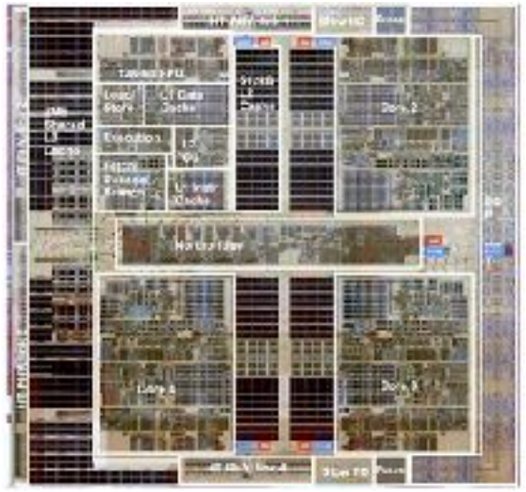
Shared-Resource-Centric Limited Preemptive Scheduling: A Comprehensive Study of Suspension-based Partitioning Approaches

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

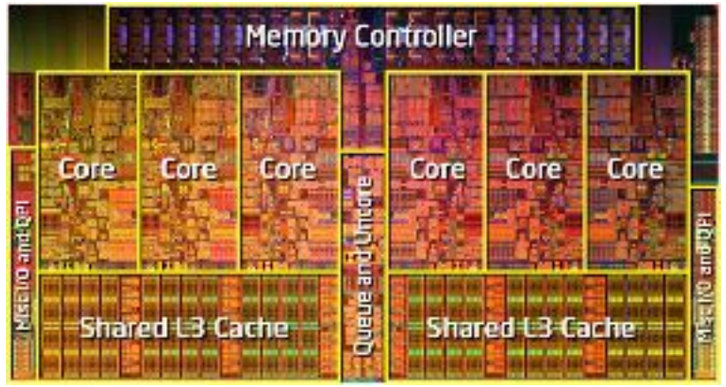
Kuan-Hsun Chen Jian-Jia Chen Georg von der Bruggen Junjie Shi
Technical University of Dortmund, Germany

Multi-Core Processor

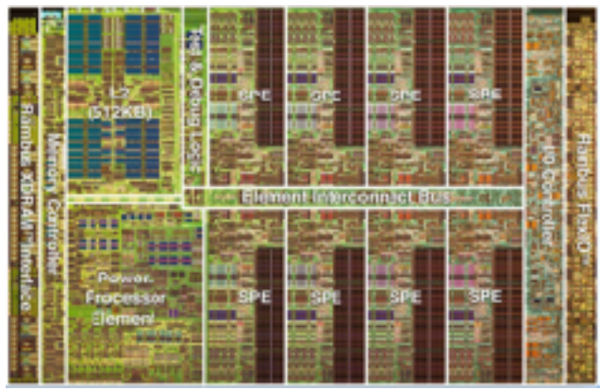
- Simpler and lower power than a single large core
- Smaller inter-core communication latency



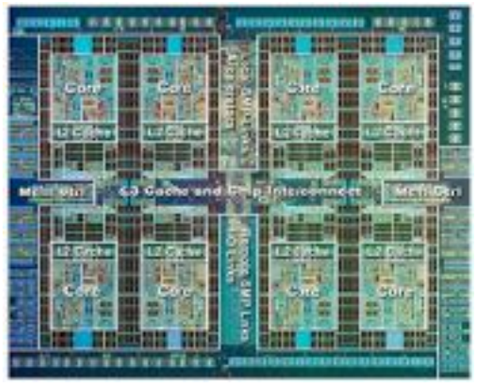
AMD Barcelona (4 cores)



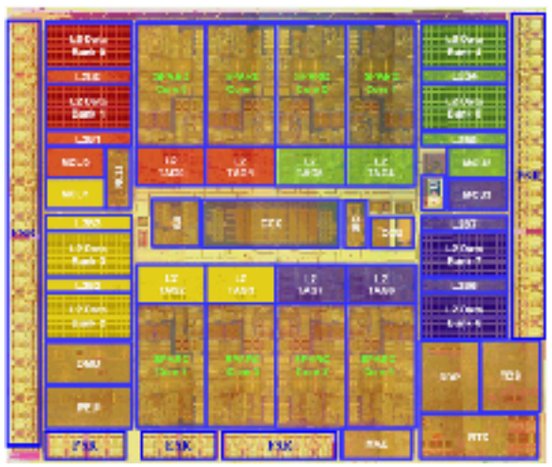
Intel Core i7 (8 cores)



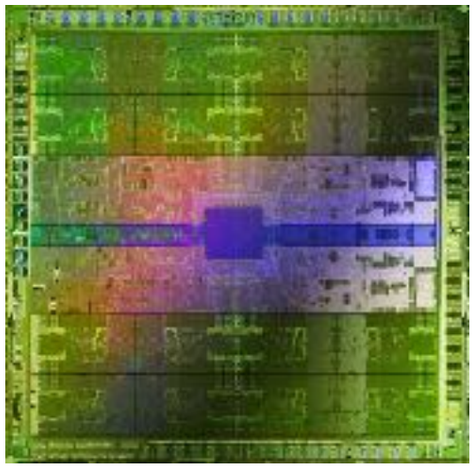
IBM Cell BE (8+1 cores)



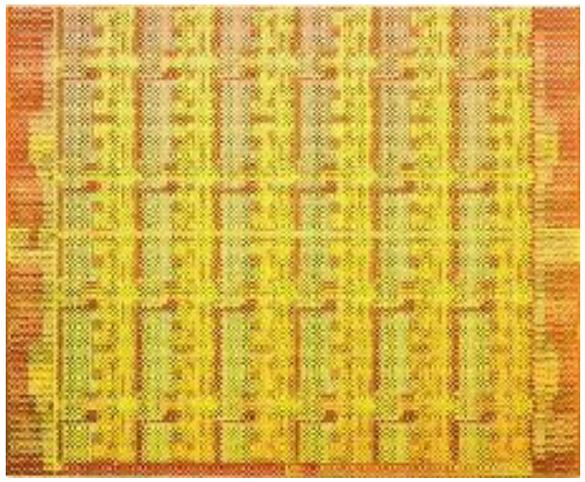
IBM POWER7 (8 cores)



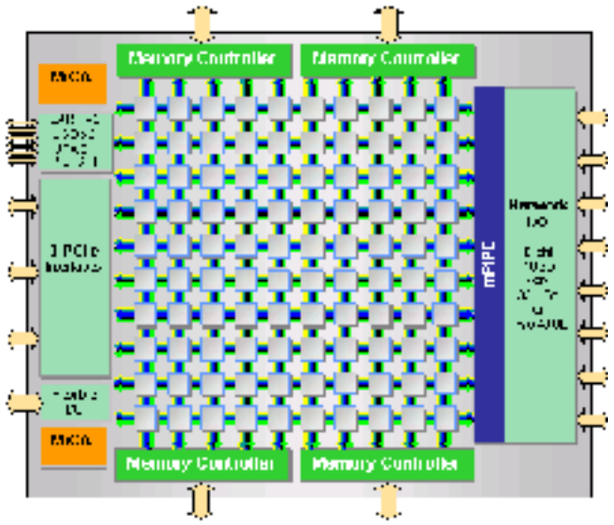
Sun Niagara 2 (8 cores)



Nvidia Fermi (448 cores)



Intel SCC (48 cores)



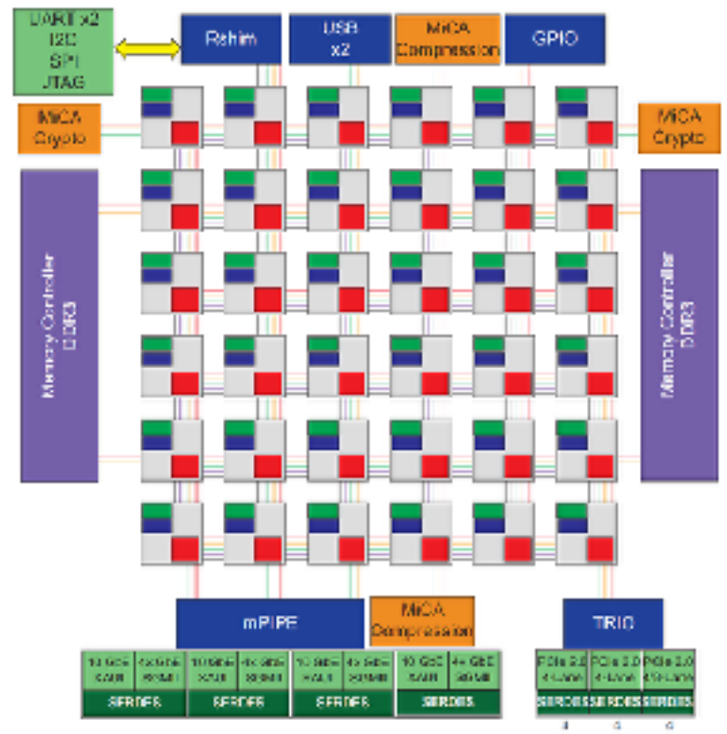
Tilera TILE Gx (100 cores)

An example of multi-phase task

Autonomous Driving Vehicle

An example of multi-phase task

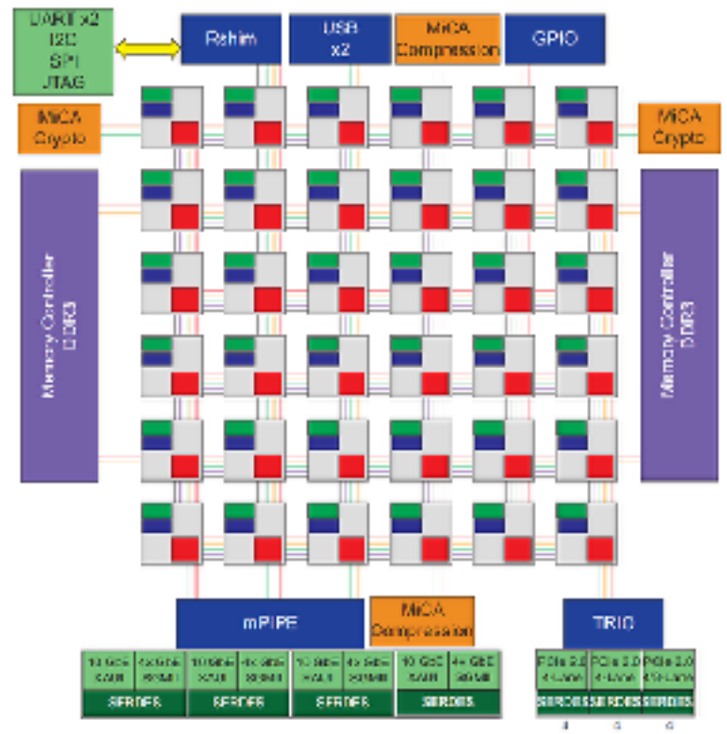
Autonomous Driving Vehicle



100 cores on chip

An example of multi-phase task

Autonomous Driving Vehicle

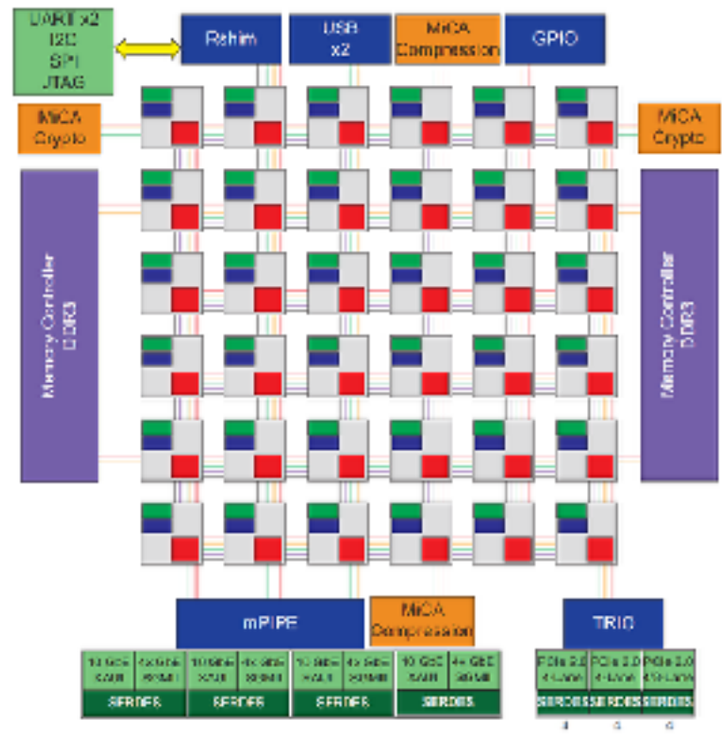


100 cores on chip

GPU (shared resource)

An example of multi-phase task

Autonomous Driving Vehicle



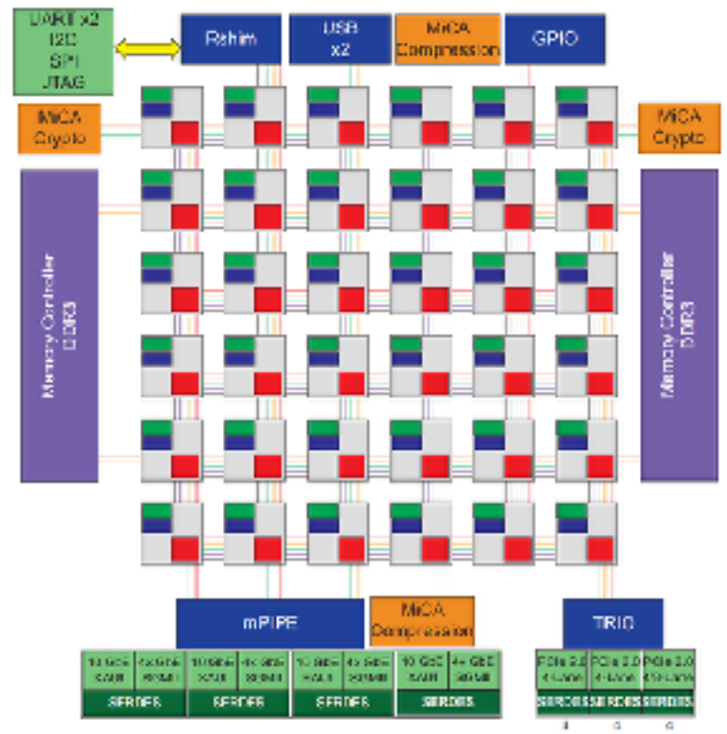
100 cores on chip

GPU (shared resource)

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

An example of multi-phase task

Autonomous Driving Vehicle



100 cores on chip

GPU (shared resource)

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

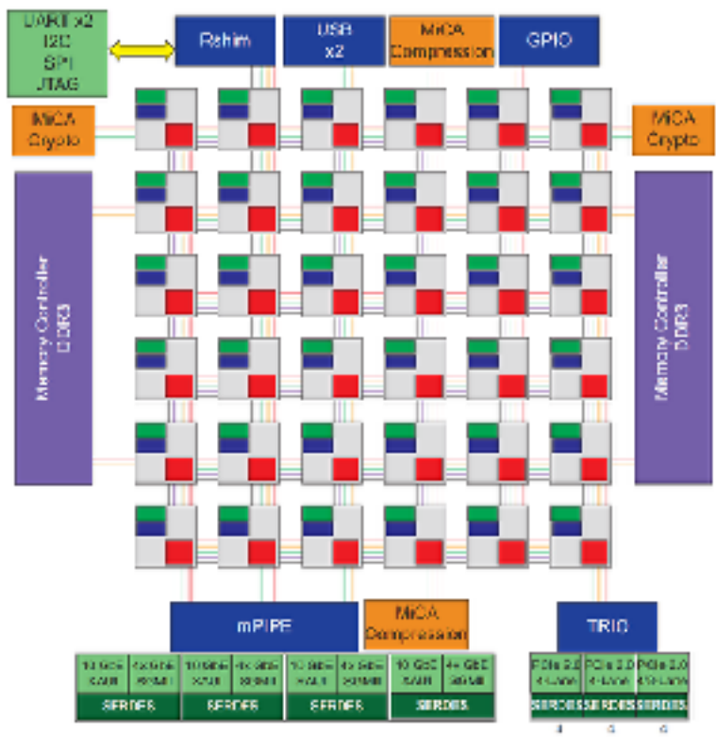
Multi-phase task:

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



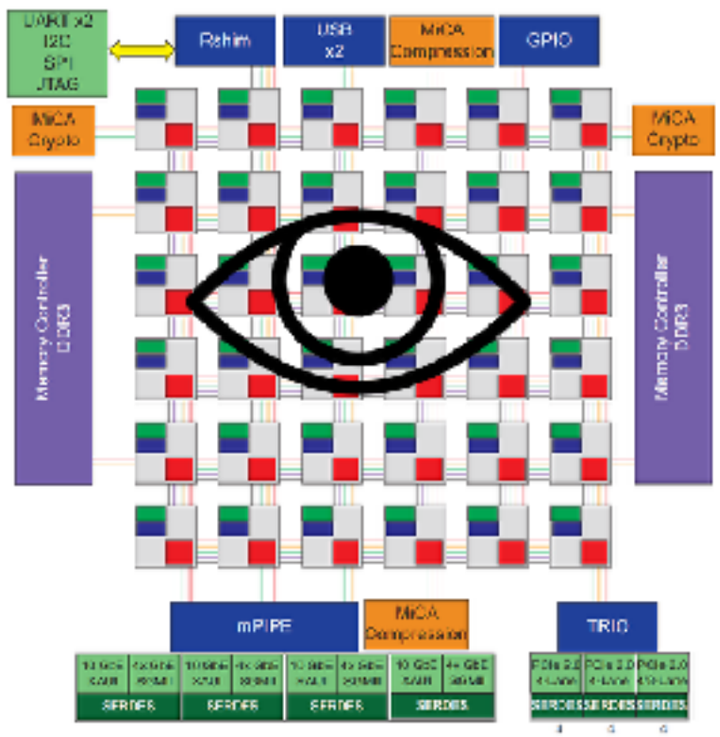
100 cores on chip

GPU (accelerator)

Traditional methods encounter multi-core system

Multi-phase task:

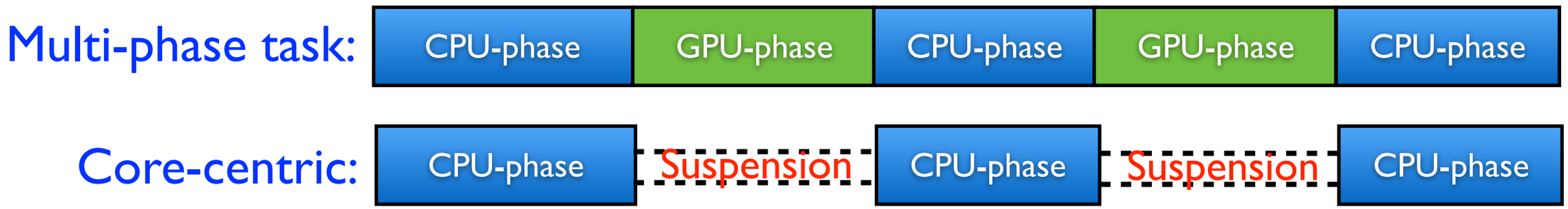
CPU-phase	GPU-phase	CPU-phase	GPU-phase	CPU-phase

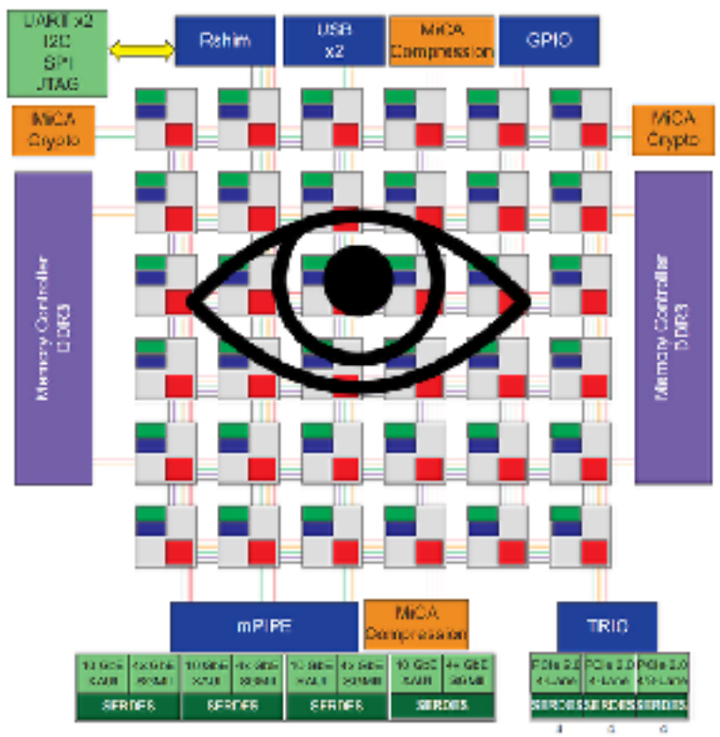


100 cores on chip

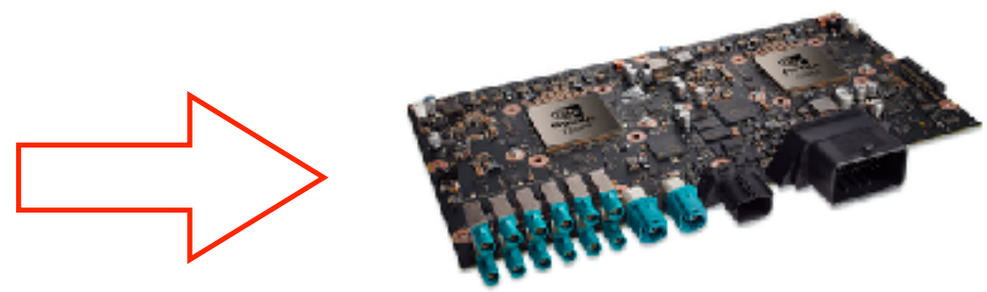
GPU (accelerator)

Traditional methods encounter multi-core system



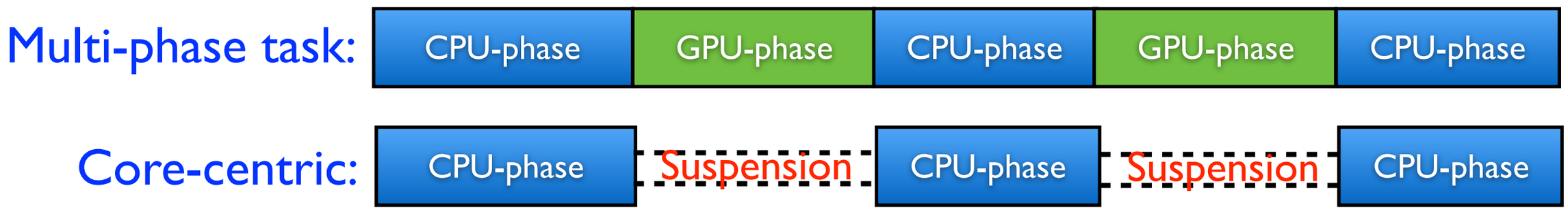


100 cores on chip

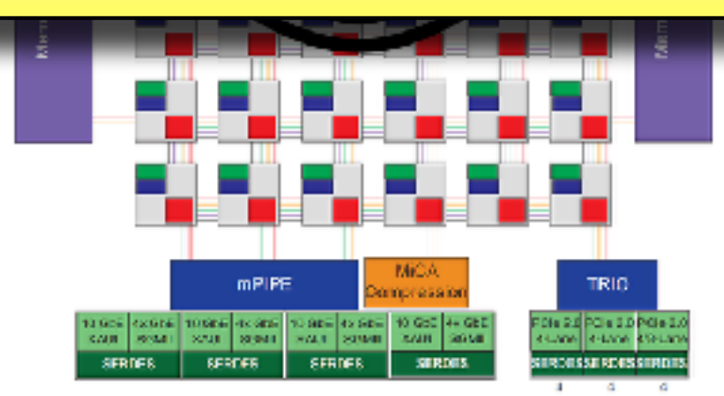


GPU (accelerator)

Traditional methods encounter multi-core system



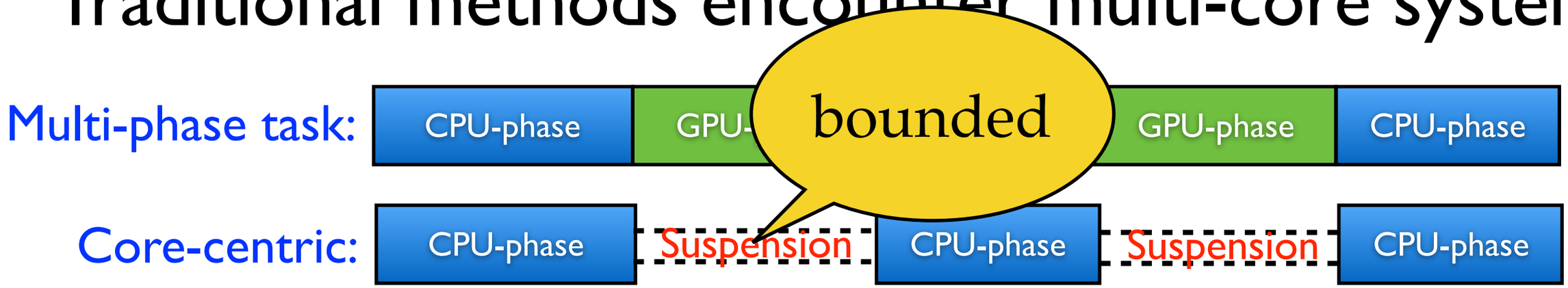
The idea is to focus on judiciously allocating CPU resources while viewing shared resources as I/O (suspensions) and bounding the worst-case latency a task may experience on such resources.



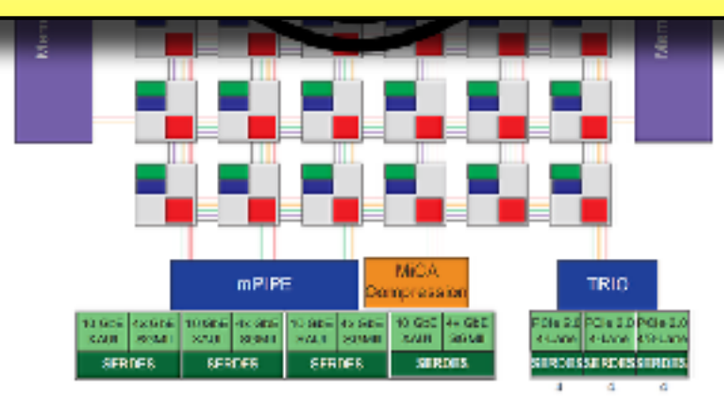
100 cores on chip

GPU (accelerator)

Traditional methods encounter multi-core system



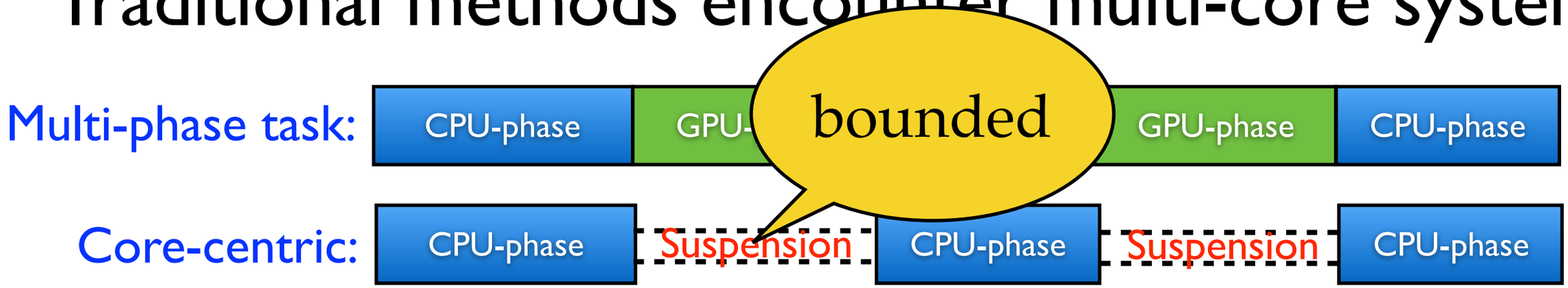
The idea is to focus on judiciously allocating CPU resources while viewing shared resources as I/O (suspensions) and bounding the worst-case latency a task may experience on such resources.



