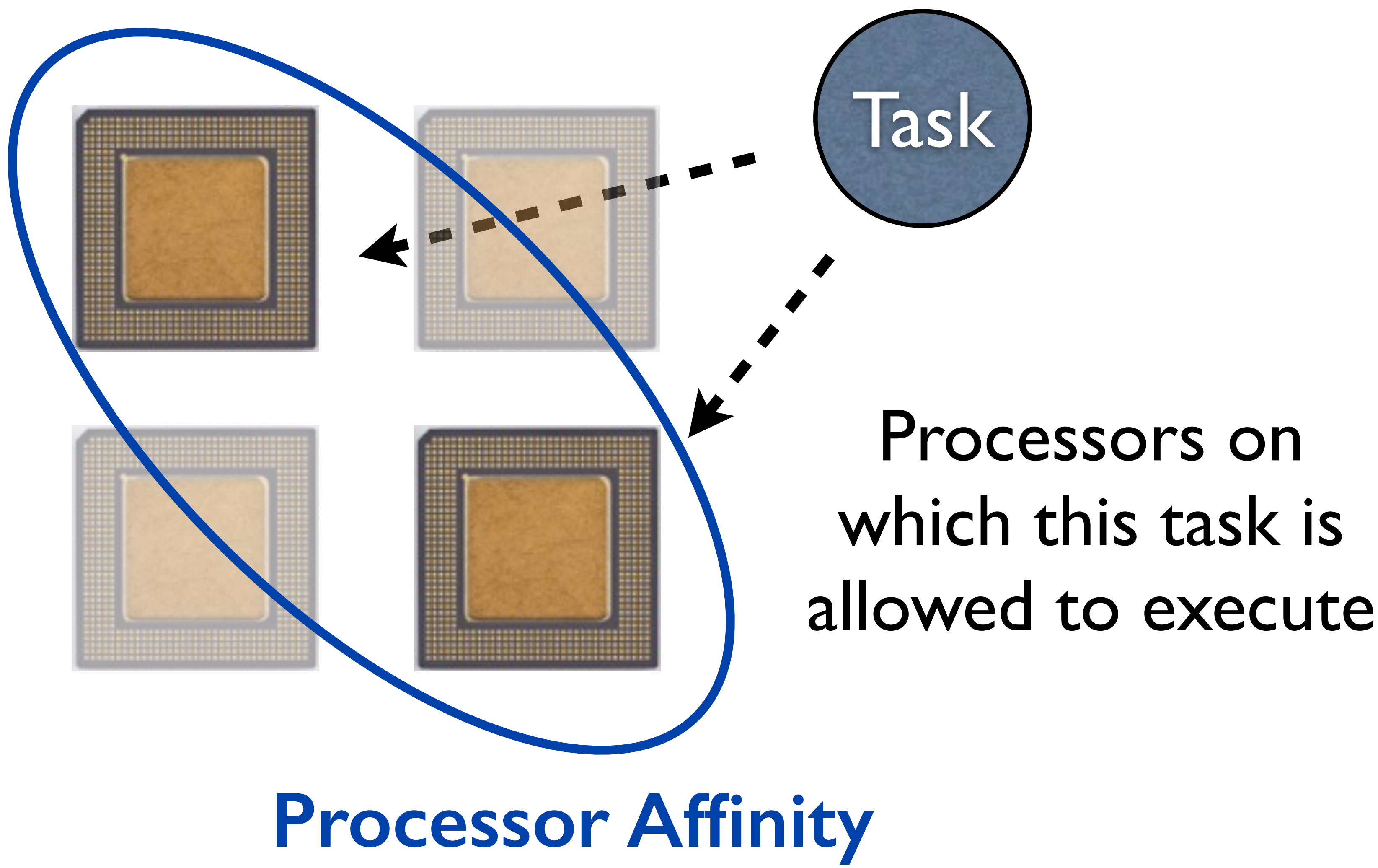
Linux's Processor Affinity API, Refined: *Shifting* Real-Time Tasks towards Higher Schedulability

Felipe Cerqueira, Arpan Gujarati, Björn Brandenburg

Max Planck Institute for Software Systems

Task Migration under Current RTOSs: **Arbitrary Processor Affinities (APA)**



Standard API provided by Linux, QNX, VxWorks, …

Use Cases of Processor Affinities

Security	Isolate tasks to prevent cache side-channel attacks
Cache Locality	Avoid migration-related cache misses
Energy Efficiency	Restrict non-critical tasks to small, power-efficient cores

and more…

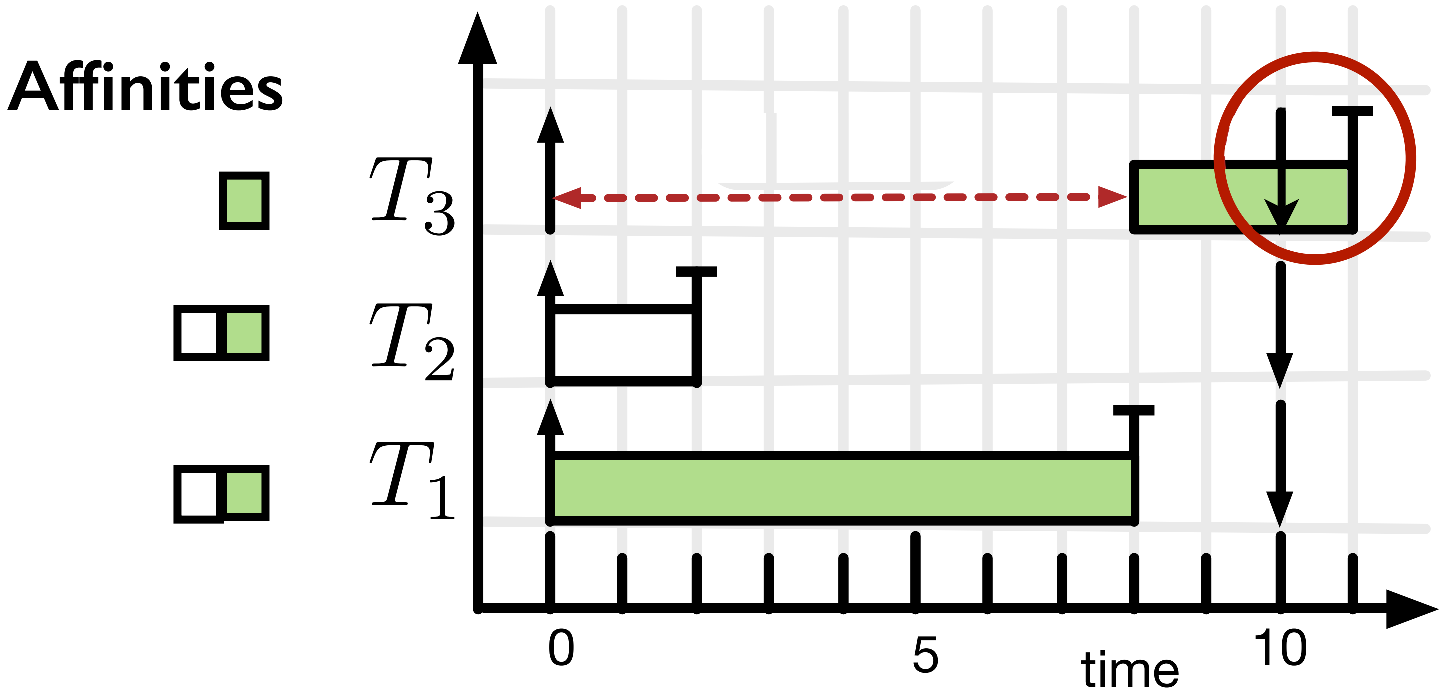
Use Cases of Processor Affinities

- Security
- Cache Locality
- Energy Efficiency

and more…

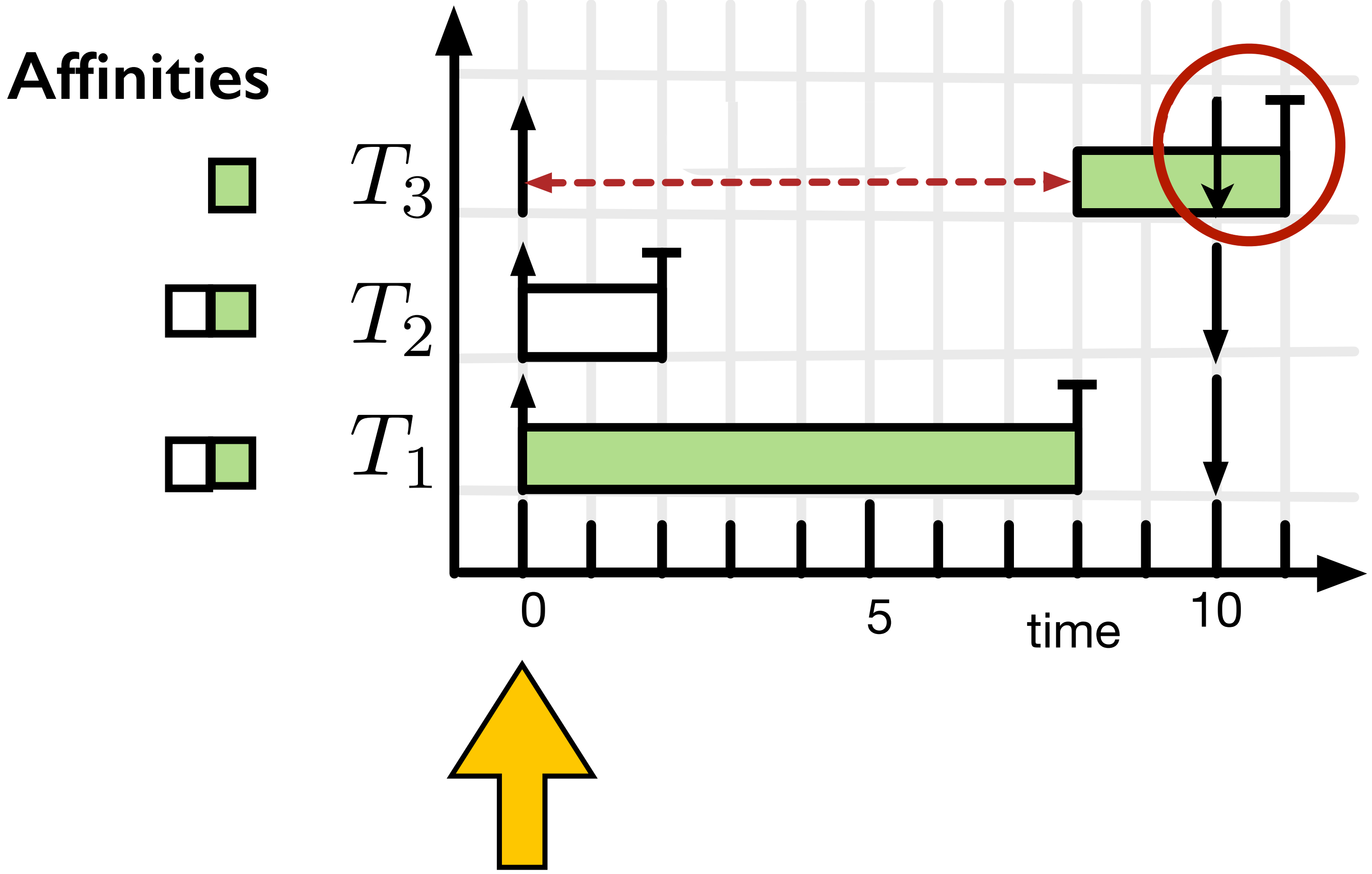
Application-specific affinity requirements may render the system unschedulable.

