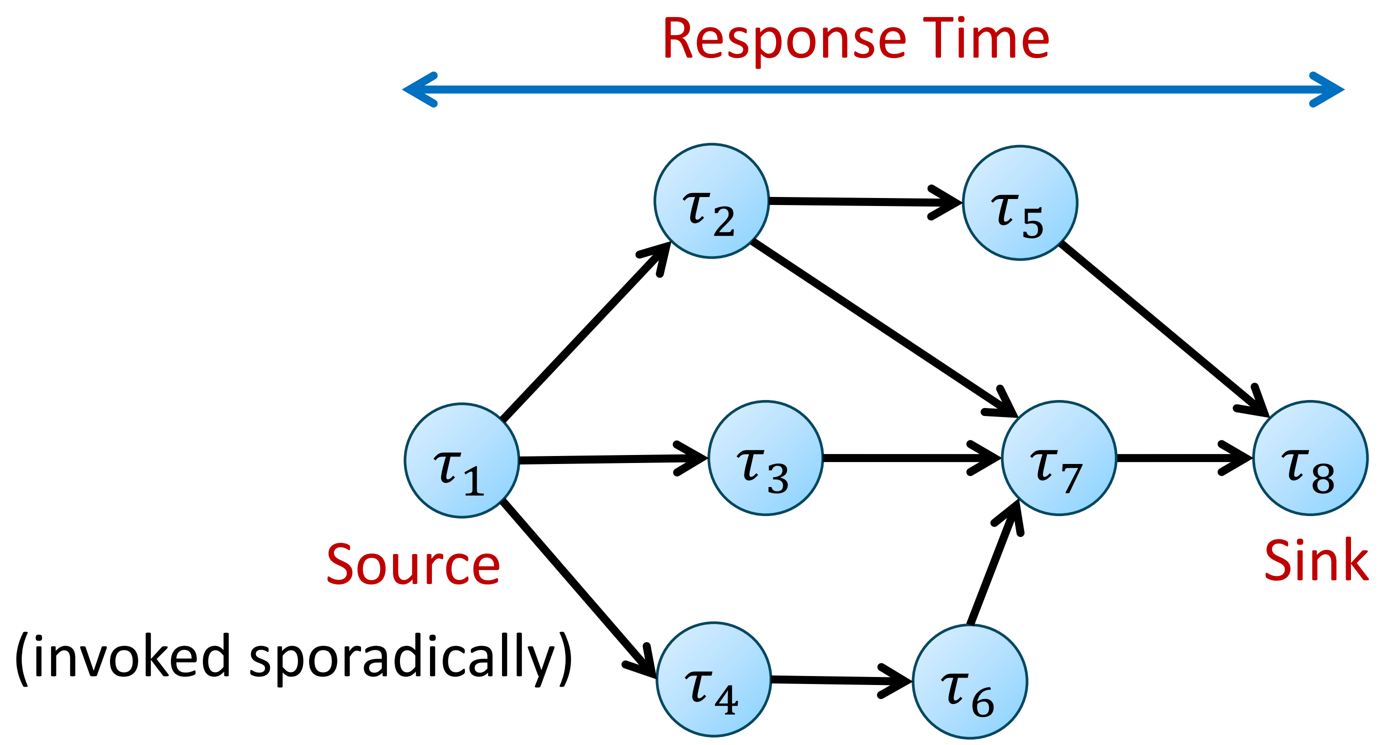
Scheduling Processing Graphs of Gang Tasks on Heterogeneous Platforms

Shareef Ahmed, Denver Massey and James H. Anderson

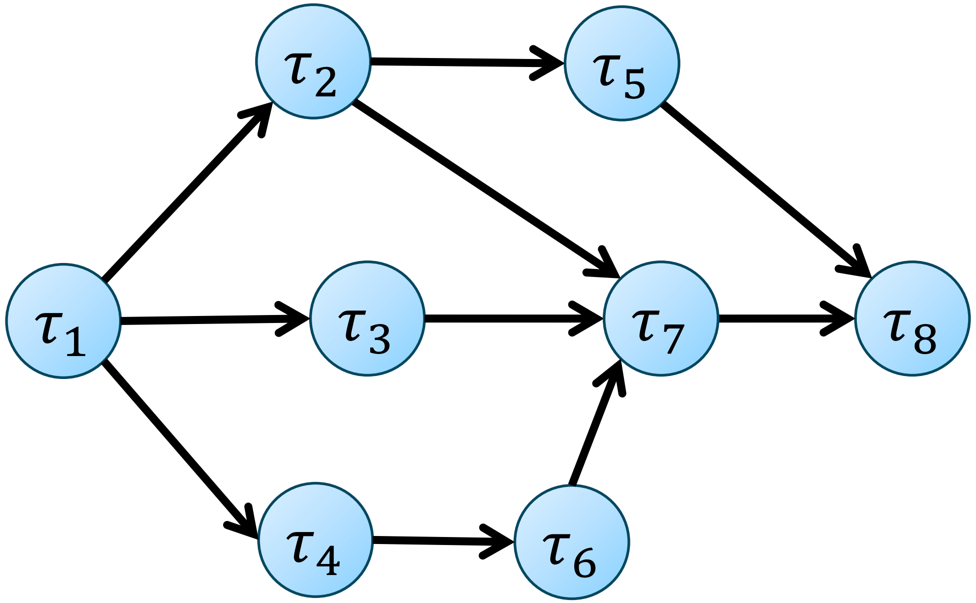
THE UNIVERSITY
of NORTH CAROLINA
at CHAPEL HILL

Processing Graphs

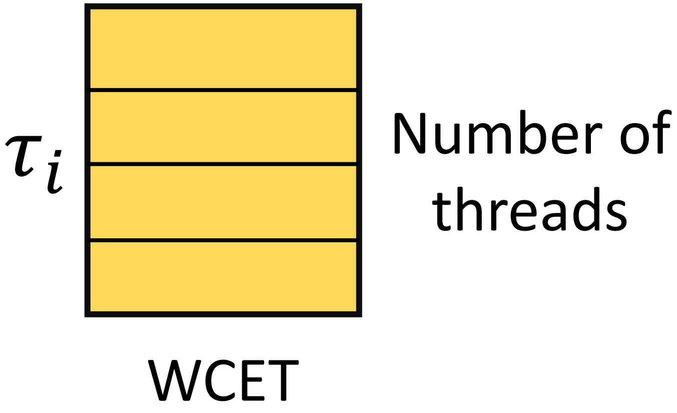


- Node = Task
- Edge = Precedence constraint
- Goal:
  - Response time ≤ Deadline