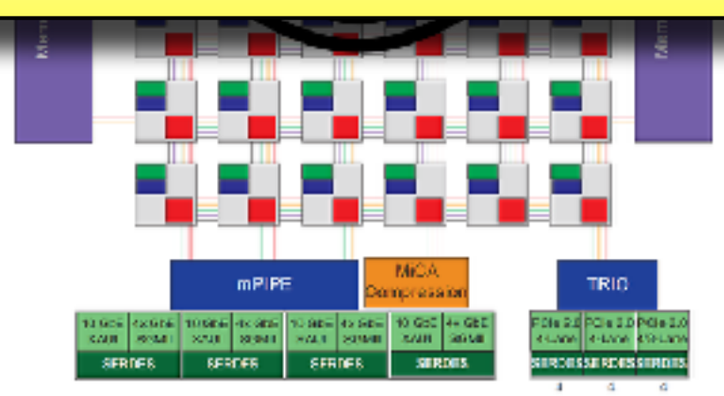
100 cores on chip

GPU (accelerator)

Traditional methods encounter multi-core system



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.



100 cores on chip

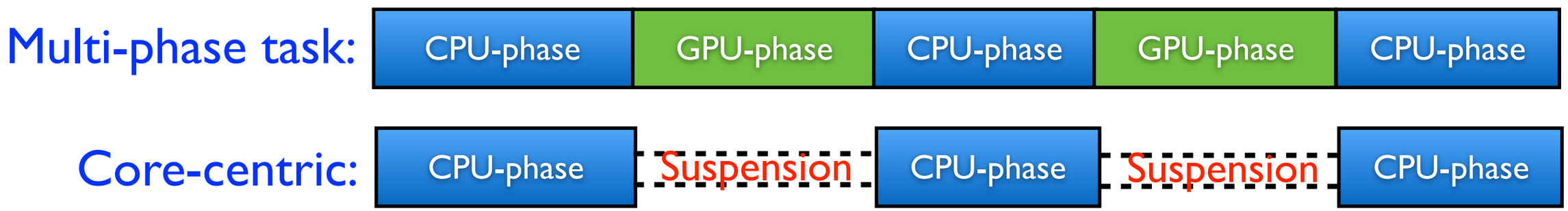
GPU (accelerator)

[1] Zheng Dong and Cong Liu. **Closing the loop for the selective conversion approach: a utilization-based test for hard real-time suspending task systems**, *Proceedings of the 37th IEEE Real-Time Systems Symposium*, RTSS 2016.

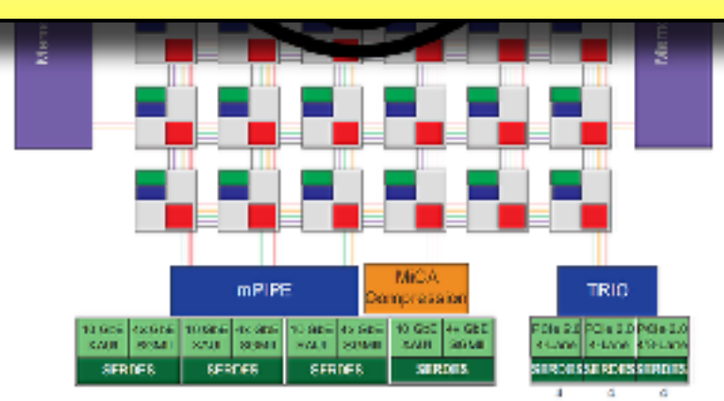
[2] Jianjia Chen, Wenhung Huang, and Cong Liu. **K2U: A General Framework from k-Point Effective Schedulabiliy Analysis to Utilization-based Tests**, *Proceedings of the 36th IEEE Real-Time Systems Symposium*, RTSS 2015.

[3] Cong Liu and James Anderson. **An O(m) Analysis Technique for Supporting Real-Time Self-Suspending Task Systems**, *Proceedings of the 33th IEEE Real-Time Systems Symposium, RTSS 2012*

Traditional methods encounter multi-core system



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

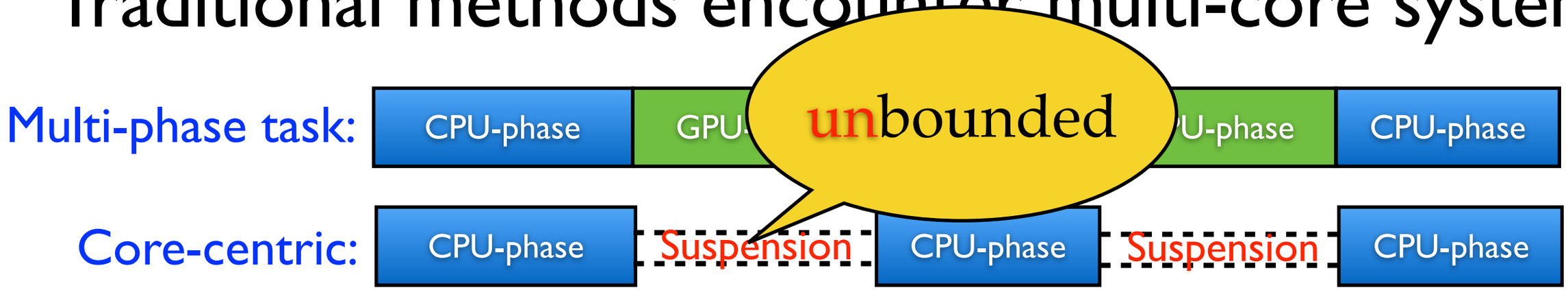
GPU (accelerator)

[1] Zheng Dong and Cong Liu. **Closing the loop for the selective conversion approach: a utilization-based test for hard real-time suspending task systems**, *Proceedings of the 37th IEEE Real-Time Systems Symposium*, RTSS 2016.

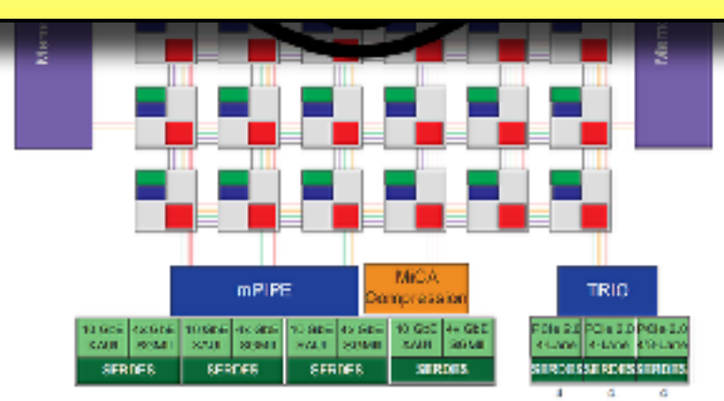
[2] Jianjia Chen, Wenhung Huang, and Cong Liu. **K2U: A General Framework from k-Point Effective Schedulabiliy Analysis to Utilization-based Tests**, *Proceedings of the 36th IEEE Real-Time Systems Symposium*, RTSS 2015.

[3] Cong Liu and James Anderson. **An O(m) Analysis Technique for Supporting Real-Time Self-Suspending Task Systems**, *Proceedings of the 33th IEEE Real-Time Systems Symposium, RTSS 2012*

Traditional methods encounter multi-core system



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

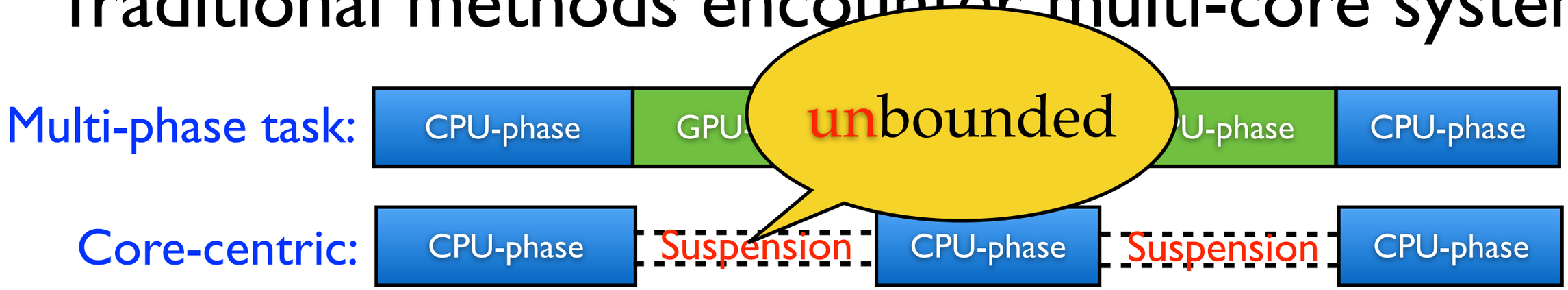
GPU (accelerator)

[1] Zheng Dong and Cong Liu. **Closing the loop for the selective conversion approach: a utilization-based test for hard real-time suspending task systems**, *Proceedings of the 37th IEEE Real-Time Systems Symposium*, RTSS 2016.

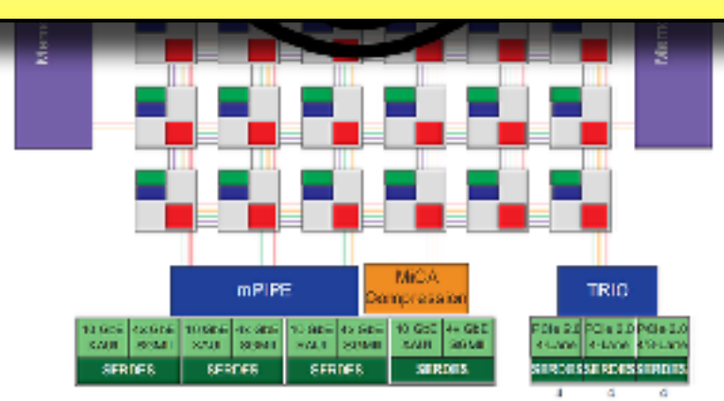
[2] Jianjia Chen, Wenhung Huang, and Cong Liu. **K2U: A General Framework from k-Point Effective Schedulabiliy Analysis to Utilization-based Tests**, *Proceedings of the 36th IEEE Real-Time Systems Symposium*, RTSS 2015.

[3] Cong Liu and James Anderson. **An O(m) Analysis Technique for Supporting Real-Time Self-Suspending Task Systems**, *Proceedings of the 33th IEEE Real-Time Systems Symposium, RTSS 2012*

Traditional methods encounter multi-core system



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

GPU (accelerator)

[1] Zheng Dong and Cong Liu. **Closing the loop for the selective conversion approach: a utilization-based test for hard real-time suspending task systems**, *Proceedings of the 37th IEEE Real-Time Systems Symposium*, RTSS 2016.

[2] Jianjia Chen, Wenhung Huang, and Cong Liu. **K2U: A General Framework from k-Point Effective Schedulabiliy Analysis to Utilization-based Tests**, *Proceedings of the 36th IEEE Real-Time Systems Symposium*, RTSS 2015.

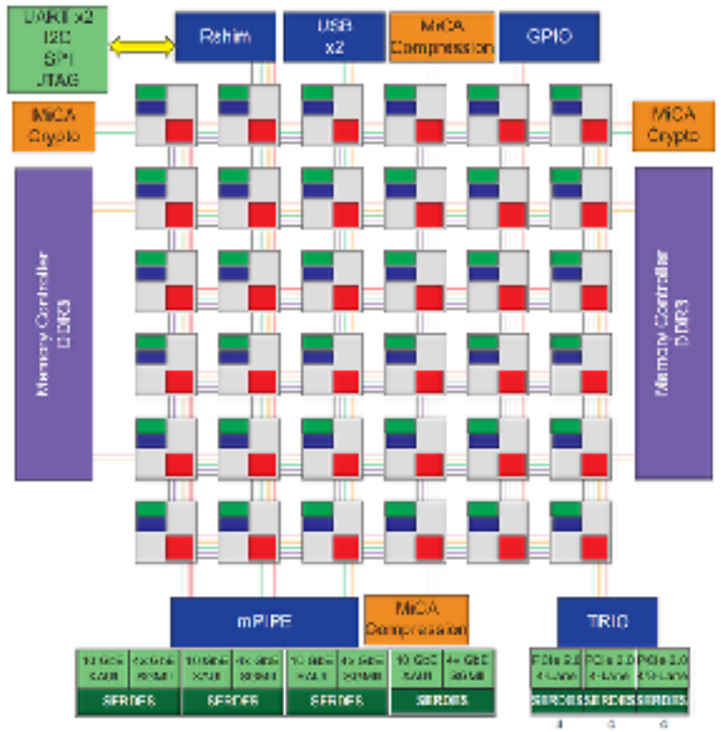
[3] Cong Liu and James Anderson. **An O(m) Analysis Technique for Supporting Real-Time Self-Suspending Task Systems**, *Proceedings of the 33th IEEE Real-Time Systems Symposium, RTSS 2012*

Not Applicable

A counter-intuitive shared-resource-centric perspective

Multi-phase task:

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



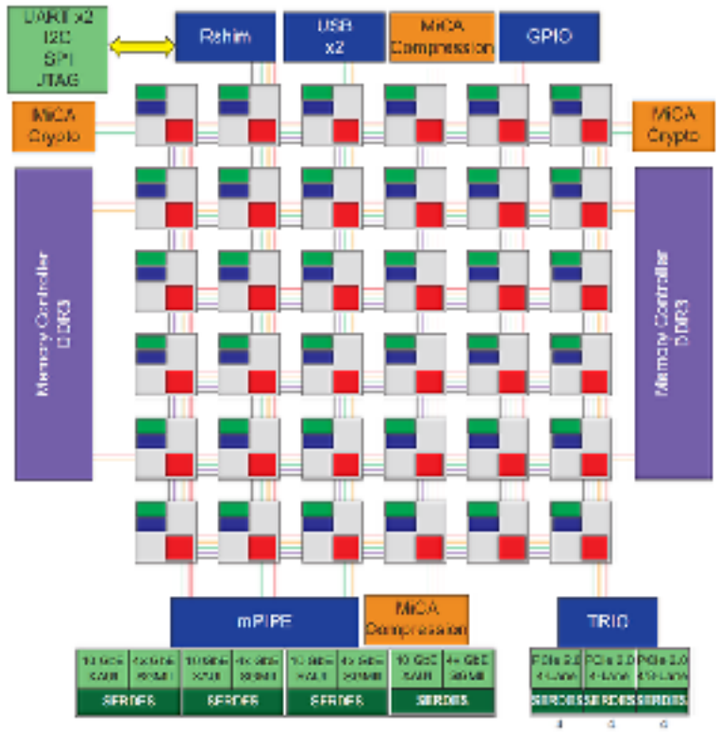
100 cores on chip

GPU (accelerator)

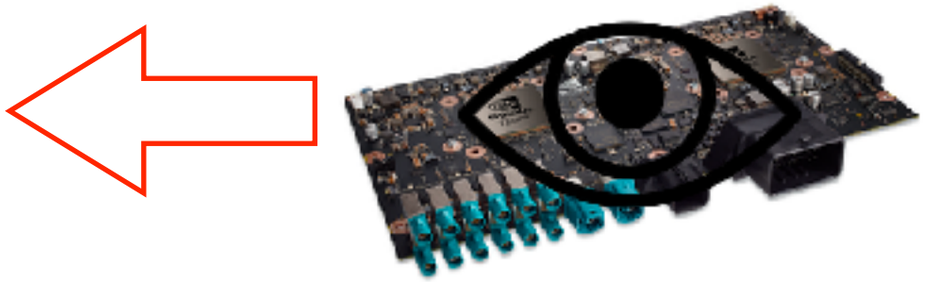
A counter-intuitive shared-resource-centric perspective

Multi-phase task:

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



100 cores on chip

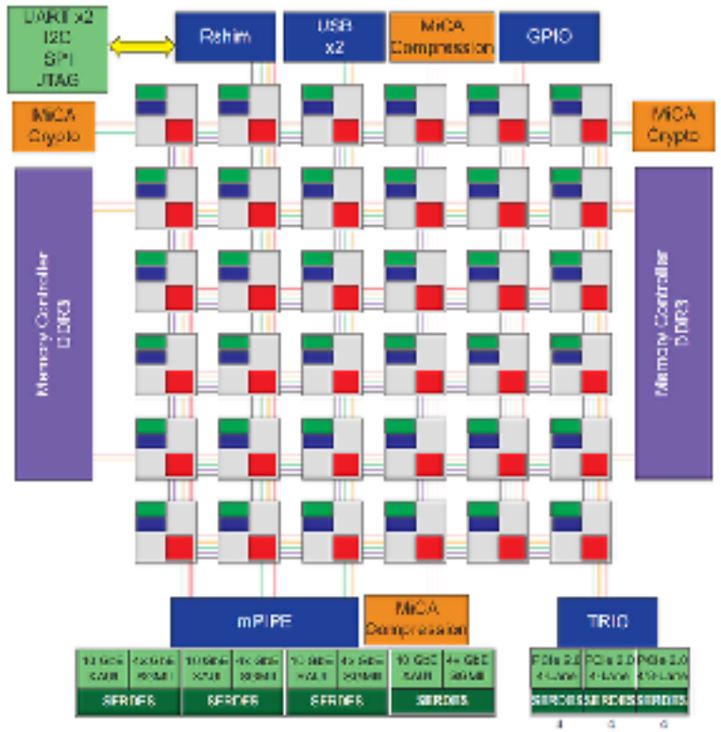


GPU (accelerator)

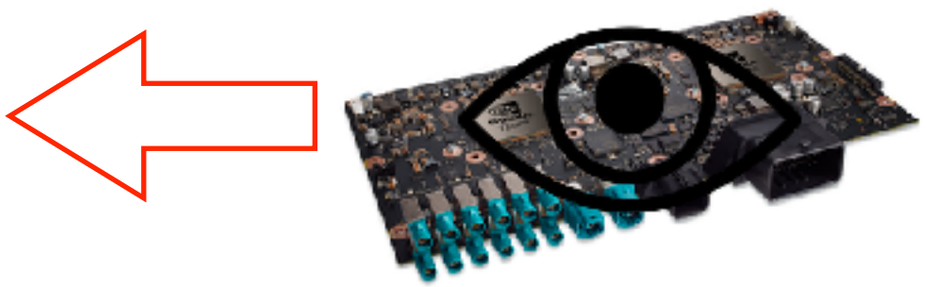
A counter-intuitive shared-resource-centric perspective

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

Shared-resource-centric: Suspension | GPU-phase | Suspension | GPU-phase | Suspension

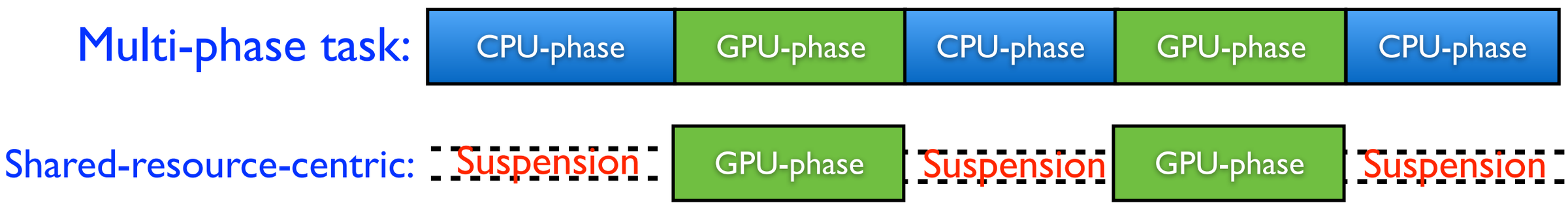


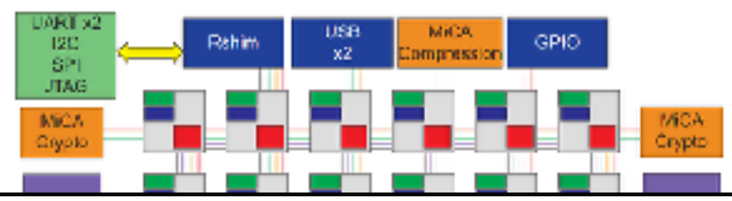
100 cores on chip



GPU (accelerator)

A counter-intuitive shared-resource-centric perspective



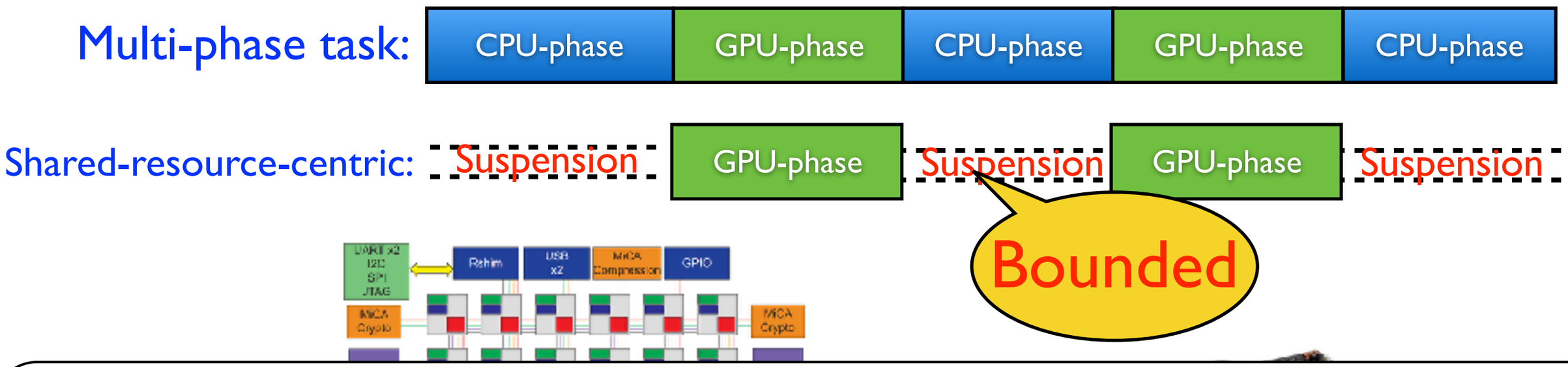


The idea is to judiciously scheduling shared-resource requests (treating shared-resource as the first class unit) and bounding the worst case latency a task may experience on the computing cores (treating cores as "I/O").

100 cores on chip

GPU (accelerator)

A counter-intuitive shared-resource-centric perspective

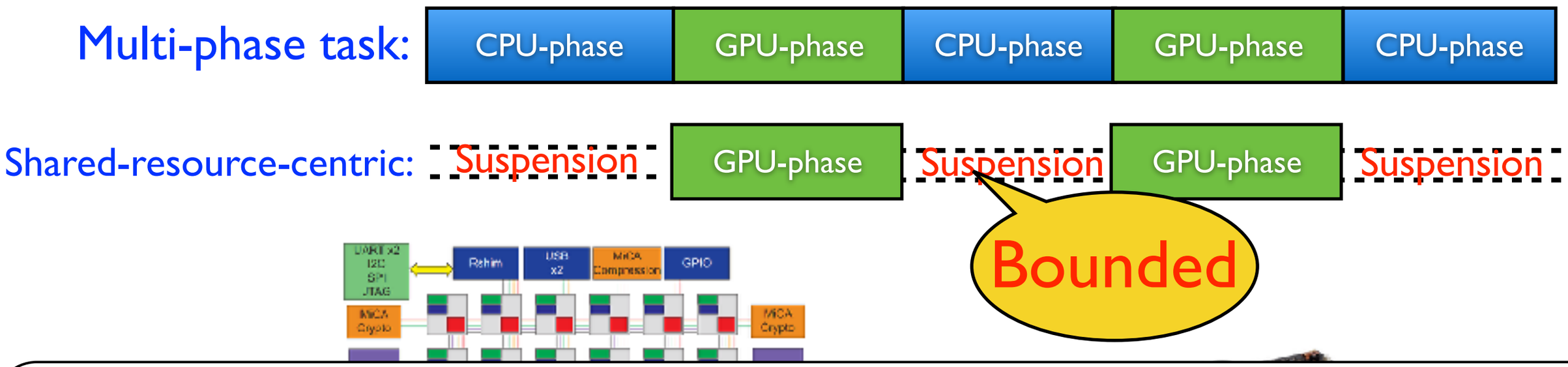


The idea is to judiciously scheduling shared-resource requests (treating shared-resource as the first class unit) and bounding the worst case latency a task may experience on the computing cores (treating cores as "I/O").

100 cores on chip

GPU (accelerator)

A counter-intuitive shared-resource-centric perspective



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").

GPU (accelerator)

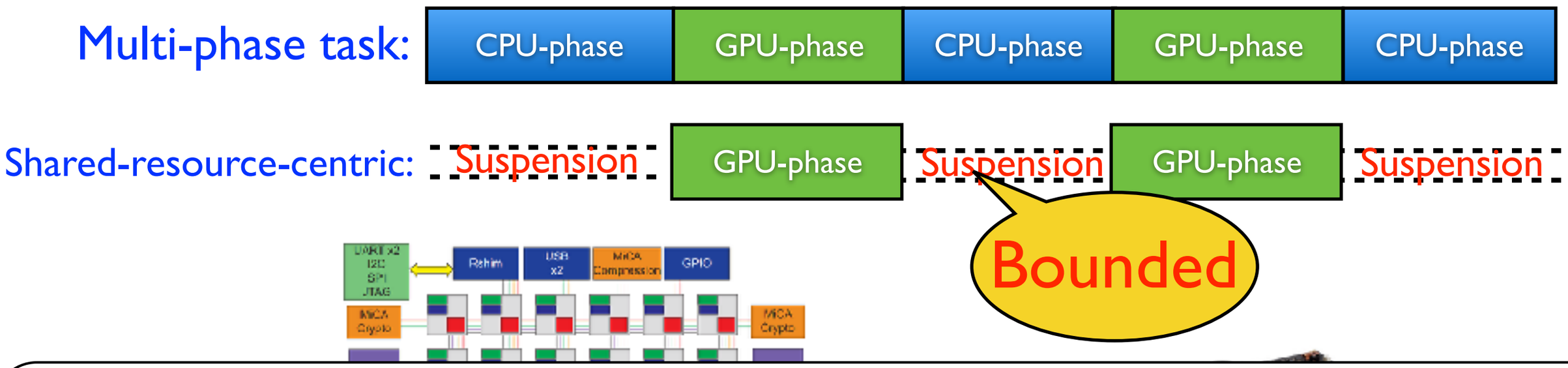
100 cores on chip

[1] Zheng Dong and Cong Liu. **Closing the loop for the selective conversion approach: a utilization-based test for hard real-time suspending task systems**, *Proceedings of the 37th IEEE Real-Time Systems Symposium*, RTSS 2016.

[2] Jianjia Chen, Wenhung Huang, and Cong Liu. **K2U: A General Framework from k-Point Effective Schedulabiliy Analysis to Utilization-based Tests**, *Proceedings of the 36th IEEE Real-Time Systems Symposium*, RTSS 2015.

[3] Cong Liu and James Anderson. **An O(m) Analysis Technique for Supporting Real-Time Self-Suspending Task Systems**, *Proceedings of the 33th IEEE Real-Time Systems Symposium, RTSS 2012*

A counter-intuitive shared-resource-centric perspective



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

GPU (accelerator)

[1] Zheng Dong and Cong Liu. **Closing the loop for the selective conversion approach: a utilization-based test for hard real-time suspending task systems**, *Proceedings of the 37th IEEE Real-Time Systems Symposium*, RTSS 2016.

[2] Jianjia Chen, Wenhung Huang, and Cong Liu. **K2U: A General Framework from k-Point Effective Schedulabiliy Analysis to Utilization-based Tests**, *Proceedings of the 36th IEEE Real-Time Systems Symposium*, RTSS 2015.

[3] Cong Liu and James Anderson. **An O(m) Analysis Technique for Supporting Real-Time Self-Suspending Task Systems**, *Proceedings of the 33th IEEE Real-Time Systems Symposium, RTSS 2012*

Not Applicable

Challenges in shared-resource-centric scheduling

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

[1] Zheng Dong and Cong Liu. **Closing the loop for the selective conversion approach: a utilization-based test for hard real-time suspending task systems**, *Proceedings of the 37th IEEE Real-Time Systems Symposium*, RTSS 2016.

[2] Jianjia Chen, Wenhung Huang, and Cong Liu. **K2U: A General Framework from k-Point Effective Schedulabiliy Analysis to Utilization-based Tests**, *Proceedings of the 36th IEEE Real-Time Systems Symposium*, RTSS 2015.

[3] Cong Liu and James Anderson. **An O(m) Analysis Technique for Supporting Real-Time Self-Suspending Task Systems**, *Proceedings of the 33th IEEE Real-Time Systems Symposium, RTSS 2012*

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

[1] Zheng Dong and Cong Liu. **Closing the loop for the selective conversion approach: a utilization-based test for hard real-time suspending task systems**, *Proceedings of the 37th IEEE Real-Time Systems Symposium*, RTSS 2016.

[2] Jianjia Chen, Wenhung Huang, and Cong Liu. **K2U: A General Framework from k-Point Effective Schedulabiliy Analysis to Utilization-based Tests**, *Proceedings of the 36th IEEE Real-Time Systems Symposium*, RTSS 2015.