Affinities can cause Deadline Miss



Linux

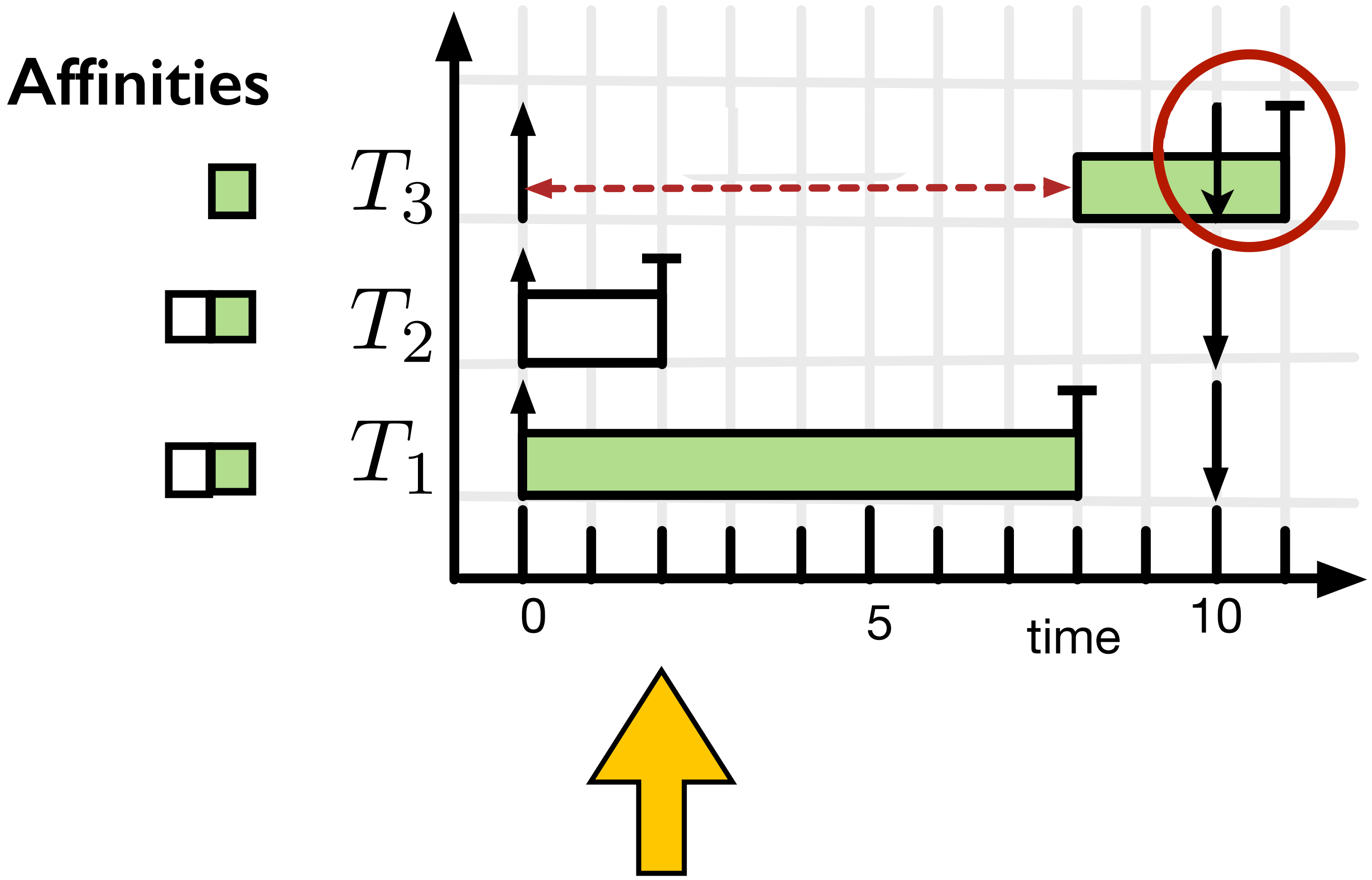
Affinities can cause Deadline Miss



Tasks are released

Linux

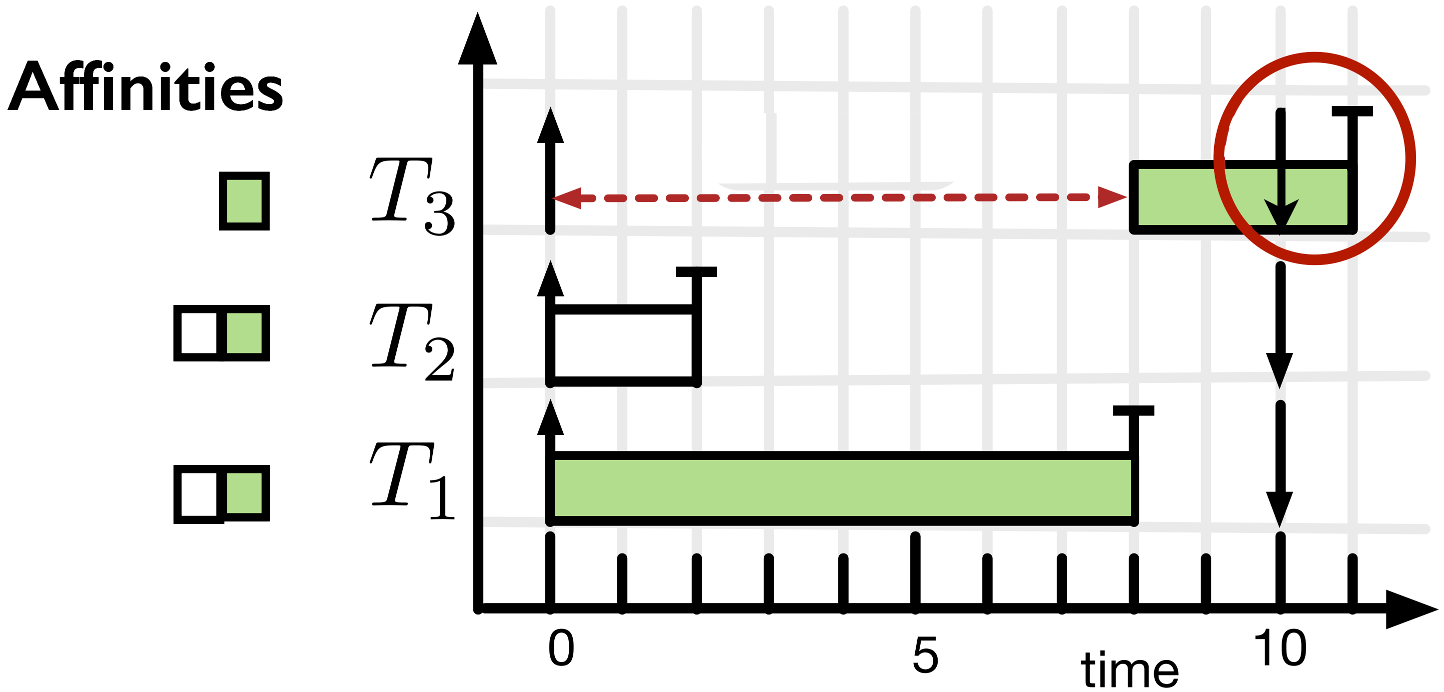
Affinities can cause Deadline Miss



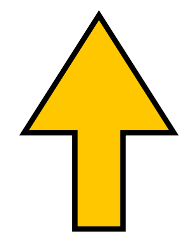
Processor idles, but Task 3 cannot execute there

Linux

Affinities can cause Deadline Miss

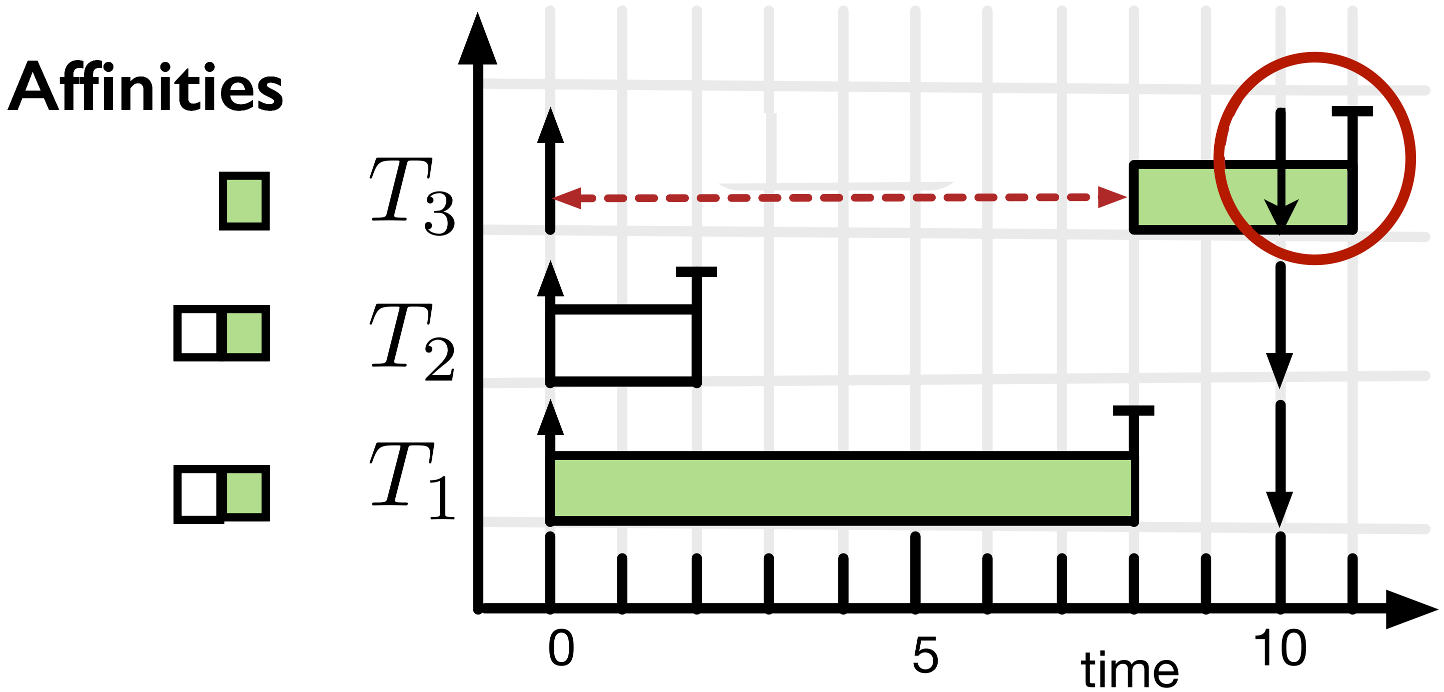


Linux



Task 3 misses deadline!

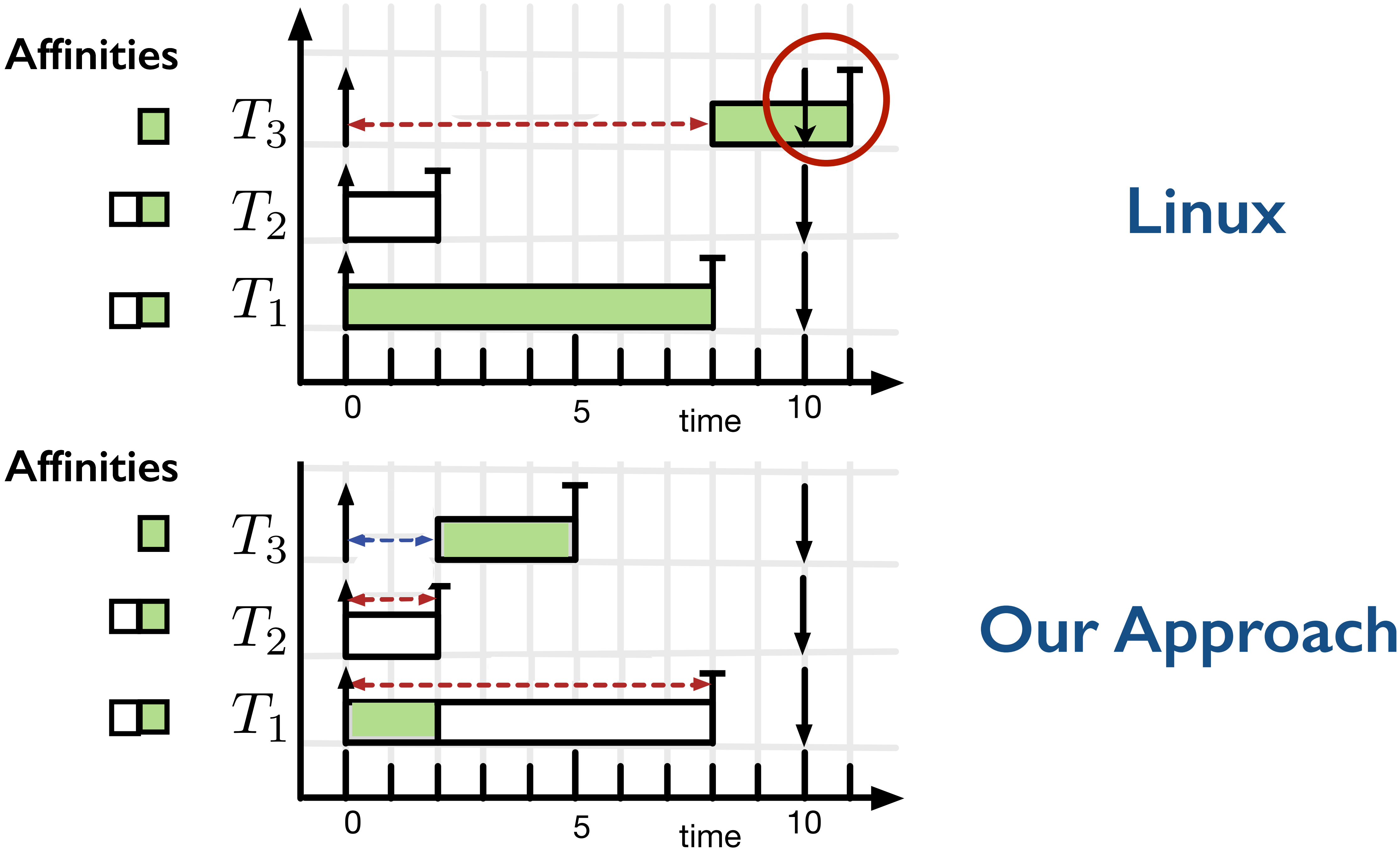
Main Question



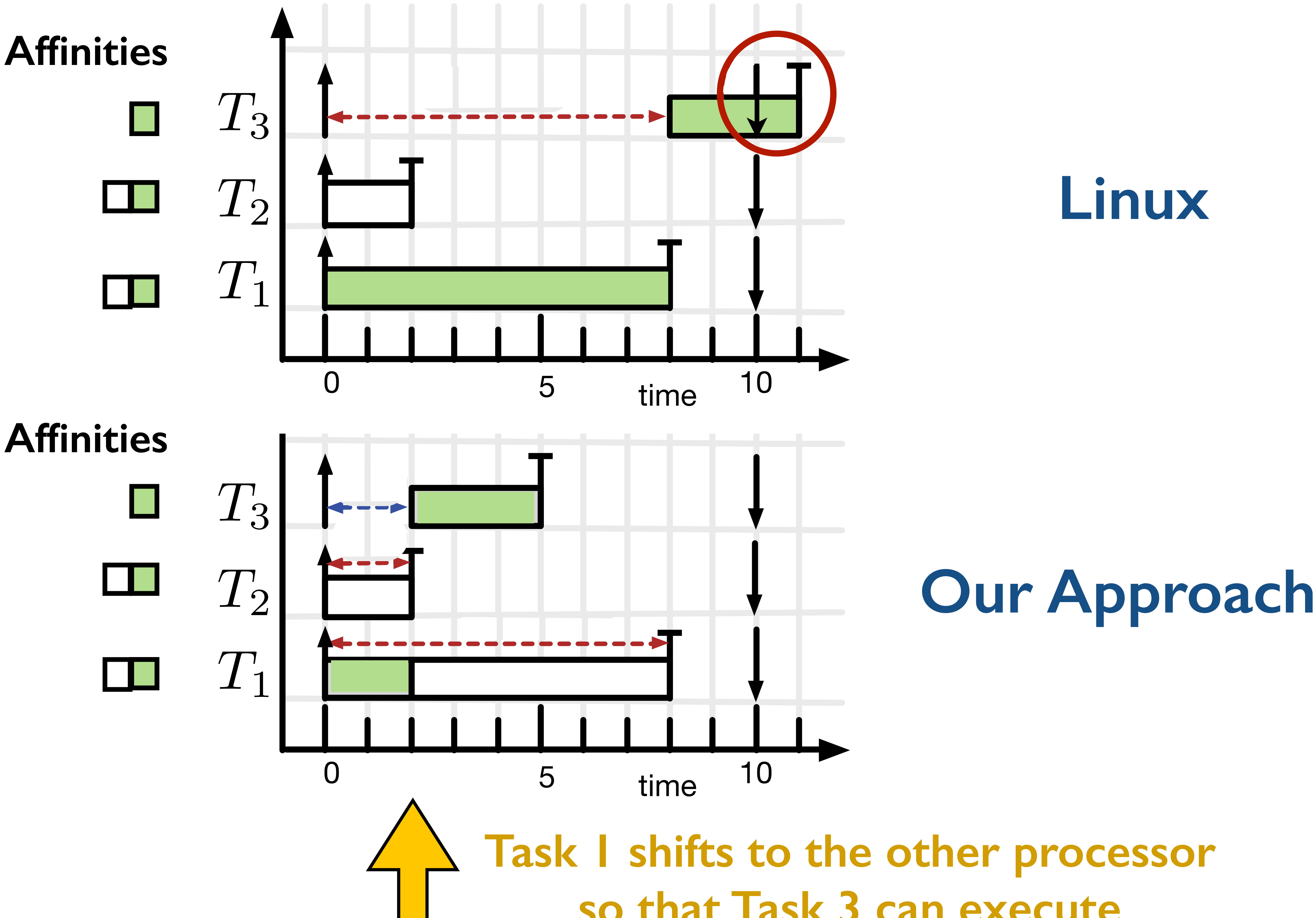
Linux

Can we improve the ability to meet deadlines **without violating** the affinity assignment?