Processing Graphs of Gang Nodes

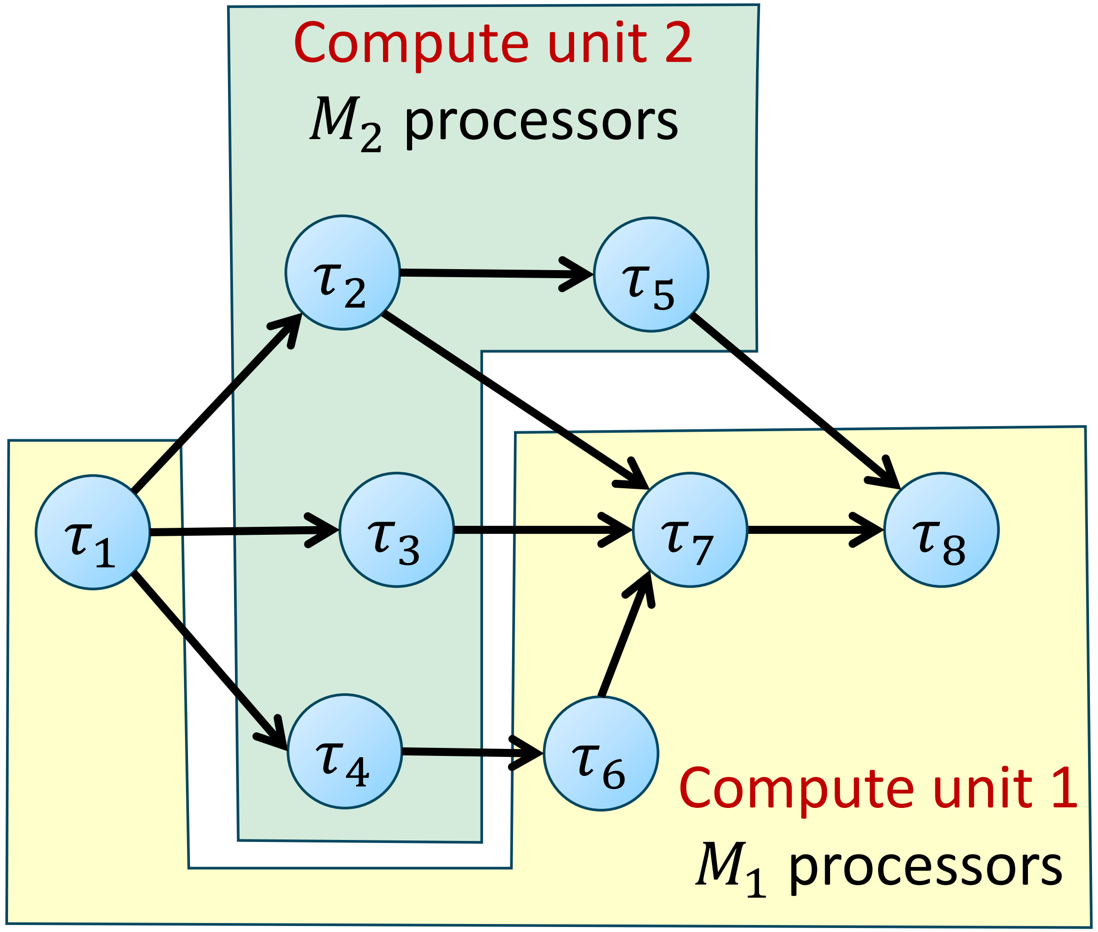


- Nodes are rigid gang tasks



- Rigid = Same number of threads for all jobs of τ_i

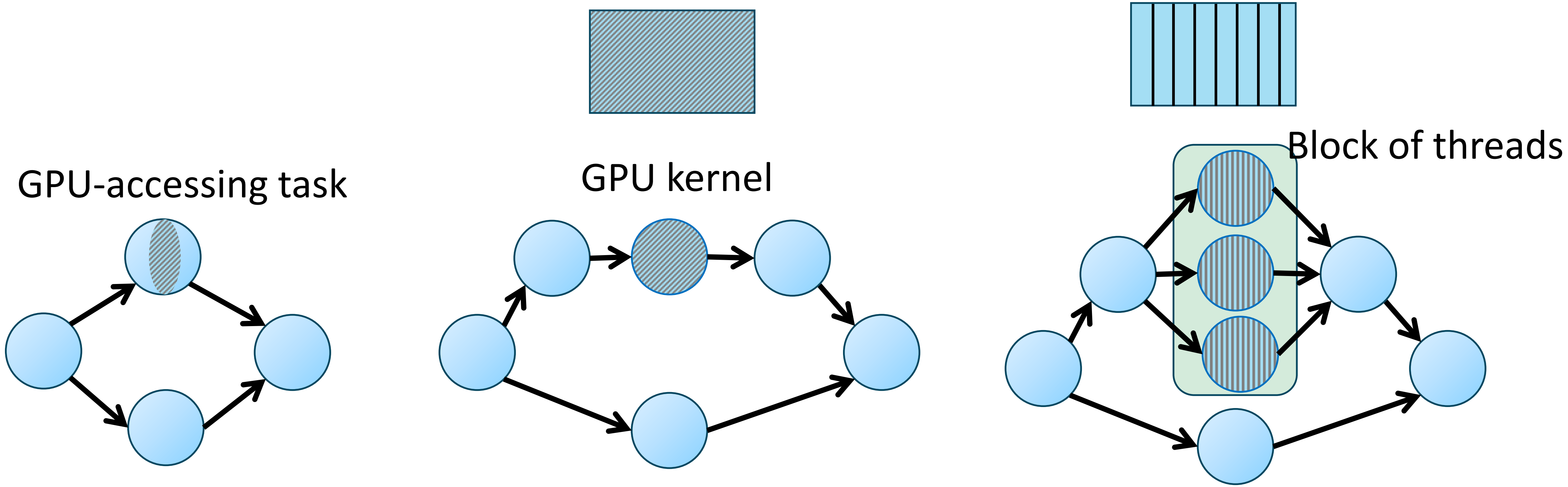
Heterogenous Compute Platform



- Multiple compute units
- Each node assigned to a compute unit
- Each compute unit has multiple same-speed processors

A Use Case

Scheduling processing graphs on multicore+GPU



Problem

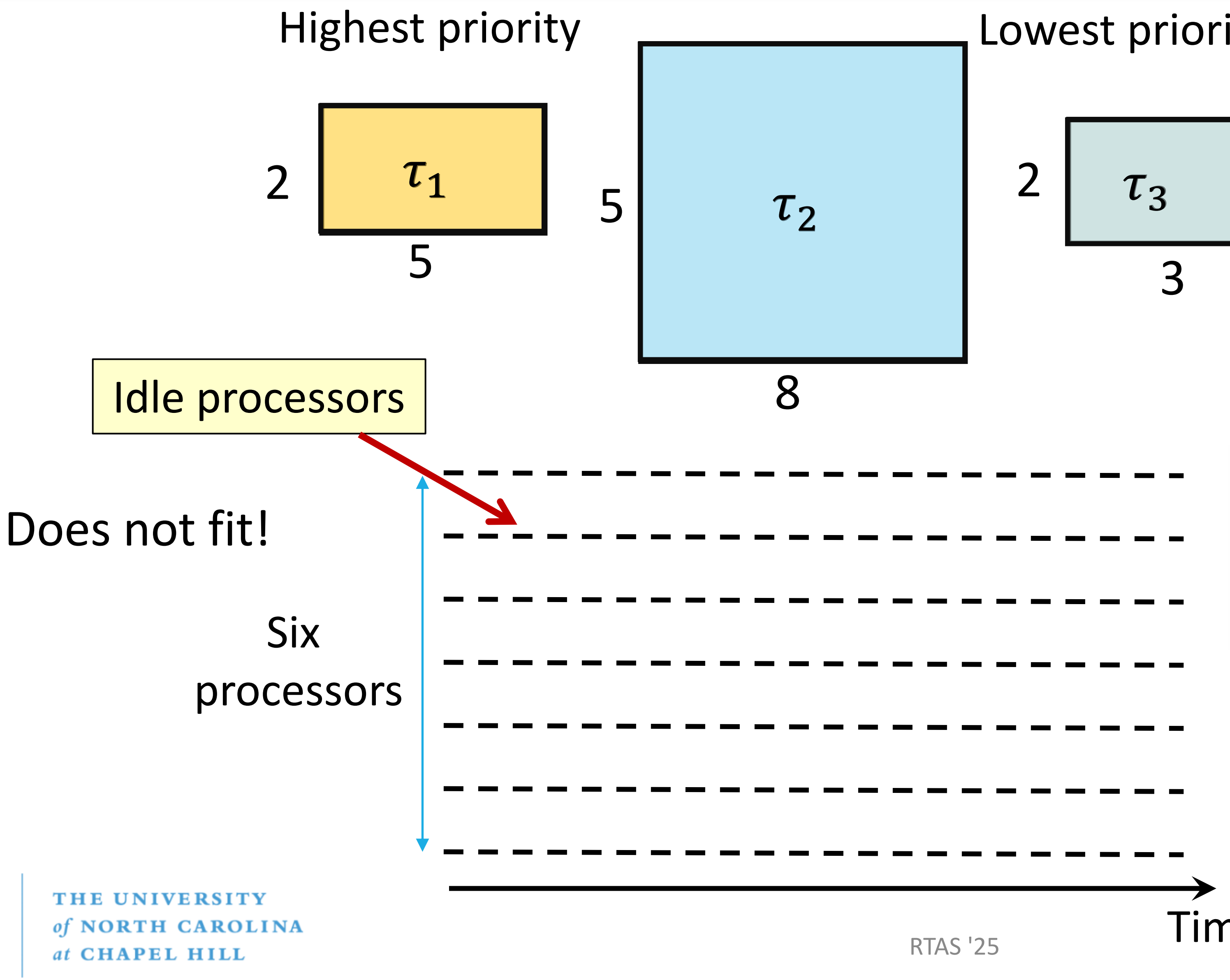
Determine response-time bound of DAGs formed by gang tasks

Assumption: Constrained deadline

Scheduling: Work-conserving, Semi work-conserving

Each DAG receives dedicated number of processors on each compute unit

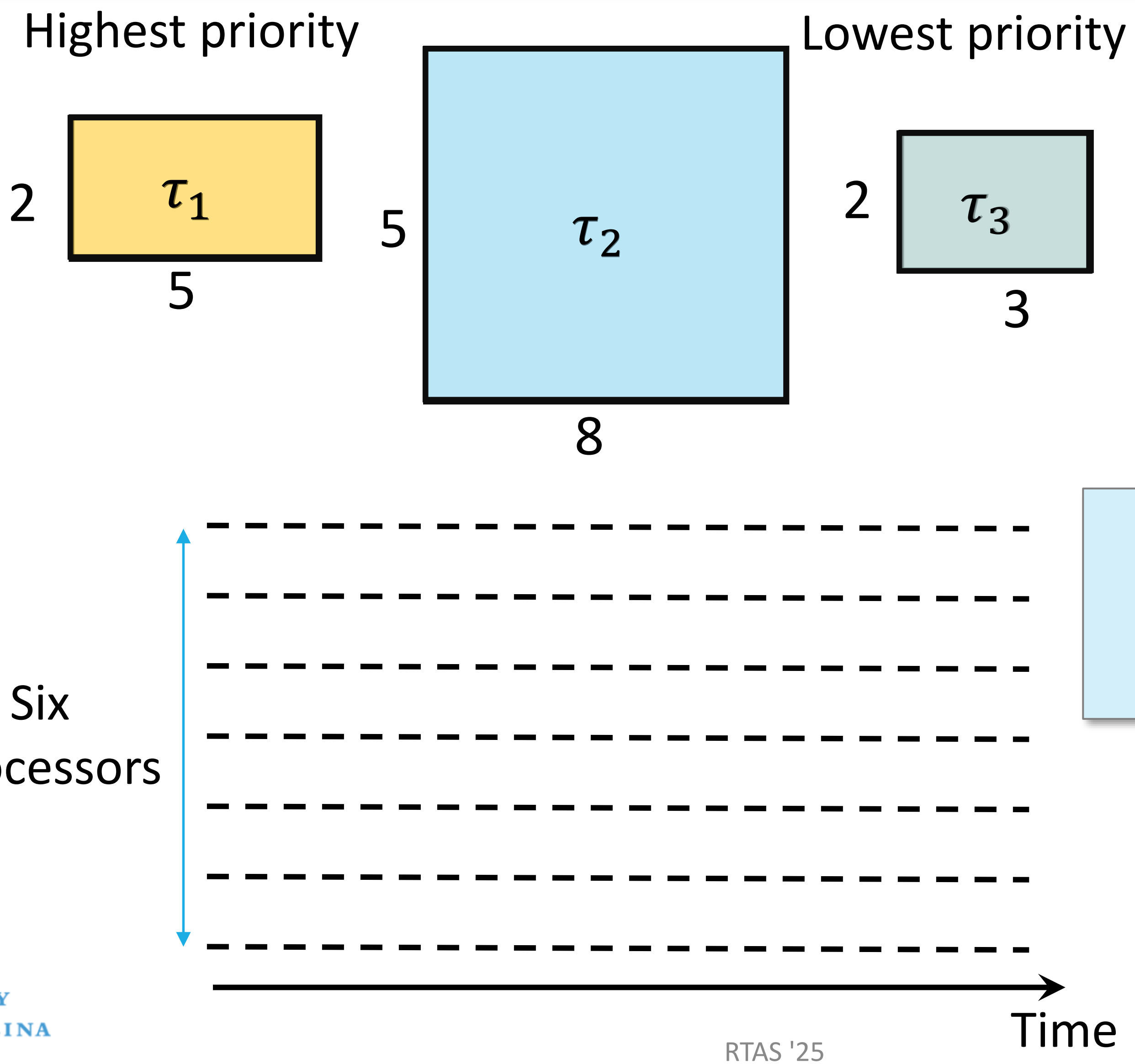
Work-Conserving Scheduling



Try out all ready jobs

∀ unscheduled ready job :
job’s processor requirement > number of idle processors