[3] Cong Liu and James Anderson. **An O(m) Analysis Technique for Supporting Real-Time Self-Suspending Task Systems**, *Proceedings of the 33th IEEE Real-Time Systems Symposium, RTSS 2012*

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

[1] Zheng Dong and Cong Liu. **Closing the loop for the selective conversion approach: a utilization-based test for hard real-time suspending task systems**, *Proceedings of the 37th IEEE Real-Time Systems Symposium*, RTSS 2016.

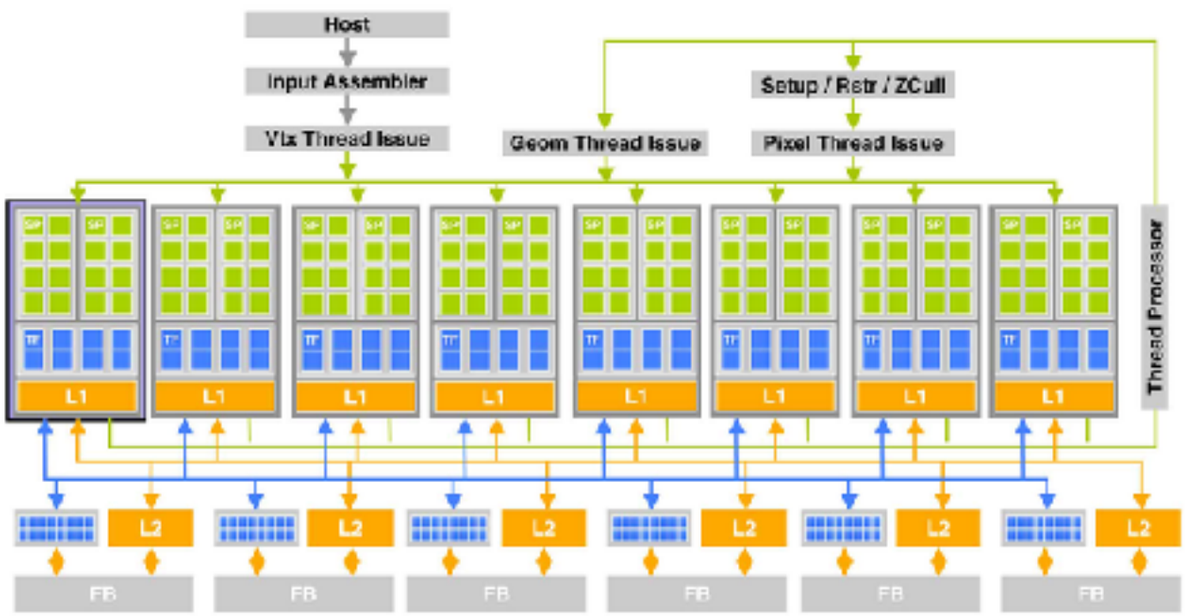
[2] Jianjia Chen, Wenhung Huang, and Cong Liu. **K2U: A General Framework from k-Point Effective Schedulabiliy Analysis to Utilization-based Tests**, *Proceedings of the 36th IEEE Real-Time Systems Symposium*, RTSS 2015.

[3] Cong Liu and James Anderson. **An O(m) Analysis Technique for Supporting Real-Time Self-Suspending Task Systems**, *Proceedings of the 33th IEEE Real-Time Systems Symposium, RTSS 2012*

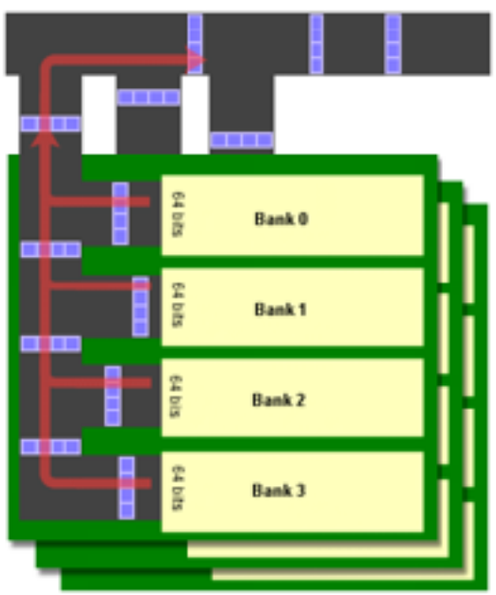
Not Applicable

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



GPU with parallel programmable units at their cores.



Memory Banking

[1] Zheng Dong and Cong Liu. **Closing the loop for the selective conversion approach: a utilization-based test for hard real-time suspending task systems**, *Proceedings of the 37th IEEE Real-Time Systems Symposium*, RTSS 2016.

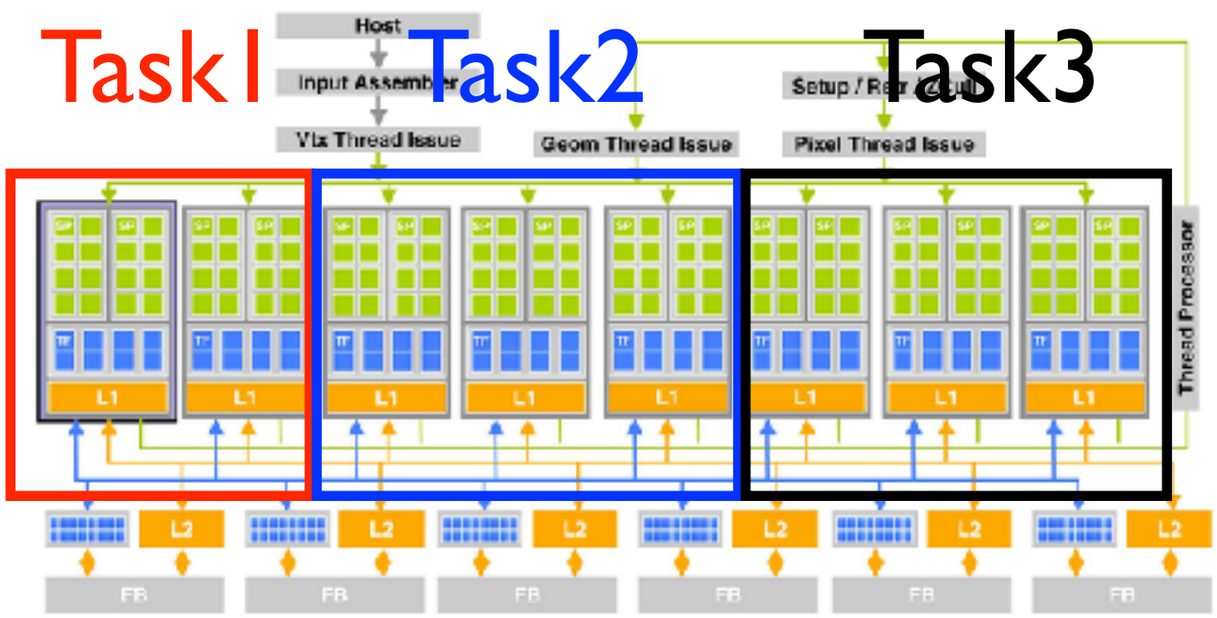
[2] Jianjia Chen, Wenhung Huang, and Cong Liu. **K2U: A General Framework from k-Point Effective Schedulabiliy Analysis to Utilization-based Tests**, *Proceedings of the 36th IEEE Real-Time Systems Symposium*, RTSS 2015.

[3] Cong Liu and James Anderson. **An O(m) Analysis Technique for Supporting Real-Time Self-Suspending Task Systems**, *Proceedings of the 33th IEEE Real-Time Systems Symposium, RTSS 2012*

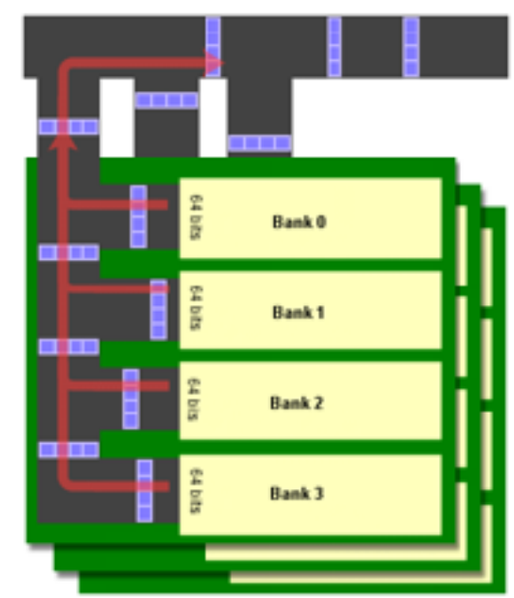
Not Applicable

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



GPU with parallel programmable units at their cores.



Memory Banking

[1] Zheng Dong and Cong Liu. **Closing the loop for the selective conversion approach: a utilization-based test for hard real-time suspending task systems**, *Proceedings of the 37th IEEE Real-Time Systems Symposium*, RTSS 2016.

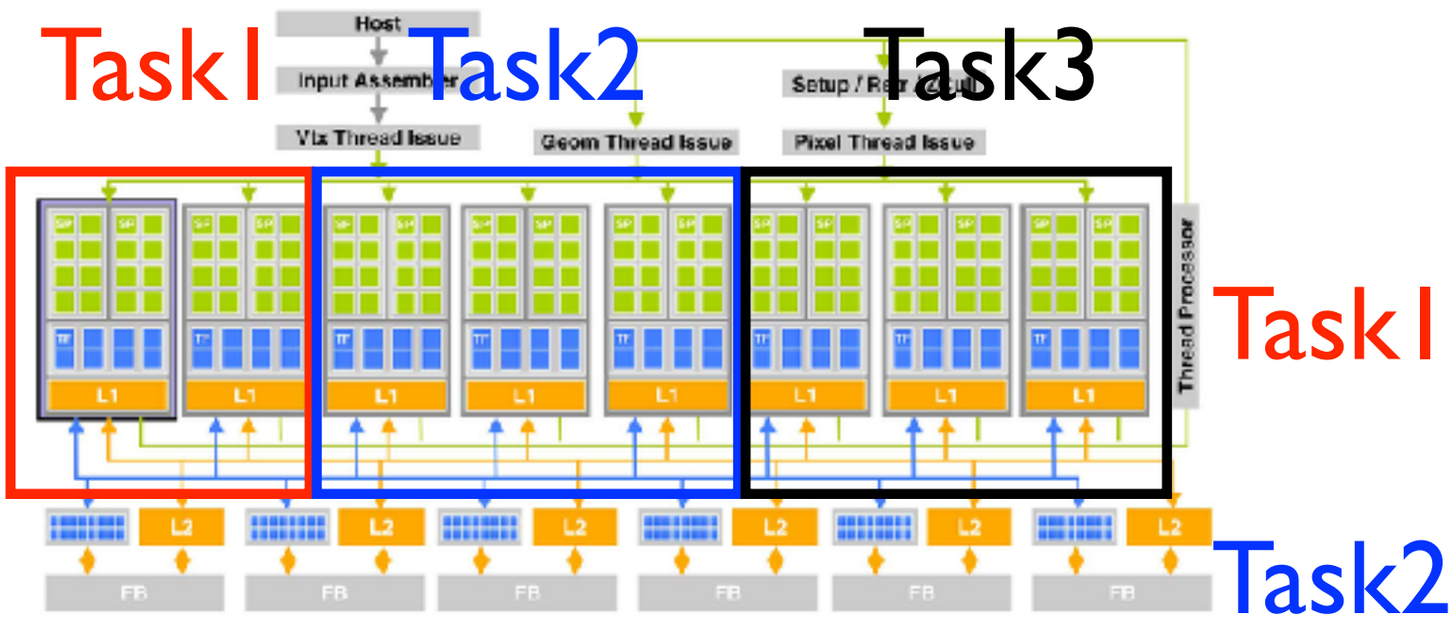
[2] Jianjia Chen, Wenhung Huang, and Cong Liu. **K2U: A General Framework from k-Point Effective Schedulabiliy Analysis to Utilization-based Tests**, *Proceedings of the 36th IEEE Real-Time Systems Symposium*, RTSS 2015.

[3] Cong Liu and James Anderson. **An O(m) Analysis Technique for Supporting Real-Time Self-Suspending Task Systems**, *Proceedings of the 33th IEEE Real-Time Systems Symposium, RTSS 2012*

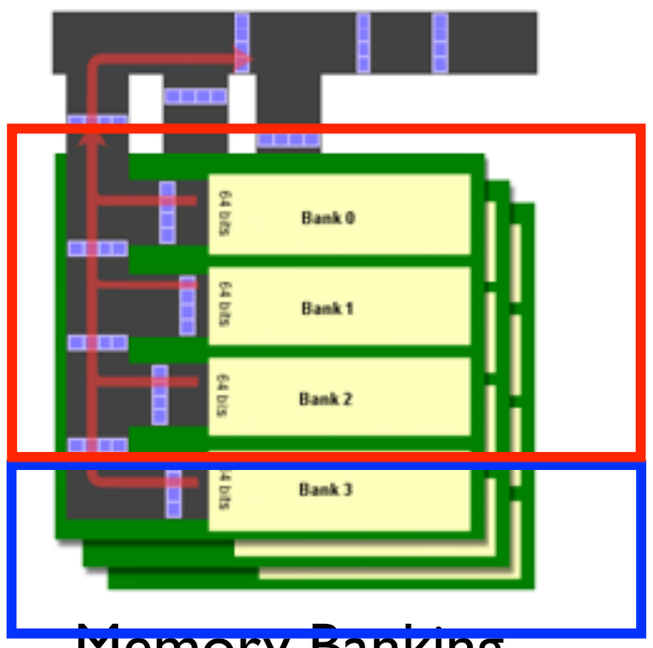
Not Applicable

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



GPU with parallel programmable units at their cores.



Memory Banking

[1] Zheng Dong and Cong Liu. **Closing the loop for the selective conversion approach: a utilization-based test for hard real-time suspending task systems**, *Proceedings of the 37th IEEE Real-Time Systems Symposium*, RTSS 2016.

[2] Jianjia Chen, Wenhung Huang, and Cong Liu. **K2U: A General Framework from k-Point Effective Schedulabiliy Analysis to Utilization-based Tests**, *Proceedings of the 36th IEEE Real-Time Systems Symposium*, RTSS 2015.

[3] Cong Liu and James Anderson. **An O(m) Analysis Technique for Supporting Real-Time Self-Suspending Task Systems**, *Proceedings of the 33th IEEE Real-Time Systems Symposium, RTSS 2012*

Not Applicable

Objectives and solution overview

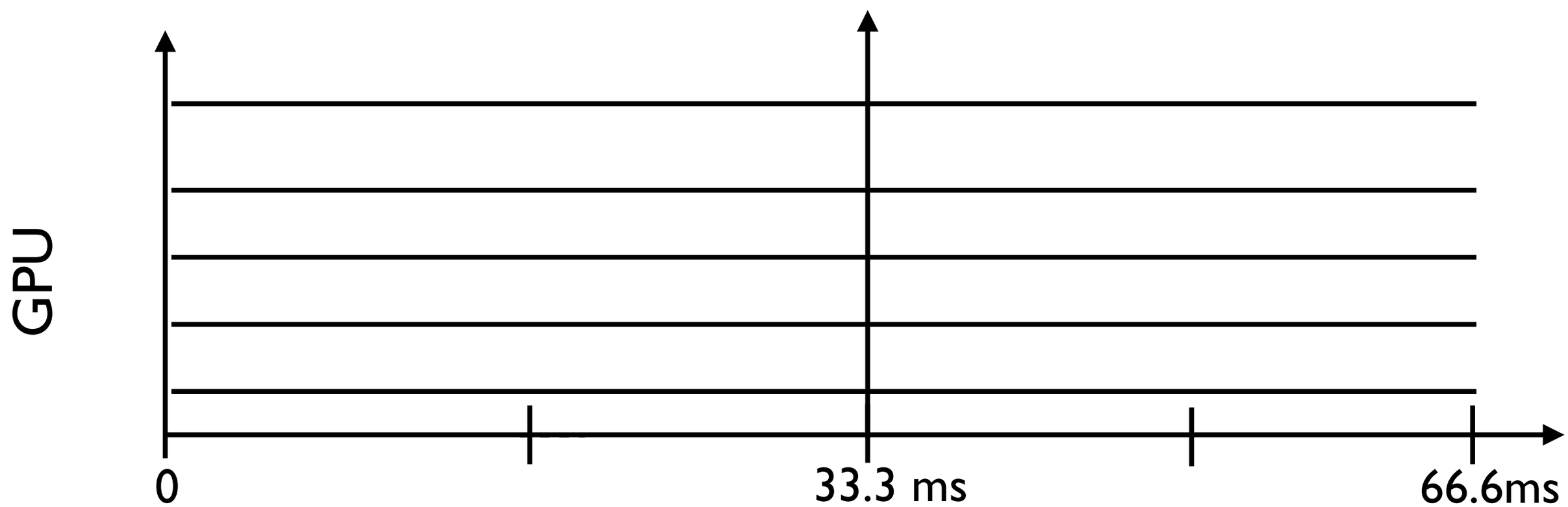
- Our goals is two-fold:
  - Schedulability tests for shared-resource-centric limited preemptive suspending task model.
  - Minimizing the number of units of required shared resource.

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.

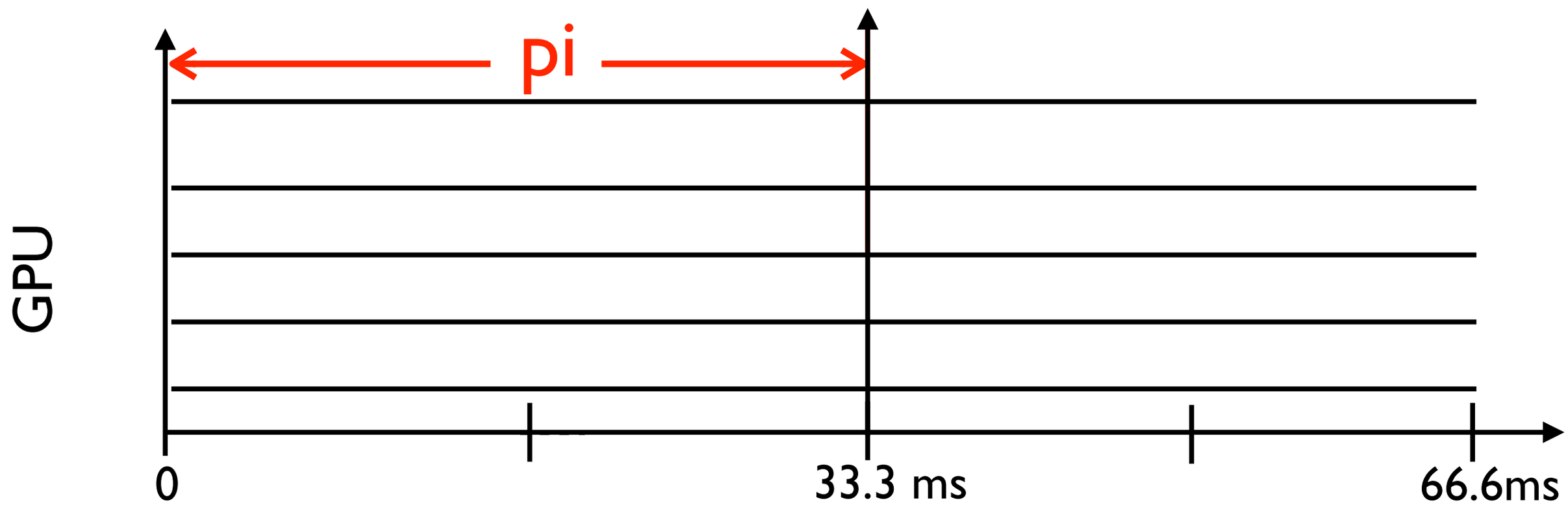
Shared-resource-centric suspending task model

- Implicit-deadline sporadic multi-phase task: τi(ei, si, pi, di, σ_i, zi)



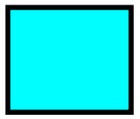
Shared-resource-centric suspending task model

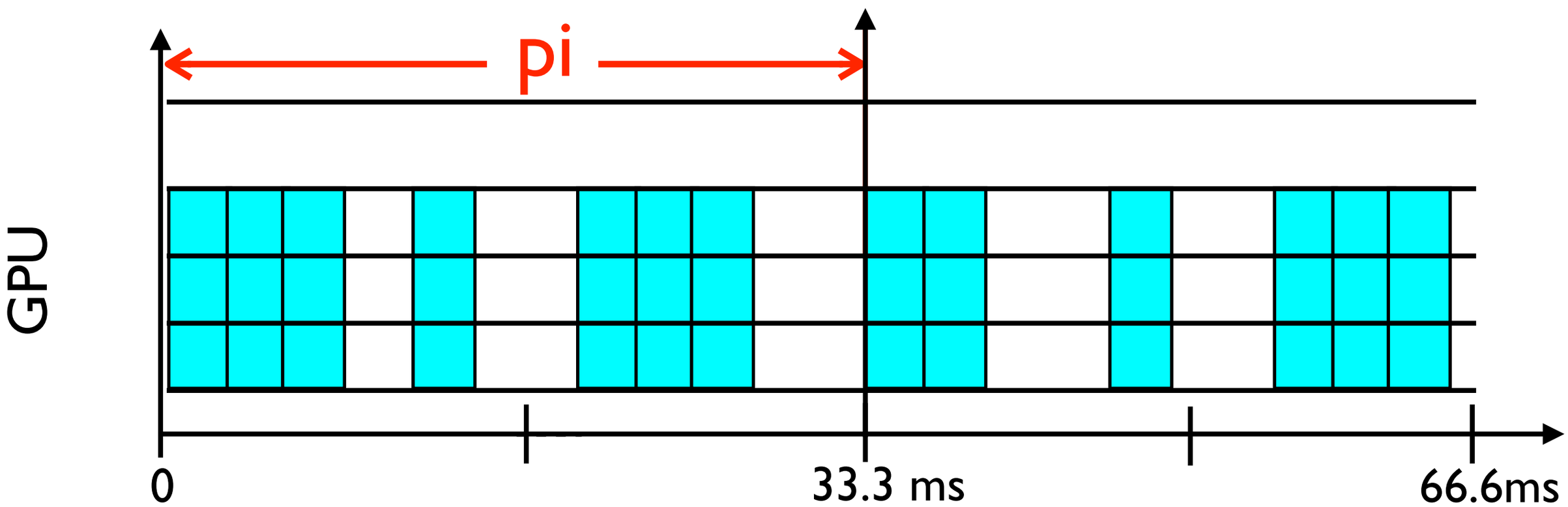
- Implicit-deadline sporadic multi-phase task: τi(ei, si, pi, di, σ_i, zi)



Shared-resource-centric suspending task model

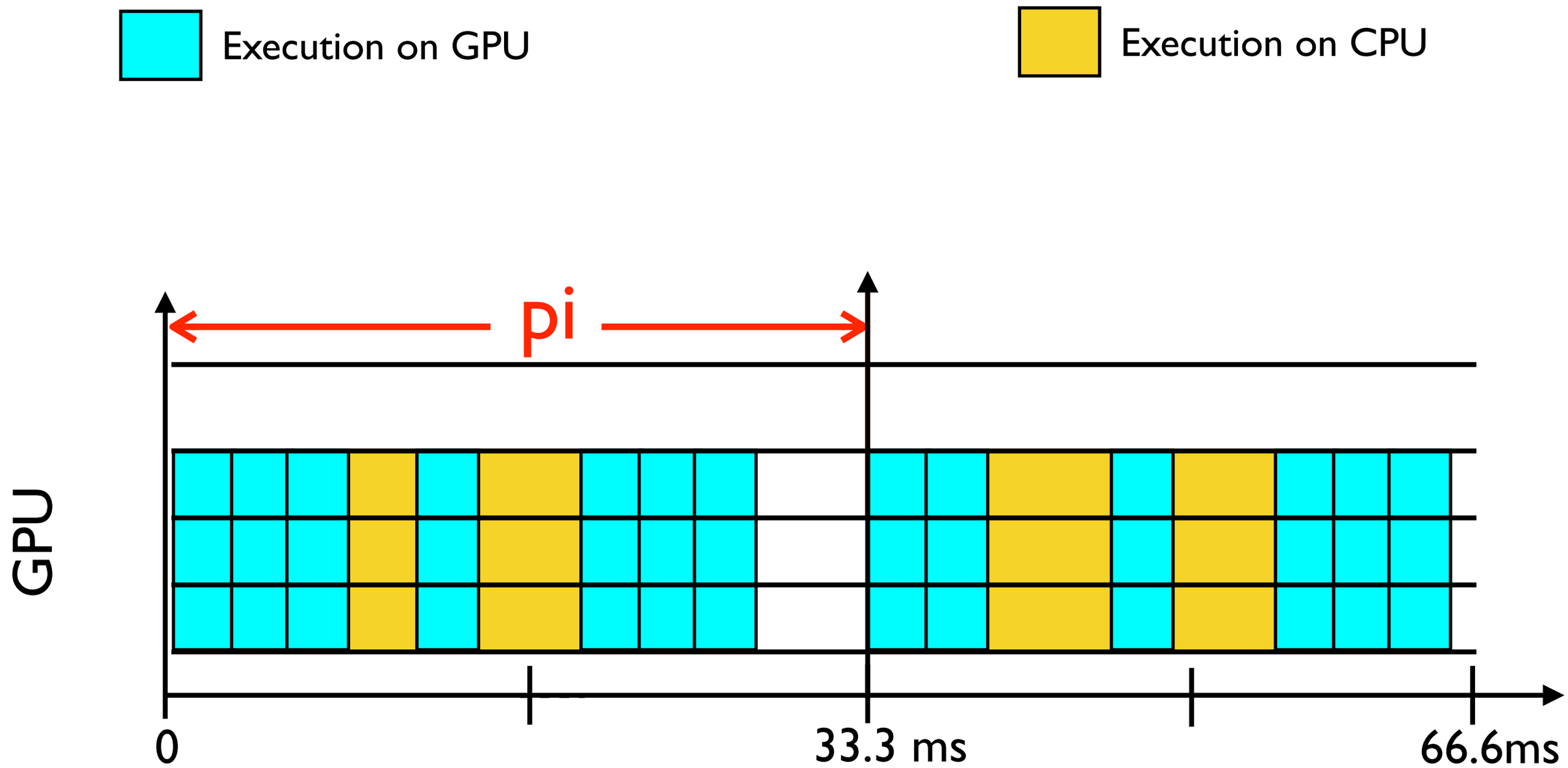
- Implicit-deadline sporadic multi-phase task: τi(ei, si, pi, di, σ_i, zi)


Execution on GPU



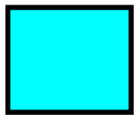
Shared-resource-centric suspending task model

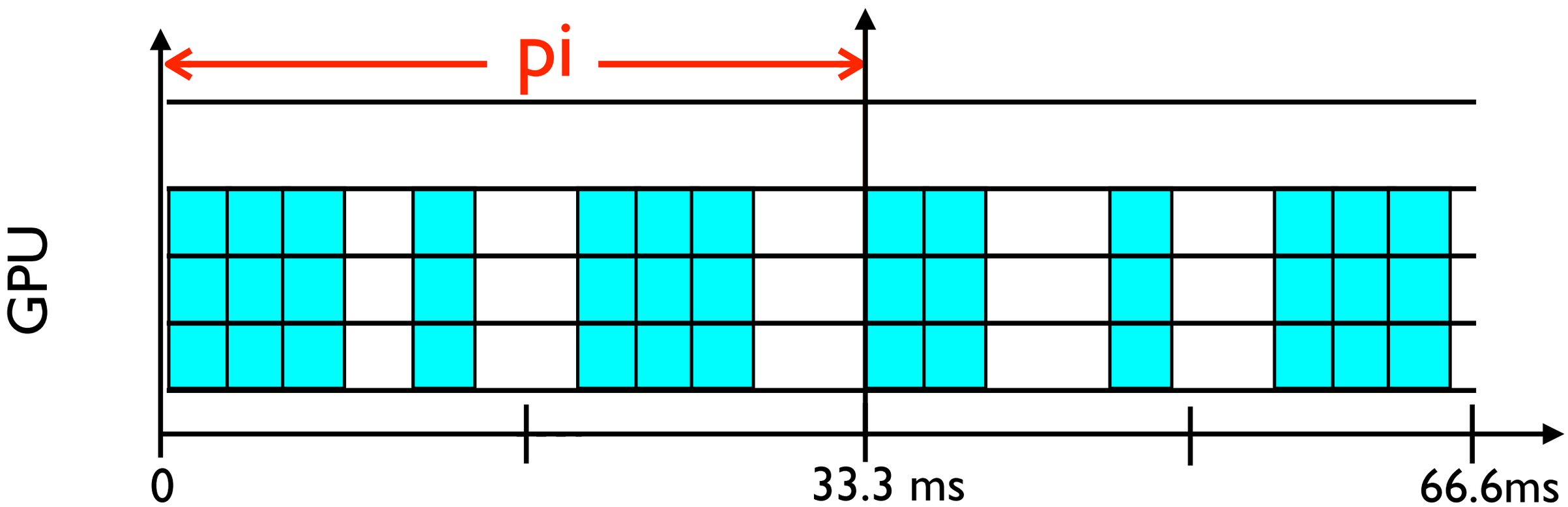
- Implicit-deadline sporadic multi-phase task: τi(ei, si, pi, di, σ_i, zi)