Shifting Tasks to Improve the Schedule

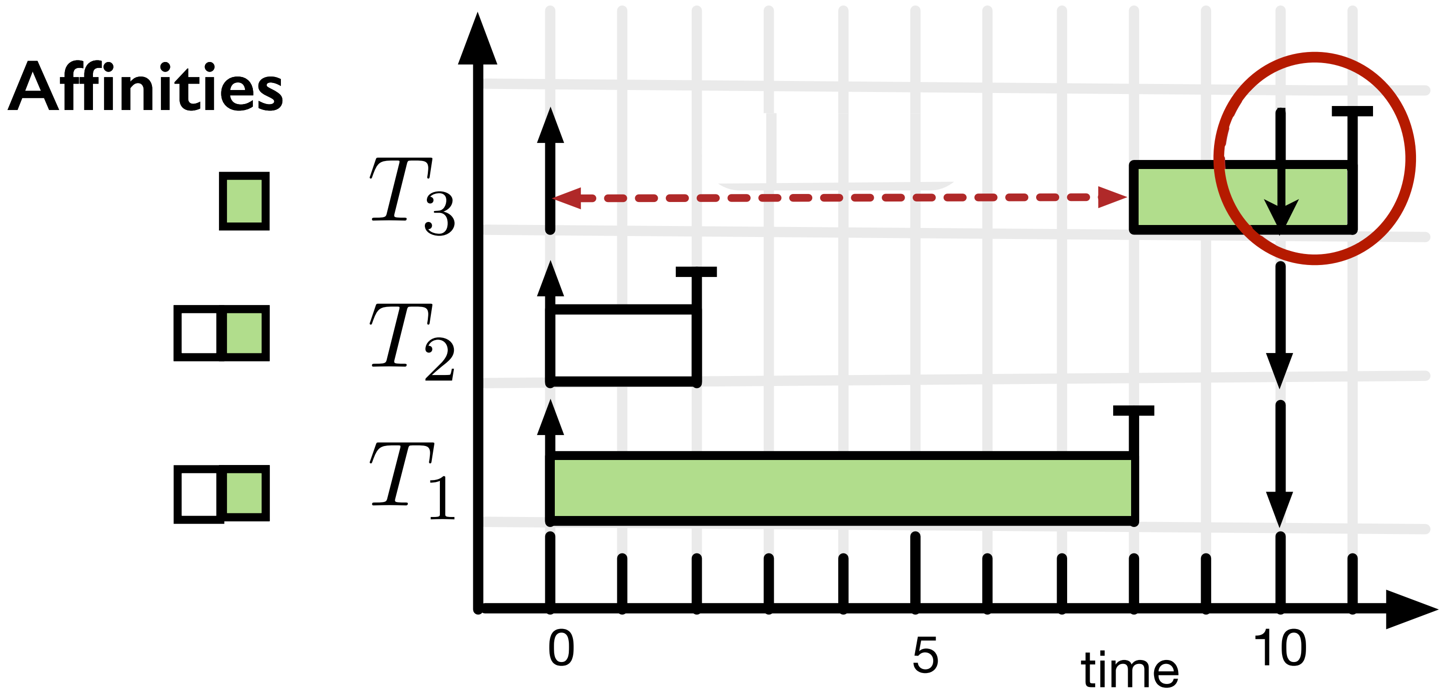


Shifting Tasks to Improve the Schedule

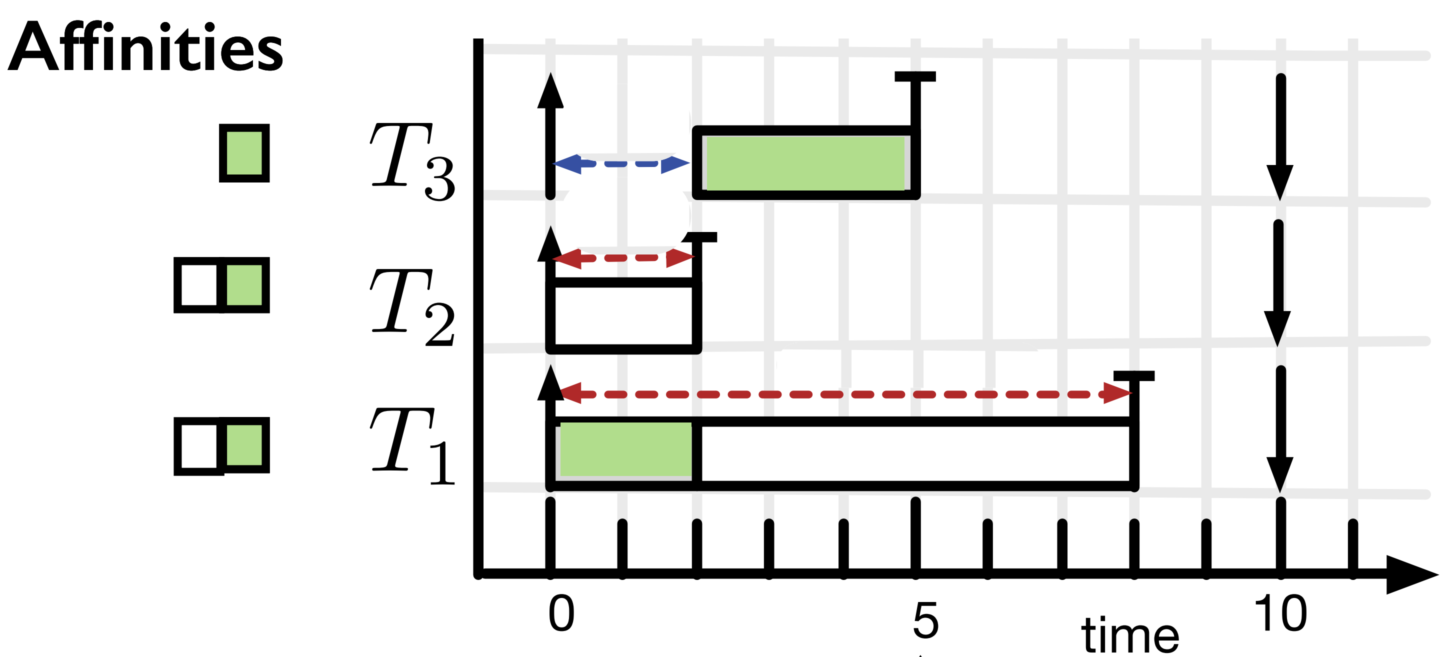


Task 1 shifts to the other processor
so that Task 3 can execute

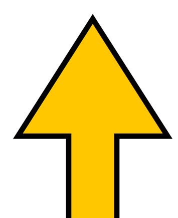
Shifting Tasks to Improve the Schedule



Linux



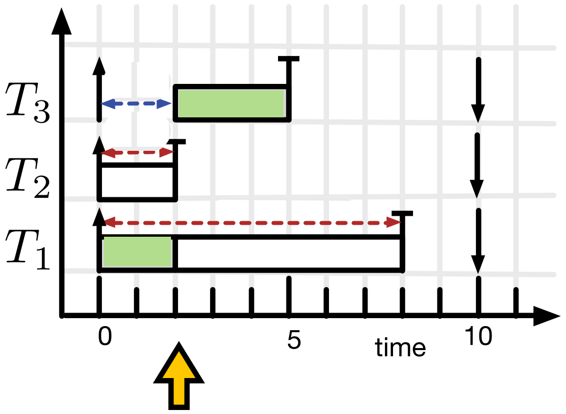
Our Approach



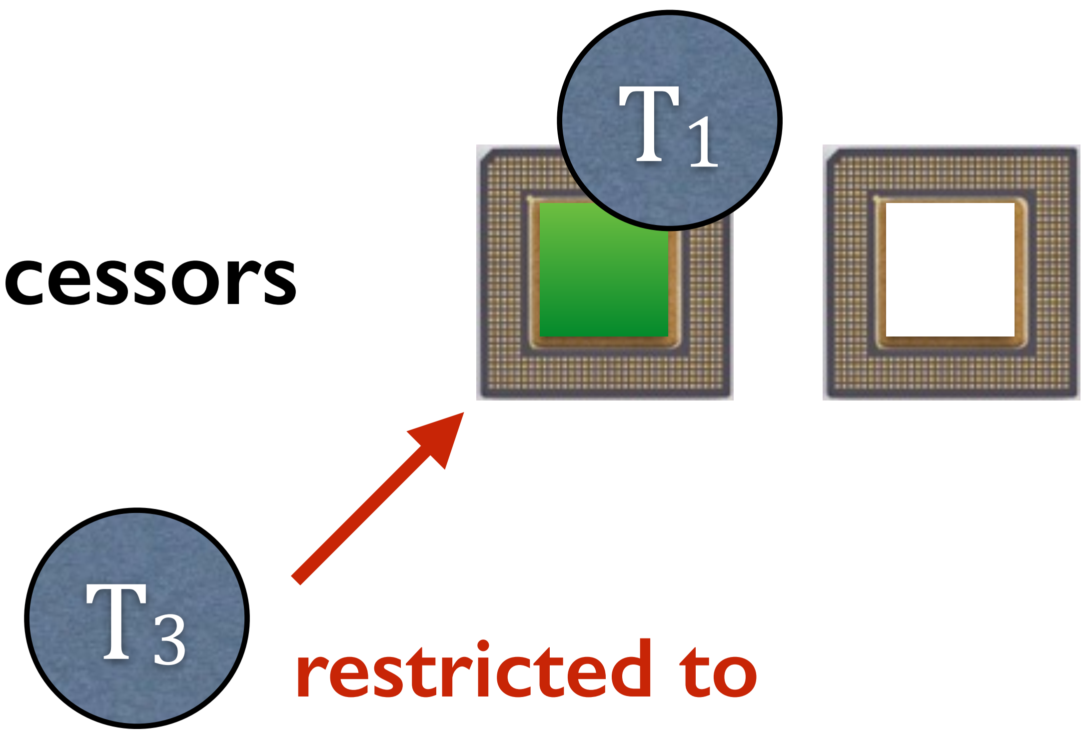
No deadline misses for Task 3!