Semi Work-Conserving Scheduling



Try until a job does not fit

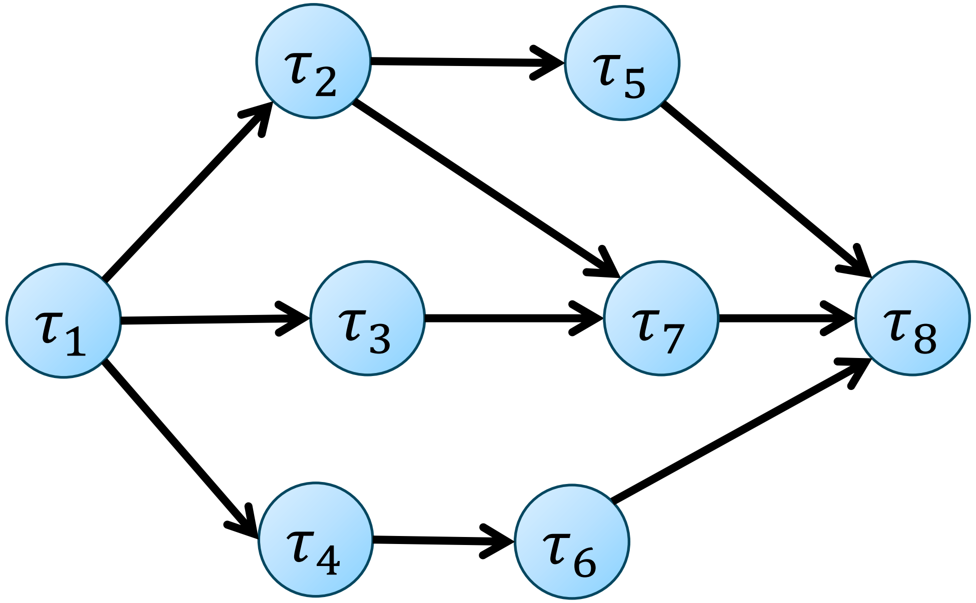
∃ unscheduled ready job :
job’s processor requirement >
number of idle processors

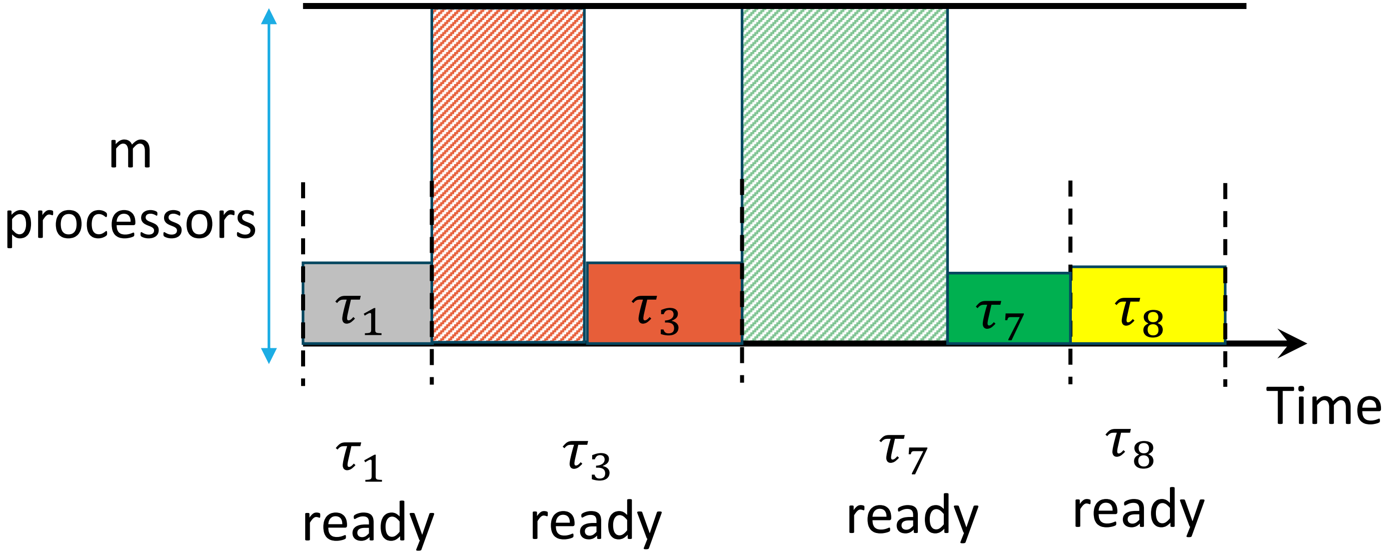
Why Semi Work-Conserving Scheduling?

Scheduling in NVIDIA GPU is semi work-conserving when all GPU work is submitted from the same address space (and some more constraints)

