tr>
<td>τ₆</td>
<td>6</td>
<td></td>
<td>7.99</td>
<td>1.00</td>
<td>1.00</td>
<td>7.5</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>τ₇</td>
<td>6</td>
<td></td>
<td>8.99</td>
<td>0.80</td>
<td>2.00</td>
<td>8.64</td>
<td>Pass</td>
<td>Fail</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>τ₈</td>
<td>1</td>
<td>11.19</td>
<td>0.50</td>
<td>2.50</td>
<td>10.5</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>τ₄</td>
<td>2</td>
<td>12.79</td>
<td>0.90</td>
<td>2.50</td>
<td>12.2</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>2</td>
</tr>
<tr>
<td>τ₃</td>
<td>2</td>
<td>12.79</td>
<td>1.00</td>
<td>2.00</td>
<td>12.6</td>
<td>Fail</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>τ₅</td>
<td>4</td>
<td>19.19</td>
<td>1.80</td>
<td>2.00</td>
<td>10.1</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>12</td>
</tr>
<tr>
<td>τ₂</td>
<td>12</td>
<td></td>
<td>25.58</td>
<td>1.60</td>
<td>3.00</td>
<td>11.8</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
</tbody>
</table>

TABLE III: Shared-resource-centric scheduling with first-fit partitioning using TDA-based schedulability tests.
GPU-centric Scheduling (First-fit partitioning)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>τ₁</td>
<td>1</td>
<td>12</td>
<td>6.40</td>
<td>0.80</td>
<td>1.00</td>
<td>5.8</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>12</td>
</tr>
<tr>
<td>τ₆</td>
<td>6</td>
<td>7.99</td>
<td>1.00</td>
<td>1.00</td>
<td>6.7</td>
<td></td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>τ₇</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>τ₈</td>
<td>2</td>
<td>8.99</td>
<td>0.80</td>
<td>2.00</td>
<td>8.0</td>
<td></td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>6</td>
</tr>
<tr>
<td>τ₄</td>
<td>1</td>
<td>11.19</td>
<td>0.50</td>
<td>2.50</td>
<td>10.1</td>
<td></td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>τ₃</td>
<td>2</td>
<td>12.79</td>
<td>0.90</td>
<td>2.50</td>
<td>11.2</td>
<td></td>
<td>Fail</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>τ₅</td>
<td>3</td>
<td>12</td>
<td>1.60</td>
<td></td>
<td></td>
<td></td>
<td>Pass</td>
<td>Fail</td>
<td>Pass</td>
<td>12</td>
</tr>
<tr>
<td>τ₂</td>
<td>12</td>
<td>25.58</td>
<td>3.00</td>
<td></td>
<td></td>
<td></td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
</tbody>
</table>

TABLE II: Shared-resource-centric scheduling with first-fit partitioning using constant-time (CT) schedulability tests.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>τ₁</td>
<td>1</td>
<td>12</td>
<td>6.40</td>
<td>0.80</td>
<td>1.00</td>
<td>6.13</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>12</td>
</tr>
<tr>
<td>τ₆</td>
<td>6</td>
<td>7.99</td>
<td>1.00</td>
<td>1.00</td>
<td>7.5</td>
<td></td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>τ₇</td>
<td>6</td>
<td>8.99</td>
<td>0.80</td>
<td>2.00</td>
<td>8.64</td>
<td></td>
<td>Pass</td>
<td>Fail</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>τ₈</td>
<td>1</td>
<td>11.19</td>
<td>0.50</td>
<td>2.50</td>
<td>10.5</td>
<td></td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>12</td>
</tr>
<tr>
<td>τ₄</td>
<td>2</td>
<td>12.79</td>
<td>0.90</td>
<td>2.50</td>
<td>12.2</td>
<td></td>
<td>Fail</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>τ₃</td>
<td>2</td>
<td>12.79</td>
<td>1.00</td>
<td>2.00</td>
<td>12.6</td>
<td></td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>2</td>
</tr>
<tr>
<td>τ₅</td>
<td>3</td>
<td>19.19</td>
<td>1.80</td>
<td>2.00</td>
<td>10.1</td>
<td></td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>τ₂</td>
<td>12</td>
<td>25.58</td>
<td>1.60</td>
<td>3.00</td>
<td>11.8</td>
<td></td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
</tbody>
</table>