New Migration Semantics for APA Scheduling

via **Task Shifting**



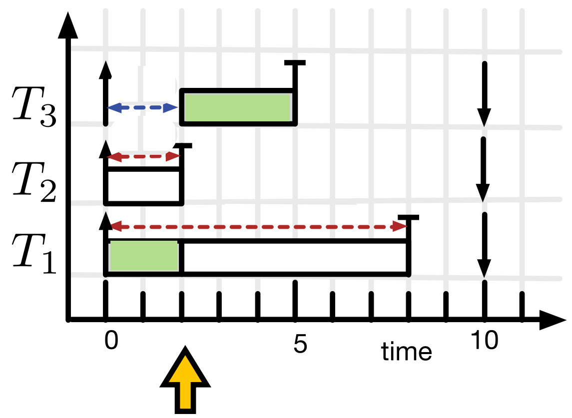
Processors

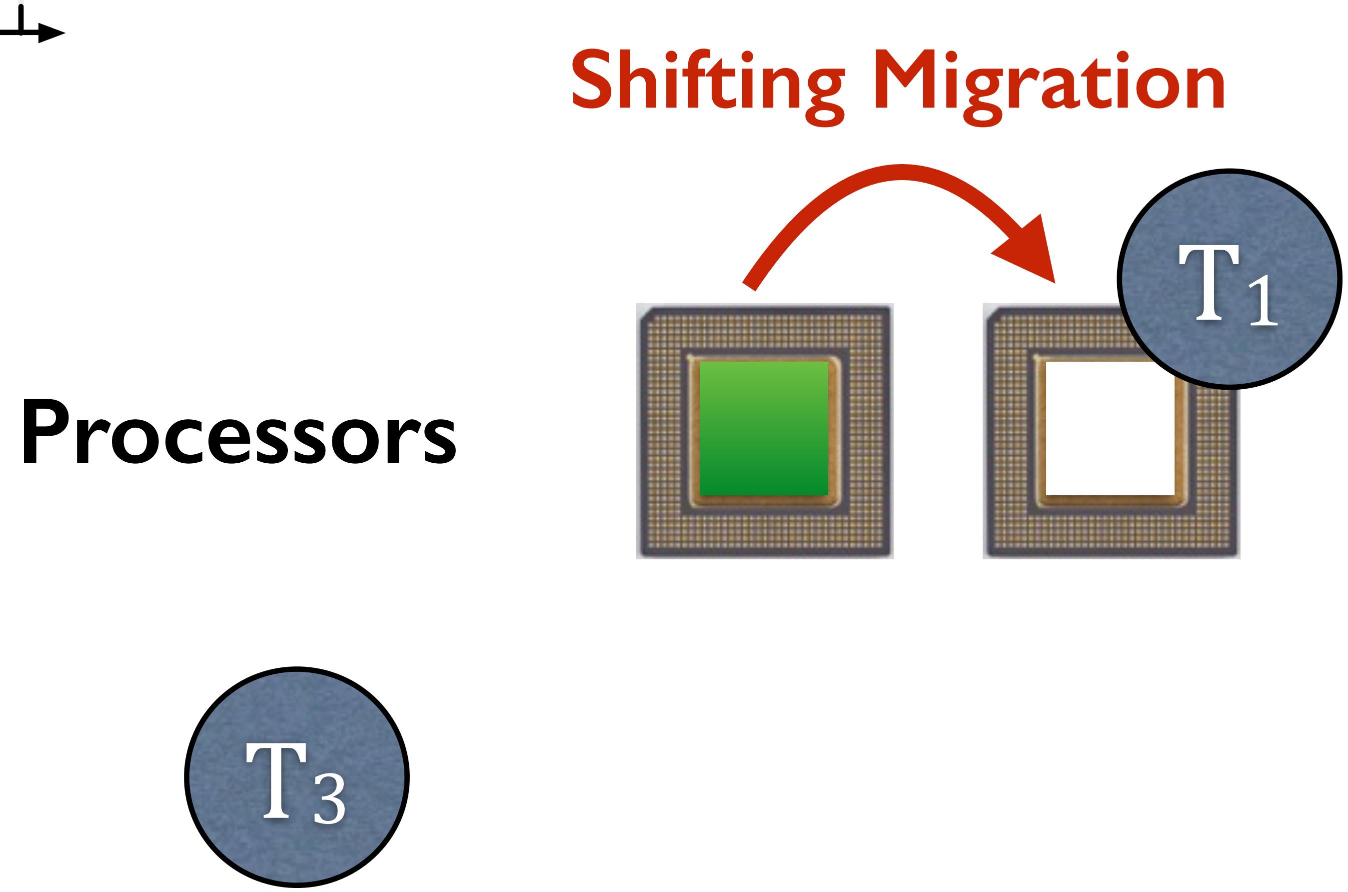


restricted to this processor

New Migration Semantics for APA Scheduling

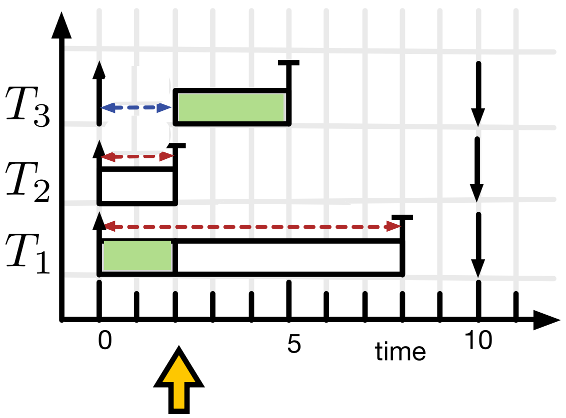
via Task Shifting





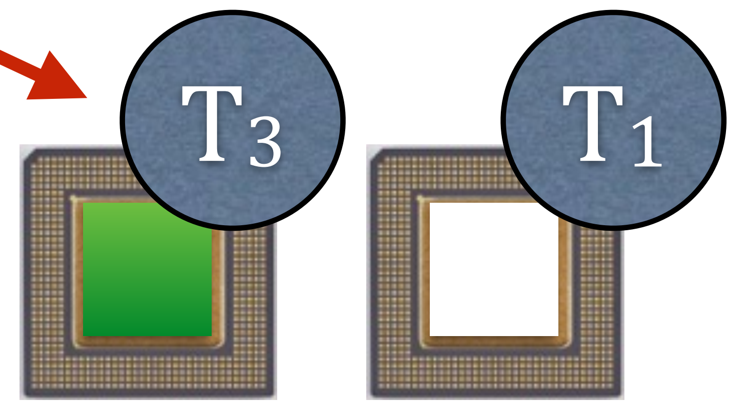
New Migration Semantics for APA Scheduling

via **Task Shifting**



Task 3 scheduled

Processors



Shifting migrations free processors for a restricted task

→ **Improved Schedulability**

Full API Compatibility

Affinity API

Scheduler

✔ **API unmodified!**

Only change:

when tasks migrate

Overall

- **API compatibility**
- **No affinity violations**
- **Improved schedulability**

Similar Problem in Operations Research

Assignment Problem with Seniority Constraints [Caron et al 1999]

Problem: Assign jobs in a hospital

Constraints:

(1) Jobs require qualification

(2) Senior employees have preference

Two variants

Weak Seniority

Strong Seniority

[Caron et al 1999] Gaetan Caron, Pierri Hansen, and Brigitte Jaumard. 1999. The Assignment Problem with Seniority and Job Priority Constraints. Oper. Res. 47, 3 (March 1999), 449-453.

Contributions of our Paper

1) Distinction between:

APA scheduling without shifting $\Longleftrightarrow$ Weak APA

APA scheduling with shifting $\Longleftrightarrow$ Strong APA

Contributions of our Paper

1) Distinction between:

APA scheduling without shifting $\Longleftrightarrow$ **Weak APA**

APA scheduling with shifting $\Longleftrightarrow$ **Strong APA**

2) Formalization of strong APA scheduling based on Bipartite Matching

Contributions of our Paper

1) Distinction between:

APA scheduling without shifting $\Longleftrightarrow$ Weak APA

APA scheduling with shifting $\Longleftrightarrow$ Strong APA

2) Formalization of strong APA scheduling based on Bipartite Matching

3) Dynamic algorithm for task shifting

Contributions of our Paper

1) Distinction between:

APA scheduling without shifting $\Longleftrightarrow$ **Weak APA**

APA scheduling with shifting $\Longleftrightarrow$ **Strong APA**

2) Formalization of strong APA scheduling based on Bipartite Matching

3) Dynamic algorithm for task shifting

4) Schedulability Analysis for strong APA Scheduling

This Talk

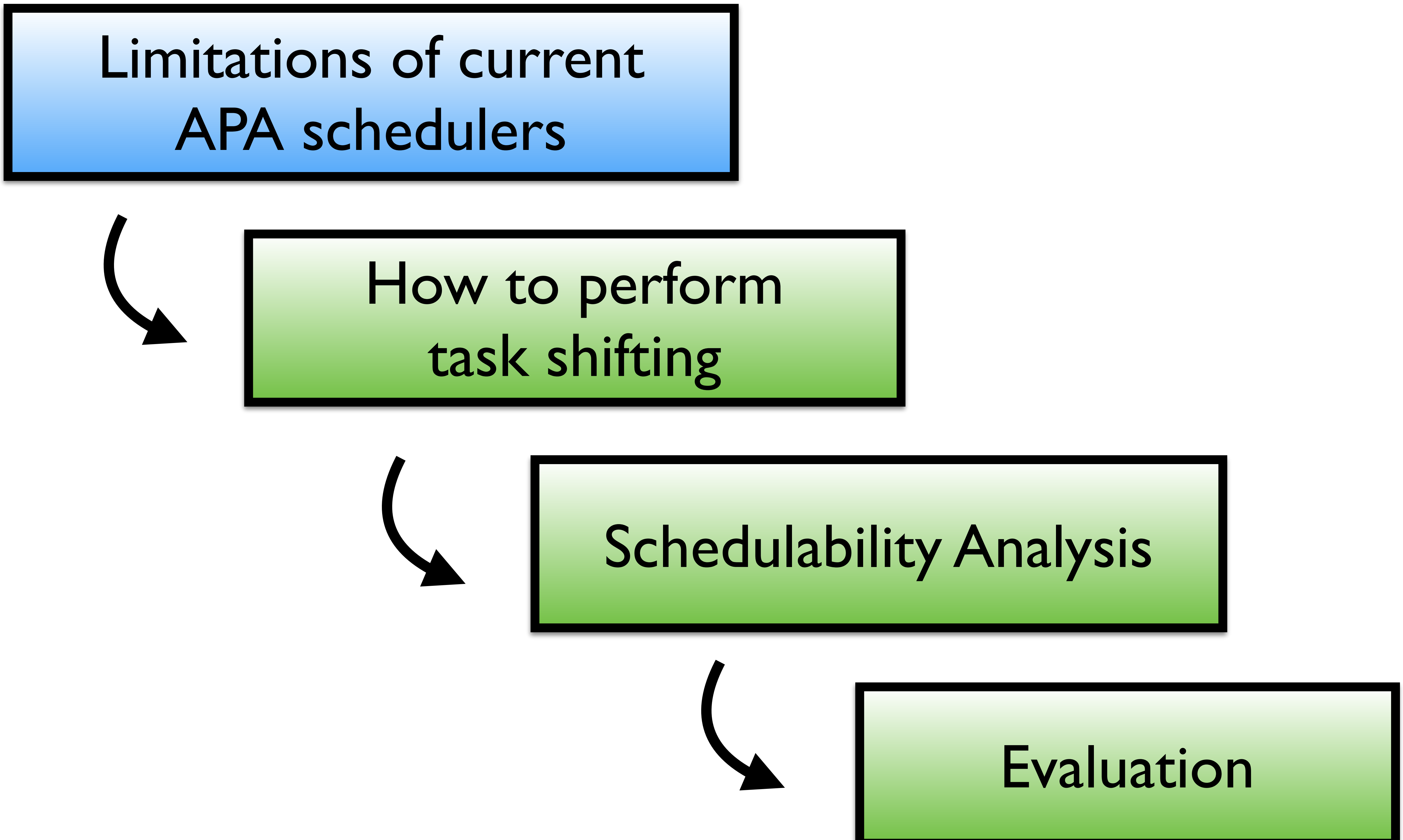
Limitations of current APA schedulers

How to perform task shifting

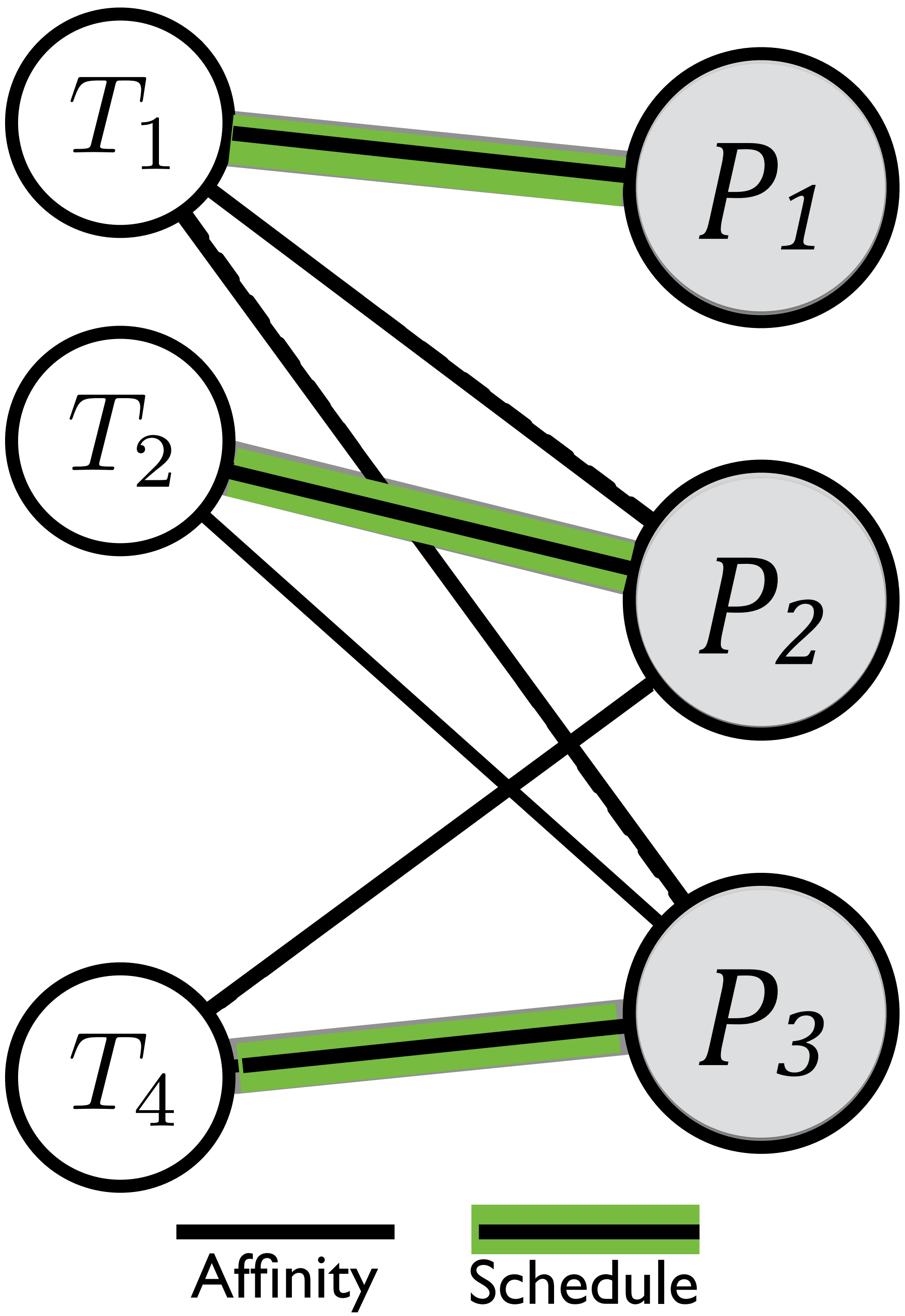
Schedulability Analysis

Evaluation

This Talk

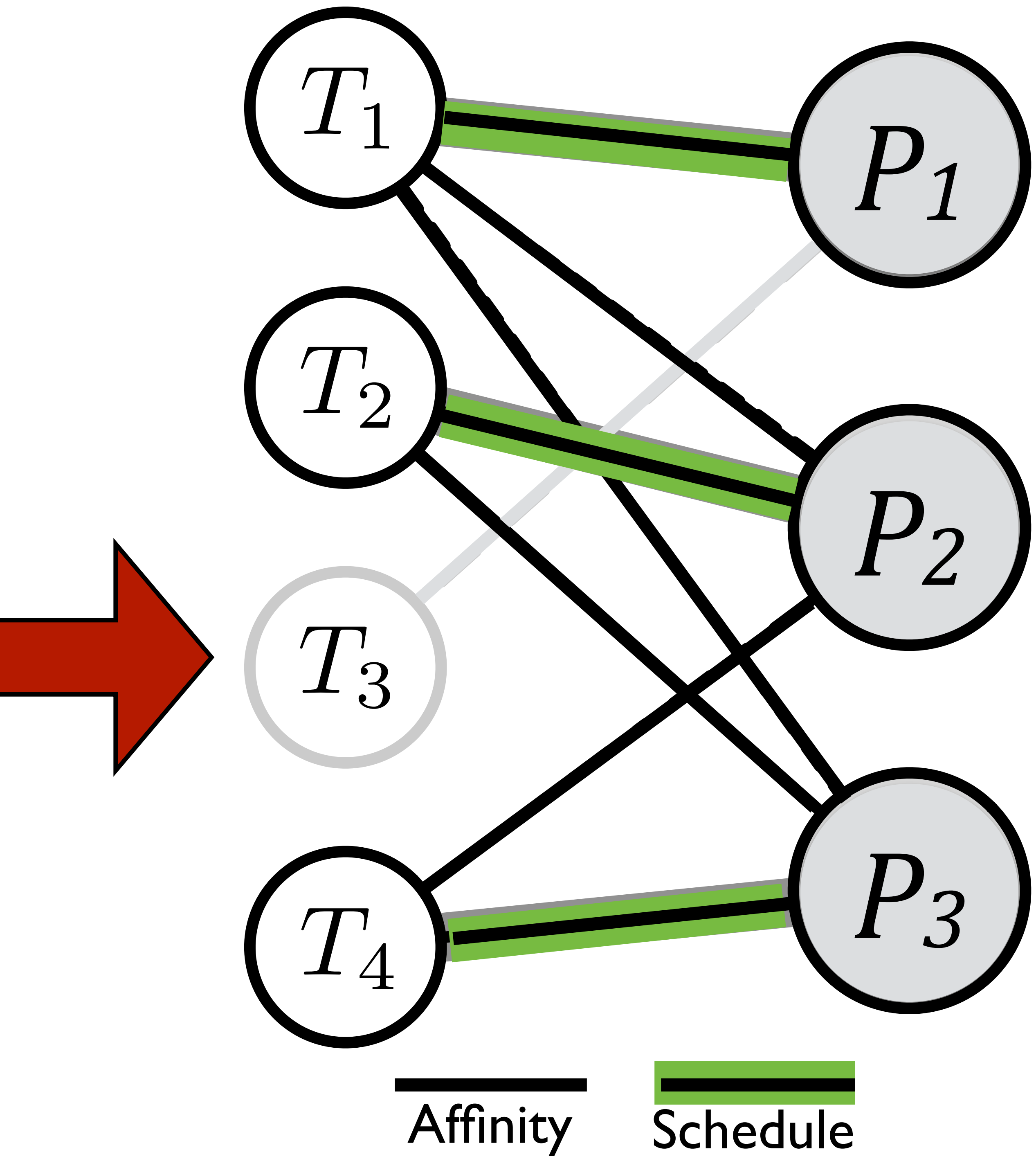

Limitations of current APA schedulers
How to perform task shifting
Schedulability Analysis
Evaluation

Limitations of Current APA Schedulers



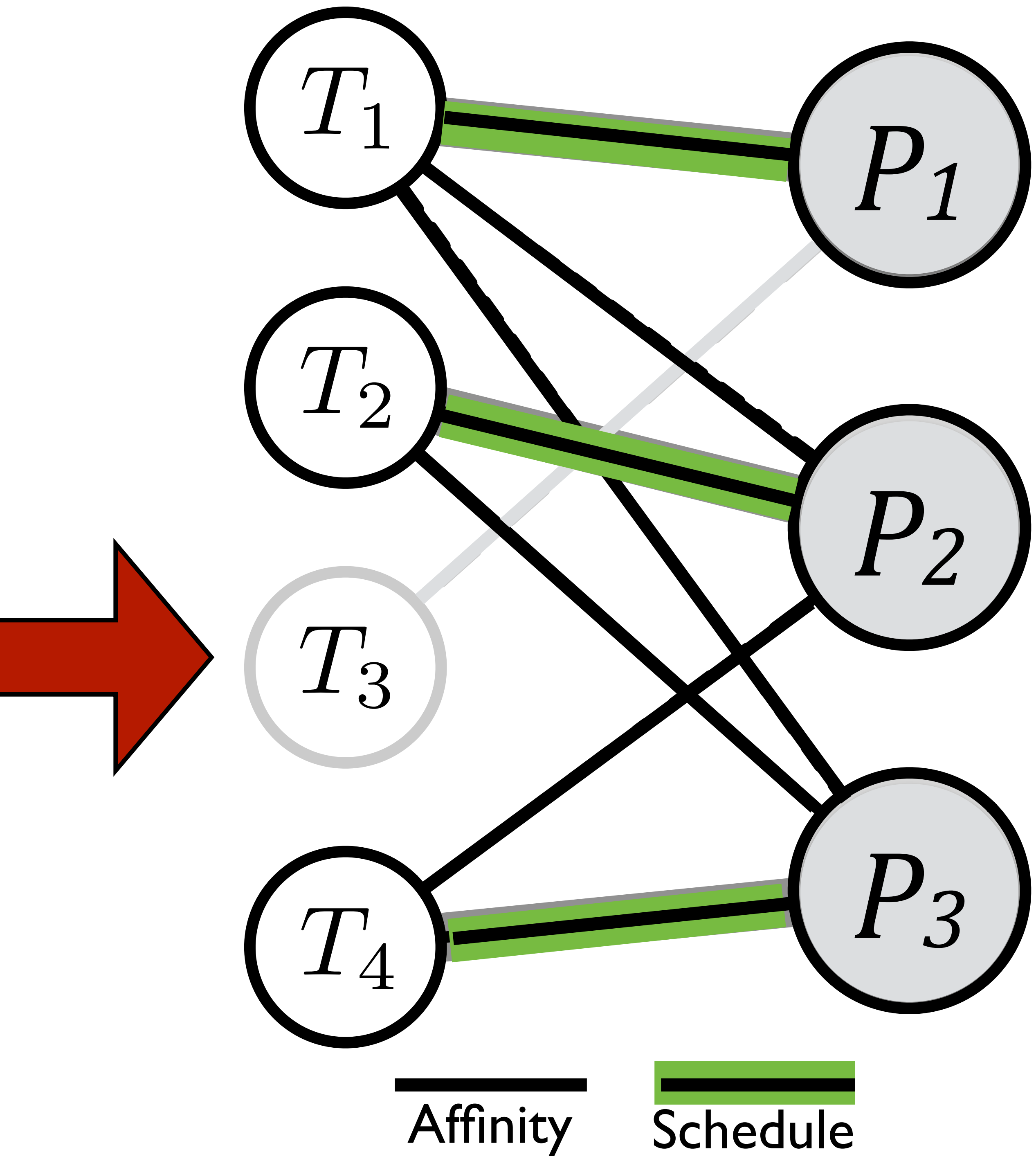
Example where Linux will **violate** task priorities

Task T_3 arrives



Linux locally checks if there is a CPU to be preempted in T_3's affinity.

Linux does not Schedule the Task!

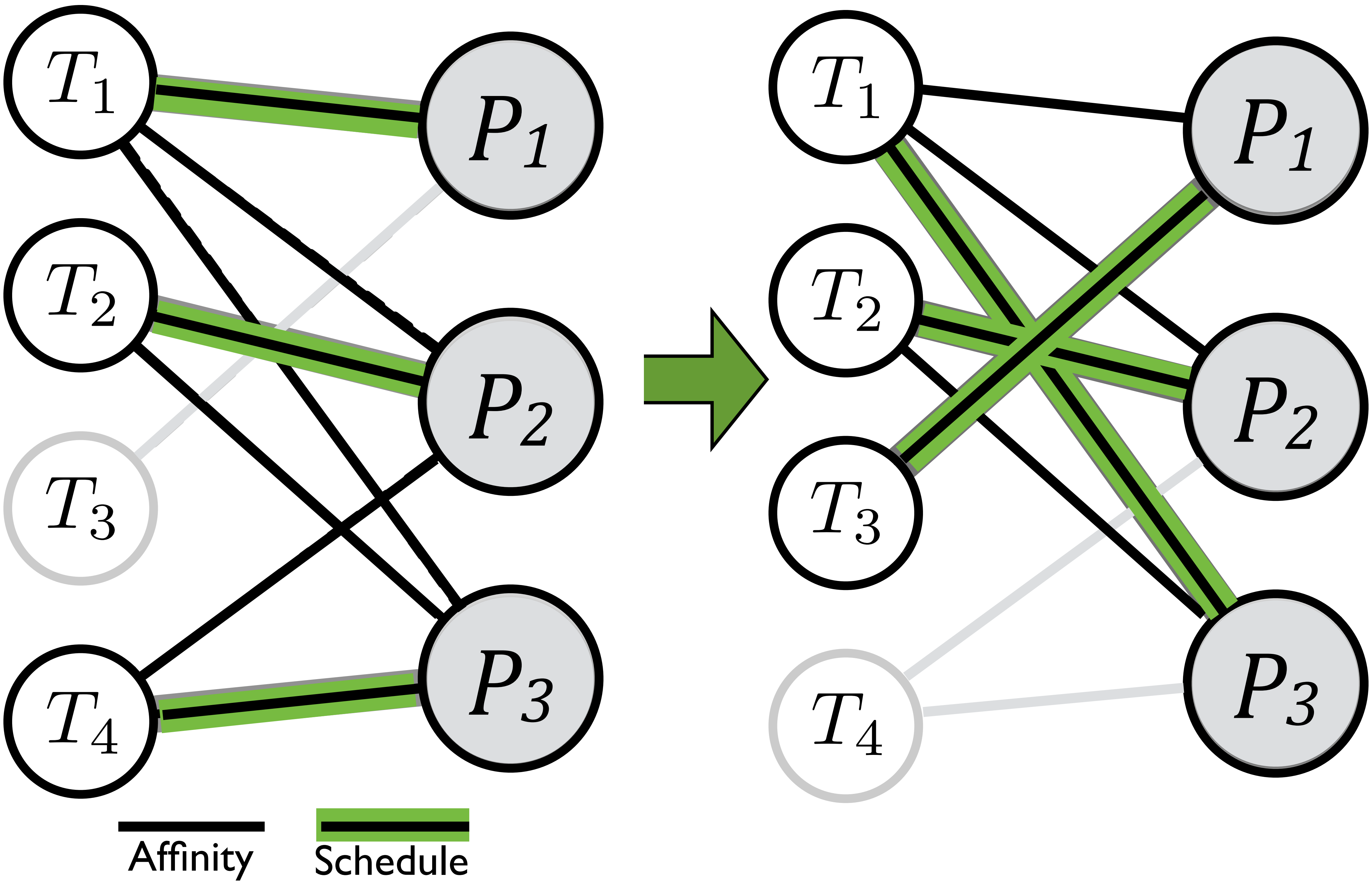


Linux locally checks if there is a CPU to be preempted in T_3's affinity.

No preemption!
CPU 1 already has a higher-priority task.

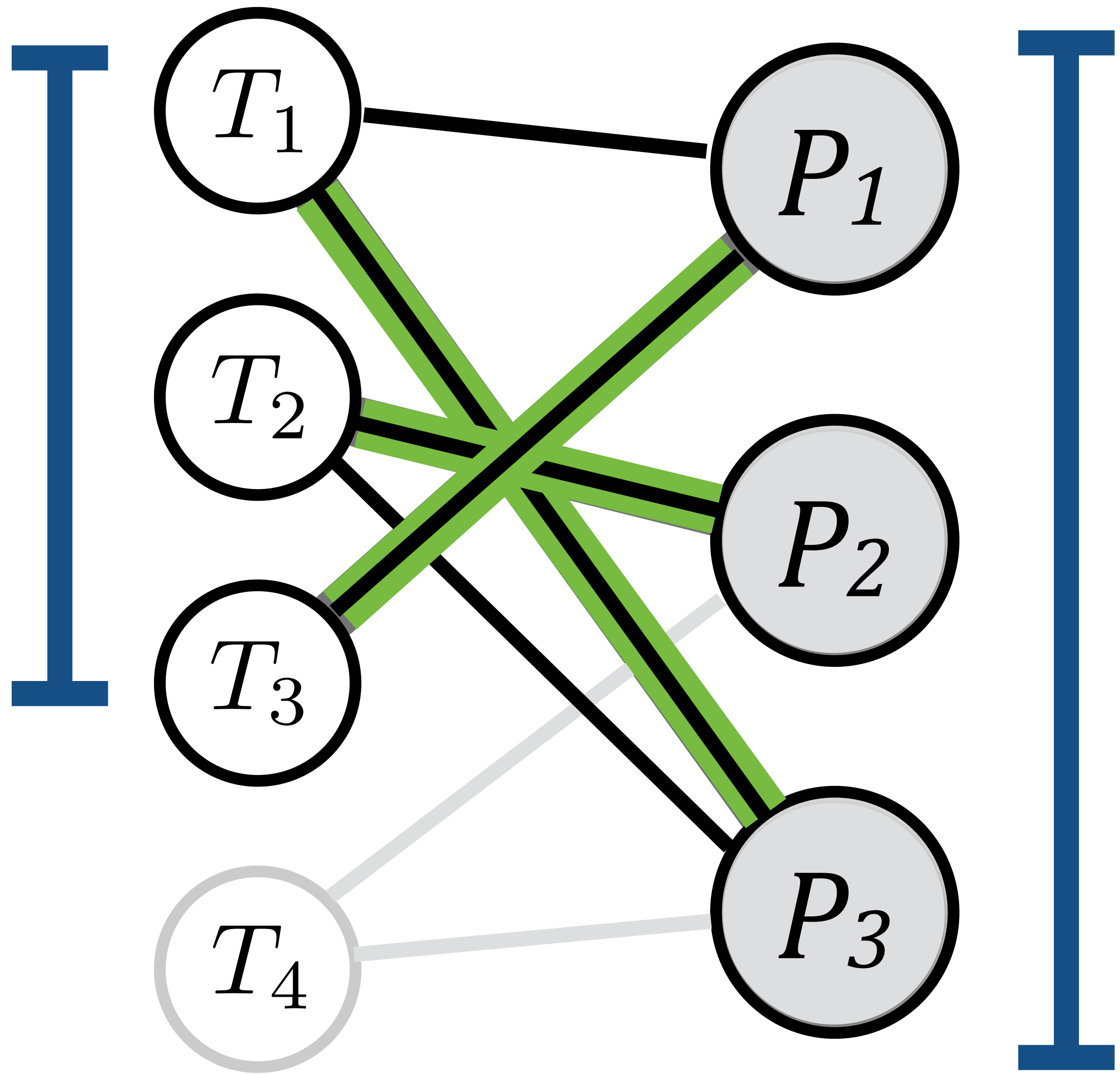
But there is a Better Schedule

(Task priorities: $T_1 < T_2 < T_3 < T_4$)



Global Decision is Required to Compute the Correct Schedule

Task priorities must be respected



Processor utilization must be maximized

Linux does not always guarantee both!