1. Amert et al., RTSS 2017
2. Bakita and Anderson, RTAS 2024

Response-Time Bound



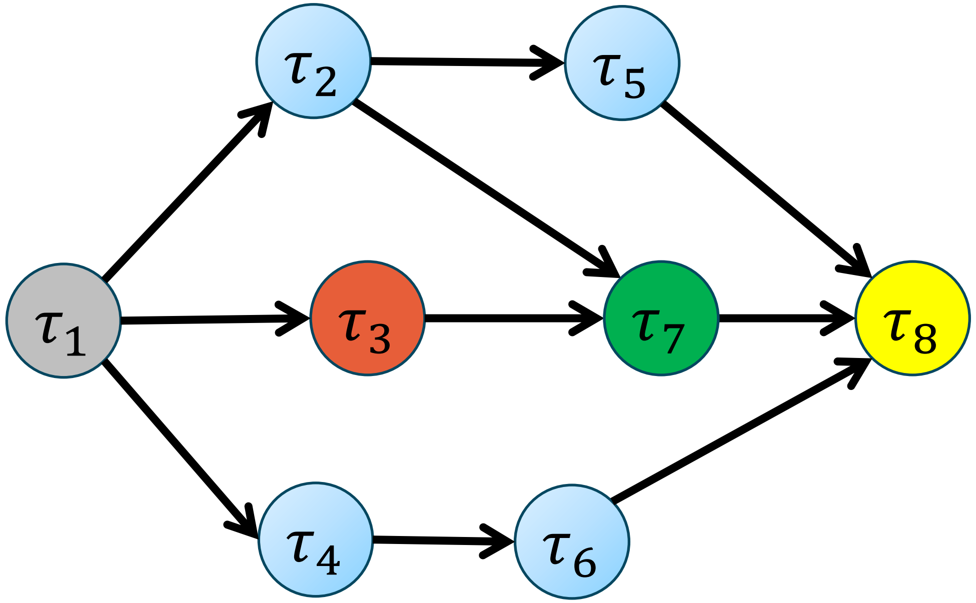


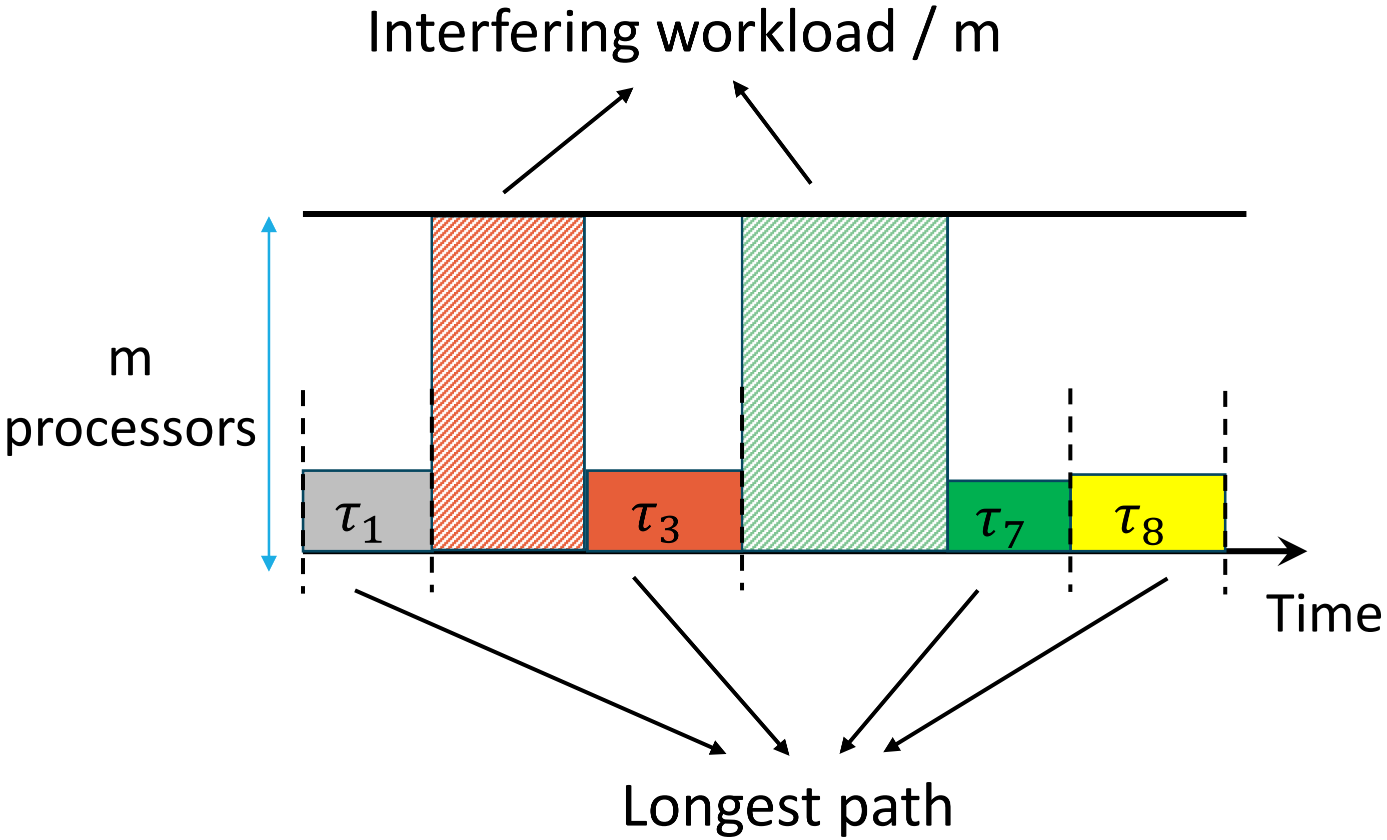
Assumption 1: One compute unit

Assumption 2: Sequential node (one thread per task)

Graham, Siam J. of Appl. Math., 1969

Response-Time Bound



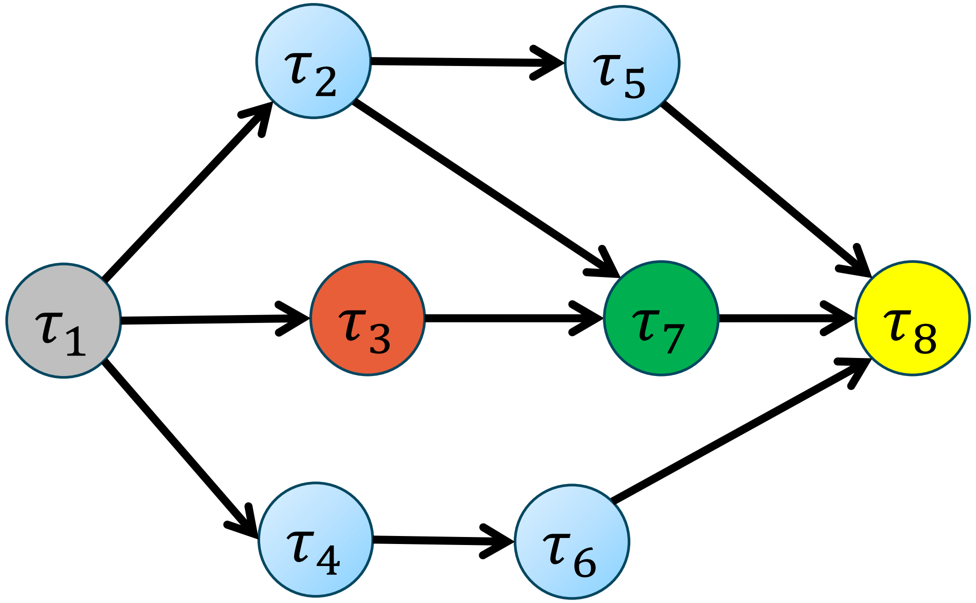


Assumption 1: One compute unit

Assumption 2: Sequential node (one thread per task)

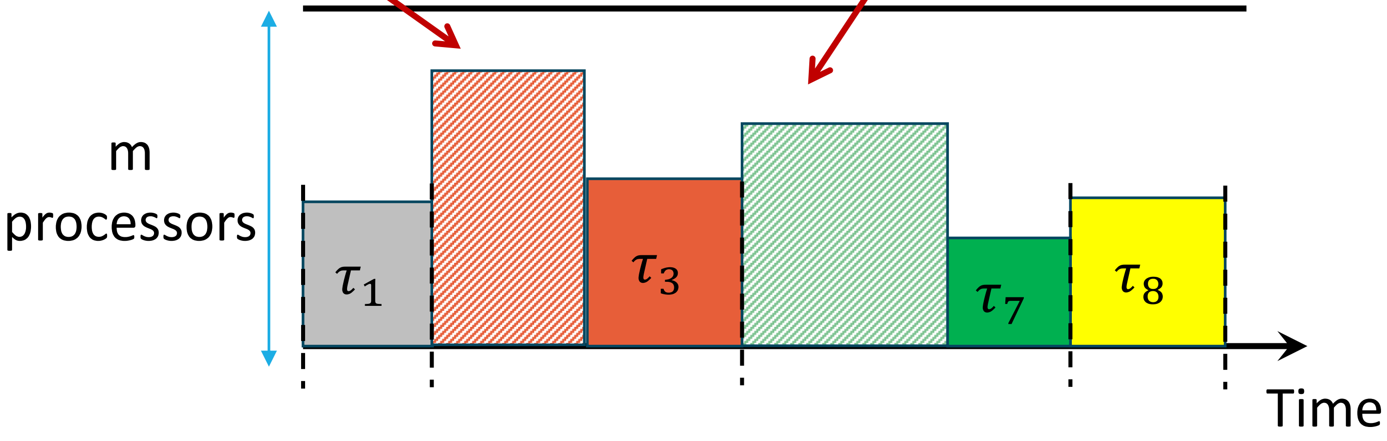
Graham, Siam J. of Appl. Math., 1969

Response-Time Bound



i. Some processors can be idle

ii. Different number of idle processors



Assumption 1: One compute unit

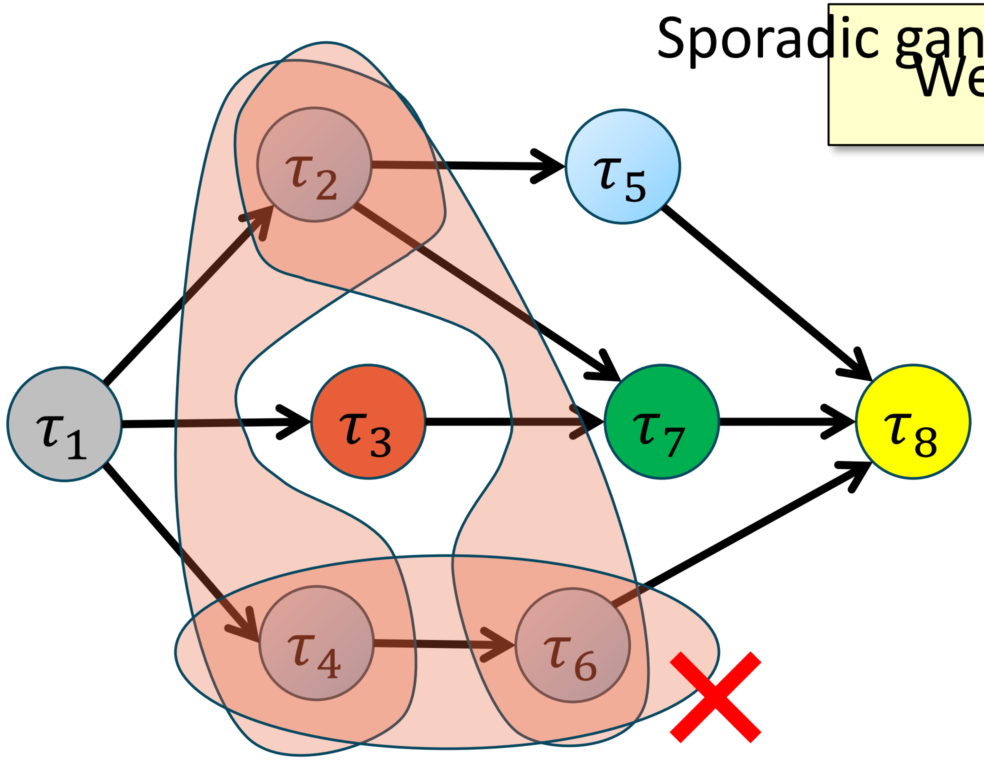
Assumption 2: ~~Sequential~~ node (~~one thread~~ per task)
Gang Multiple threads