Shared-resource-centric suspending task model

- Implicit-deadline sporadic multi-phase task: τi(ei, si, pi, di, σ_i, zi)


Execution on GPU

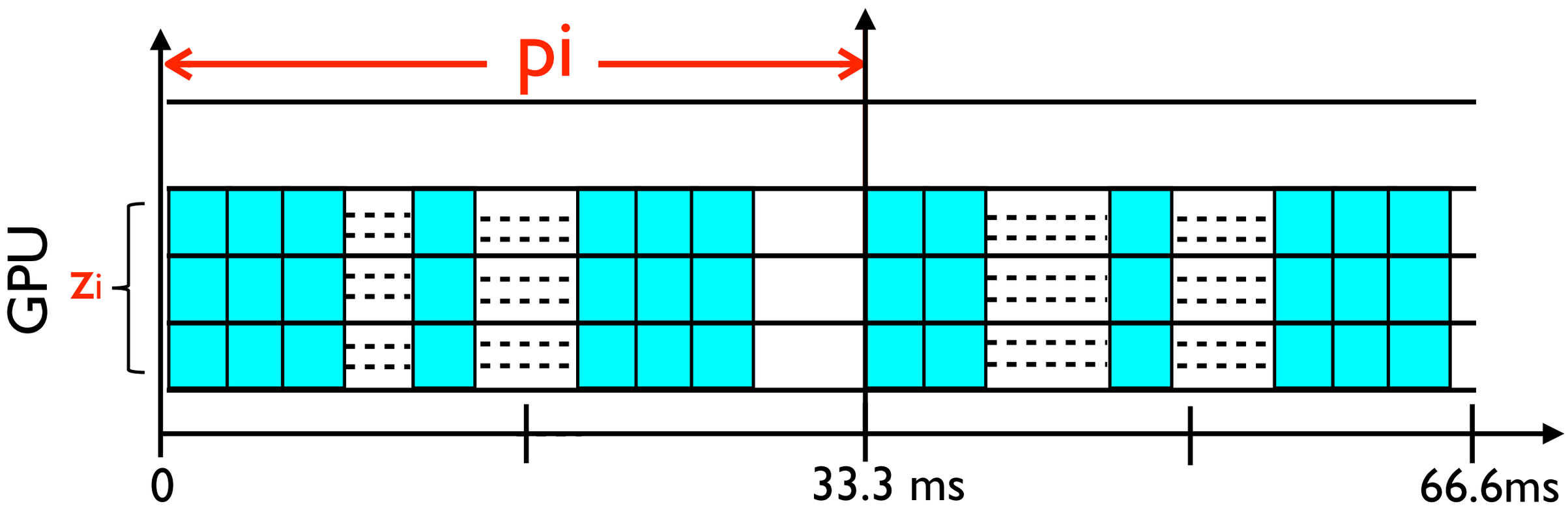


Shared-resource-centric suspending task model

- Implicit-deadline sporadic multi-phase task: τi(ei, si, pi, di, σ_i, zi)

Execution on GPU

Suspensions from shared-resource-centric perspective

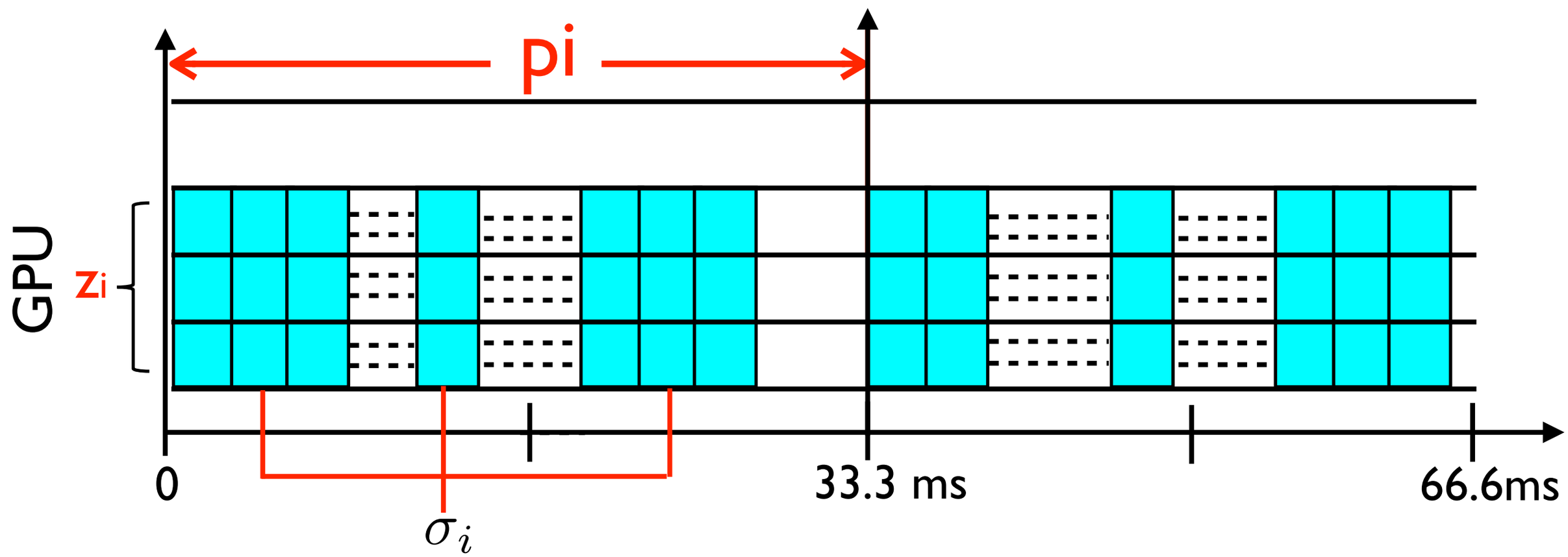


Shared-resource-centric suspending task model

- Implicit-deadline sporadic multi-phase task: τi(ei, si, pi, di, σ_i, zi)

Execution on GPU

Suspensions from shared-resource-centric perspective



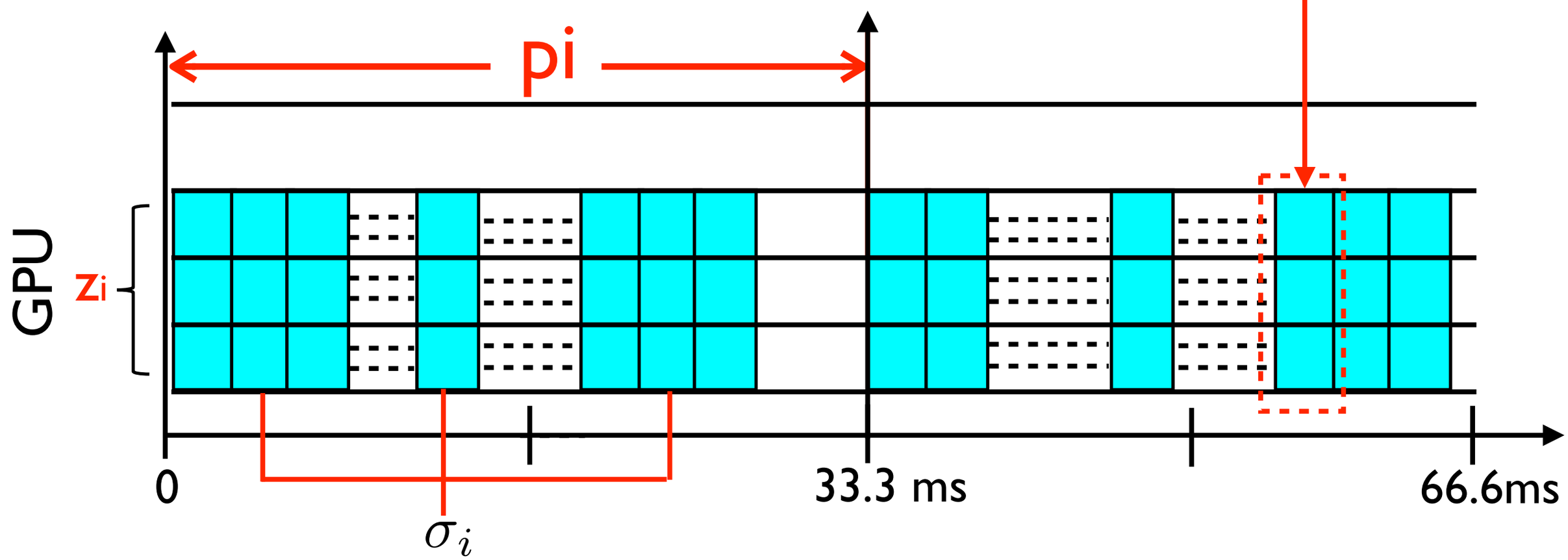
Shared-resource-centric suspending task model

- Implicit-deadline sporadic multi-phase task: τi(ei, si, pi, di, σ_i, zi)

Execution on GPU

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

Suspensions from shared-resource-centric perspective



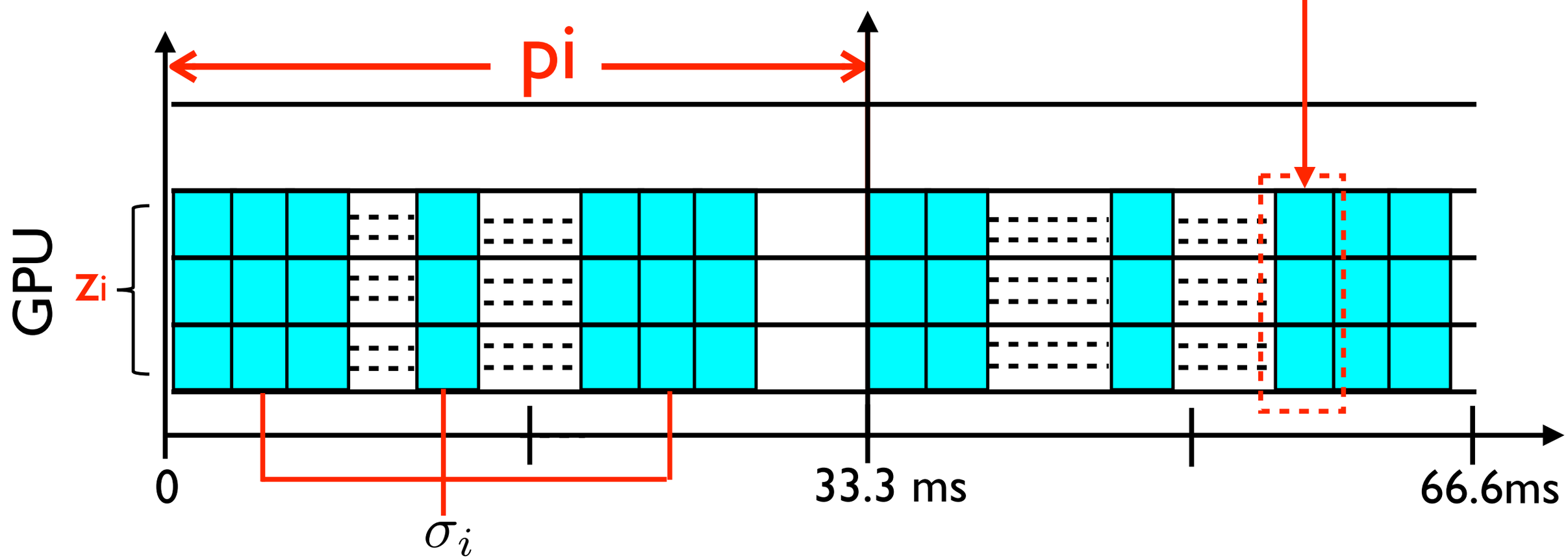
Shared-resource-centric suspending task model

- Implicit-deadline sporadic multi-phase task: τi(ei, si, pi, di, σ_i, zi)

Execution on GPU

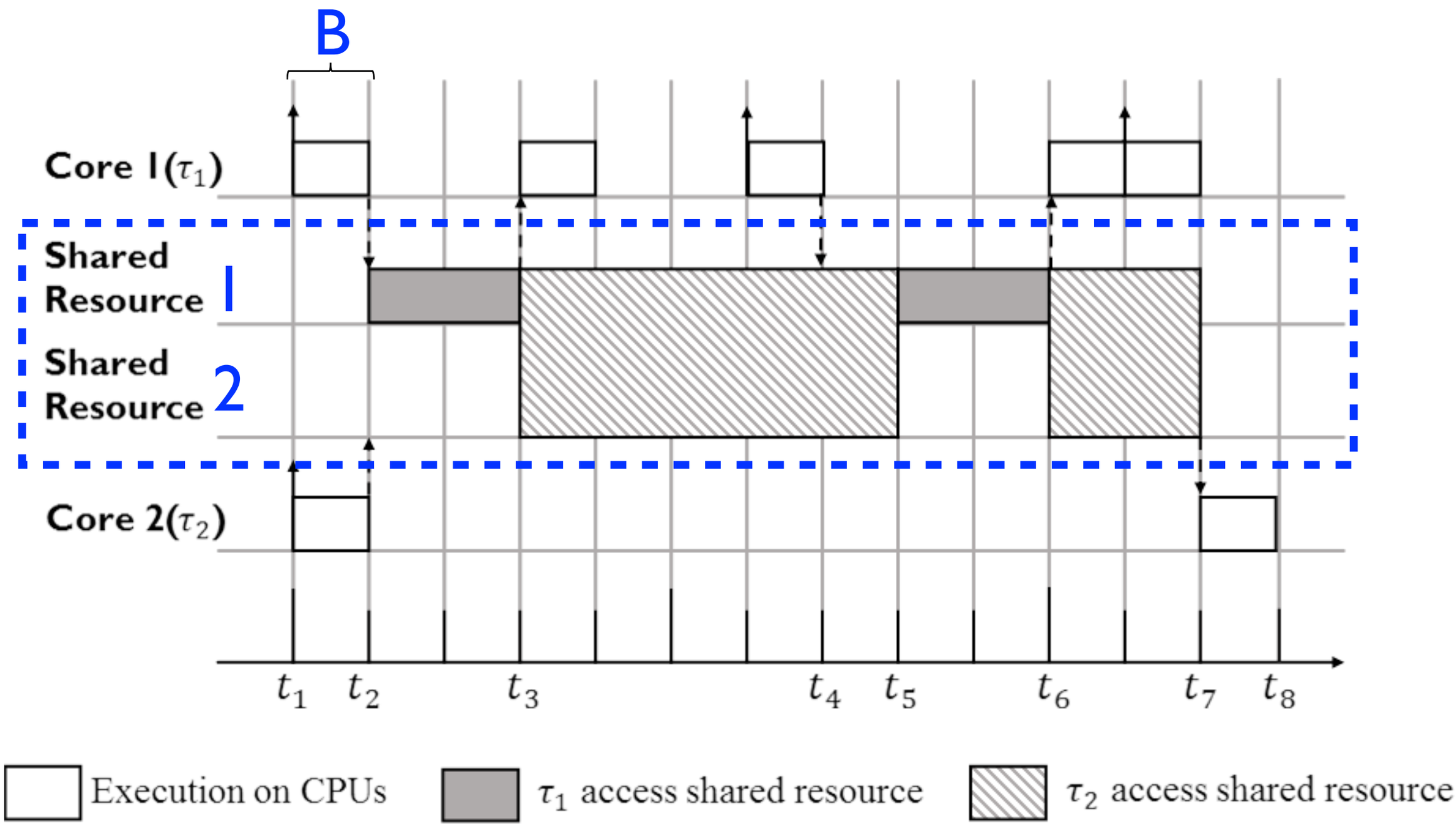
Each operation on the shared resource is atomic, with a processing time upper - bounded by B

Suspensions from shared-resource-centric perspective

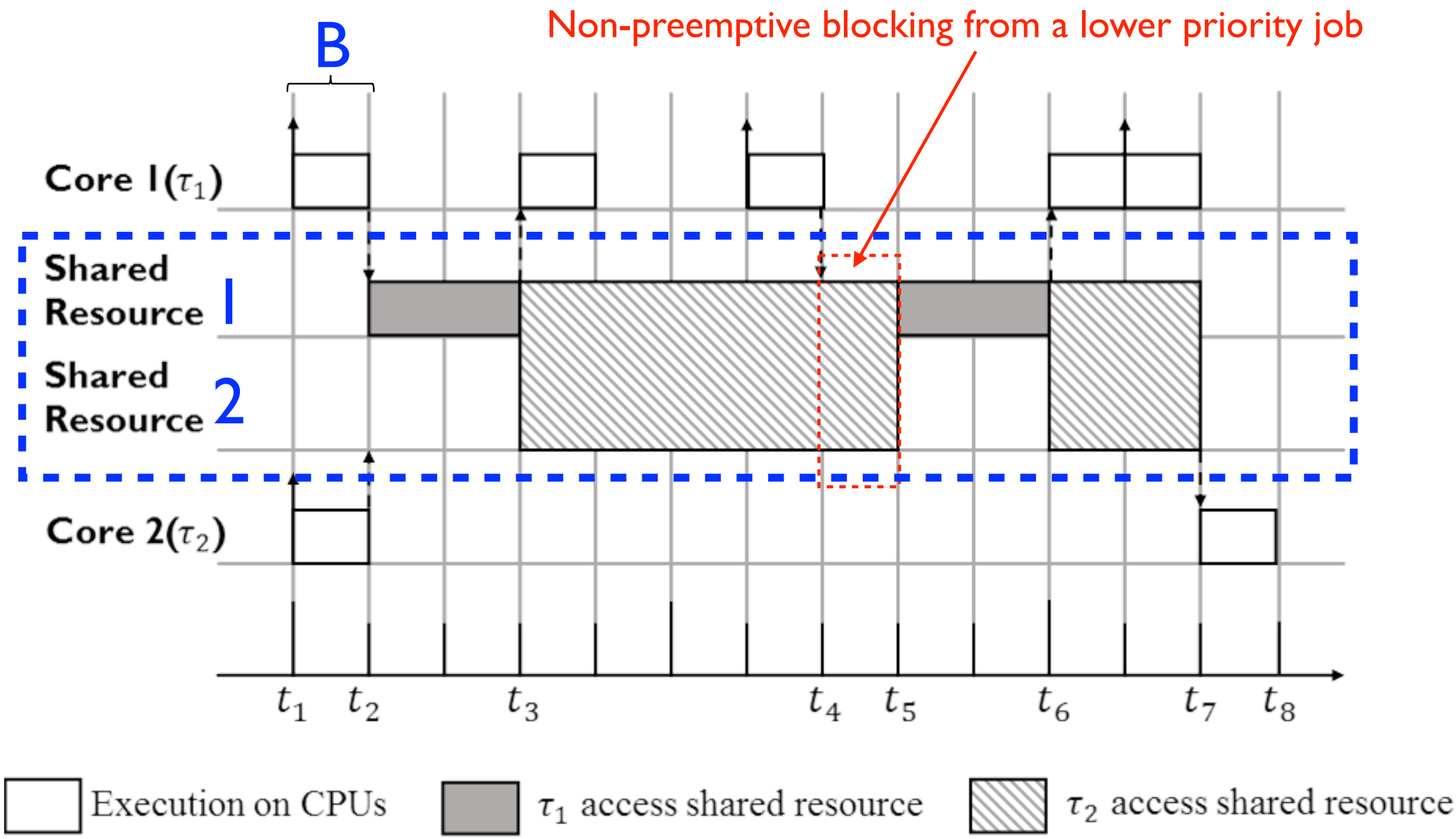


Rate-monotonic (RM) Scheduling on the shared resource.

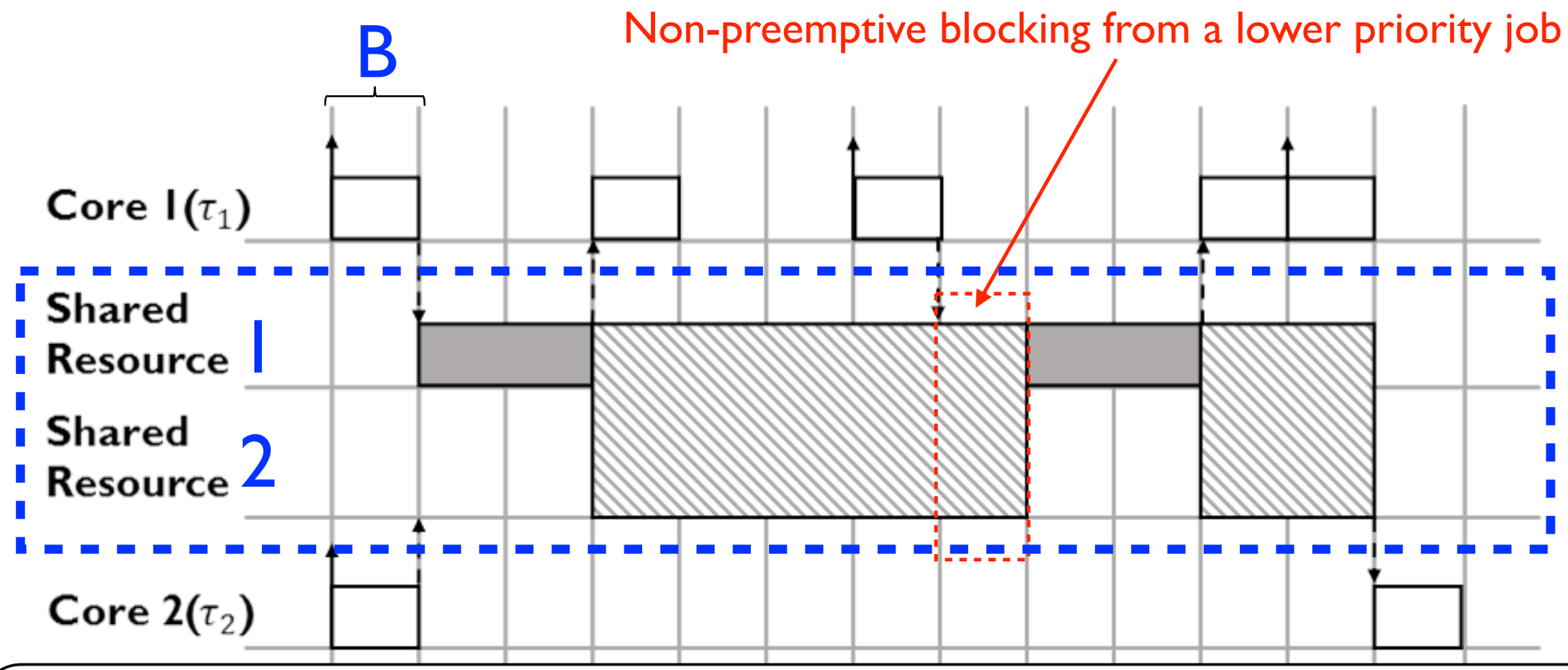
Step 1: Developing schedulability tests for Limited preemptive suspending task scheduling



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



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



Observation

The maximum blocking time experienced by any job released by τ_i due to non-preemptive blocking on shared resource is at most σ_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).

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

Lemma 2 (Suspension as Jitter).

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

[2] Jianjia Chen, Wenhung Huang, and Cong Liu. **K2U: A General Framework from k-Point Effective Schedulabiliy Analysis to Utilization-based Tests**, *Proceedings of the 36th IEEE Real-Time Systems Symposium*, RTSS 2015.

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).

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

Lemma 2 (Suspension as Jitter).

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

[2] Jianjia Chen, Wenhung Huang, and Cong Liu. **K2U: A General Framework from k-Point Effective Schedulabiliy Analysis to Utilization-based Tests**, *Proceedings of the 36th IEEE Real-Time Systems Symposium*, RTSS 2015.

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).

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

Lemma 2 (Suspension as Jitter).

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

Observation

The maximum blocking time experienced by any job released by τ_i is due to non-preemptive blocking on shared resource is at most σ_i times B.

[2] Jianjia Chen, Wenhung Huang, and Cong Liu. **K2U: A General Framework from k-Point Effective Schedulabiliy Analysis to Utilization-based Tests**, *Proceedings of the 36th IEEE Real-Time Systems Symposium*, RTSS 2015.

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).

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

Lemma 2 (Suspension as Jitter).

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

Time-demand Analysis(TDA)

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

$$\exists t \mid 0 < t \leq p_k, \quad s_k + e_k + \sigma_k \times B + \sum_{\tau_i \in hp(k)} \left(\left\lceil \frac{t}{p_i} \right\rceil + 1 \right) s_i \leq t \tag{3}$$

Lemma 4 (Suspension and Blocking as Jitter).

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

Observation

The maximum blocking time experienced by any job released by τ_i is due to non-preemptive blocking on shared resource is at most σ_i times B.

[2] Jianjia Chen, Wenhung Huang, and Cong Liu. **K2U: A General Framework from k-Point Effective Schedulabiliy Analysis to Utilization-based Tests**, *Proceedings of the 36th IEEE Real-Time Systems Symposium*, RTSS 2015.

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).

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

Lemma 2 (Suspension as Jitter).

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

Time-demand Analysis(TDA)

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

$$\exists t \mid 0 < t \leq p_k, \quad s_k + e_k + \boxed{\sigma_k \times B} + \sum_{\tau_i \in hp(k)} \left(\left\lceil \frac{t}{p_i} \right\rceil + 1 \right) s_i \leq t \tag{3}$$

Lemma 4 (Suspension and Blocking as Jitter).

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

Observation

The maximum blocking time experienced by any job released by τ_i is due to non-preemptive blocking on shared resource is at most σ_i times B.

[2] Jianjia Chen, Wenhung Huang, and Cong Liu. **K2U: A General Framework from k-Point Effective Schedulabiliy Analysis to Utilization-based Tests**, *Proceedings of the 36th IEEE Real-Time Systems Symposium*, RTSS 2015.

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).

$$\exists t \mid 0 < t \leq p_k, \quad s_k + e_k + \sigma_k \times B + \sum_{\tau_i \in hp(k)} \left(\left\lceil \frac{t}{p_i} \right\rceil + 1 \right) s_i \leq t \tag{3}$$

Lemma 4 (Suspension and Blocking as Jitter).

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

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).

$$\exists t \mid 0 < t \leq p_k, \quad s_k + e_k + \sigma_k \times B + \sum_{\tau_i \in hp(k)} \left(\left\lceil \frac{t}{p_i} \right\rceil + 1 \right) s_i \leq t \tag{3}$$

Lemma 4 (Suspension and Blocking as Jitter).

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

- Time-demand Analysis(TDA)
  - 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).

$$\exists t \mid 0 < t \leq p_k, \quad s_k + e_k + \sigma_k \times B + \sum_{\tau_i \in hp(k)} \left(\left\lceil \frac{t}{p_i} \right\rceil + 1 \right) s_i \leq t \tag{3}$$

Lemma 4 (Suspension and Blocking as Jitter).

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

Constant-time Tests (CT)

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

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

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

Lemma 6 (Suspension and Blocking as Jitter).

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

- Time-demand Analysis(TDA)
  - 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).

$$\exists t \mid 0 < t \leq p_k, \quad s_k + e_k + \sigma_k \times B + \sum_{\tau_i \in hp(k)} \left(\left\lceil \frac{t}{p_i} \right\rceil + 1 \right) s_i \leq t \tag{3}$$

Lemma 4 (Suspension and Blocking as Jitter).