This Talk

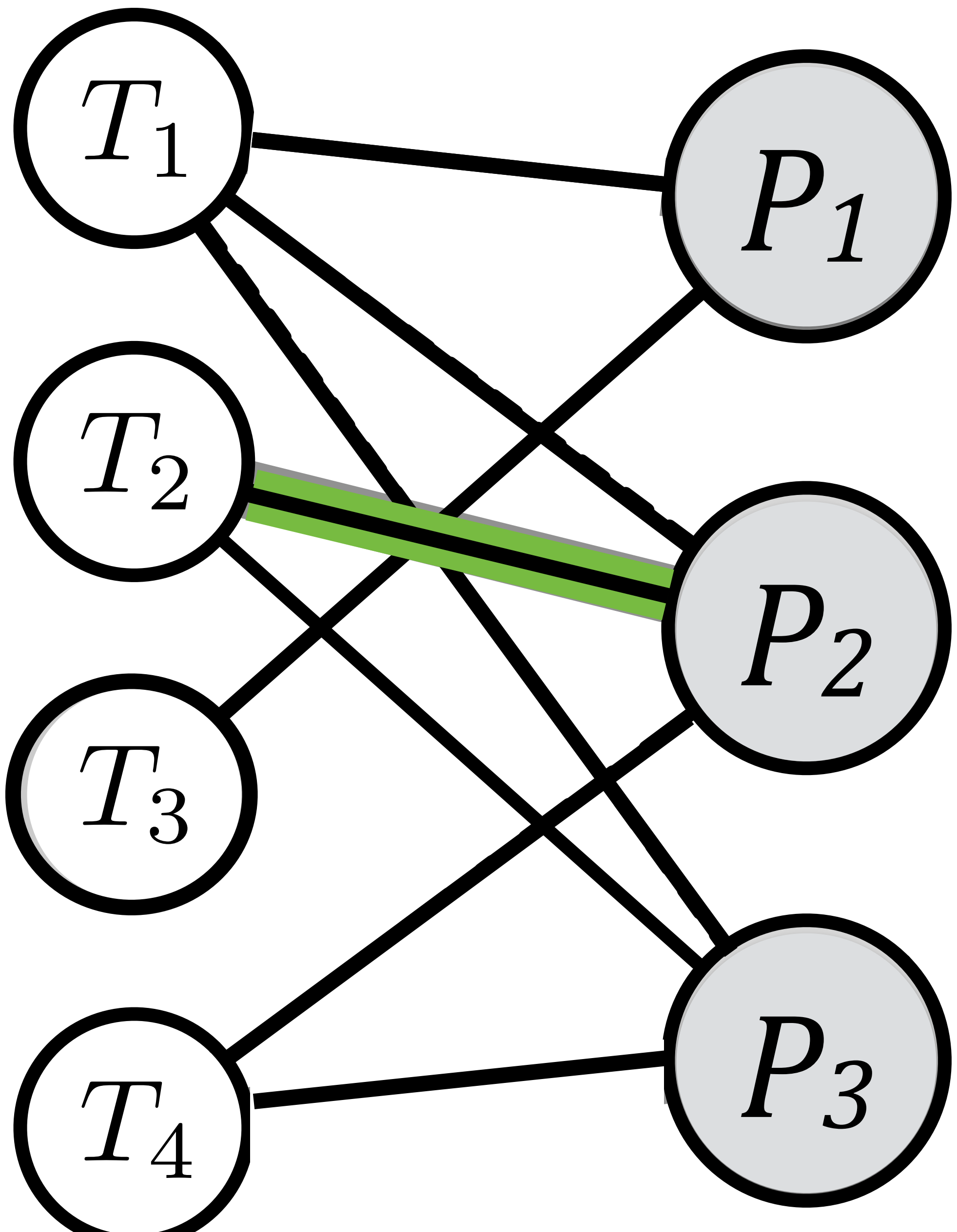
Limitations of current APA schedulers

How to perform task shifting

Schedulability Analysis

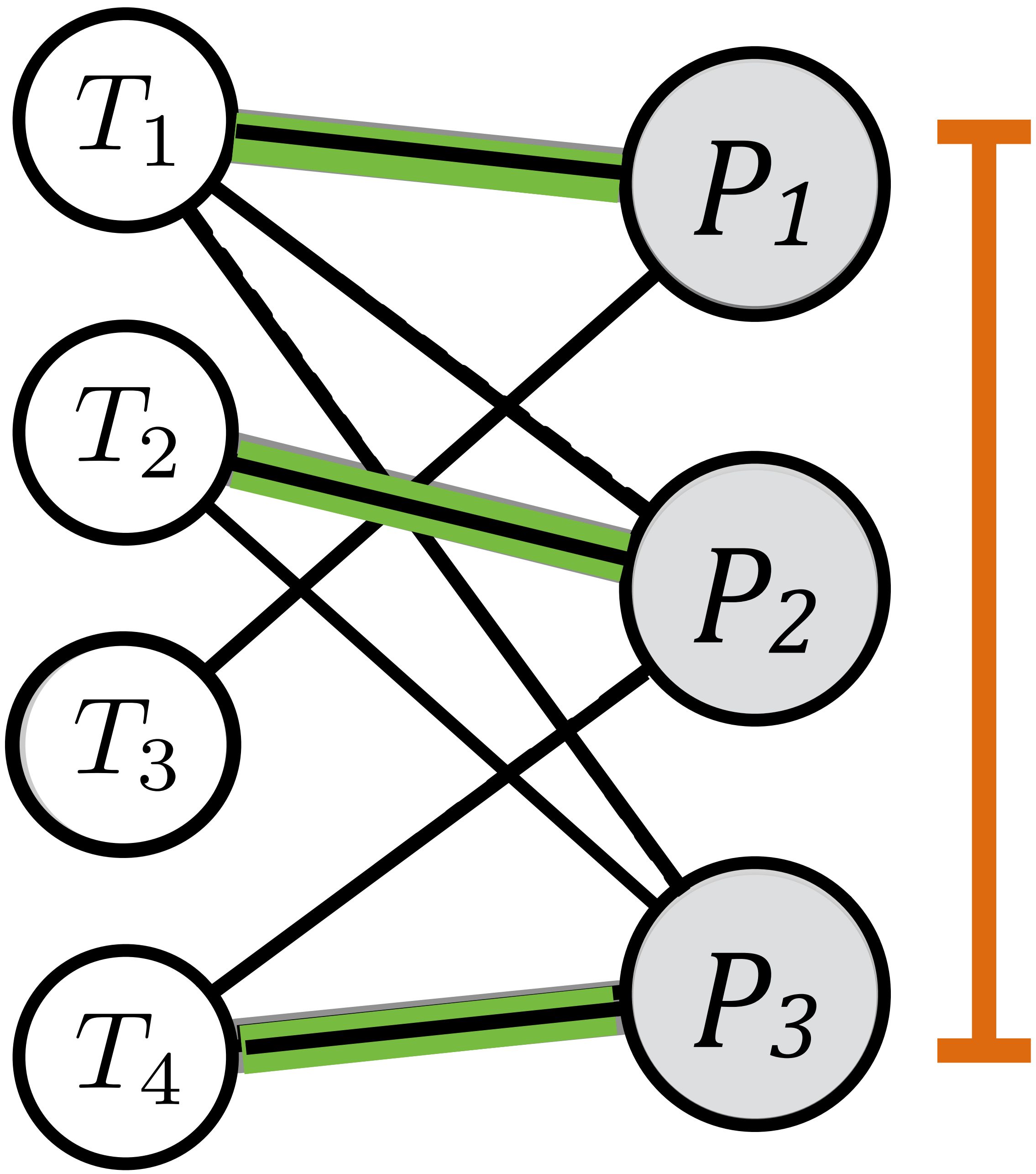
Evaluation

Scheduling as a Bipartite Matching



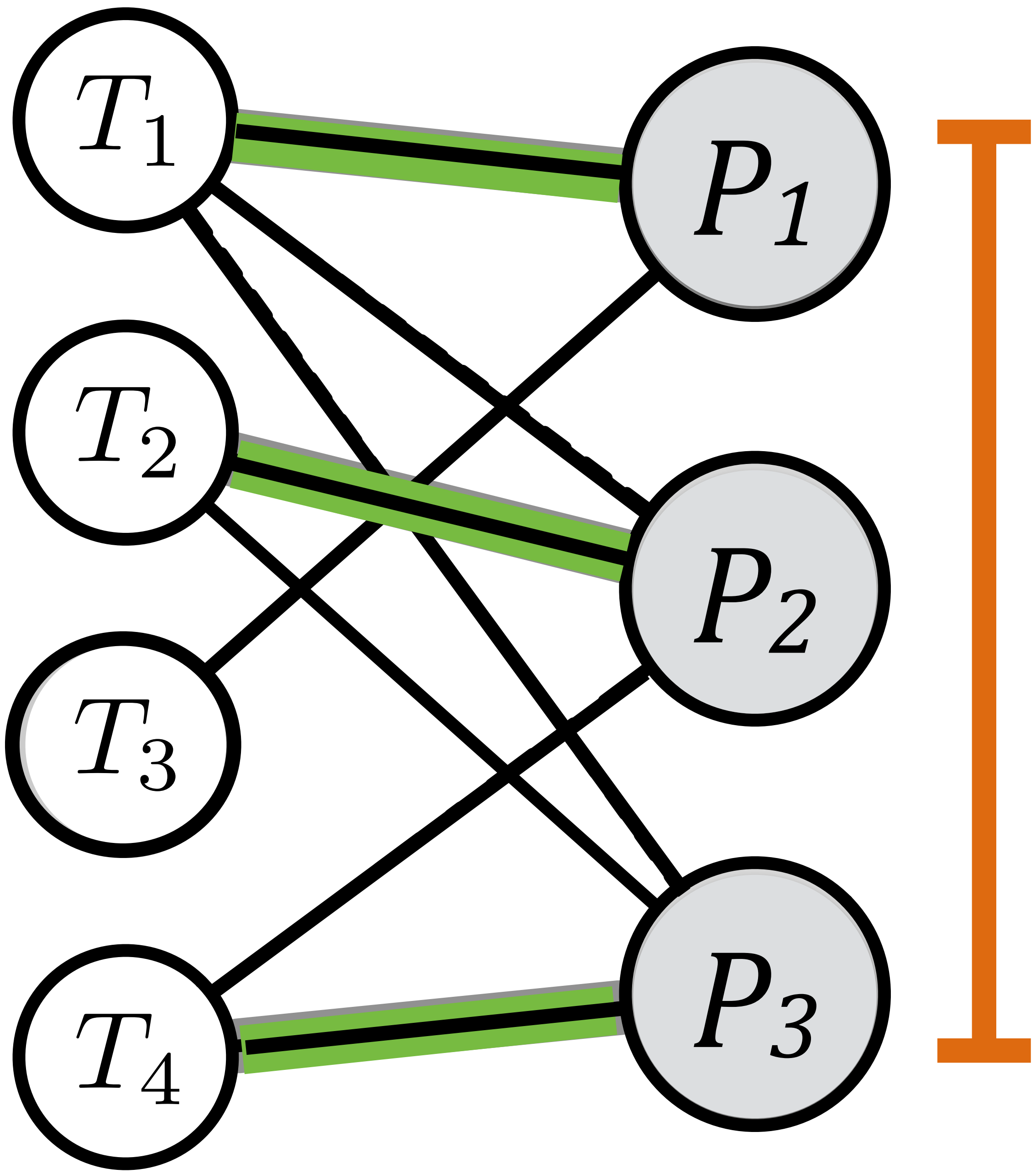
Any matching in the graph
is a valid scheduler state

Maximum Bipartite Matching?