Response-Time Bound

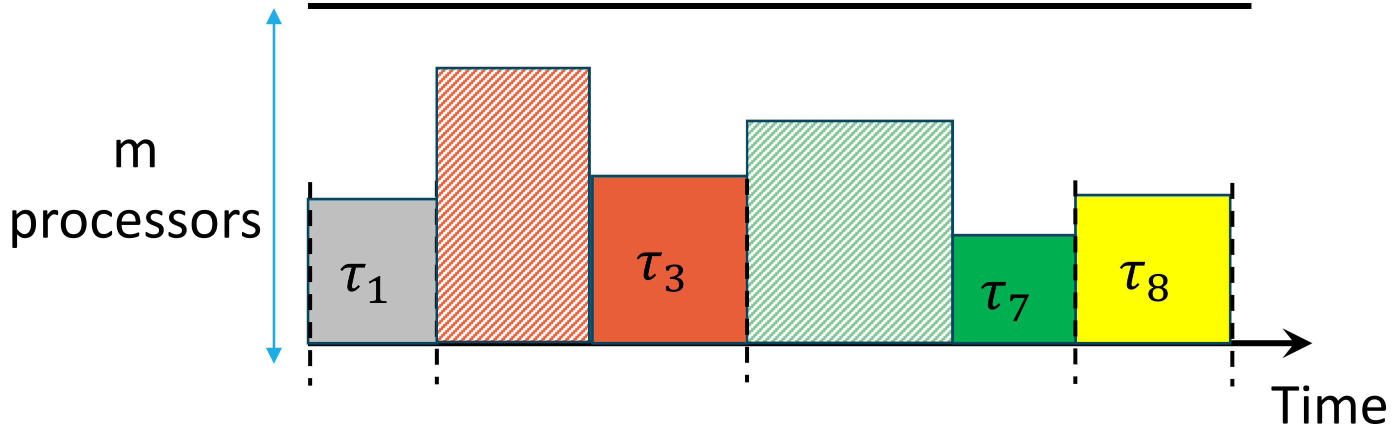
Step 1: Determine the minimum number of busy processors when τ_i is ready but unscheduled