TABLE III: Shared-resource-centric scheduling with first-fit partitioning using TDA-based schedulability tests.
Shared-Resource-Centric Limited Preemptive Scheduling

**GPU-centric Scheduling (ILP-based partitioning)**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>12</td>
<td>6.40</td>
<td>0.80</td>
<td>1.00</td>
<td>6.03</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>12</td>
<td>25.58</td>
<td>1.60</td>
<td>3.00</td>
<td>17.71</td>
<td>Fail</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_7$</td>
<td>6</td>
<td>8.99</td>
<td>0.80</td>
<td>2.00</td>
<td>8.32</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_6$</td>
<td>2</td>
<td>7.99</td>
<td>1.00</td>
<td>1.00</td>
<td>7.45</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_5$</td>
<td>4</td>
<td>19.19</td>
<td>1.80</td>
<td>2.00</td>
<td>16.6</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>2</td>
<td>12.79</td>
<td>2.00</td>
<td>2.00</td>
<td>12.6</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_4$</td>
<td>2</td>
<td>12.79</td>
<td>0.90</td>
<td>2.50</td>
<td>12.2</td>
<td>Fail</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_8$</td>
<td>1</td>
<td>11.19</td>
<td>0.50</td>
<td>2.50</td>
<td>10.5</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TABLE IV:** Shared-resource-centric scheduling with Algorithm 2 using CT schedulability tests.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>12</td>
<td>6.40</td>
<td>0.80</td>
<td>1.00</td>
<td>6.03</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>12</td>
<td>25.58</td>
<td>1.60</td>
<td>3.00</td>
<td>17.71</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_7$</td>
<td>6</td>
<td>8.99</td>
<td>0.80</td>
<td>2.00</td>
<td>8.32</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_6$</td>
<td>2</td>
<td>7.99</td>
<td>1.00</td>
<td>1.00</td>
<td>7.45</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_5$</td>
<td>4</td>
<td>19.19</td>
<td>1.80</td>
<td>2.00</td>
<td>16.6</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>2</td>
<td>12.79</td>
<td>2.00</td>
<td>2.00</td>
<td>12.6</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_4$</td>
<td>2</td>
<td>12.79</td>
<td>0.90</td>
<td>2.50</td>
<td>12.2</td>
<td>Fail</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_8$</td>
<td>1</td>
<td>11.19</td>
<td>0.50</td>
<td>2.50</td>
<td>10.5</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TABLE V:** Shared-resource-centric scheduling with Algorithm 2 using TDA-based schedulability tests.
### GPU-centric Scheduling (ILP-based partitioning)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>1</td>
<td>12</td>
<td>6.40</td>
<td>0.80</td>
<td>1.00</td>
<td>6.03</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>12</td>
<td>25.58</td>
<td>1.60</td>
<td>3.00</td>
<td></td>
<td>17.71</td>
<td>Fail</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>$\tau_7$</td>
<td>6</td>
<td>8.99</td>
<td>0.80</td>
<td>2.00</td>
<td></td>
<td>8.32</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>$\tau_6$</td>
<td>2</td>
<td>6</td>
<td>7.99</td>
<td>1.00</td>
<td>1.00</td>
<td>7.45</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>12</td>
</tr>
<tr>
<td>$\tau_5$</td>
<td>4</td>
<td>19.19</td>
<td>1.80</td>
<td>2.00</td>
<td></td>
<td>16.6</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>2</td>
<td>12.79</td>
<td>2.00</td>
<td></td>
<td>2.00</td>
<td>12.6</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>$\tau_4$</td>
<td>2</td>
<td>12.79</td>
<td>0.90</td>
<td>2.50</td>
<td></td>
<td>12.2</td>
<td>Fail</td>
<td>Pass</td>
<td>Pass</td>
<td>2</td>
</tr>
<tr>
<td>$\tau_8$</td>
<td>1</td>
<td>11.19</td>
<td>0.50</td>
<td>2.50</td>
<td></td>
<td>10.5</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
</tbody>
</table>