A maximum bipartite matching maximizes processor utilization

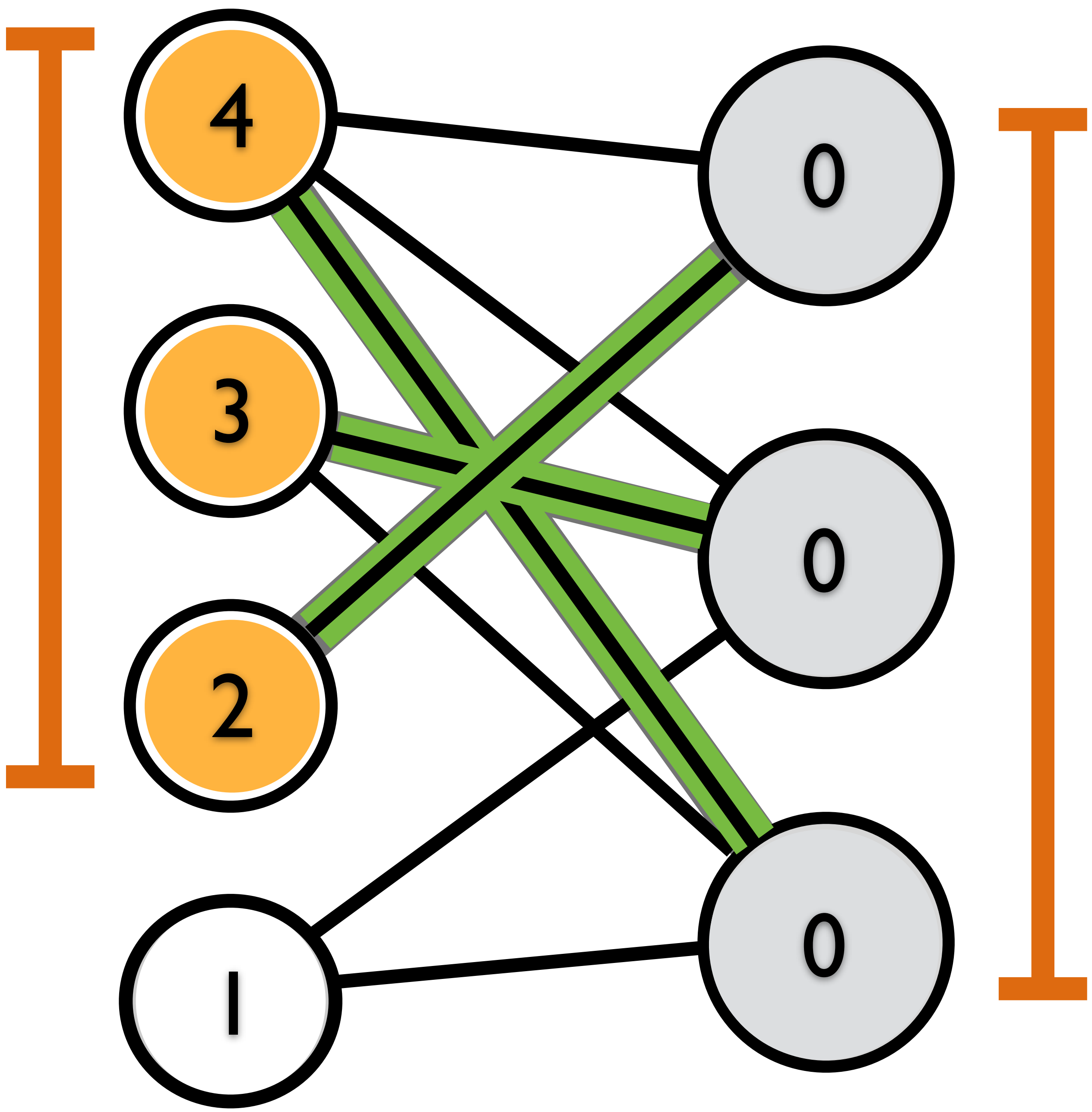
Maximum Bipartite Matching?



A maximum bipartite matching maximizes processor utilization

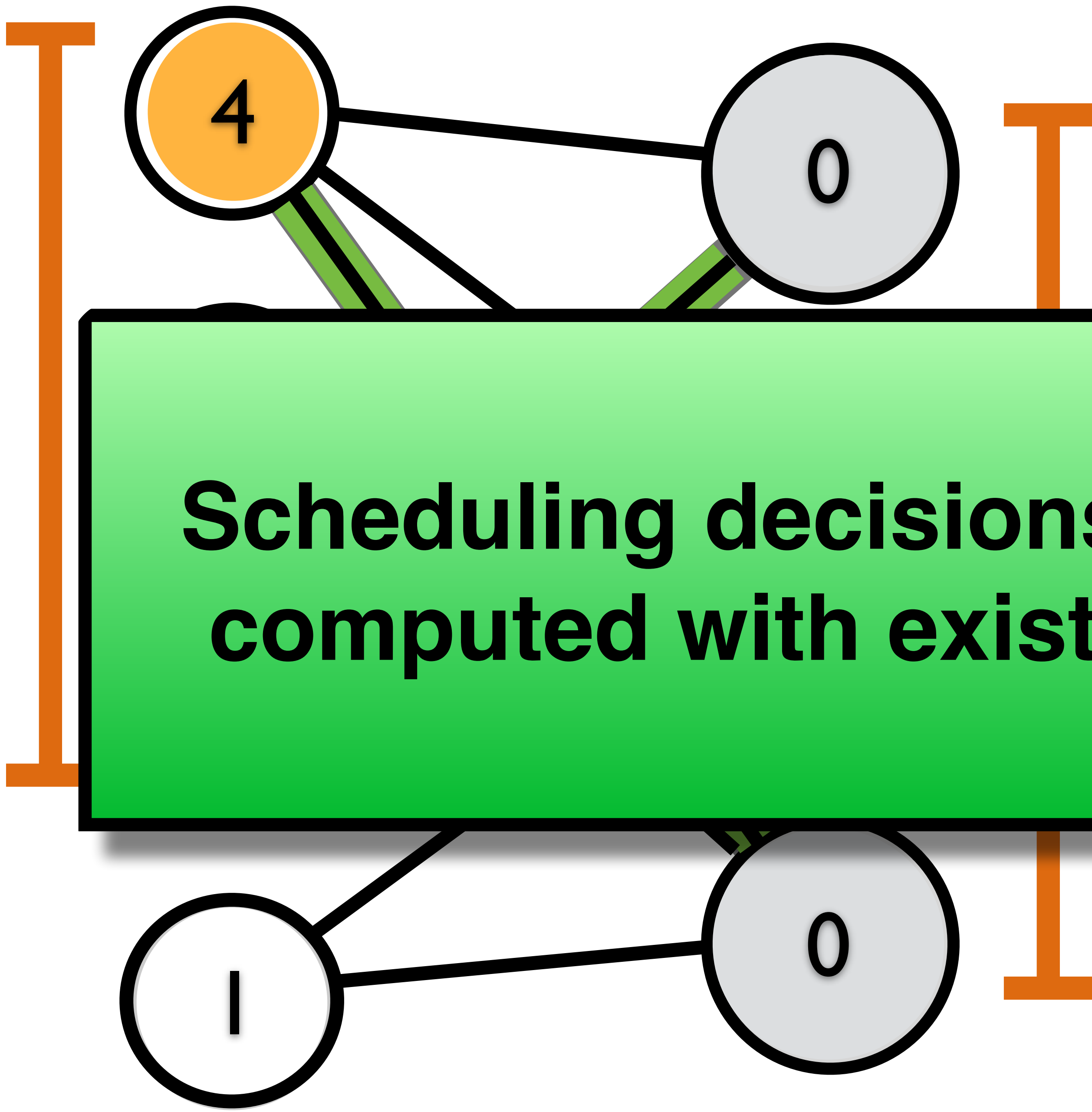
…but does not enforce task priorities.

Maximum Vertex-Weighted Bipartite Matching (MVM)



If we map task priorities to vertex weights, MVM is the optimal scheduling decision.

Maximum Vertex-Weighted Bipartite Matching (MVM)



Scheduling decisions for strong APA can be computed with existing graph algorithms.

Scheduling Decisions must be Fast!

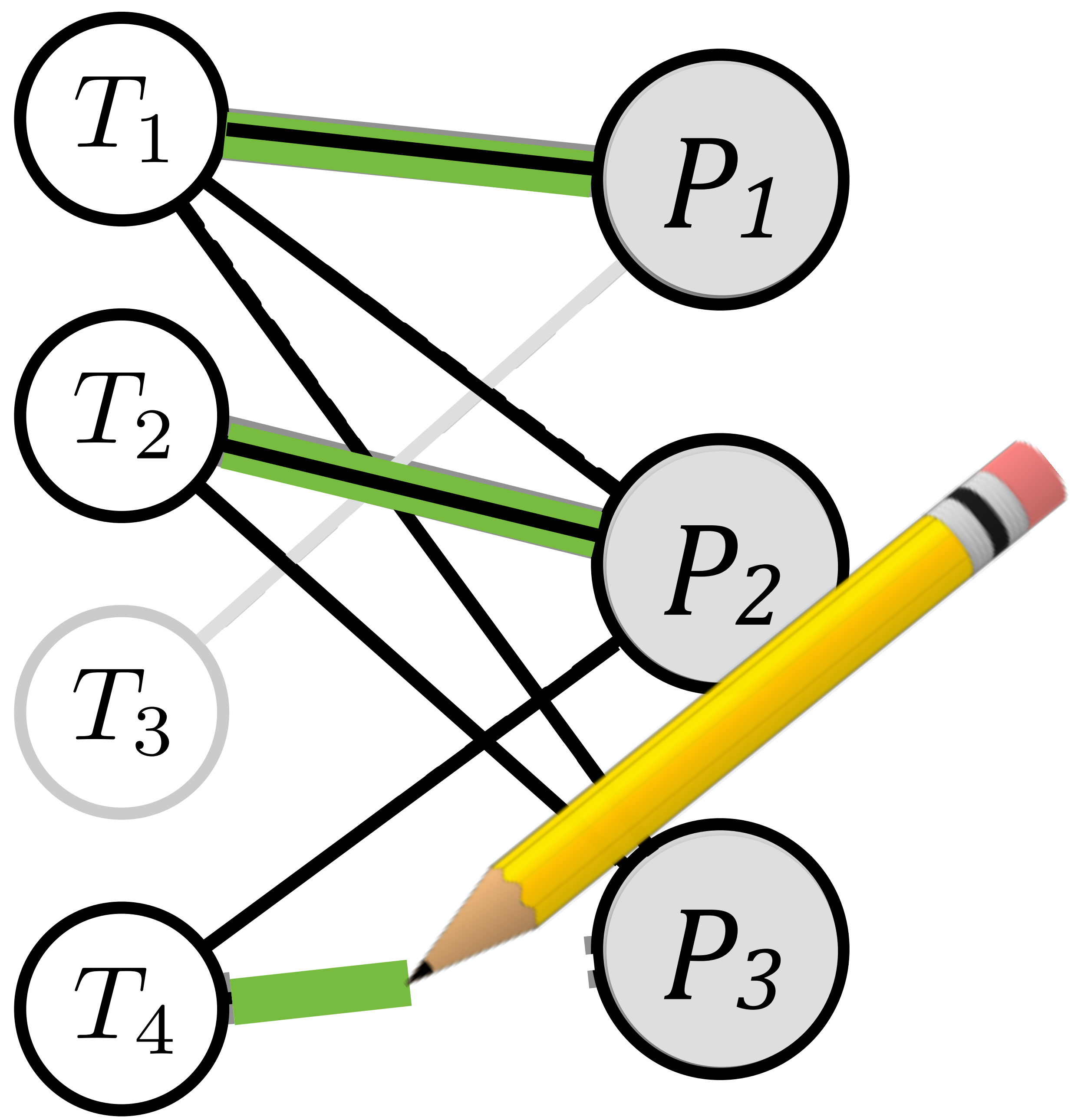
- Scheduler is a critical part of an OS
- Computing an MVM from scratch is costly

Scheduling Decisions must be Fast!

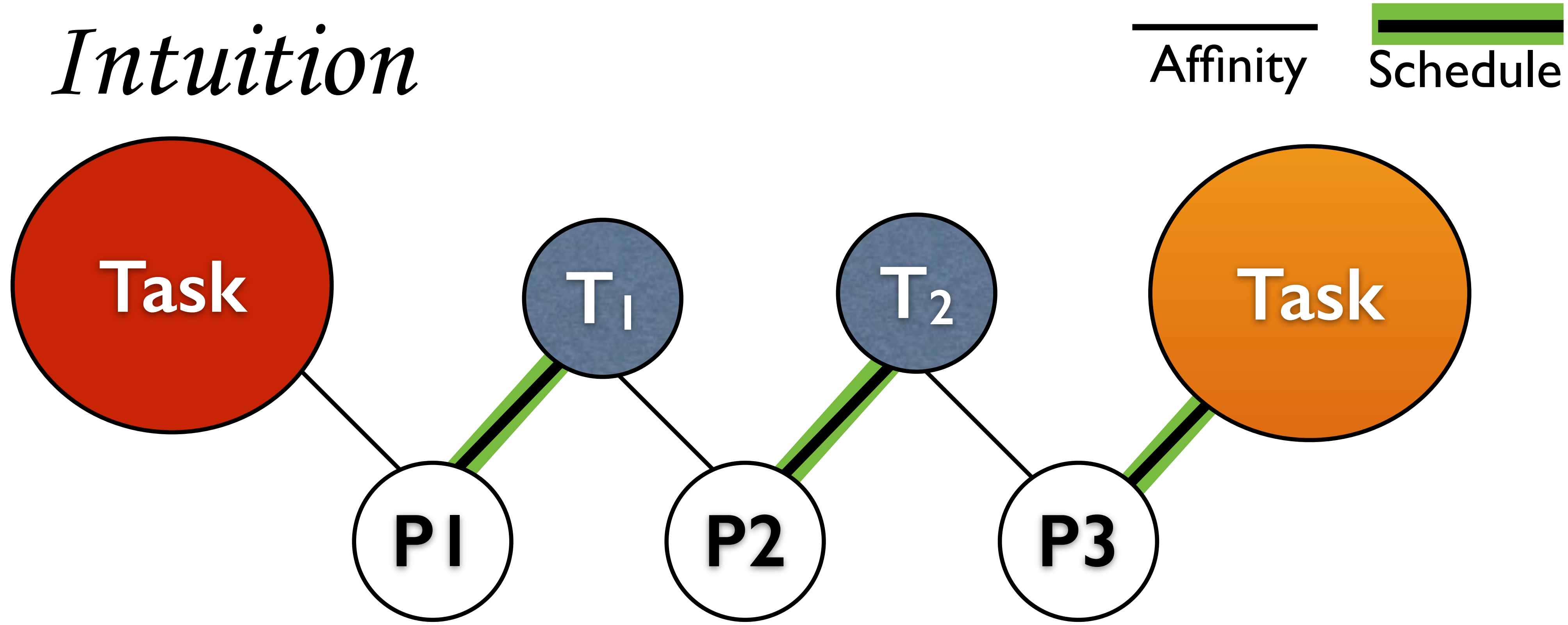
- Scheduler is a critical part of an OS
- Computing an MVM from scratch is costly

Previous schedules are not just discarded.
We need a **dynamic** algorithm!

Recomputing MVM is Inefficient!