Sporadic gang task: Dong and Liu, RTSS 2017

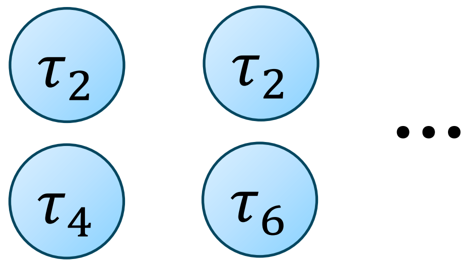
We give a dynamic programming algorithm



Assumption 1: One compute unit

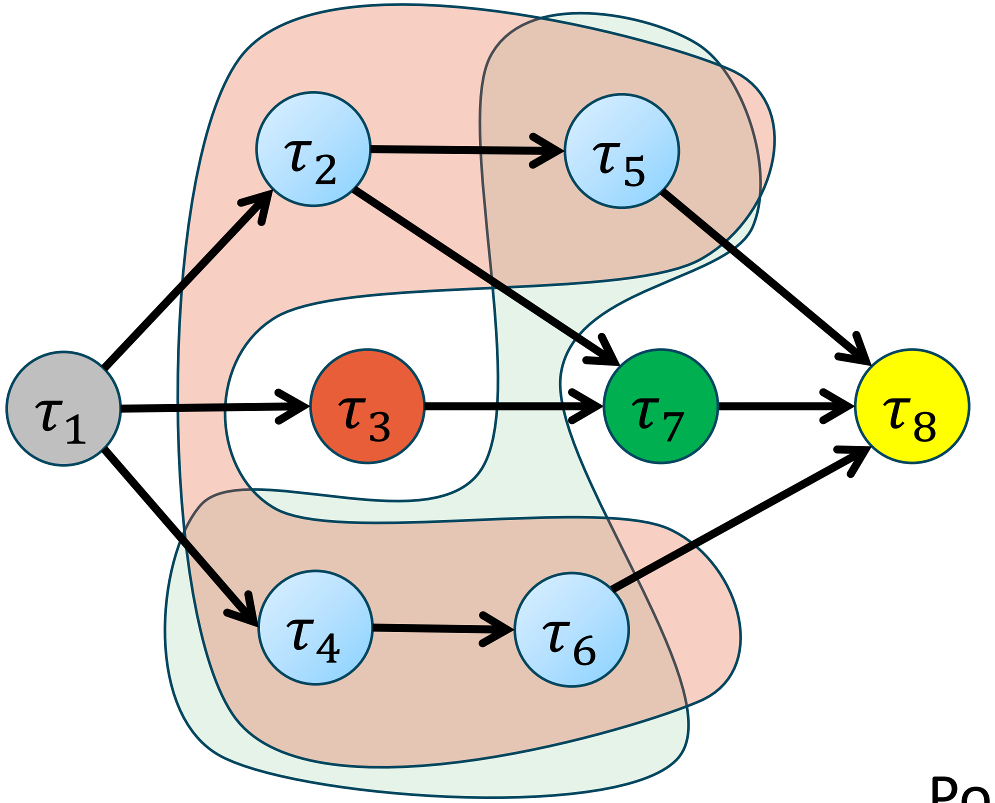


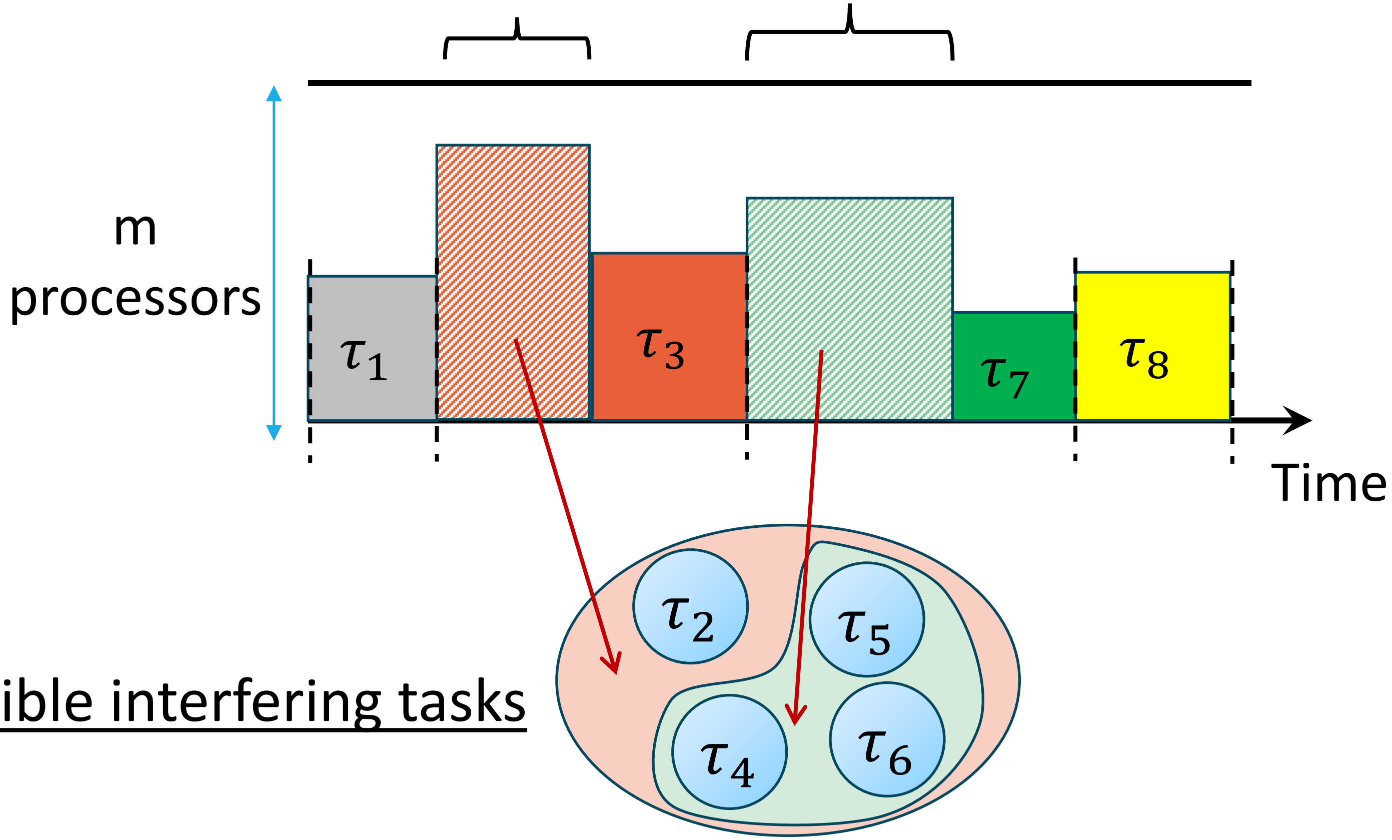
Possible execution



Response-Time Bound

Step 2: Upper bound total interference time



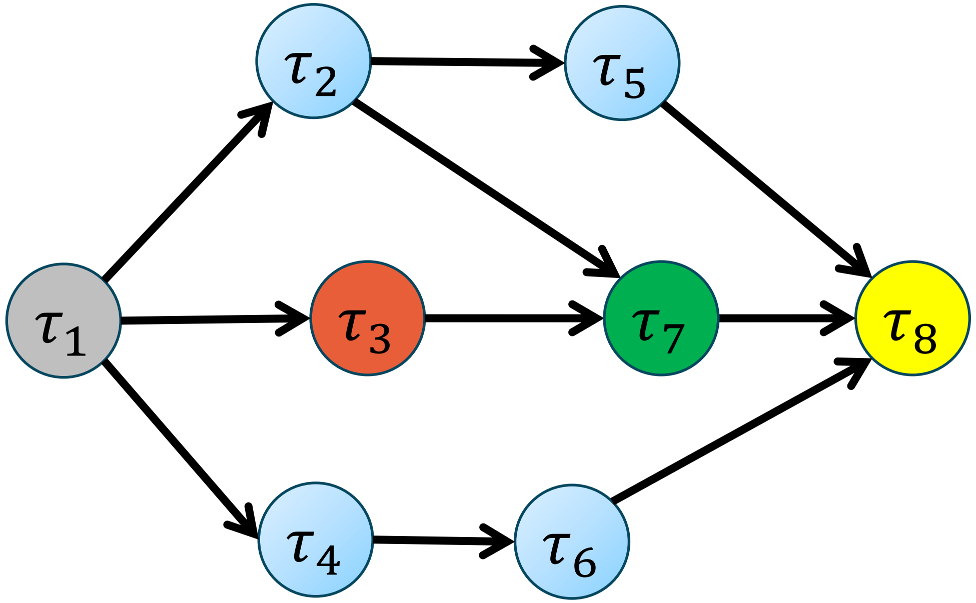


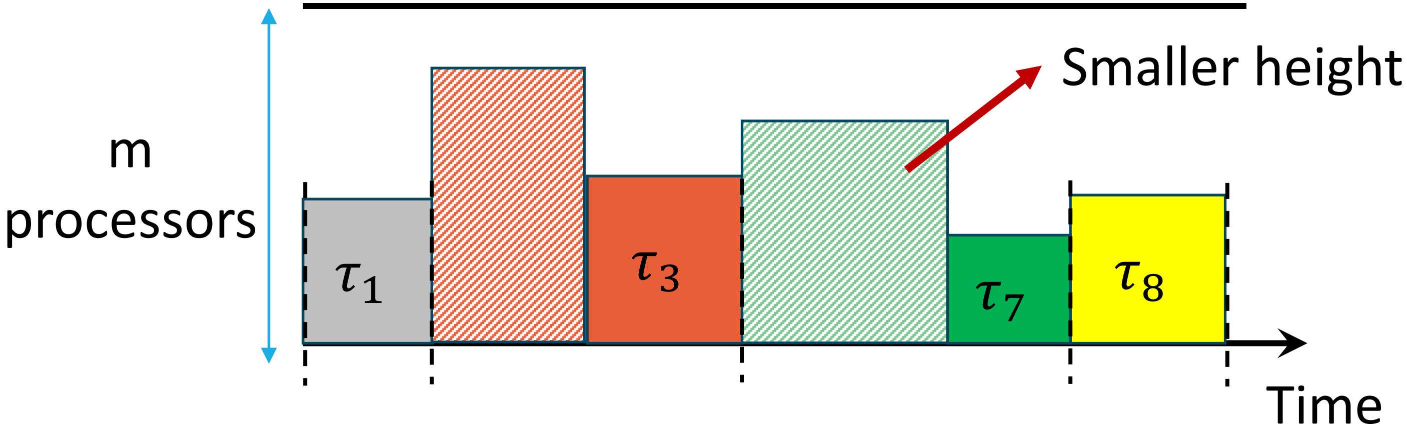
Possible interfering tasks

Response-Time Bound

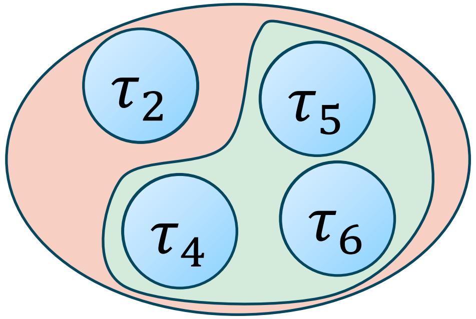
Any path can be a critical path

Determine a set of nodes (not necessarily on a path) that upper bounds interference time