**TABLE IV:** Shared-resource-centric scheduling with Algorithm 2 using CT schedulability tests.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>1</td>
<td>12</td>
<td>6.40</td>
<td>0.80</td>
<td>1.00</td>
<td>6.03</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>12</td>
<td>25.58</td>
<td>1.60</td>
<td>3.00</td>
<td></td>
<td>17.71</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>12</td>
</tr>
<tr>
<td>$\tau_7$</td>
<td>6</td>
<td>8.99</td>
<td>0.80</td>
<td>2.00</td>
<td></td>
<td>8.32</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>$\tau_6$</td>
<td>2</td>
<td>6</td>
<td>7.99</td>
<td>1.00</td>
<td>1.00</td>
<td>7.45</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>$\tau_5$</td>
<td>4</td>
<td>19.19</td>
<td>1.80</td>
<td>2.00</td>
<td></td>
<td>16.6</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>2</td>
<td>12.79</td>
<td>2.00</td>
<td></td>
<td>2.00</td>
<td>12.6</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>$\tau_4$</td>
<td>2</td>
<td>12.79</td>
<td>0.90</td>
<td>2.50</td>
<td></td>
<td>12.2</td>
<td>Fail</td>
<td>Pass</td>
<td>Pass</td>
<td>2</td>
</tr>
<tr>
<td>$\tau_8$</td>
<td>1</td>
<td>11.19</td>
<td>0.50</td>
<td>2.50</td>
<td></td>
<td>10.5</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
</tbody>
</table>