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

Constant-time Tests (CT)

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

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

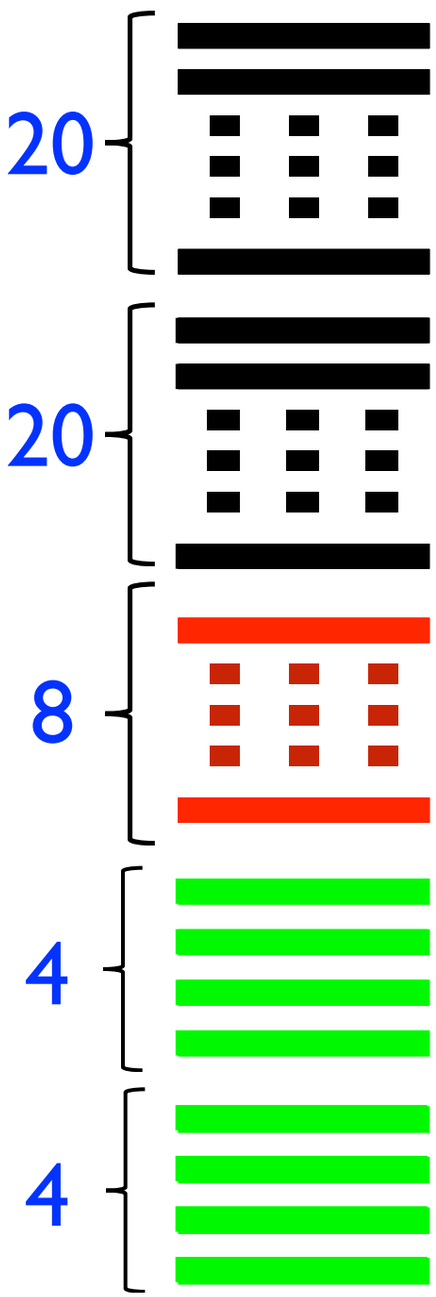
$$\sum_{\tau_i \in hp(k)} \frac{s_i}{p_i} \leq \ln \left(\frac{3}{\frac{s_k + e_k + \sigma_k \times B}{p_k} + 2} \right) \tag{5b}$$

Lemma 6 (Suspension and Blocking as Jitter).

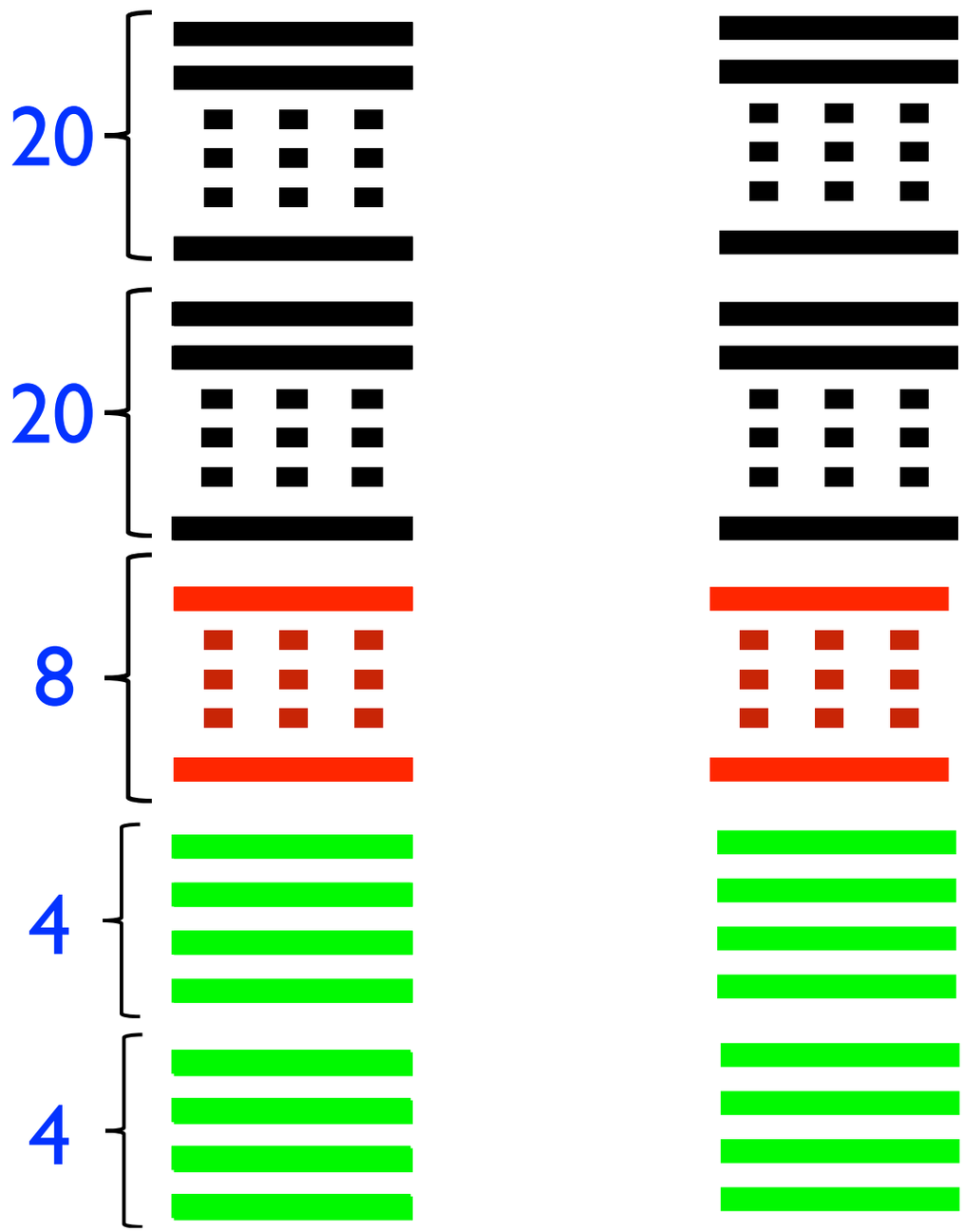
$$\frac{e_k + s_k + \sigma_k \times B}{p_k} + \frac{\sum_{\tau_i \in hp(k)} \left(2 s_i - \frac{s_i^2}{p_i} \right)}{p_k} + \sum_{\tau_i \in hp(k)} \frac{s_i}{p_i} \leq 1 \tag{6}$$

- Time-demand Analysis(TDA)
  - Higher complexity
  - Higher accuracy
- Constant-time Tests (CT)
  - Lower complexity
  - Lower accuracy

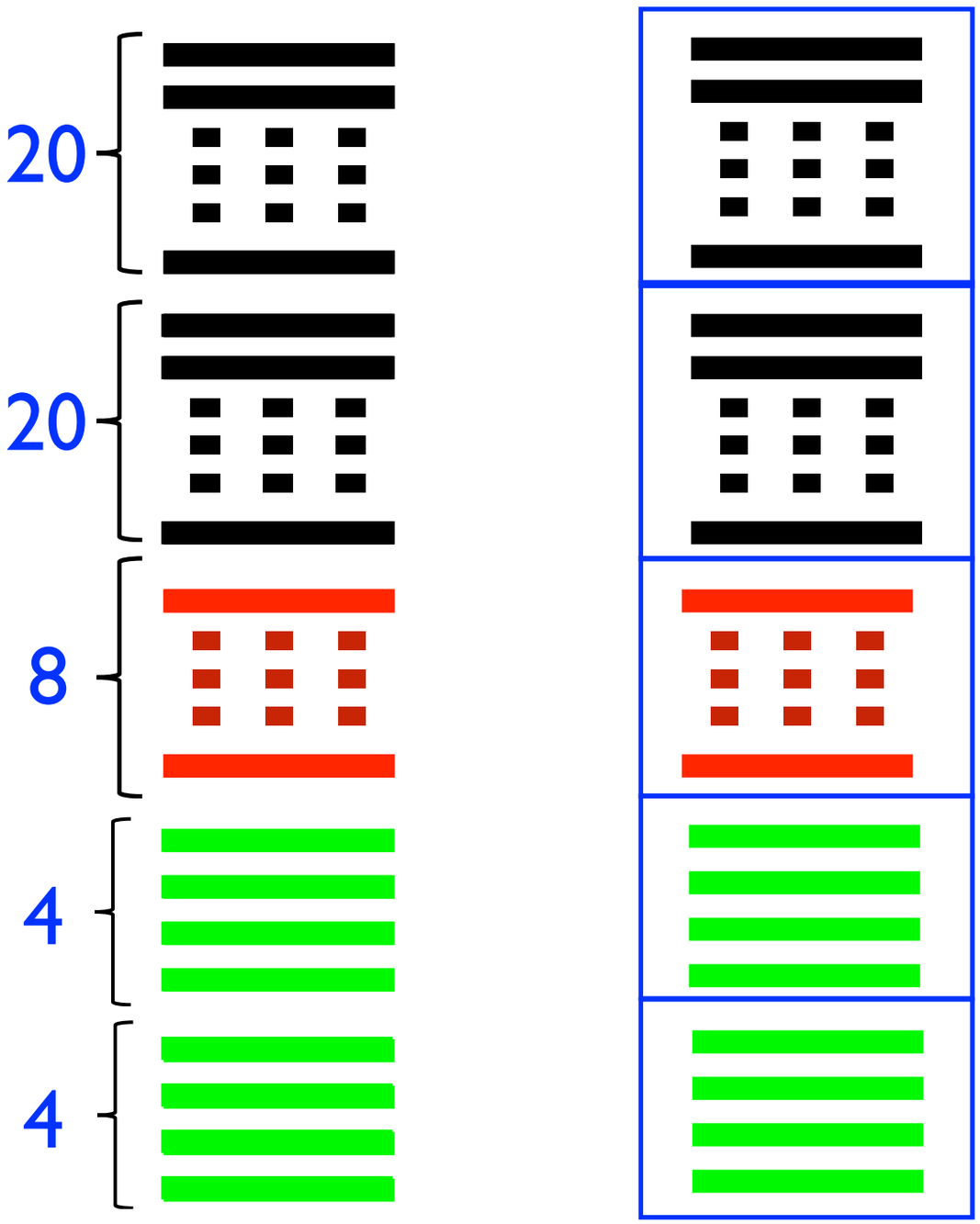
Step 2： Partitioning tasks onto a multi-unit shared resource



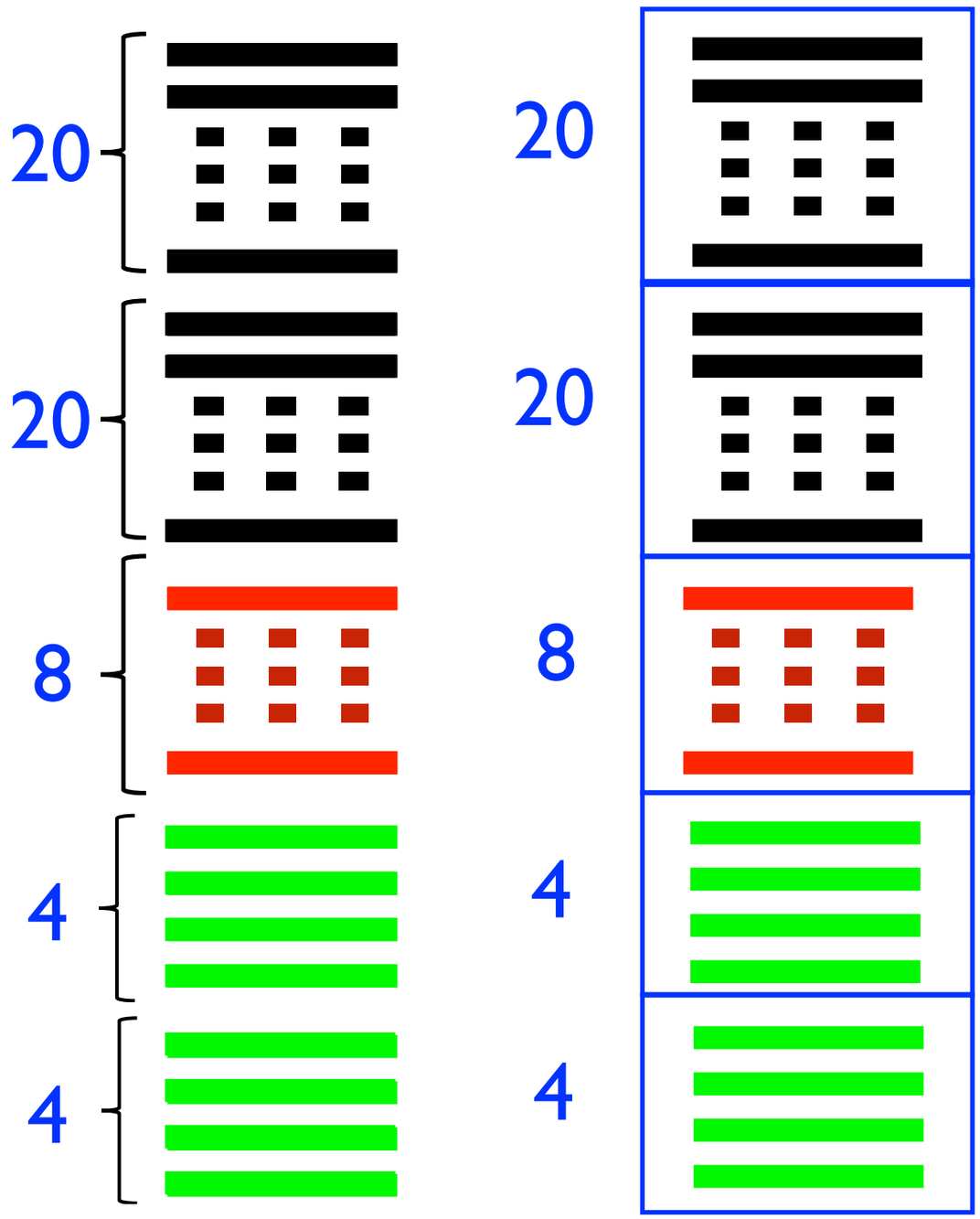
Step 2： Partitioning tasks onto a multi-unit shared resource



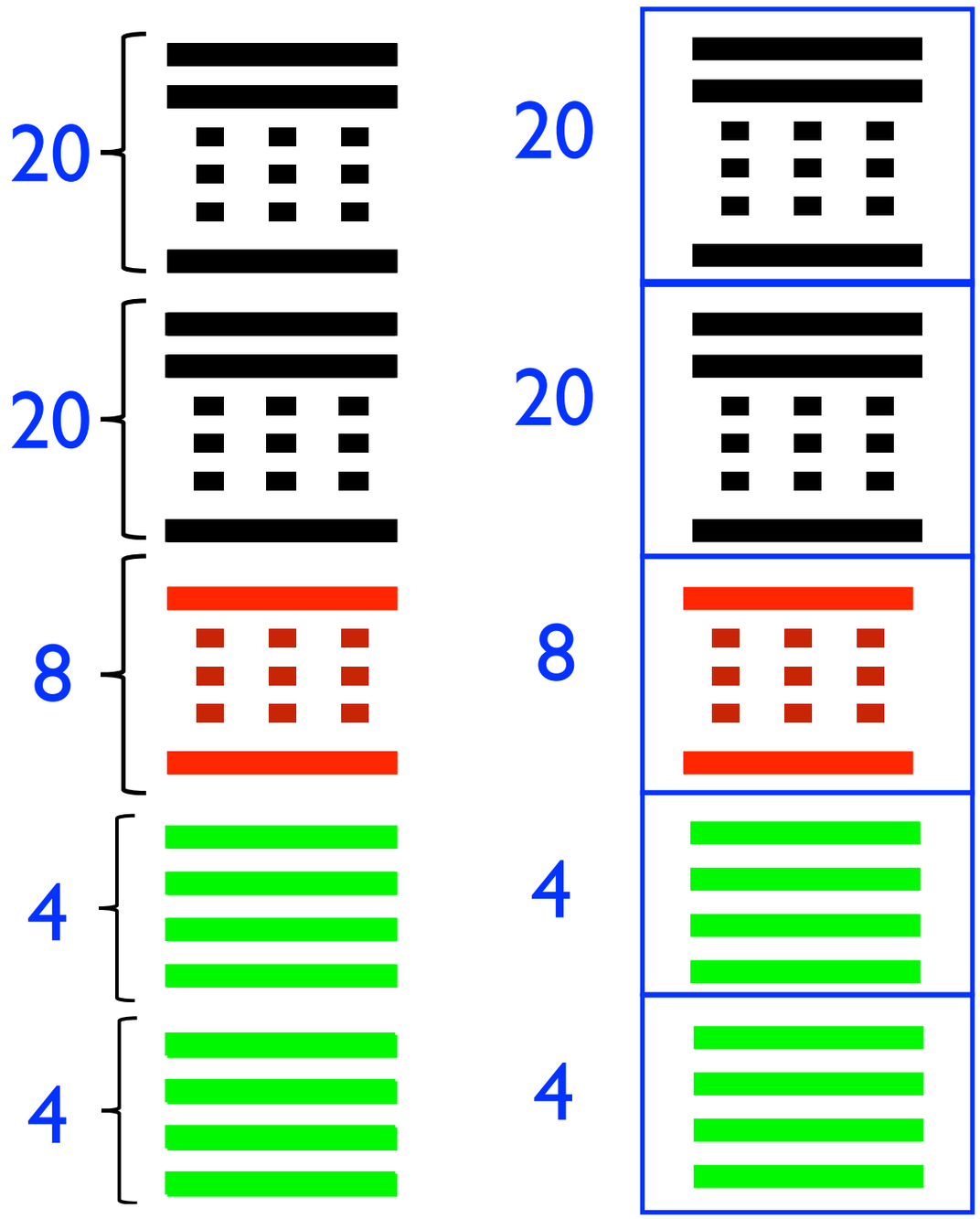
Step 2： Partitioning tasks onto a multi-unit shared resource



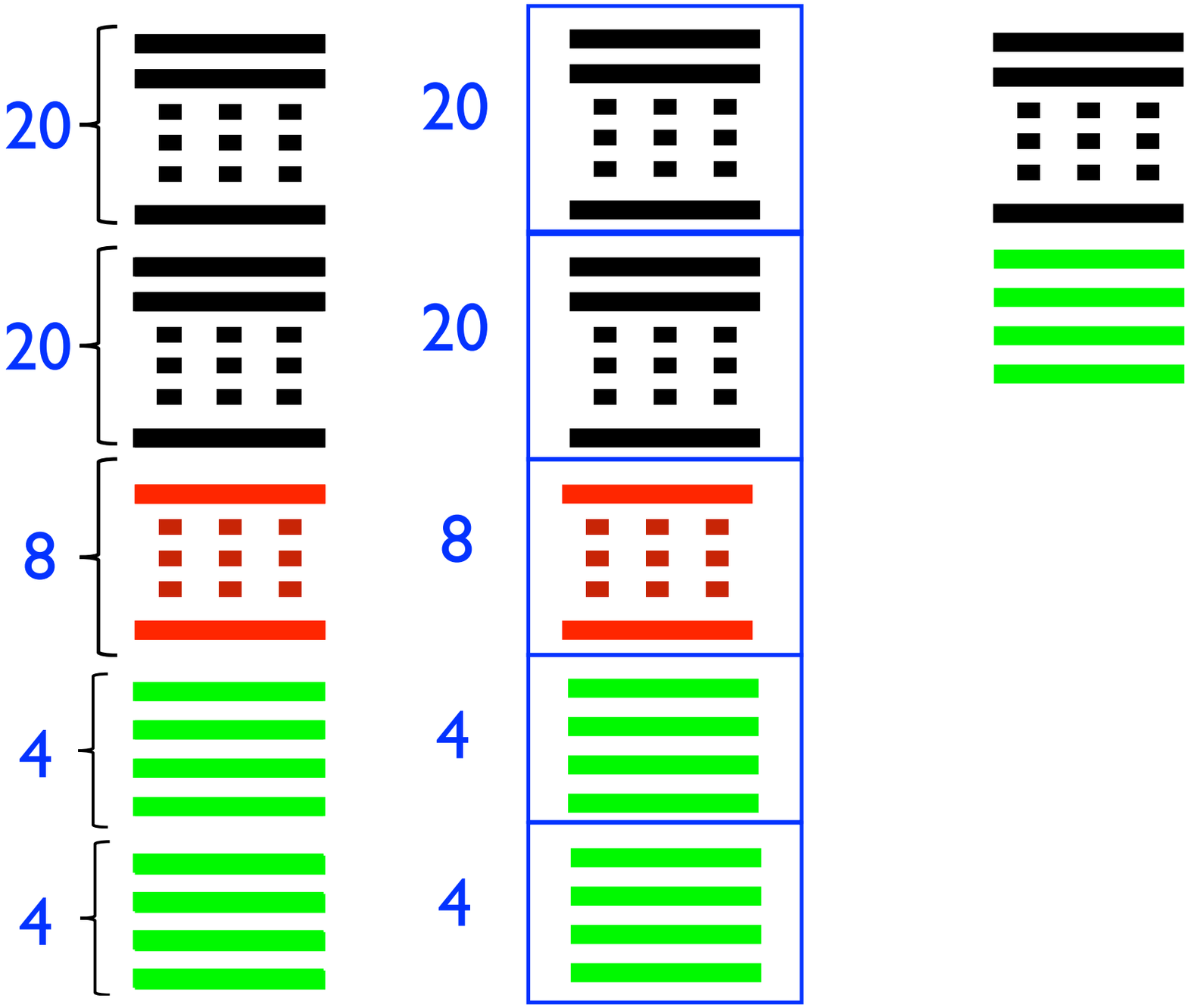
Step 2： Partitioning tasks onto a multi-unit shared resource



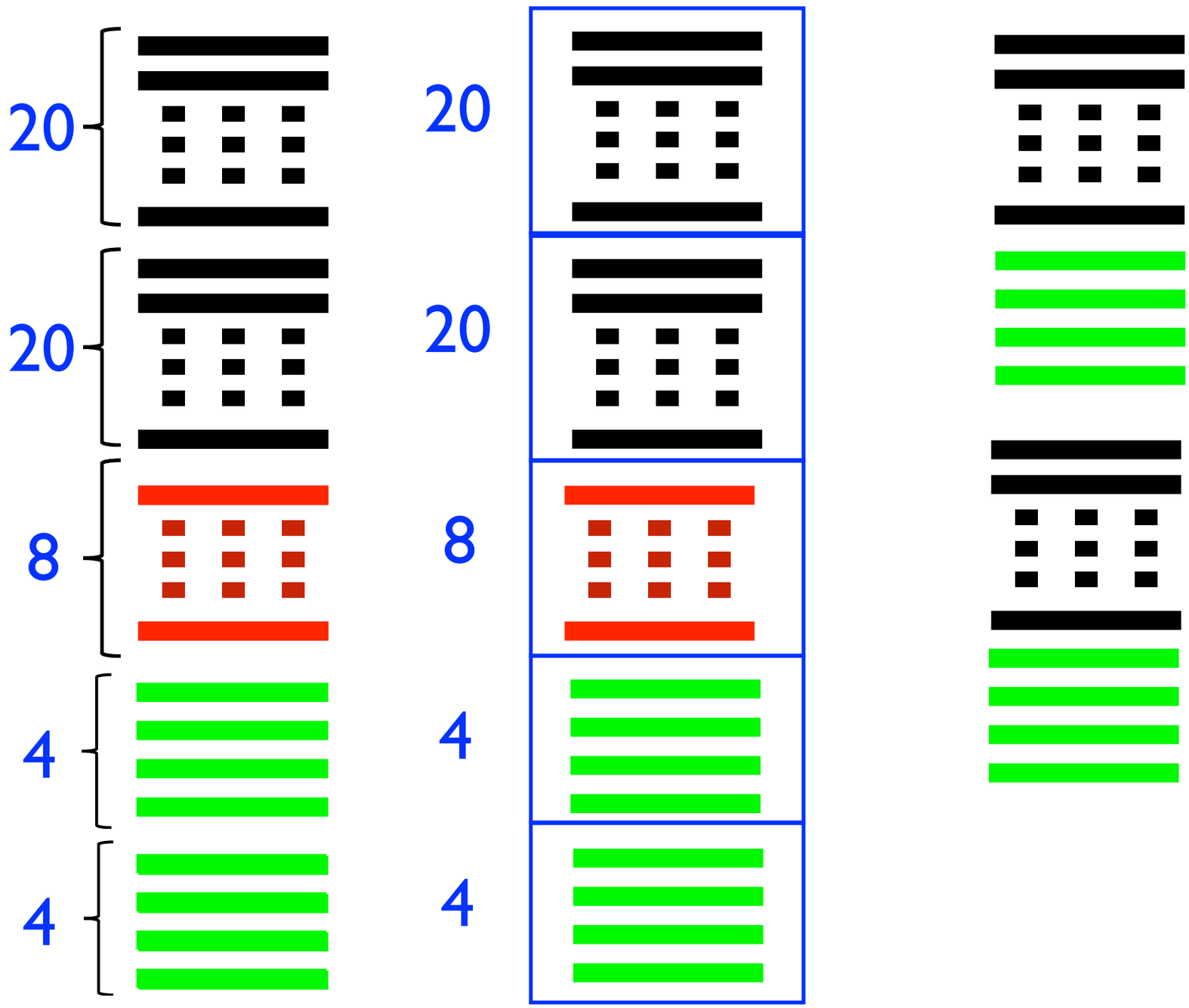
Step 2： Partitioning tasks onto a multi-unit shared resource



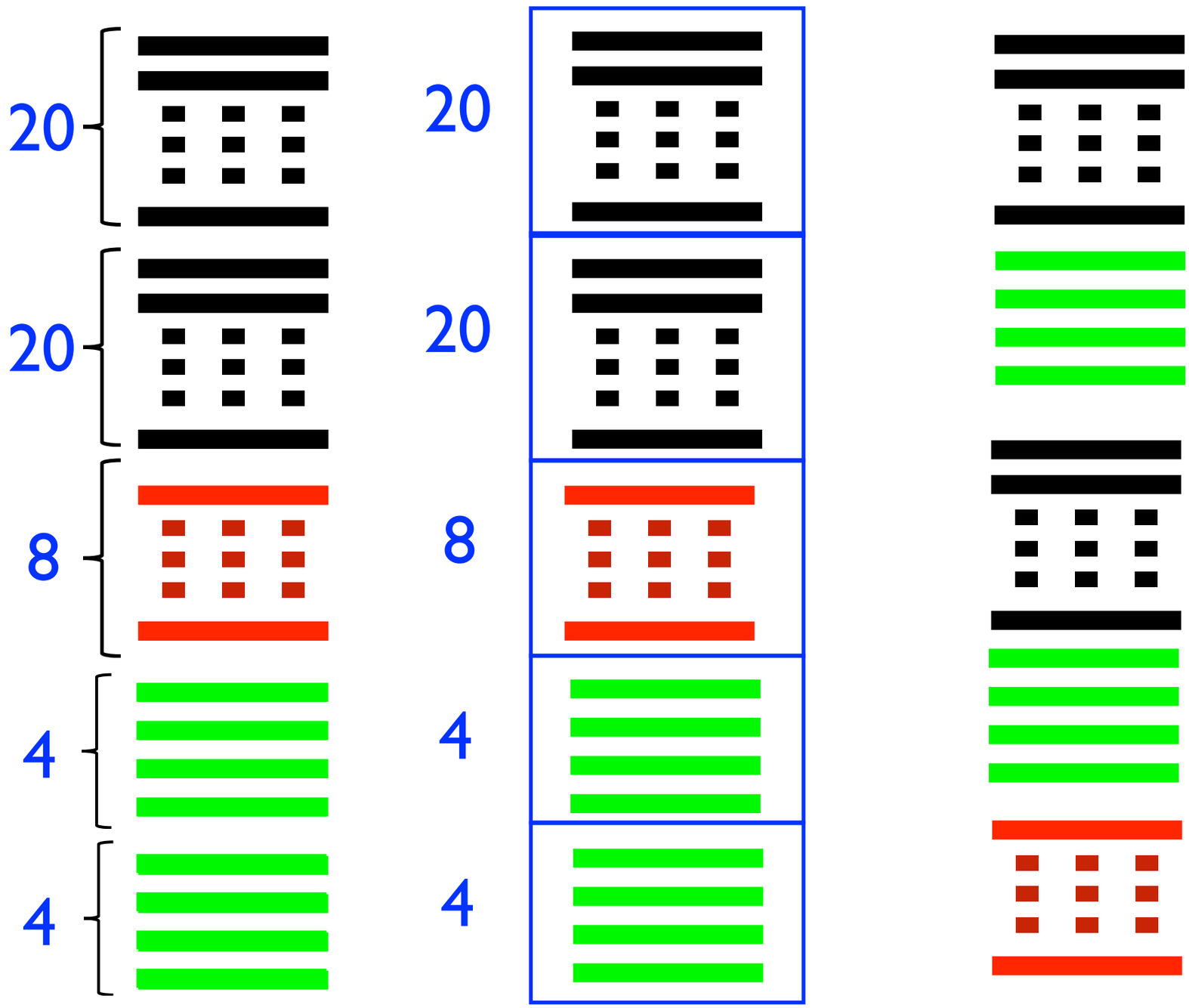
Step 2： Partitioning tasks onto a multi-unit shared resource