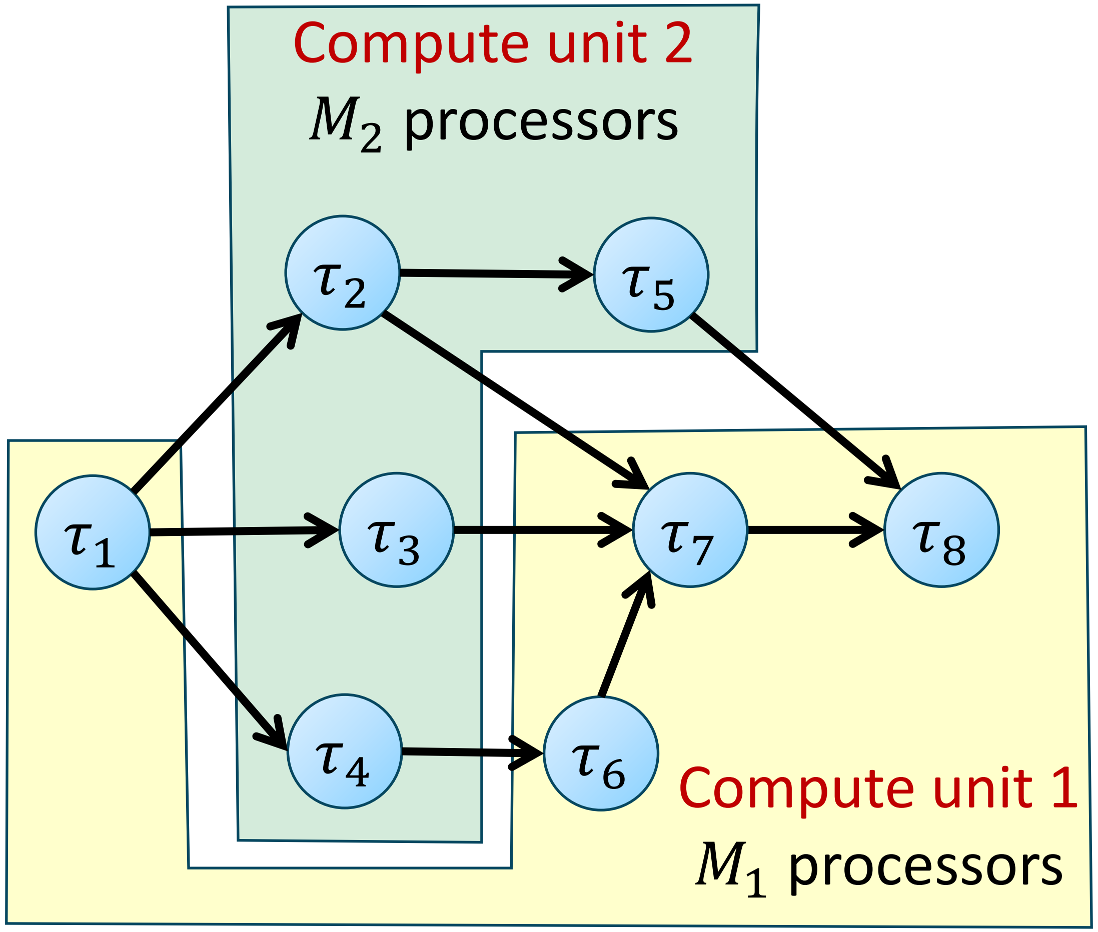




Possible interfering tasks

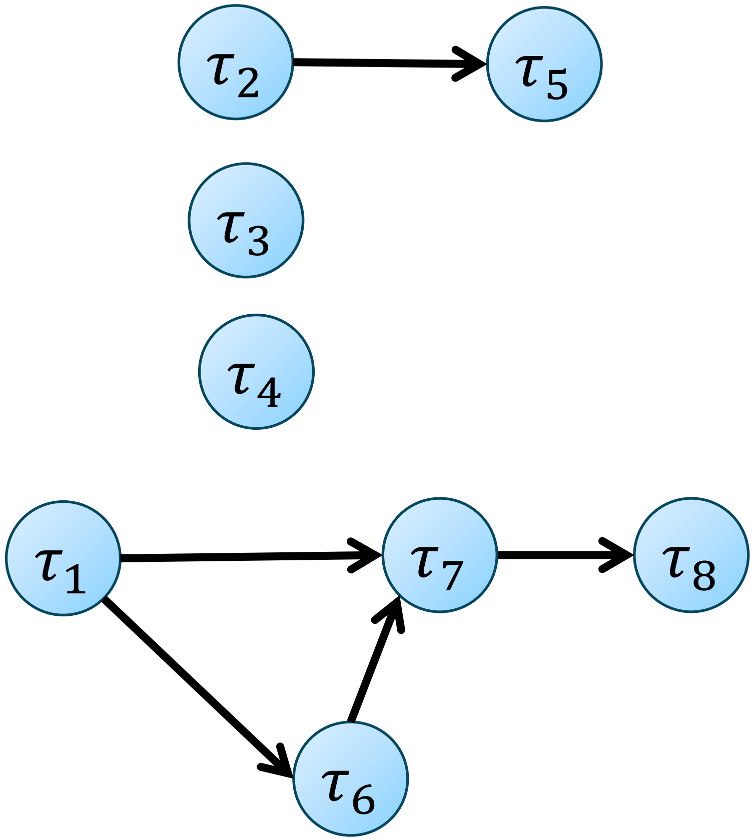


Response-Time Bound



Compute unit 2
M_2 processors

Compute unit 1
M_1 processors



Assumption 1: ~~One~~ Multiple compute ~~unit~~ units

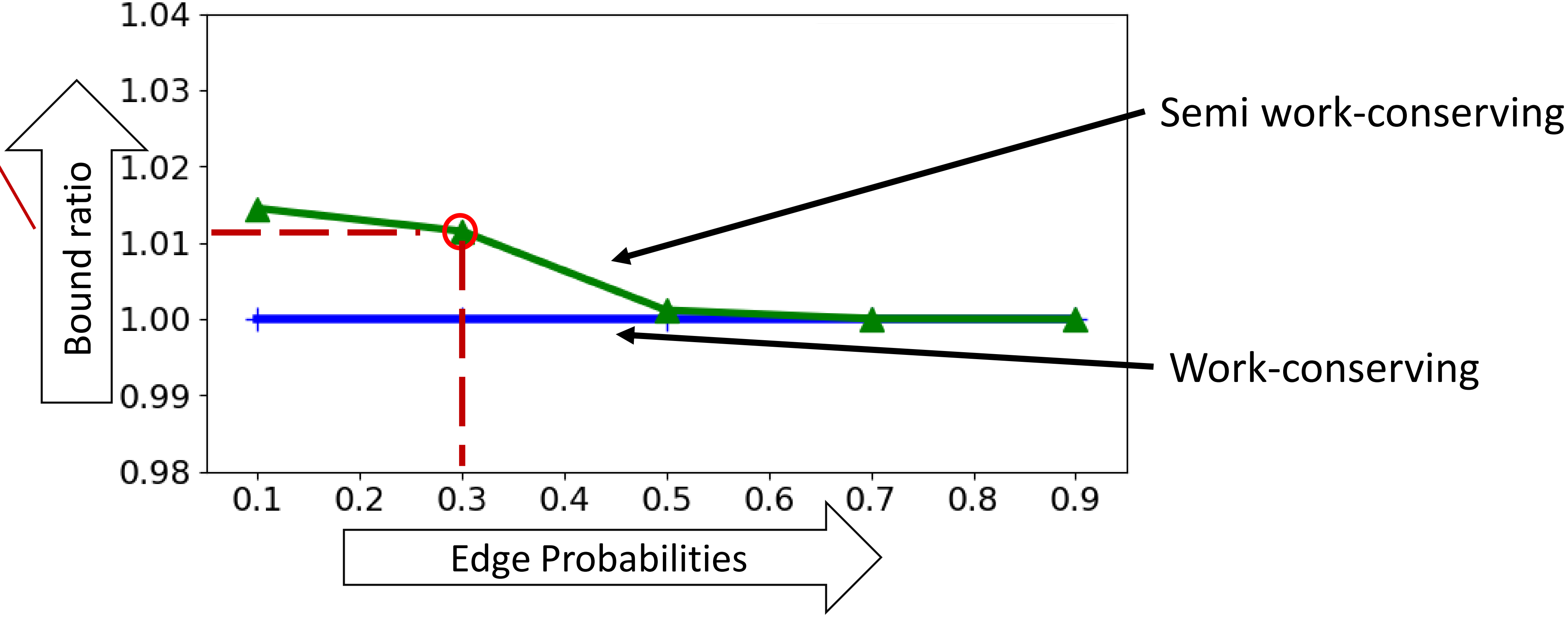
Scheduling in different compute units can be different

Evaluation

$$\frac{\text{Bound under X}}{\text{Bound under work-conserving}}$$

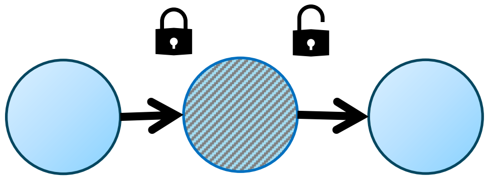
Work-conserving vs. Semi work-conserving scheduling

≥ 1 means higher bound



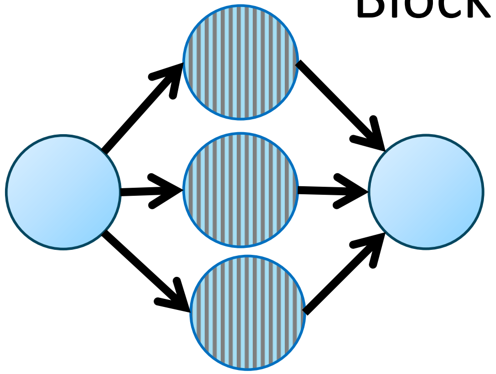
Evaluation

GPU as a shared resource vs. scheduling platform



With locking

CPU-only DAG response-time bound



Without locking

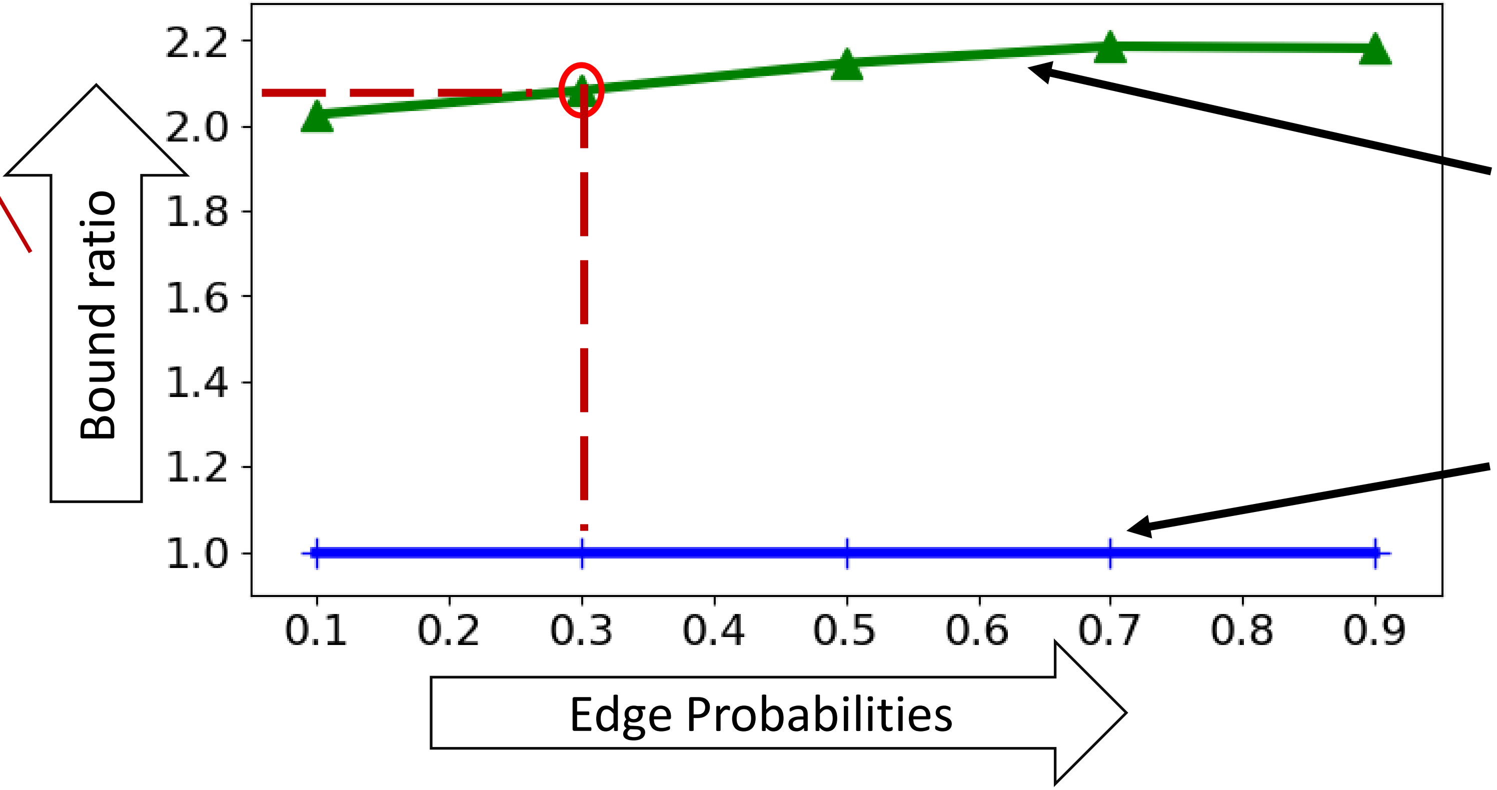
DAG of gang tasks response-time bound

Evaluation

$$\frac{\text{Bound under X}}{\text{Bound under default scheduler}}$$

Locking-based GPU access vs. Default GPU scheduling

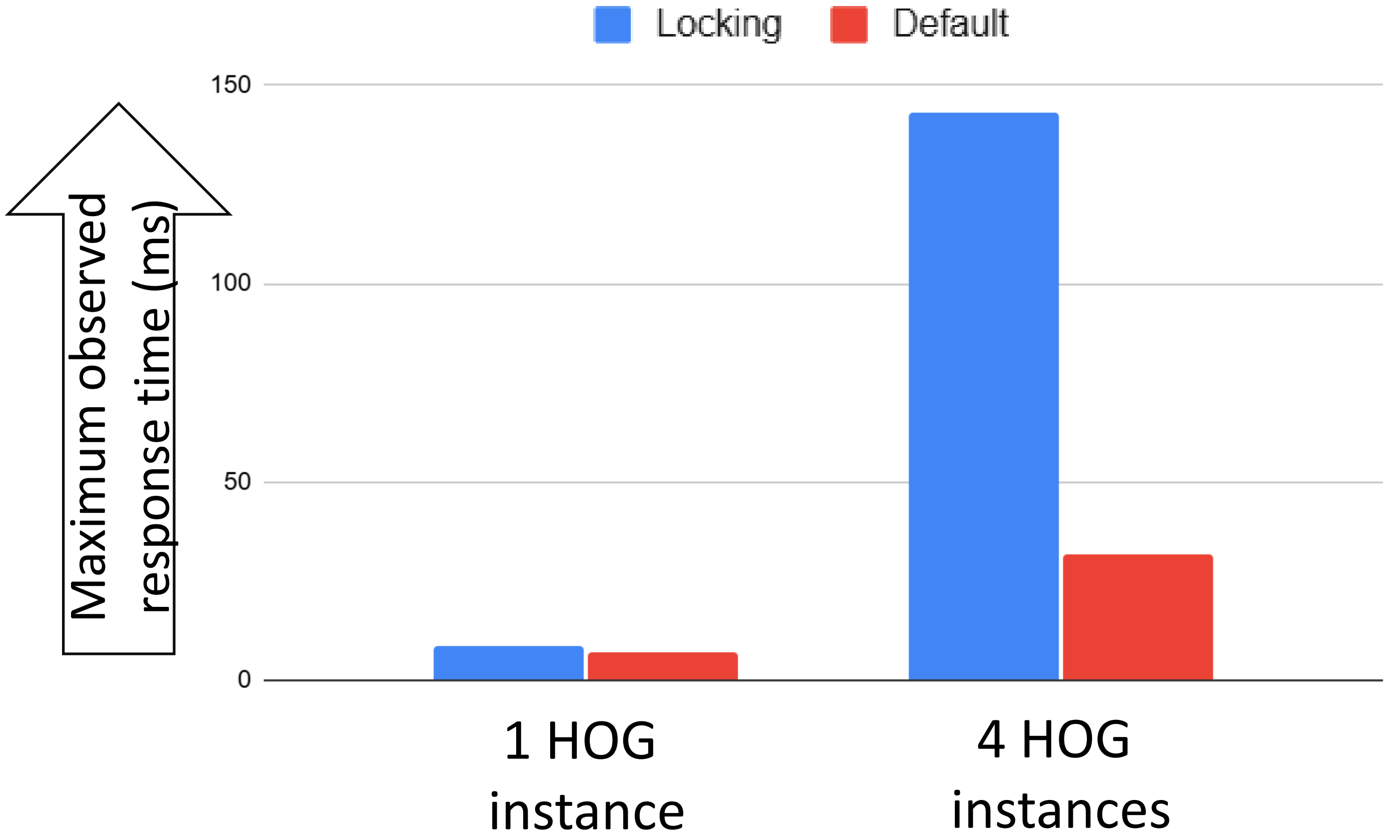
≥ 1 means higher bound



Locking-based
[He et al., RTSS 2022]