**TABLE V:** Shared-resource-centric scheduling with Algorithm 2 using TDA-based schedulability tests.
## GPU-centric Scheduling (ILP-based partitioning)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>1</td>
<td>12</td>
<td>6.40</td>
<td>0.80</td>
<td>1.00</td>
<td>6.03</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>12</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>1</td>
<td>12</td>
<td>25.58</td>
<td>1.60</td>
<td>3.00</td>
<td>17.71</td>
<td><strong>Fail</strong></td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>$\tau_7$</td>
<td>6</td>
<td>8.99</td>
<td>0.80</td>
<td>2.00</td>
<td>8.32</td>
<td><strong>Fail</strong></td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>$\tau_6$</td>
<td>2</td>
<td>6</td>
<td>7.99</td>
<td>1.00</td>
<td>1.00</td>
<td>7.45</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>6</td>
</tr>
<tr>
<td>$\tau_5$</td>
<td>4</td>
<td>19.19</td>
<td>1.80</td>
<td>2.00</td>
<td>16.6</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>2</td>
<td>2</td>
<td>12.79</td>
<td>2.00</td>
<td>2.00</td>
<td>12.6</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>2</td>
</tr>
<tr>
<td>$\tau_4$</td>
<td>2</td>
<td>12.79</td>
<td>0.90</td>
<td>2.50</td>
<td>12.2</td>
<td><strong>Fail</strong></td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>$\tau_8$</td>
<td>1</td>
<td>11.19</td>
<td>0.50</td>
<td>2.50</td>
<td>10.5</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TABLE IV:** Shared-resource-centric scheduling with Algorithm 2 using CT schedulability tests.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>1</td>
<td>12</td>
<td>6.40</td>
<td>0.80</td>
<td>1.00</td>
<td>6.03</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>12</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>12</td>
<td>25.58</td>
<td>1.60</td>
<td>3.00</td>
<td>17.71</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_7$</td>
<td>6</td>
<td>8.99</td>
<td>0.80</td>
<td>2.00</td>
<td>8.32</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_6$</td>
<td>2</td>
<td>6</td>
<td>7.99</td>
<td>1.00</td>
<td>1.00</td>
<td>7.45</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>6</td>
</tr>
<tr>
<td>$\tau_5$</td>
<td>4</td>
<td>19.19</td>
<td>1.80</td>
<td>2.00</td>
<td>16.6</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>2</td>
<td>2</td>
<td>12.79</td>
<td>2.00</td>
<td>2.00</td>
<td>12.6</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>2</td>
</tr>
<tr>
<td>$\tau_4$</td>
<td>2</td>
<td>12.79</td>
<td>0.90</td>
<td>2.50</td>
<td>12.2</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_8$</td>
<td>1</td>
<td>11.19</td>
<td>0.50</td>
<td>2.50</td>
<td>10.5</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TABLE V:** Shared-resource-centric scheduling with Algorithm 2 using TDA-based schedulability tests.
## GPU-centric Scheduling (ILP-based partitioning)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>12</td>
<td>6.40</td>
<td>0.80</td>
<td>1.00</td>
<td>6.03</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>12</td>
<td>25.58</td>
<td>1.60</td>
<td>3.00</td>
<td>17.71</td>
<td>Fail</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>6</td>
<td>8.99</td>
<td>0.80</td>
<td>2.00</td>
<td>8.32</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_4$</td>
<td>2</td>
<td>7.99</td>
<td>1.00</td>
<td>1.00</td>
<td>7.45</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>$\tau_5$</td>
<td>4</td>
<td>19.19</td>
<td>1.80</td>
<td>2.00</td>
<td>16.6</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_6$</td>
<td>2</td>
<td>12.79</td>
<td>2.00</td>
<td>2.00</td>
<td>12.6</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_7$</td>
<td>2</td>
<td>12.79</td>
<td>0.90</td>
<td>2.50</td>
<td>12.2</td>
<td>Fail</td>
<td>Pass</td>
<td>Pass</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$\tau_8$</td>
<td>1</td>
<td>11.19</td>
<td>0.50</td>
<td>2.50</td>
<td>10.5</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TABLE IV:** Shared-resource-centric scheduling with Algorithm 2 using CT schedulability tests.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>12</td>
<td>6.40</td>
<td>0.80</td>
<td>1.00</td>
<td>6.03</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>12</td>
<td>25.58</td>
<td>1.60</td>
<td>3.00</td>
<td>17.71</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>6</td>
<td>8.99</td>
<td>0.80</td>
<td>2.00</td>
<td>8.32</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_4$</td>
<td>2</td>
<td>7.99</td>
<td>1.00</td>
<td>1.00</td>
<td>7.45</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>$\tau_5$</td>
<td>4</td>
<td>19.19</td>
<td>1.80</td>
<td>2.00</td>
<td>16.6</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_6$</td>
<td>2</td>
<td>12.79</td>
<td>2.00</td>
<td>2.00</td>
<td>12.6</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_7$</td>
<td>2</td>
<td>12.79</td>
<td>0.90</td>
<td>2.50</td>
<td>12.2</td>
<td>Fail</td>
<td>Pass</td>
<td>Pass</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$\tau_8$</td>
<td>1</td>
<td>11.19</td>
<td>0.50</td>
<td>2.50</td>
<td>10.5</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TABLE V:** Shared-resource-centric scheduling with Algorithm 2 using TDA-based schedulability tests.

Zheng Dong @ RTAS'18, Porto
Simulation evaluation

❖ The UUnifast-Discard method was applied to generate task sets.

❖ Evaluated methods
  ➡ Comparison with first-fit partition using TDA-based tests.
  ➡ Comparison with first-fit partition using constant time tests.
  ➡ Comparison among different ILP-based approaches.
  ➡ Representative Approaches Comparison.
First-Fit partition using TDA-based tests (CT tests).

![Graph showing normalized number of required shared resource vs utilization](image-url)
ILP-based approaches and representative approaches comparison

- **Limited-Gmeanratio-20Tasks-e(L)-b(L)**

![Graph showing normalized number of required shared resource vs utilization (in %). The graph compares different ILP-based approaches and a baseline approach.](image-url)
Conclusion

❖ Develop a HRT multi-phase task scheduling framework

- Resolve this problem from a counter-intuitive shared-resource-centric perspective.
- Develop a rather comprehensive set of suspending task partitioning algorithms that minimize the required size of the shared resource.
- A GPU-based prototype case study and extensive simulation-based experiments have been conducted.
Thank you!