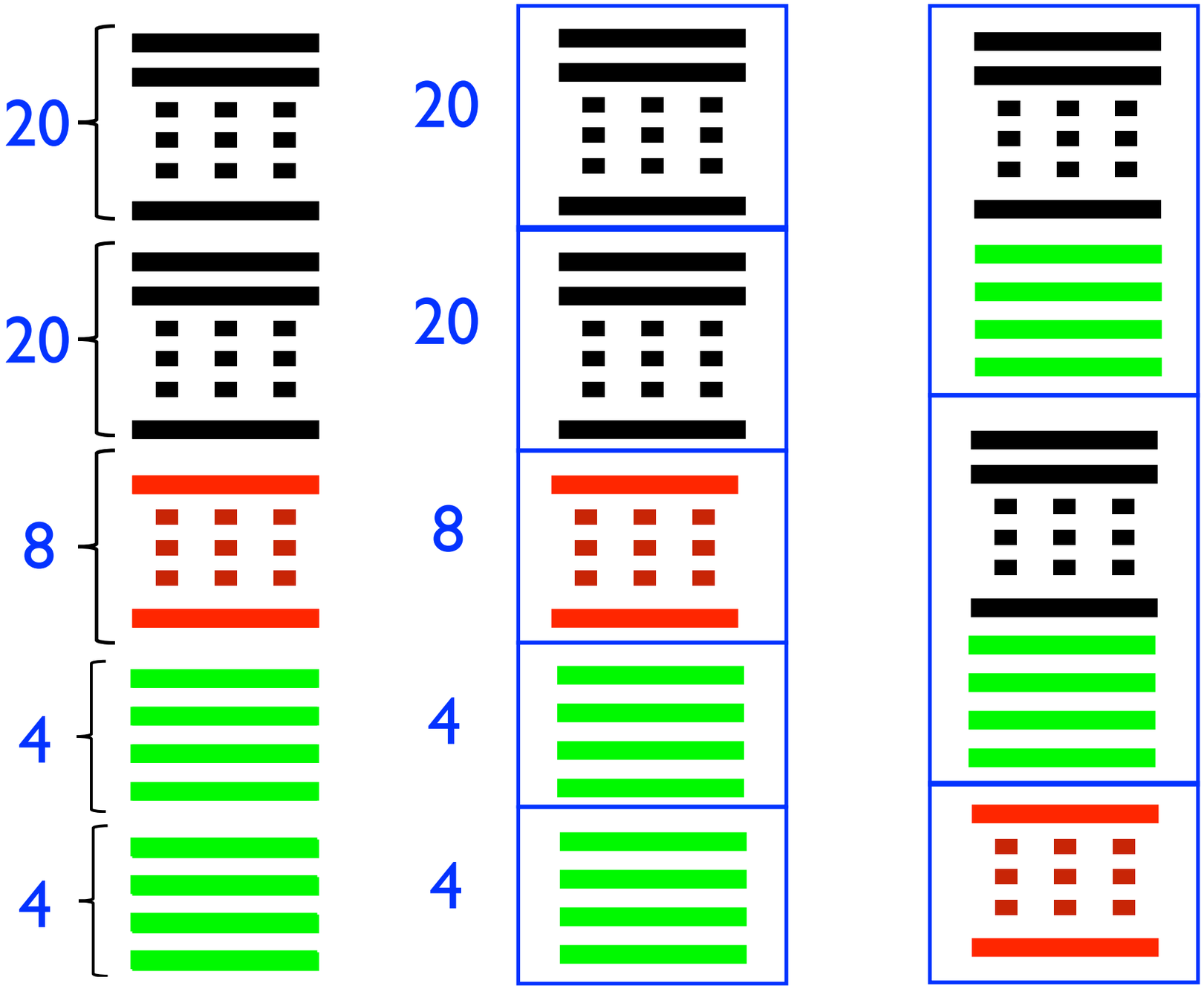
Step 2: Partitioning tasks onto a multi-unit shared resource



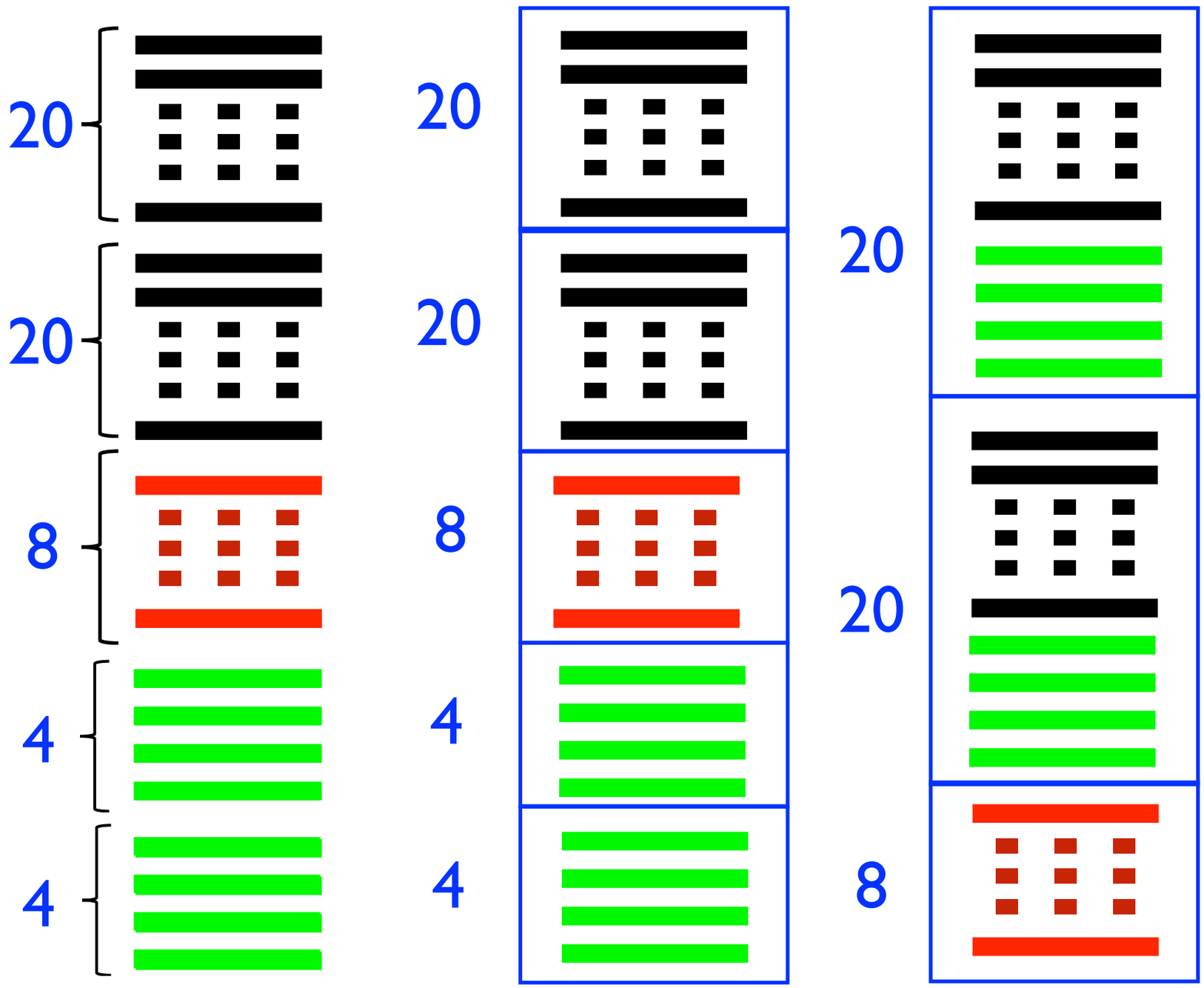
Step 2： Partitioning tasks onto a multi-unit shared resource



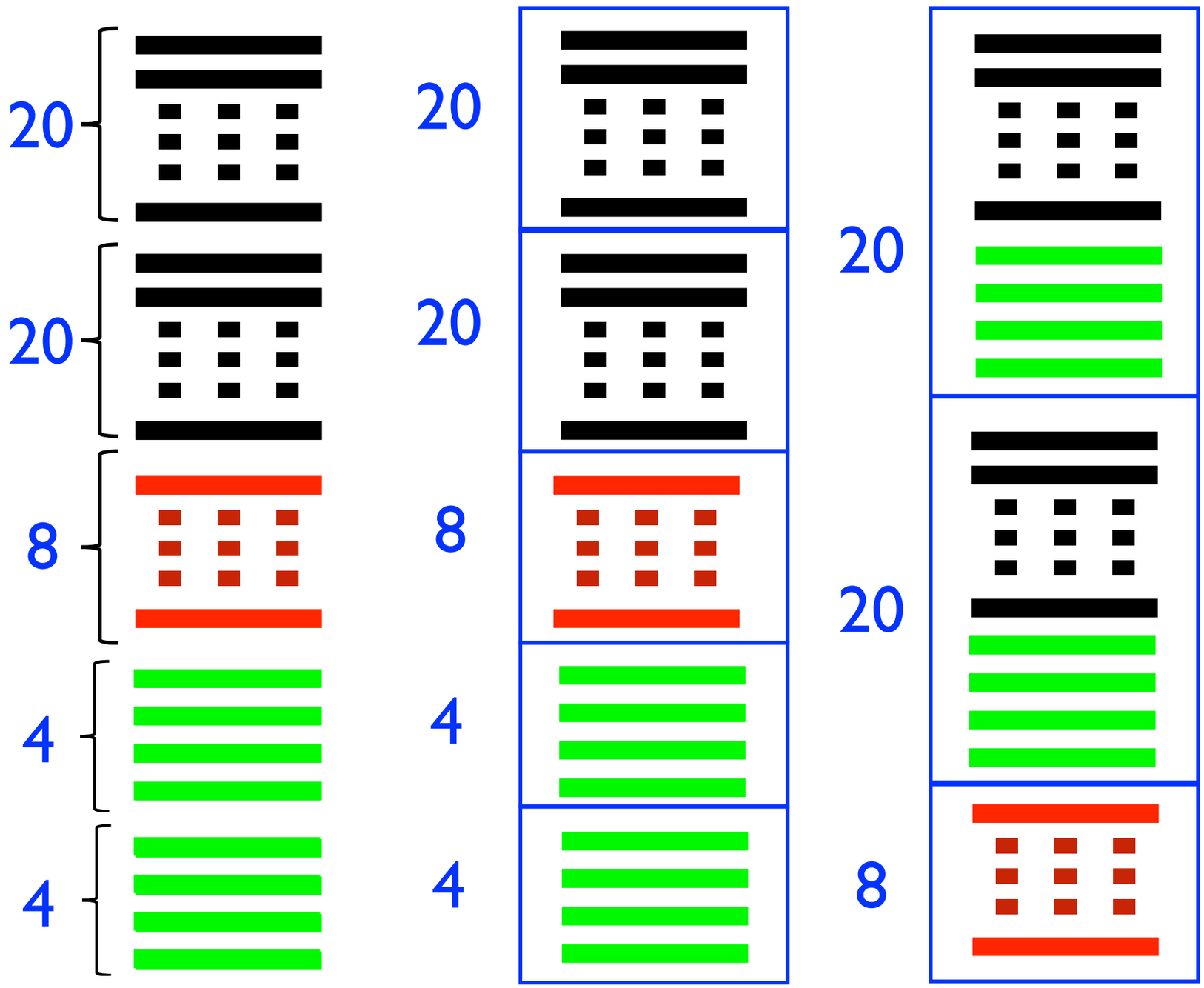
Step 2: Partitioning tasks onto a multi-unit shared resource



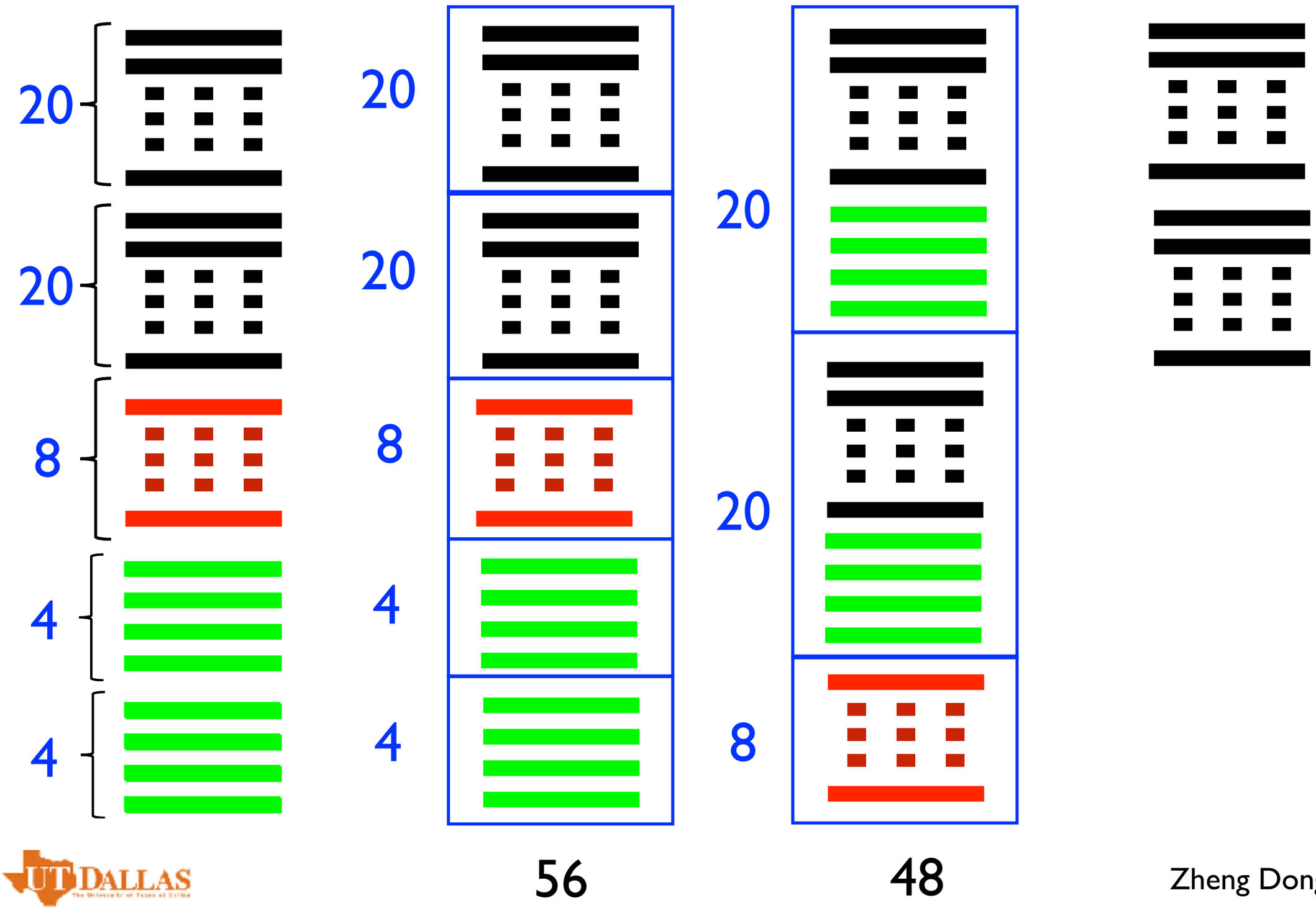
Step 2： Partitioning tasks onto a multi-unit shared resource



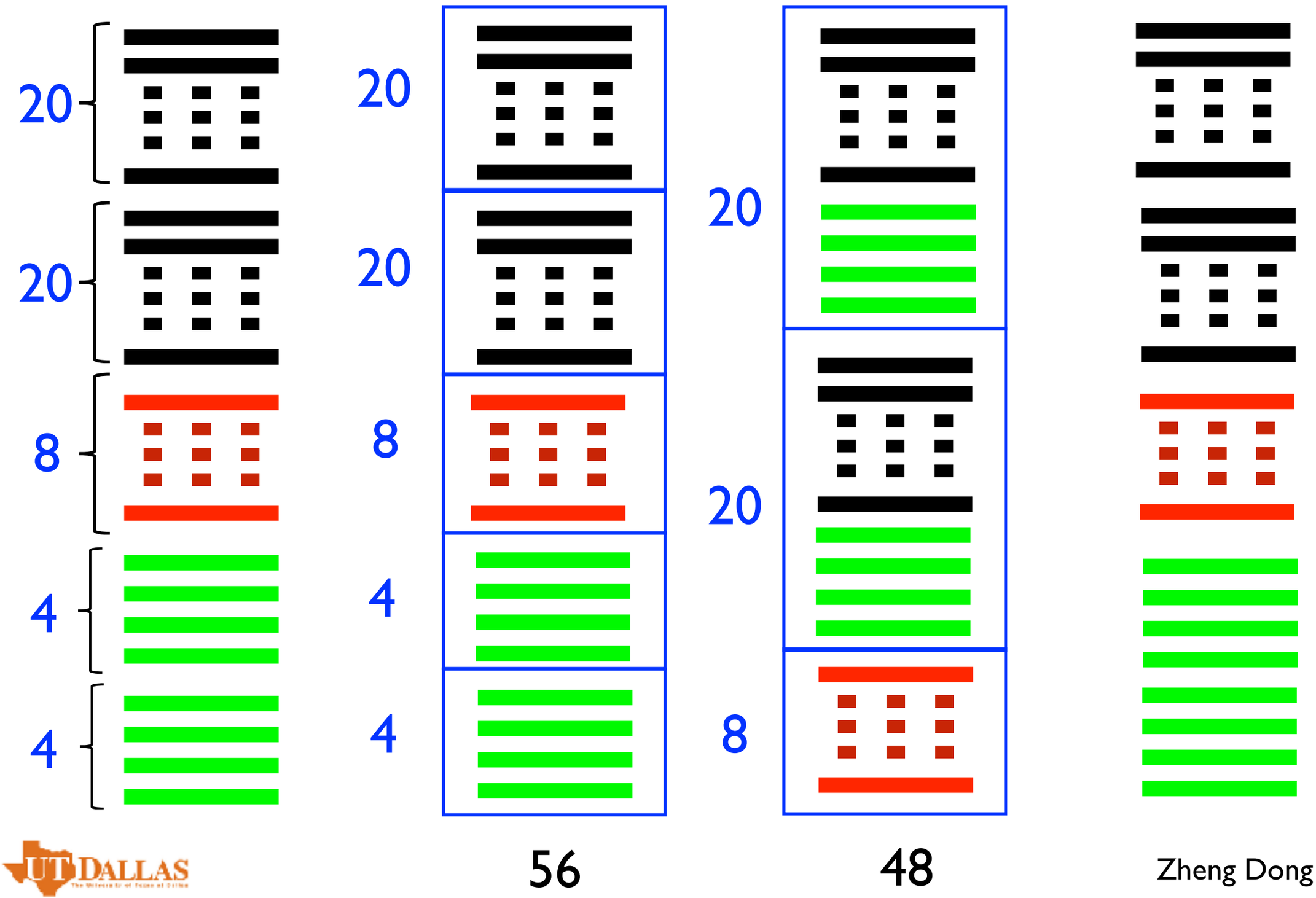
Step 2： Partitioning tasks onto a multi-unit shared resource



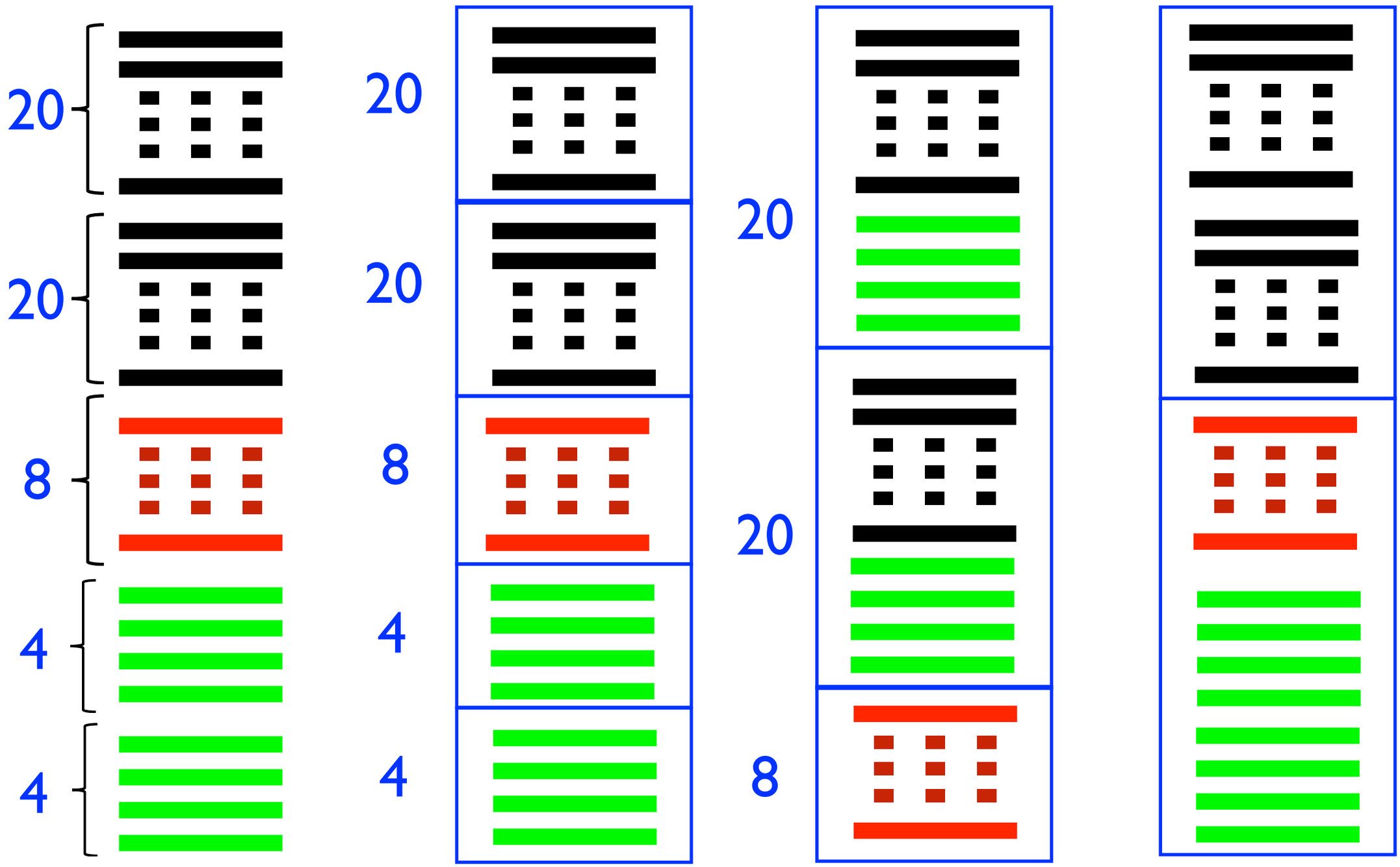
Step 2: Partitioning tasks onto a multi-unit shared resource



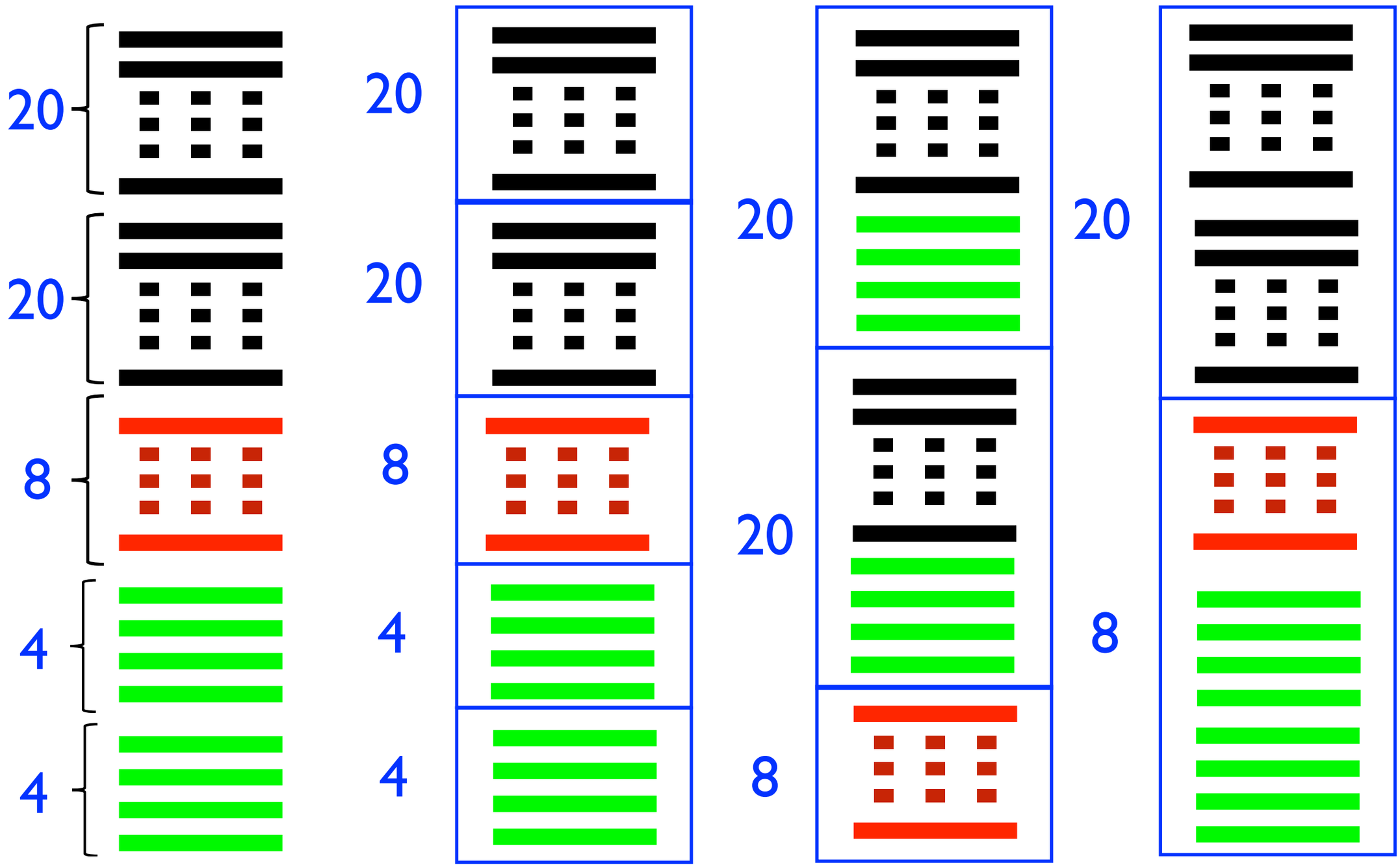
Step 2: Partitioning tasks onto a multi-unit shared resource



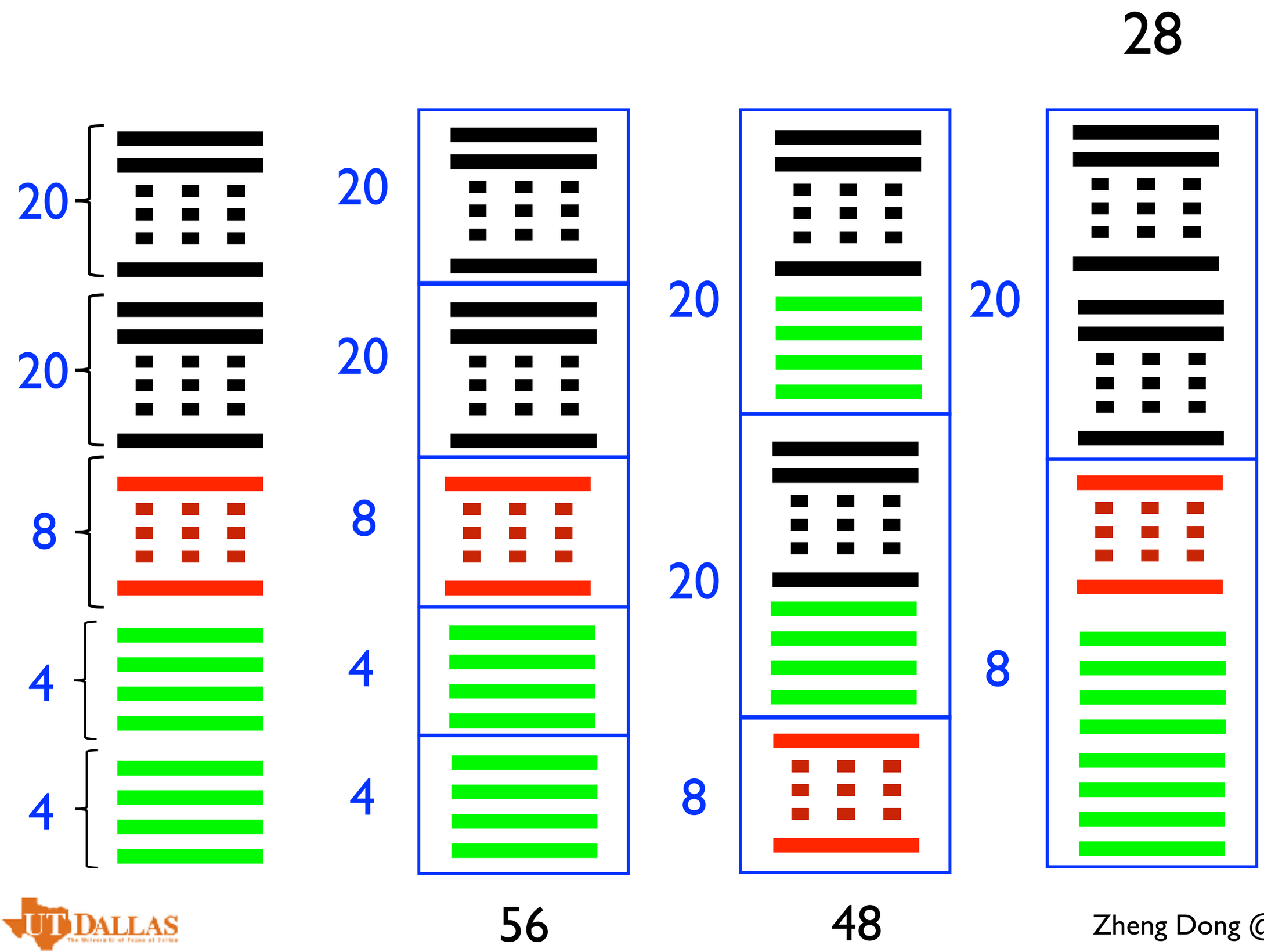
Step 2: Partitioning tasks onto a multi-unit shared resource