Default

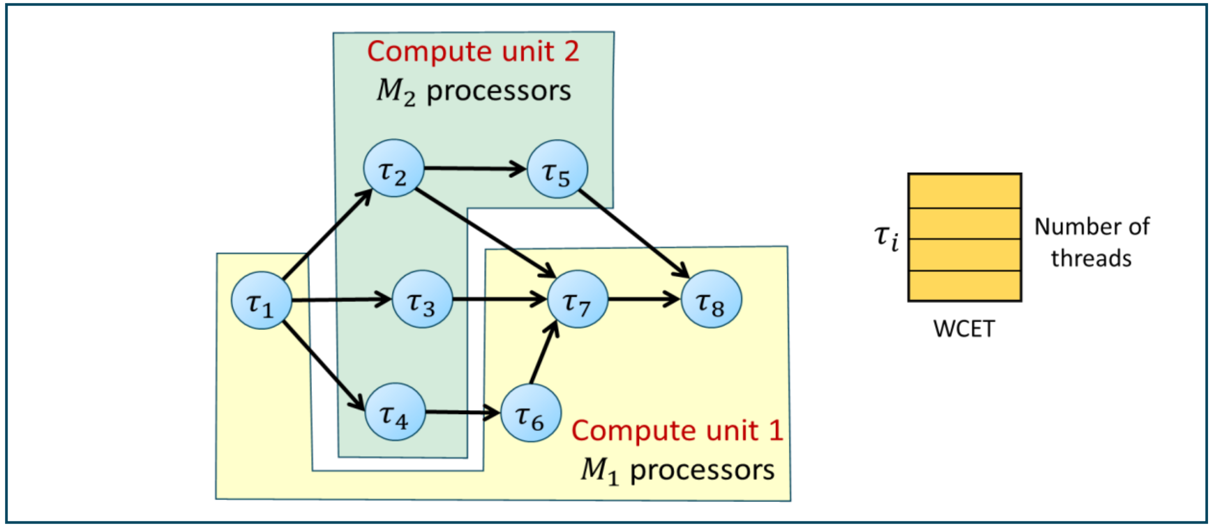
Evaluation



Histogram of Oriented Gradients

GPU partitioning using `libsmctrl`
[Bakita and Anderson, RTAS 2023]

Conclusion & Thank You!

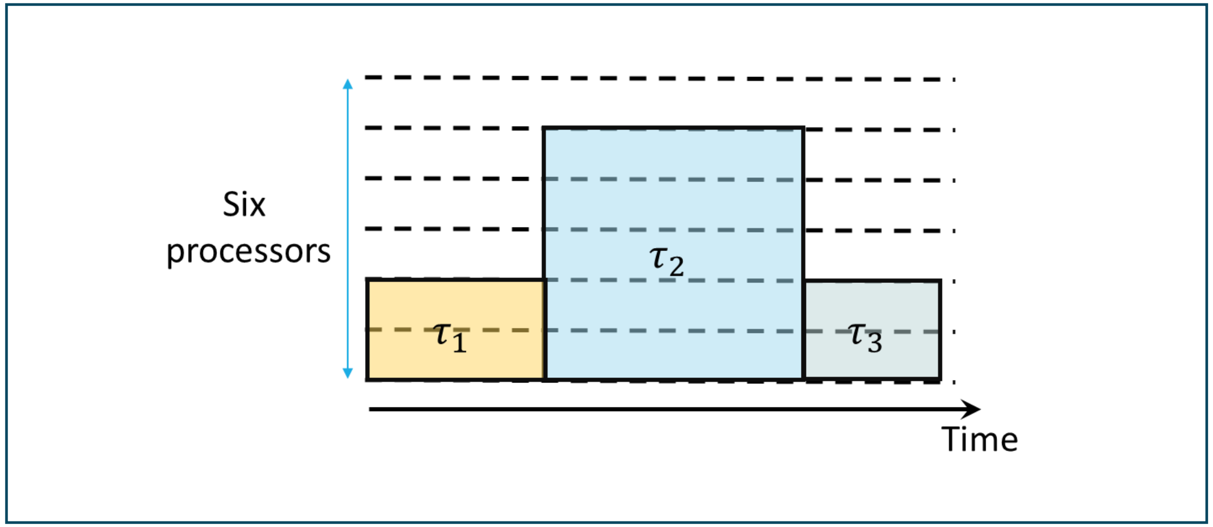


Compute unit 2
M_2 processors

Compute unit 1
M_1 processors

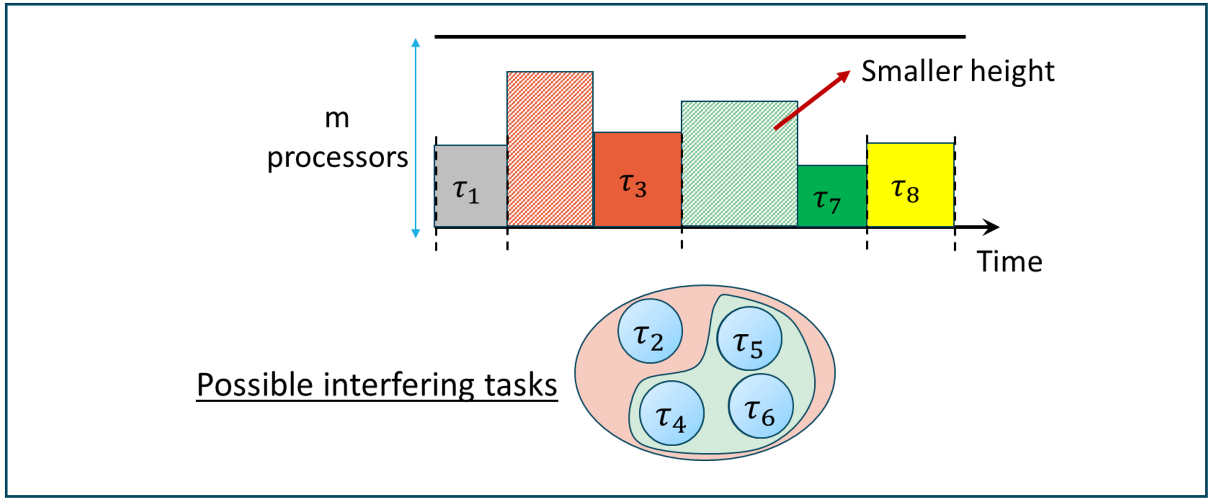
Number of threads

WCET



Six processors

Time

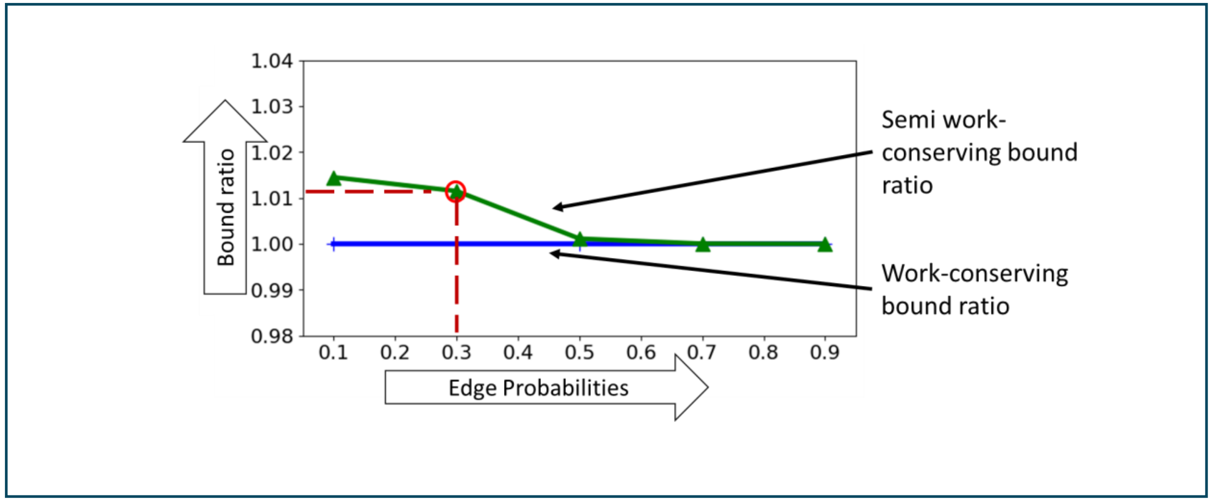


m processors

Smaller height

Time

Possible interfering tasks



Semi work-conserving bound ratio

Work-conserving bound ratio