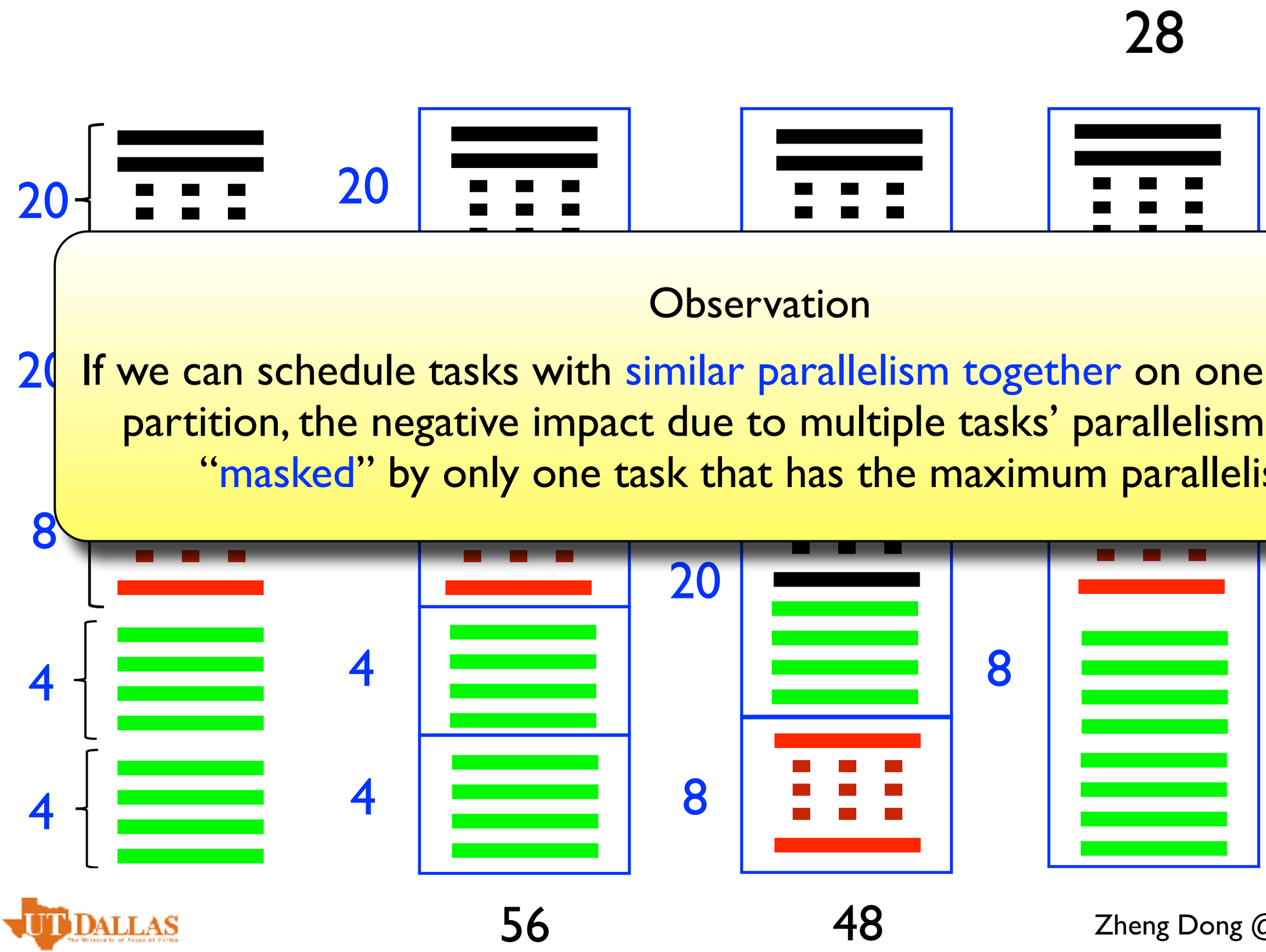
Step 2: Partitioning tasks onto a multi-unit shared resource



Step 2: Partitioning tasks onto a multi-unit shared resource



Step 2: Partitioning tasks onto a multi-unit shared resource



Observation

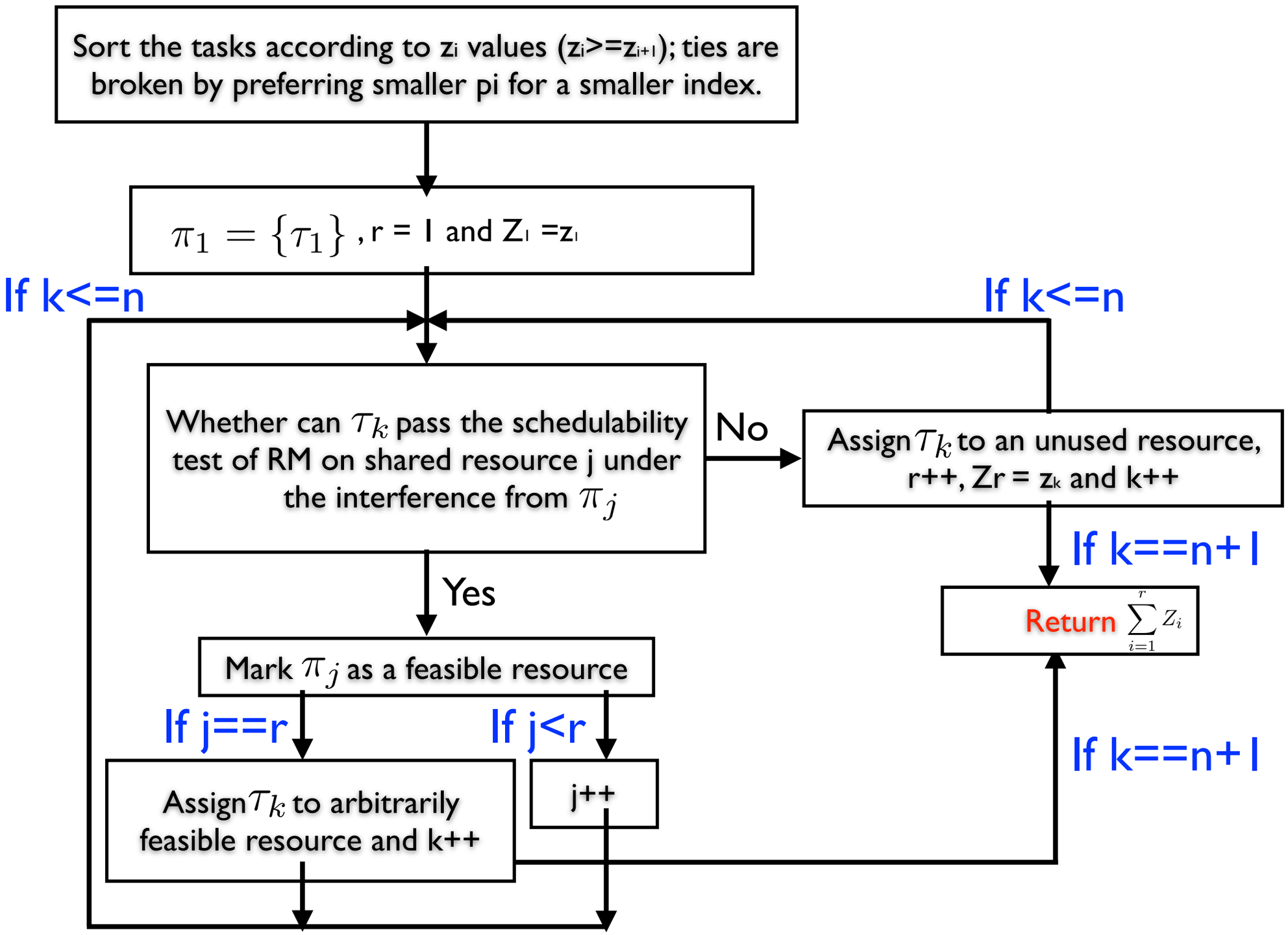
If we can schedule tasks with similar parallelism together on one resource partition, the negative impact due to multiple tasks' parallelism can be "masked" by only one task that has the maximum parallelism.

Step 2： Partitioning tasks onto a multi-unit shared resource

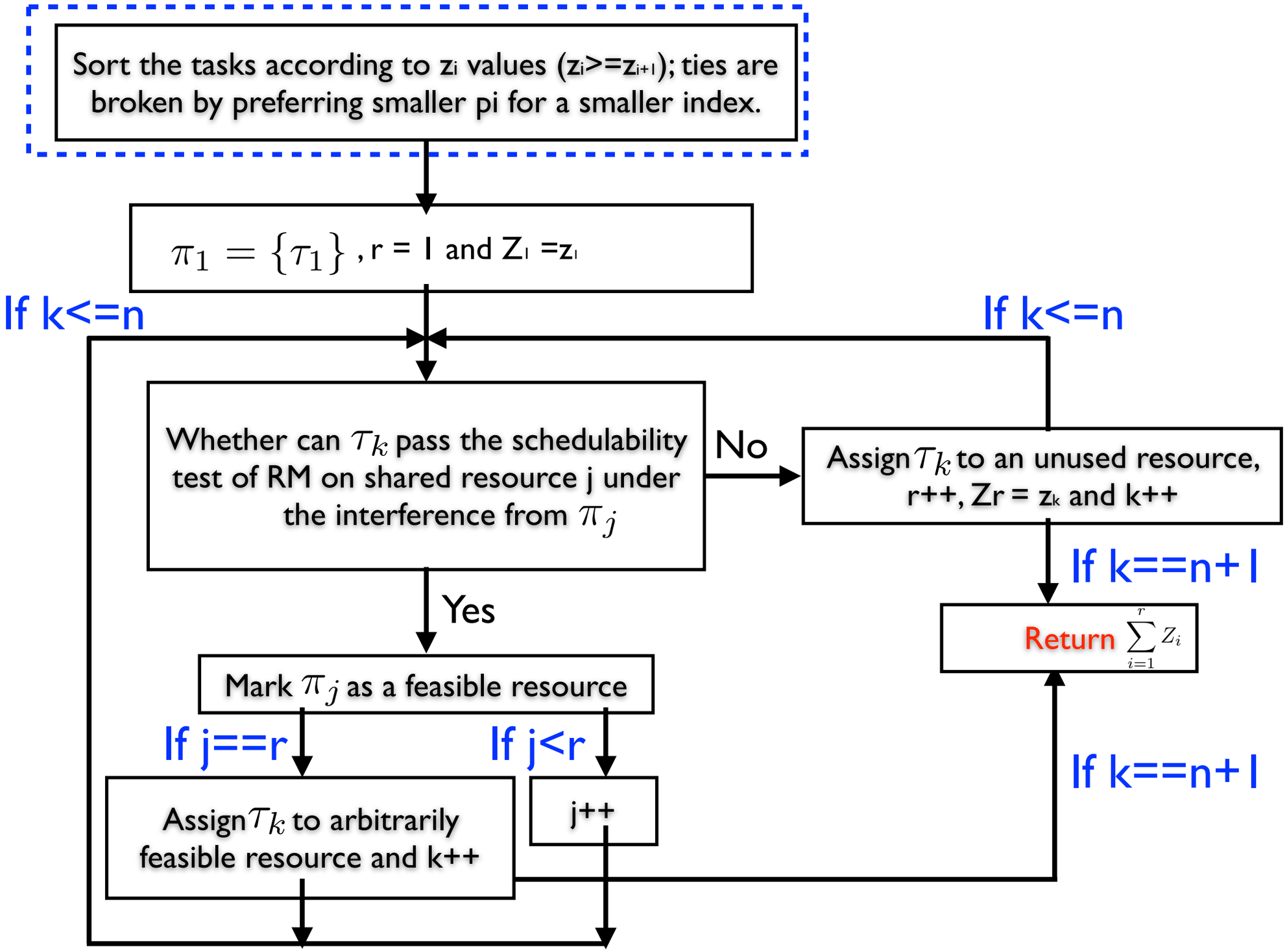
Step 2： Partitioning tasks onto a multi-unit shared resource

- 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.

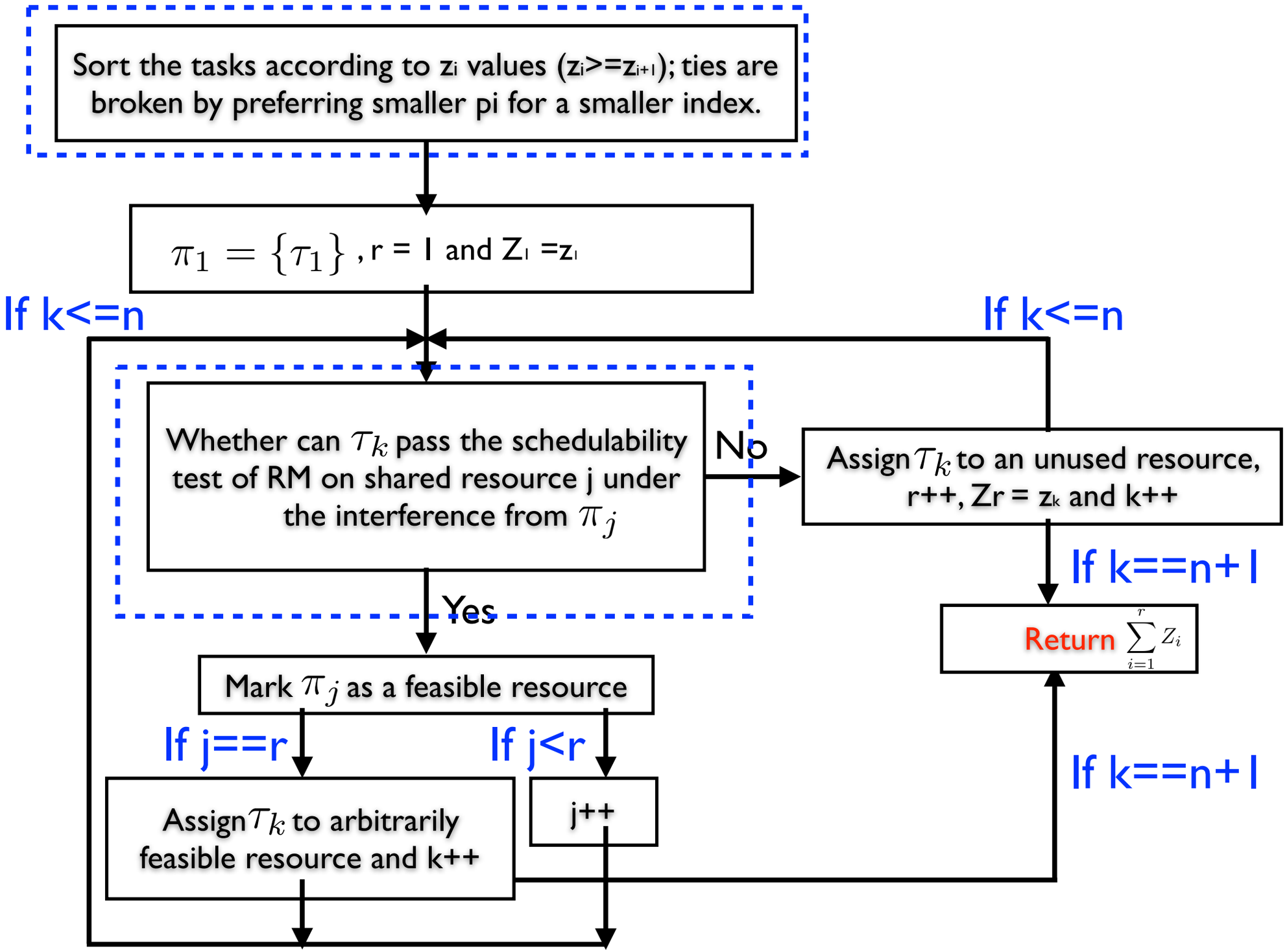
Step 2： Partitioning tasks onto a multi-unit shared resource



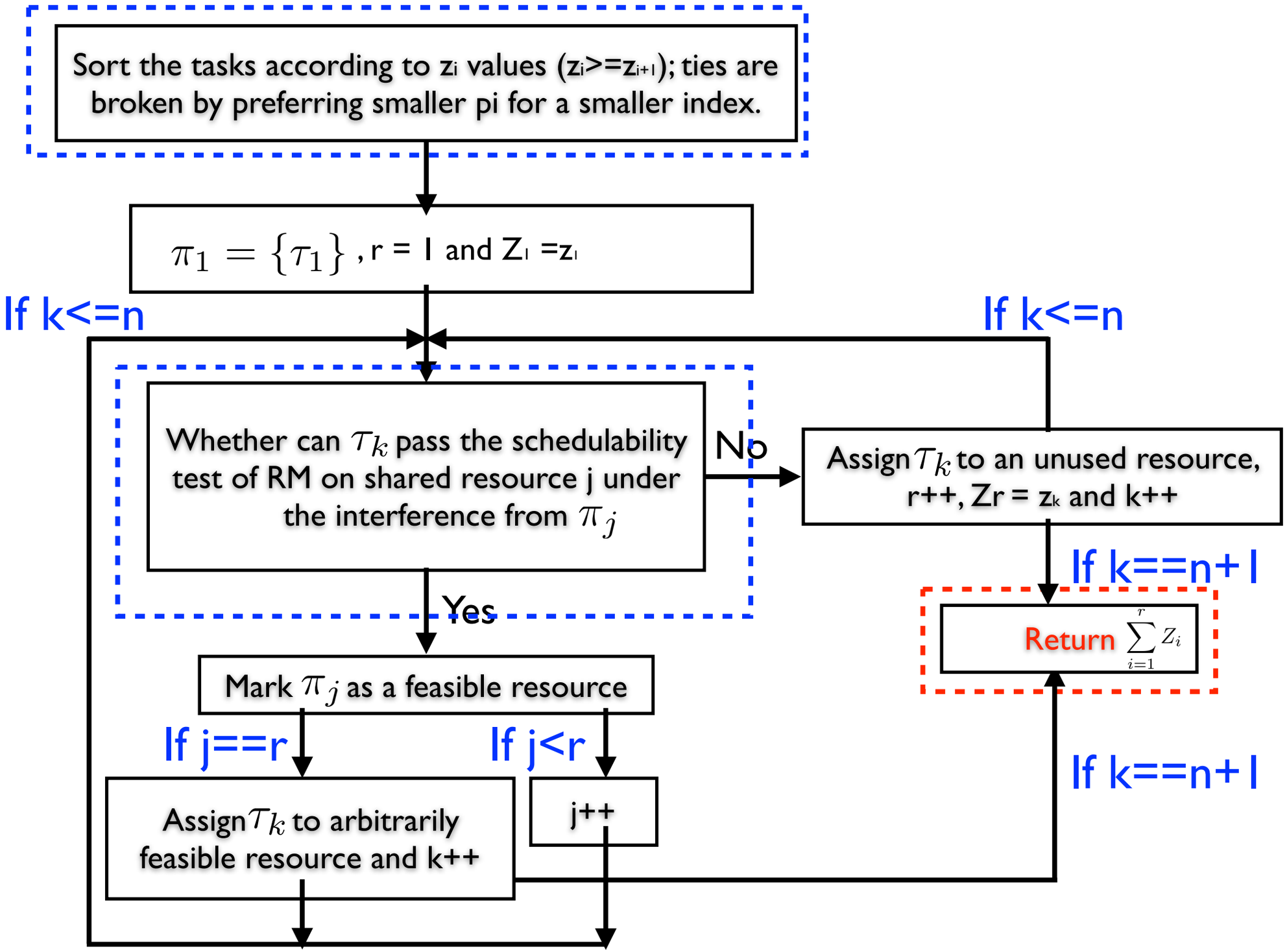
Step 2： Partitioning tasks onto a multi-unit shared resource



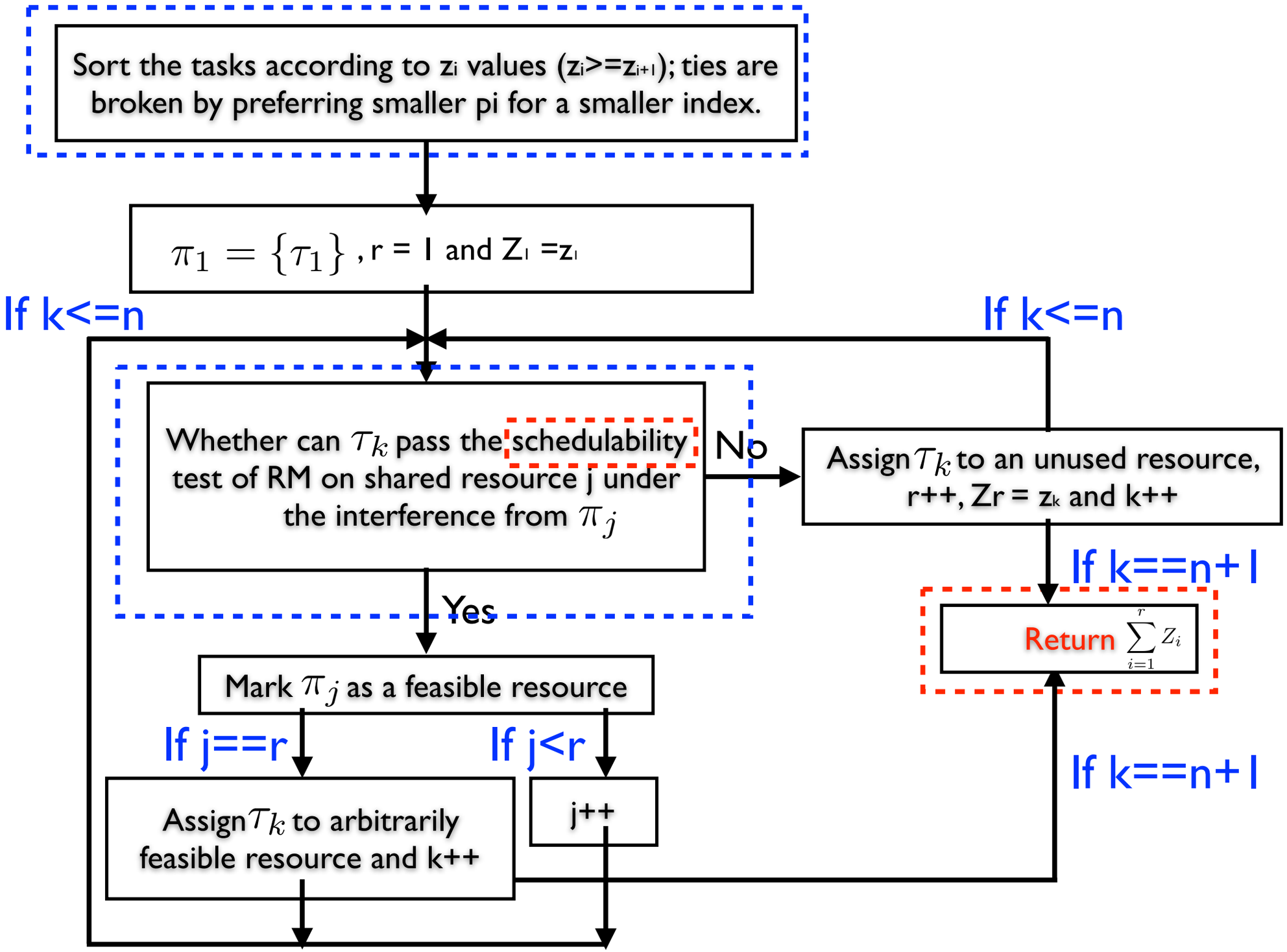
Step 2： Partitioning tasks onto a multi-unit shared resource



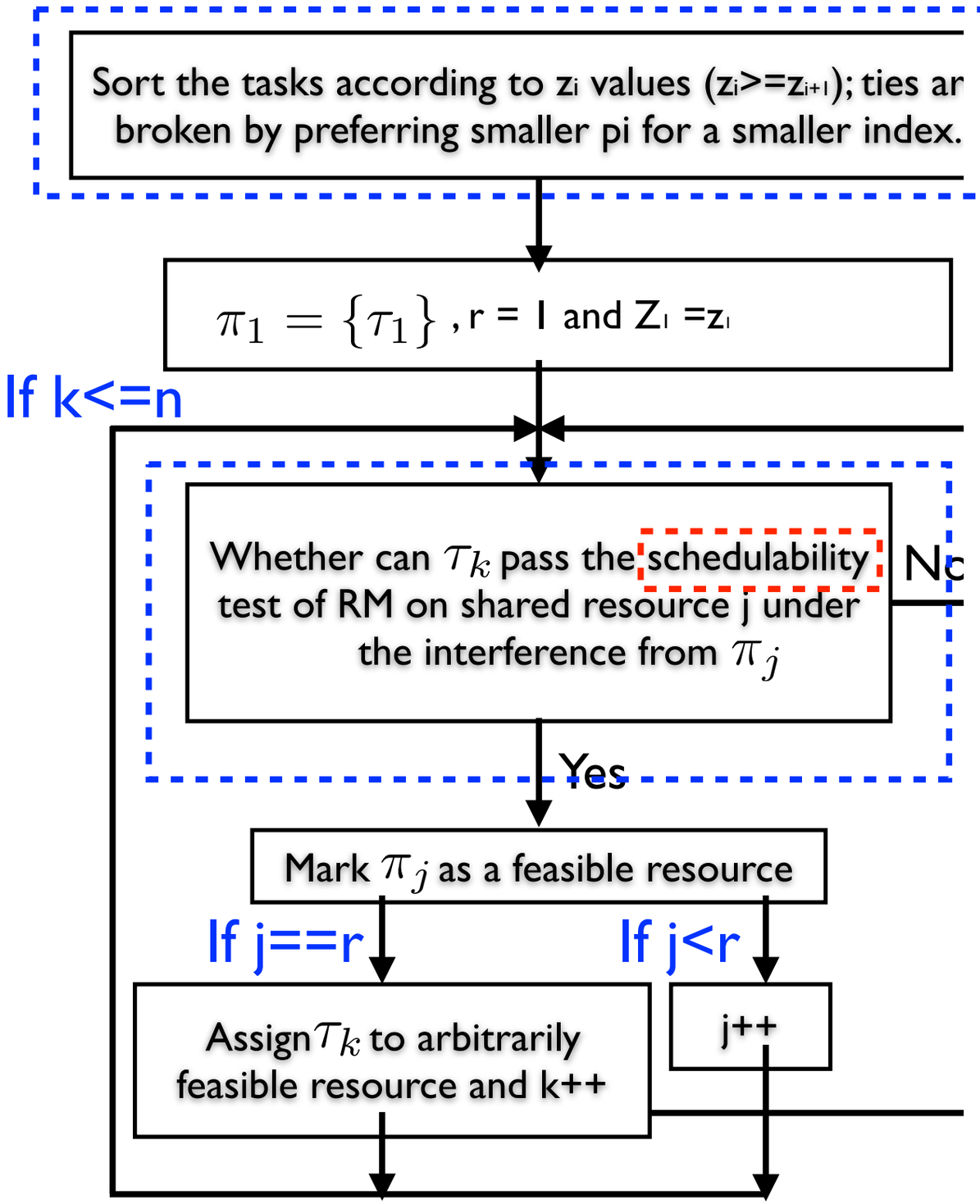
Step 2： Partitioning tasks onto a multi-unit shared resource



Step 2： Partitioning tasks onto a multi-unit shared resource



Step 2： Partitioning tasks onto a multi-unit shared resource



Time-demand Analysis(TDA)

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

$$\exists t \mid 0 < t \le p_k, \quad s_k + e_k + \sigma_k \times B + \sum_{\tau_i \in hp(k)} \left(\left\lceil \frac{t}{p_i} \right\rceil + 1 \right) s_i \le t \tag{3}$$

Lemma 4 (Suspension and Blocking as Jitter).

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

Constant-time Tests (CT)

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

$$\left(\frac{s_k + e_k + \sigma_k \times B}{p_k} + 2 \right) \prod_{\tau_i \in hp(k)} \left(\frac{s_i}{p_i} + 1 \right) \le 3 \tag{5a}$$

$$\sum_{\tau_i \in hp(k)} \frac{s_i}{p_i} \le \ln \left(\frac{3}{\frac{s_k + e_k + \sigma_k \times B}{p_k} + 2} \right) \tag{5b}$$

Lemma 6 (Suspension and Blocking as Jitter).

$$\frac{e_k + s_k + \sigma_k \times B}{p_k} + \frac{\sum_{\tau_i \in hp(k)} \left(2 s_i - \frac{s_i^2}{p_i} \right)}{p_k} + \sum_{\tau_i \in hp(k)} \frac{s_i}{p_i} \le 1 \tag{6}$$

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).

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

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

Lemma 6 (Suspension and Blocking as Jitter).

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

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).

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

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

Lemma 6 (Suspension and Blocking as Jitter).

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

ST-FF-CT(Carry)

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

$$where\ \overline{u_{max}} = \max_{i=m+1}^{n} \frac{e_i + s_i + \sigma_i \times B}{p_i}$$

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).

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

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

Lemma 6 (Suspension and Blocking as Jitter).

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

ST-FF-CT(Carry)

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

$$where\ \overline{u_{max}} = \max_{i=m+1}^{n} \frac{e_i + s_i + \sigma_i \times B}{p_i}$$

ST-FF-CT(Jitter)

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

$$where\ \overline{u_{max}} = \max_{i=m+1}^{n} \frac{e_i + s_i + \sigma_i \times B}{p_i}$$

Step 2： Partitioning tasks onto a multi-unit shared resource

Upper bounding the system utilization loss

ST-FF-CT(Carry)

$$U_{sum} \leq m \times \ln\left(\frac{3}{\ldots}\right)$$

Intuitively, task partitioning methods always suffer from both scheduling algorithms and bin-packing induced utilization loss.

$$\frac{e_k + s_k + \sigma_k \times B}{p_k} + \frac{\sum_{\tau_i \in hp(k)} (\ldots)}{p_k} + \sum_{\tau_i \in hp(k)} \frac{s_i}{p_i} \leq 1 \quad (6)$$

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

$$where\ \overline{u_{max}} = \max_{i=m+1}^{n} \frac{e_i + s_i + \sigma_i \times B}{p_i}$$

Step 2: Partitioning tasks onto a multi-unit shared resource

An Integer linear programming based approach.

Step 2: Partitioning tasks onto a multi-unit shared resource

An Integer linear programming based approach.

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.

min. the number of units of the shared resource

s.t. All Possible Partitions

Step 2: Partitioning tasks onto a multi-unit shared resource

An Integer linear programming based approach.

min. the number of units of the shared resource

s.t. All Possible Partitions

Time-demand Analysis(TDA)

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

$$\exists t \mid 0 < t \leq p_k, \quad s_k + e_k + \sigma_k \times B + \sum_{\tau_i \in hp(k)} \left(\left\lceil \frac{t}{p_i} \right\rceil + 1 \right) s_i \leq t \tag{3}$$

Lemma 4 (Suspension and Blocking as Jitter).

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

Constant-time Tests (CT)

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

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

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

Lemma 6 (Suspension and Blocking as Jitter).

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

Case Study

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

Task	Para.	Period	GPU	CPU	Res. Tim
τ_1	12	6.40	0.80	1.00	N/A
τ_2	12	25.58	1.60	3.00	20.5
τ_3	2	12.79	1.00	2.00	N/A
τ_4	2	12.79	0.90	2.50	N/A
τ_5	4	19.19	1.80	2.00	17.70
τ_6	6	7.99	1.00	1.00	N/A
τ_7	6	8.99	0.80	2.00	N/A
τ_8	1	11.19	0.50	2.50	N/A

TABLE I: CPU-centric scheduling.

Traditional CPU-centric Scheduling

Task	Para.	Period	GPU	CPU	Res. Tim
τ_1	12	6.40	0.80	1.00	N/A
τ_2	12	25.58	1.60	3.00	20.5
τ_3	2	12.79	1.00	2.00	N/A
τ_4	2	12.79	0.90	2.50	N/A
τ_5	4	19.19	1.80	2.00	17.70
τ_6	6	7.99	1.00	1.00	N/A
τ_7	6	8.99	0.80	2.00	N/A
τ_8	1	11.19	0.50	2.50	N/A

TABLE I: CPU-centric scheduling.

Traditional CPU-centric Scheduling

Task	Para.	Period	GPU	CPU	Res. Tim
τ_1	12	6.40	0.80	1.00	N/A
τ_2	12	25.58	1.60	3.00	20.5
τ_3	2	12.79	1.00	2.00	N/A
τ_4	2	12.79	0.90	2.50	N/A
τ_5	4	19.19	1.80	2.00	17.70
τ_6	6	7.99	1.00	1.00	N/A
τ_7	6	8.99	0.80	2.00	N/A
τ_8	1	11.19	0.50	2.50	N/A

TABLE I: CPU-centric scheduling.

GPU-centric Scheduling (First-fit partitioning)

Task	Part.	Para.	Period	GPU	CPU	Res. Time	Carry	Jitter	Mixed	Warps
τ_1	1	12	6.40	0.80	1.00	5.8	Pass	Pass	Pass	12
τ_6		6	7.99	1.00	1.00	6.7	Pass	Pass	Pass	
τ_7		6	8.99	0.80	2.00	8.0	Pass	Pass	Pass	
τ_8	2	1	11.19	0.50	2.50	10.1	Pass	Pass	Pass	6
τ_4		2	12.79	0.90	2.50	11.2	Fail	Pass	Pass	
τ_3		2	12.79	1.00	2.00	11.9	Pass	Pass	Pass	
τ_5	3	4	19.19	1.80	2.00	14	Pass	Pass	Pass	12
τ_2		12	25.58	1.60	3.00	18.3	Pass	Fail	Pass	

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

Task	Part.	Para.	Period	GPU	CPU	Res. Time	Carry	Jitter	Mixed	Warps
τ_1		12	6.40	0.80	1.00	6.13	Pass	Pass	Pass	
τ_6	1	6	7.99	1.00	1.00	7.5	Pass	Pass	Pass	12
τ_7		6	8.99	0.80	2.00	8.64	Pass	Fail	Pass	
τ_8		1	11.19	0.50	2.50	10.5	Pass	Pass	Pass	
τ_4	2	2	12.79	0.90	2.50	12.2	Pass	Pass	Pass	2
τ_3		2	12.79	1.00	2.00	12.6	Fail	Pass	Pass	
τ_5	3	4	19.19	1.80	2.00	10.1	Pass	Pass	Pass	12
τ_2		12	25.58	1.60	3.00	11.8	Pass	Pass	Pass	

TABLE III: Shared-resource-centric scheduling with first-fit partitioning using TDA-based schedulability tests.

GPU-centric Scheduling (First-fit partitioning)

Task	Part.	Para.	Period	GPU	CPU	Res. Time	Carry	Jitter	Mixed	Warps
τ_1	1	12	6.40	0.80	1.00	5.8	Pass	Pass	Pass	12
τ_6		6	7.99	1.00	1.00	6.7	Pass	Pass	Pass	
τ_7		6	8.99	0.80	2.00	8.0	Pass	Pass	Pass	
τ_8	2	1	11.19	0.50	2.50	10.1	Pass	Pass	Pass	6
τ_4		2	12.79	0.90	2.50	11.2	Fail	Pass	Pass	
τ_3		2	12.79	1.00	2.00	11.9	Pass	Pass	Pass	
τ_5	3	4	19.19	1.80	2.00	14	Pass	Pass	Pass	12
τ_2		12	25.58	1.60	3.00	18.3	Pass	Fail	Pass	

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

Task	Part.	Para.	Period	GPU	CPU	Res. Time	Carry	Jitter	Mixed	Warps
τ_1		12	6.40	0.80	1.00	6.13	Pass	Pass	Pass	
τ_6	1	6	7.99	1.00	1.00	7.5	Pass	Pass	Pass	12
τ_7		6	8.99	0.80	2.00	8.64	Pass	Fail	Pass	
τ_8		1	11.19	0.50	2.50	10.5	Pass	Pass	Pass	
τ_4	2	2	12.79	0.90	2.50	12.2	Pass	Pass	Pass	2
τ_3		2	12.79	1.00	2.00	12.6	Fail	Pass	Pass	
τ_5	3	4	19.19	1.80	2.00	10.1	Pass	Pass	Pass	12
τ_2		12	25.58	1.60	3.00	11.8	Pass	Pass	Pass	

TABLE III: Shared-resource-centric scheduling with first-fit partitioning using TDA-based schedulability tests.

GPU-centric Scheduling (First-fit partitioning)

Task	Part.	Para.	Period	GPU	CPU	Res. Time	Carry	Jitter	Mixed	Warps
τ_1	1	12	6.40	0.80	1.00	5.8	Pass	Pass	Pass	12
τ_6		6	7.99	1.00	1.00	6.7	Pass	Pass	Pass	
τ_7		6	8.99	0.80	2.00	8.0	Pass	Pass	Pass	
τ_8	2	1	11.19	0.50	2.50	10.1	Pass	Pass	Pass	6
τ_4		2	12.79	0.90	2.50	11.2	Fail	Pass	Pass	
τ_3		2	12.79	1.00	2.00	11.9	Pass	Pass	Pass	
τ_5	3	4	19.19	1.80	2.00	14	Pass	Pass	Pass	12
τ_2		12	25.58	1.60	3.00	18.3	Pass	Fail	Pass	

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

Task	Part.	Para.	Period	GPU	CPU	Res. Time	Carry	Jitter	Mixed	Warps
τ_1		12	6.40	0.80	1.00	6.13	Pass	Pass	Pass	
τ_6	1	6	7.99	1.00	1.00	7.5	Pass	Pass	Pass	12
τ_7		6	8.99	0.80	2.00	8.64	Pass	Fail	Pass	
τ_8		1	11.19	0.50	2.50	10.5	Pass	Pass	Pass	
τ_4	2	2	12.79	0.90	2.50	12.2	Pass	Pass	Pass	2
τ_3		2	12.79	1.00	2.00	12.6	Fail	Pass	Pass	
τ_5	3	4	19.19	1.80	2.00	10.1	Pass	Pass	Pass	12
τ_2		12	25.58	1.60	3.00	11.8	Pass	Pass	Pass	

TABLE III: Shared-resource-centric scheduling with first-fit partitioning using TDA-based schedulability tests.

GPU-centric Scheduling (First-fit partitioning)

Task	Part.	Para.	Period	GPU	CPU	Res. Time	Carry	Jitter	Mixed	Warps
τ_1	1	12	6.40	0.80	1.00	5.8	Pass	Pass	Pass	12
τ_6		6	7.99	1.00	1.00	6.7	Pass	Pass	Pass	
τ_7		6	8.99	0.80	2.00	8.0	Pass	Pass	Pass	
τ_8	2	1	11.19	0.50	2.50	10.1	Pass	Pass	Pass	6
τ_4		2	12.79	0.90	2.50	11.2	Fail	Pass	Pass	
τ_3		2	12.79	1.00	2.00	11.9	Pass	Pass	Pass	
τ_5	3	4	19.19	1.80	2.00	14	Pass	Pass	Pass	12
τ_2		12	25.58	1.60	3.00	18.3	Pass	Fail	Pass	

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

Task	Part.	Para.	Period	GPU	CPU	Res. Time	Carry	Jitter	Mixed	Warps
τ_1		12	6.40	0.80	1.00	6.13	Pass	Pass	Pass	
τ_6	1	6	7.99	1.00	1.00	7.5	Pass	Pass	Pass	12
τ_7		6	8.99	0.80	2.00	8.64	Pass	Fail	Pass	
τ_8		1	11.19	0.50	2.50	10.5	Pass	Pass	Pass	
τ_4	2	2	12.79	0.90	2.50	12.2	Pass	Pass	Pass	2
τ_3		2	12.79	1.00	2.00	12.6	Fail	Pass	Pass	
τ_5	3	4	19.19	1.80	2.00	10.1	Pass	Pass	Pass	12
τ_2		12	25.58	1.60	3.00	11.8	Pass	Pass	Pass	

TABLE III: Shared-resource-centric scheduling with first-fit partitioning using TDA-based schedulability tests.

GPU-centric Scheduling (First-fit partitioning)

30

Task	Part.	Para.	Period	GPU	CPU	Res. Time	Carry	Jitter	Mixed	Warps
τ_1	1	12	6.40	0.80	1.00	5.8	Pass	Pass	Pass	12
τ_6		6	7.99	1.00	1.00	6.7	Pass	Pass	Pass	
τ_7		6	8.99	0.80	2.00	8.0	Pass	Pass	Pass	
τ_8	2	1	11.19	0.50	2.50	10.1	Pass	Pass	Pass	6
τ_4		2	12.79	0.90	2.50	11.2	Fail	Pass	Pass	
τ_3		2	12.79	1.00	2.00	11.9	Pass	Pass	Pass	
τ_5	3	4	19.19	1.80	2.00	14	Pass	Pass	Pass	12
τ_2		12	25.58	1.60	3.00	18.3	Pass	Fail	Pass	

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

26

Task	Part.	Para.	Period	GPU	CPU	Res. Time	Carry	Jitter	Mixed	Warps
τ_1		12	6.40	0.80	1.00	6.13	Pass	Pass	Pass	
τ_6	1	6	7.99	1.00	1.00	7.5	Pass	Pass	Pass	12
τ_7		6	8.99	0.80	2.00	8.64	Pass	Fail	Pass	
τ_8		1	11.19	0.50	2.50	10.5	Pass	Pass	Pass	
τ_4	2	2	12.79	0.90	2.50	12.2	Pass	Pass	Pass	2
τ_3		2	12.79	1.00	2.00	12.6	Fail	Pass	Pass	
τ_5	3	4	19.19	1.80	2.00	10.1	Pass	Pass	Pass	12
τ_2		12	25.58	1.60	3.00	11.8	Pass	Pass	Pass	

TABLE III: Shared-resource-centric scheduling with first-fit partitioning using TDA-based schedulability tests.

GPU-centric Scheduling (ILP-based partitioning)

Task	Part.	Para.	Period	GPU	CPU	Res. Time	Carry	Jitter	Mixed	Warps
τ_1		12	6.40	0.80	1.00	6.03	Pass	Pass	Pass	
τ_2	1	12	25.58	1.60	3.00	17.71	Fail	Pass	Pass	12
τ_7		6	8.99	0.80	2.00	8.32	Pass	Pass	Pass	
τ_6	2	6	7.99	1.00	1.00	7.45	Pass	Pass	Pass	6
τ_5		4	19.19	1.80	2.00	16.6	Pass	Pass	Pass	
τ_3		2	12.79	2.00	2.00	12.6	Pass	Pass	Pass	
τ_4	3	2	12.79	0.90	2.50	12.2	Fail	Pass	Pass	2
τ_8		1	11.19	0.50	2.50	10.5	Pass	Pass	Pass	

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

Task	Part.	Para.	Period	GPU	CPU	Res. Time	Carry	Jitter	Mixed	Warps
τ_1		12	6.40	0.80	1.00	6.03	Pass	Pass	Pass	
τ_2	1	12	25.58	1.60	3.00	17.71	Pass	Pass	Pass	12
τ_7		6	8.99	0.80	2.00	8.32	Pass	Pass	Pass	
τ_6	2	6	7.99	1.00	1.00	7.45	Pass	Pass	Pass	6
τ_5		4	19.19	1.80	2.00	16.6	Pass	Pass	Pass	
τ_3		2	12.79	2.00	2.00	12.6	Pass	Pass	Pass	
τ_4	3	2	12.79	0.90	2.50	12.2	Fail	Pass	Pass	2
τ_8		1	11.19	0.50	2.50	10.5	Pass	Pass	Pass	

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

GPU-centric Scheduling (ILP-based partitioning)

Task	Part.	Para.	Period	GPU	CPU	Res. Time	Carry	Jitter	Mixed	Warps
τ_1		12	6.40	0.80	1.00	6.03	Pass	Pass	Pass	
τ_2	1	12	25.58	1.60	3.00	17.71	Fail	Pass	Pass	12
τ_7		6	8.99	0.80	2.00	8.32	Pass	Pass	Pass	
τ_6	2	6	7.99	1.00	1.00	7.45	Pass	Pass	Pass	6
τ_5		4	19.19	1.80	2.00	16.6	Pass	Pass	Pass	
τ_3		2	12.79	2.00	2.00	12.6	Pass	Pass	Pass	
τ_4	3	2	12.79	0.90	2.50	12.2	Fail	Pass	Pass	2
τ_8		1	11.19	0.50	2.50	10.5	Pass	Pass	Pass	

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

Task	Part.	Para.	Period	GPU	CPU	Res. Time	Carry	Jitter	Mixed	Warps
τ_1		12	6.40	0.80	1.00	6.03	Pass	Pass	Pass	
τ_2	1	12	25.58	1.60	3.00	17.71	Pass	Pass	Pass	12
τ_7		6	8.99	0.80	2.00	8.32	Pass	Pass	Pass	
τ_6	2	6	7.99	1.00	1.00	7.45	Pass	Pass	Pass	6
τ_5		4	19.19	1.80	2.00	16.6	Pass	Pass	Pass	
τ_3		2	12.79	2.00	2.00	12.6	Pass	Pass	Pass	
τ_4	3	2	12.79	0.90	2.50	12.2	Fail	Pass	Pass	2
τ_8		1	11.19	0.50	2.50	10.5	Pass	Pass	Pass	

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

GPU-centric Scheduling (ILP-based partitioning)

Task	Part.	Para.	Period	GPU	CPU	Res. Time	Carry	Jitter	Mixed	Warps
τ_1		12	6.40	0.80	1.00	6.03	Pass	Pass	Pass	
τ_2	1	12	25.58	1.60	3.00	17.71	Fail	Pass	Pass	12
τ_7		6	8.99	0.80	2.00	8.32	Pass	Pass	Pass	
τ_6	2	6	7.99	1.00	1.00	7.45	Pass	Pass	Pass	6
τ_5		4	19.19	1.80	2.00	16.6	Pass	Pass	Pass	
τ_3		2	12.79	2.00	2.00	12.6	Pass	Pass	Pass	
τ_4	3	2	12.79	0.90	2.50	12.2	Fail	Pass	Pass	2
τ_8		1	11.19	0.50	2.50	10.5	Pass	Pass	Pass	

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

Task	Part.	Para.	Period	GPU	CPU	Res. Time	Carry	Jitter	Mixed	Warps
τ_1		12	6.40	0.80	1.00	6.03	Pass	Pass	Pass	
τ_2	1	12	25.58	1.60	3.00	17.71	Pass	Pass	Pass	12
τ_7		6	8.99	0.80	2.00	8.32	Pass	Pass	Pass	
τ_6	2	6	7.99	1.00	1.00	7.45	Pass	Pass	Pass	6
τ_5		4	19.19	1.80	2.00	16.6	Pass	Pass	Pass	
τ_3		2	12.79	2.00	2.00	12.6	Pass	Pass	Pass	
τ_4	3	2	12.79	0.90	2.50	12.2	Fail	Pass	Pass	2
τ_8		1	11.19	0.50	2.50	10.5	Pass	Pass	Pass	

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

GPU-centric Scheduling (ILP-based partitioning)

Task	Part.	Para.	Period	GPU	CPU	Res. Time	Carry	Jitter	Mixed	Warps
τ_1		12	6.40	0.80	1.00	6.03	Pass	Pass	Pass	
τ_2	1	12	25.58	1.60	3.00	17.71	Fail	Pass	Pass	12
τ_7		6	8.99	0.80	2.00	8.32	Pass	Pass	Pass	
τ_6	2	6	7.99	1.00	1.00	7.45	Pass	Pass	Pass	6
τ_5		4	19.19	1.80	2.00	16.6	Pass	Pass	Pass	
τ_3		2	12.79	2.00	2.00	12.6	Pass	Pass	Pass	
τ_4	3	2	12.79	0.90	2.50	12.2	Fail	Pass	Pass	2
τ_8		1	11.19	0.50	2.50	10.5	Pass	Pass	Pass	

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

20

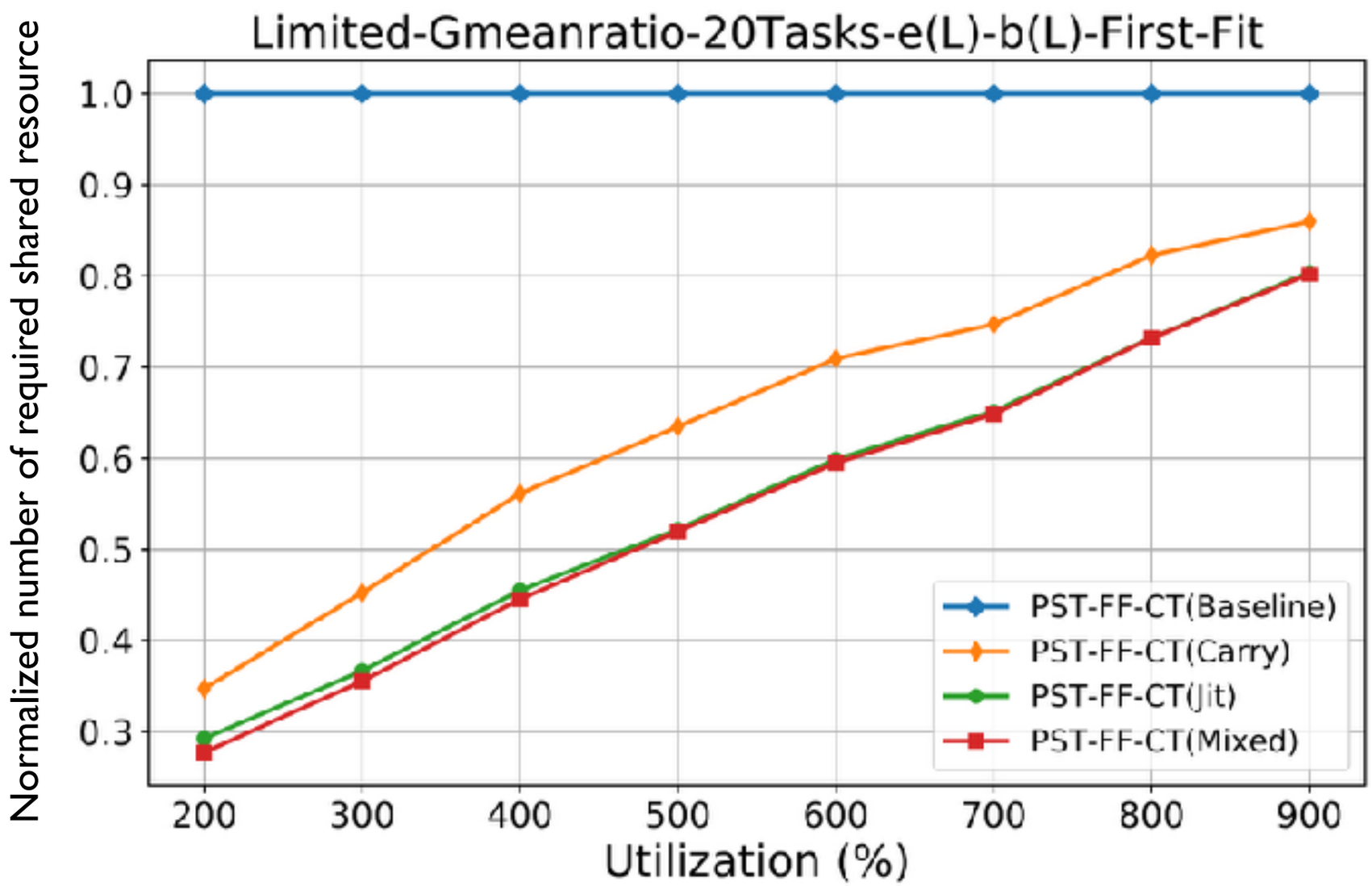
Task	Part.	Para.	Period	GPU	CPU	Res. Time	Carry	Jitter	Mixed	Warps
τ_1		12	6.40	0.80	1.00	6.03	Pass	Pass	Pass	
τ_2	1	12	25.58	1.60	3.00	17.71	Pass	Pass	Pass	12
τ_7		6	8.99	0.80	2.00	8.32	Pass	Pass	Pass	
τ_6	2	6	7.99	1.00	1.00	7.45	Pass	Pass	Pass	6
τ_5		4	19.19	1.80	2.00	16.6	Pass	Pass	Pass	
τ_3		2	12.79	2.00	2.00	12.6	Pass	Pass	Pass	
τ_4	3	2	12.79	0.90	2.50	12.2	Fail	Pass	Pass	2
τ_8		1	11.19	0.50	2.50	10.5	Pass	Pass	Pass	

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

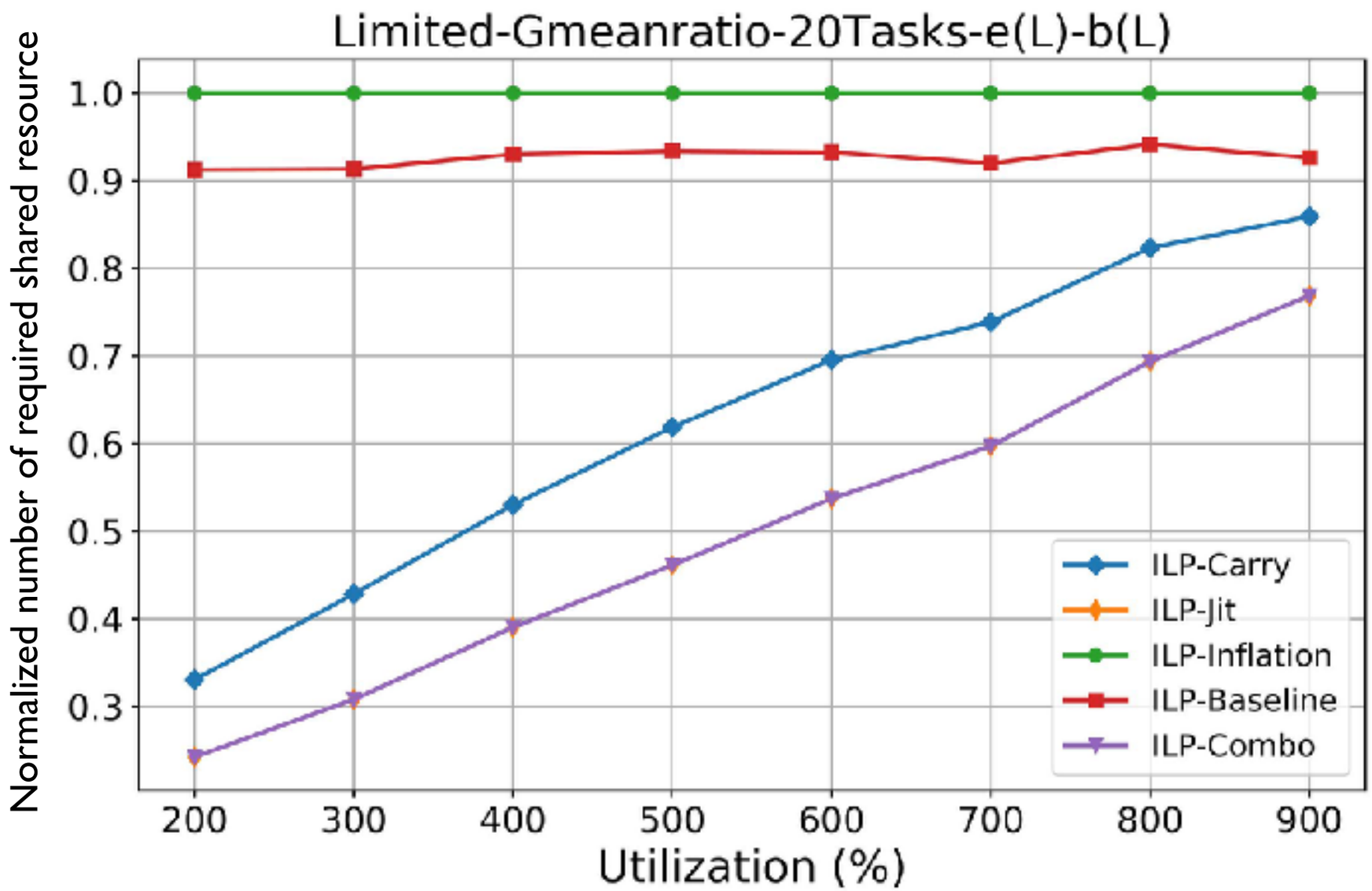
Simulation evaluation

- 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).



ILP-based approaches and representative approaches comparison



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!