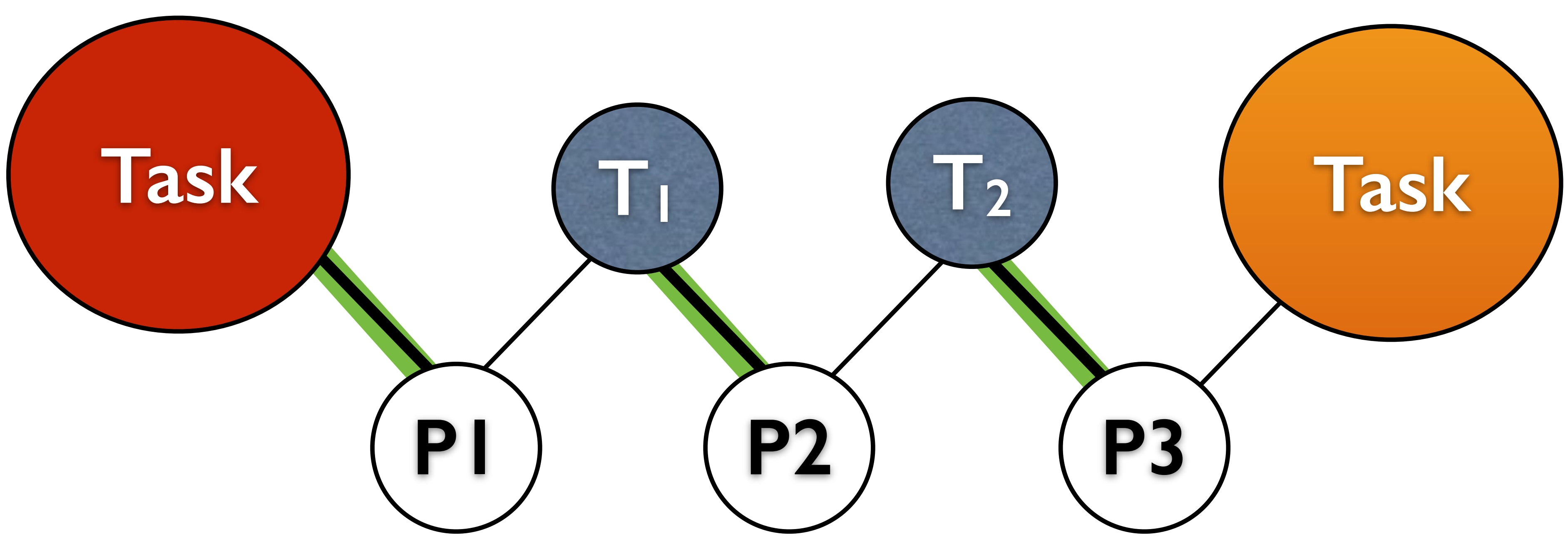
Task Migration in the Graph



For some task that just arrived,
any reachable task can be preempted

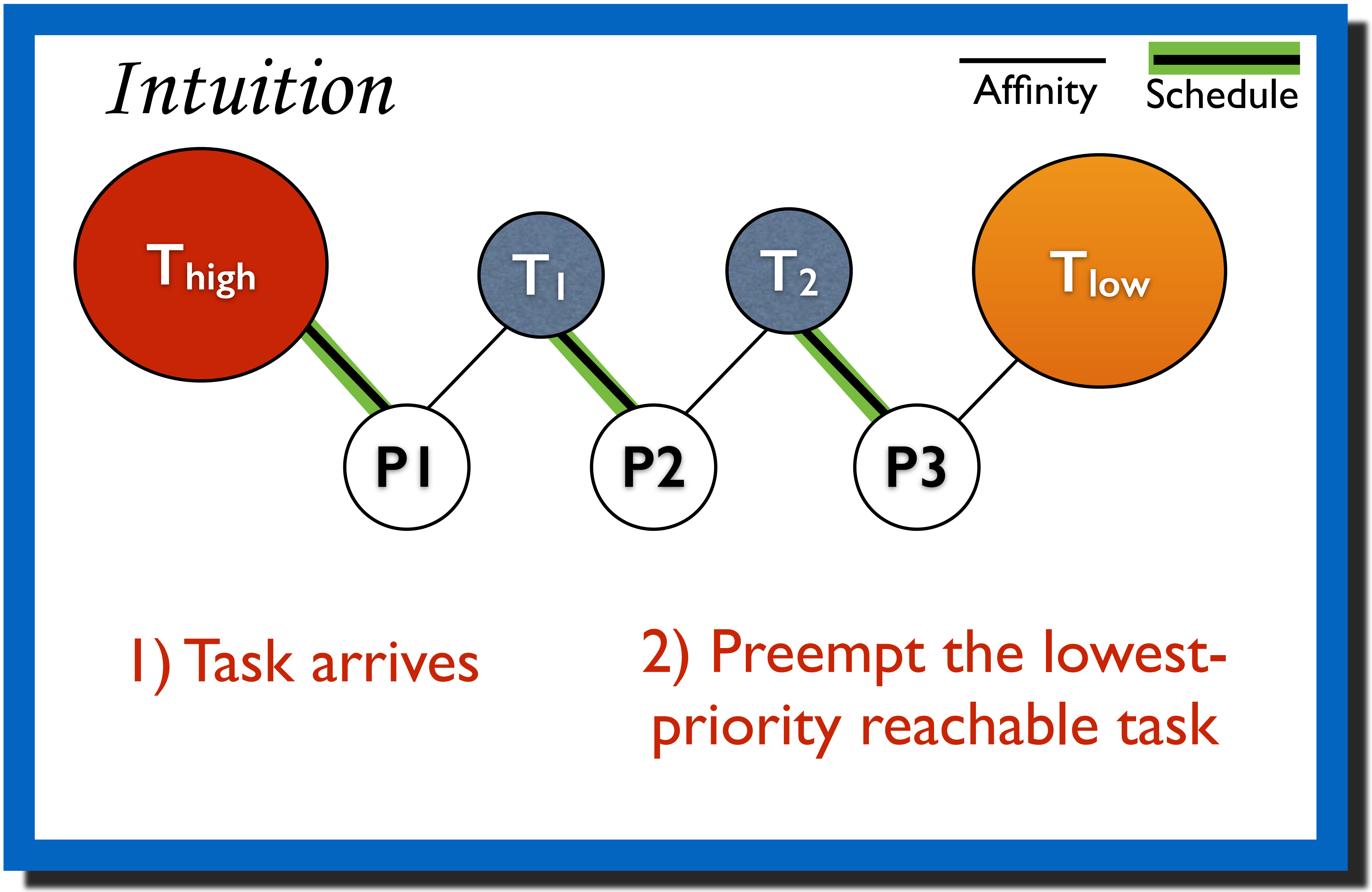
Task Migration in the Graph

Intuition

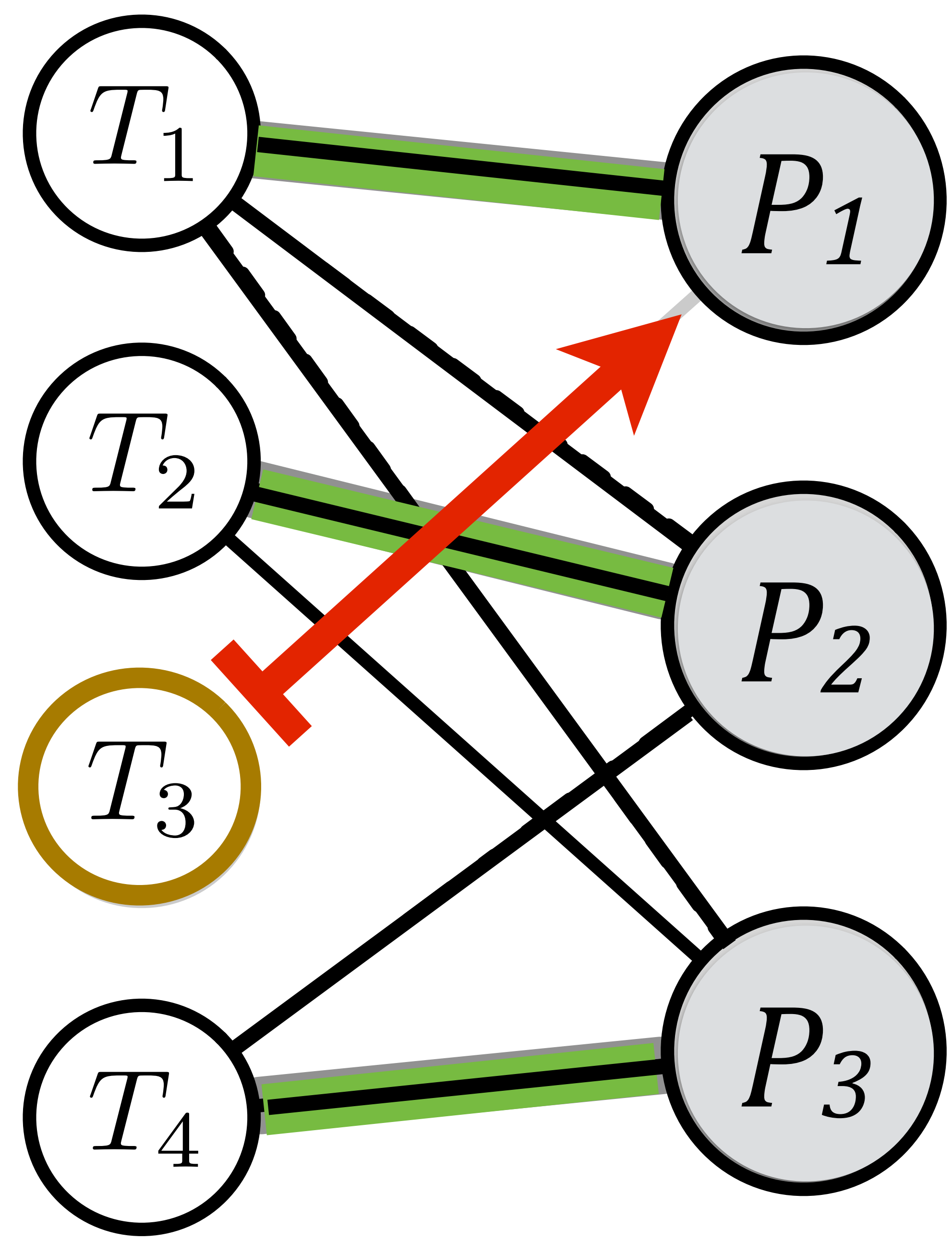


We just need to shift tasks by taking the complementary edges in the path

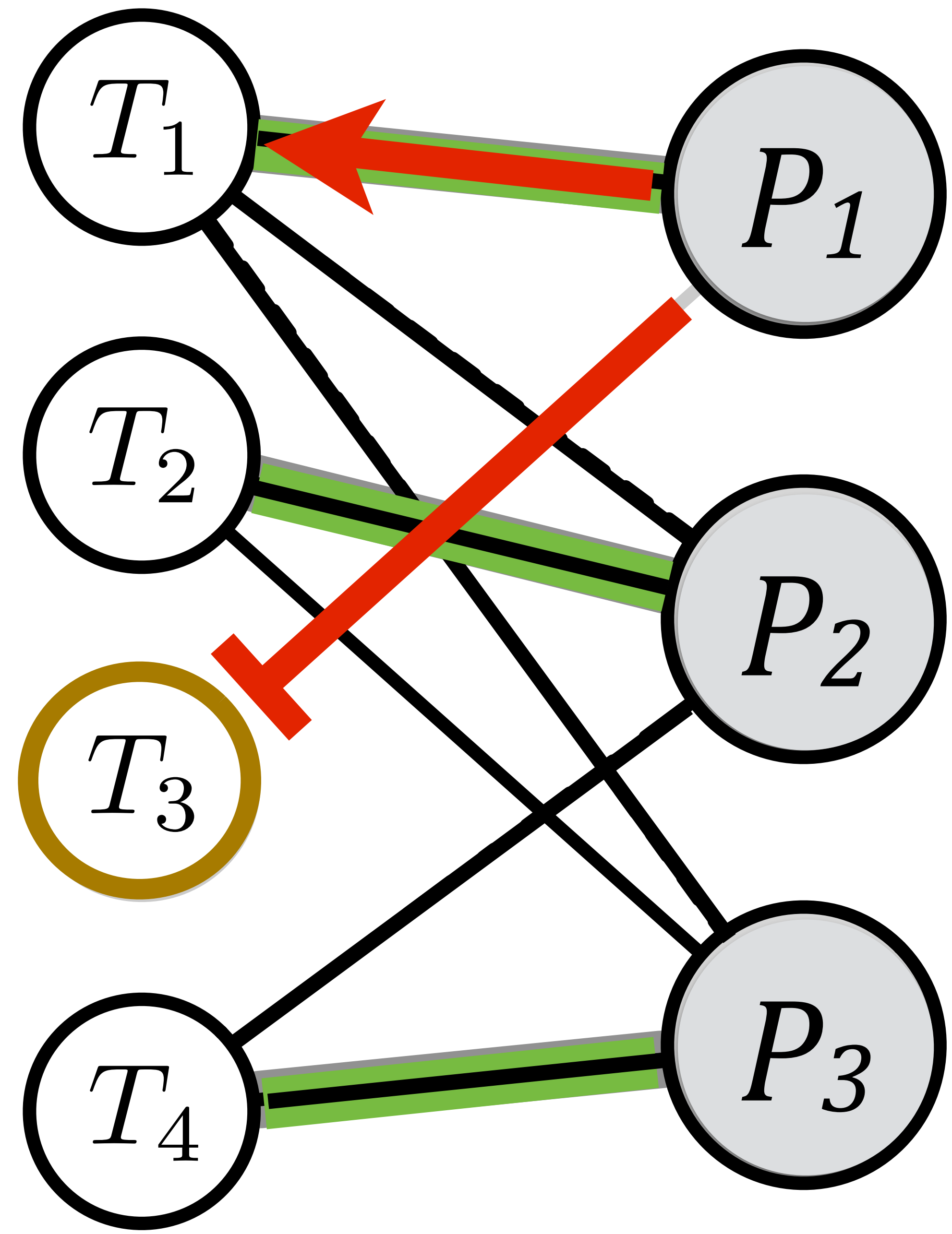
Updating the Matching



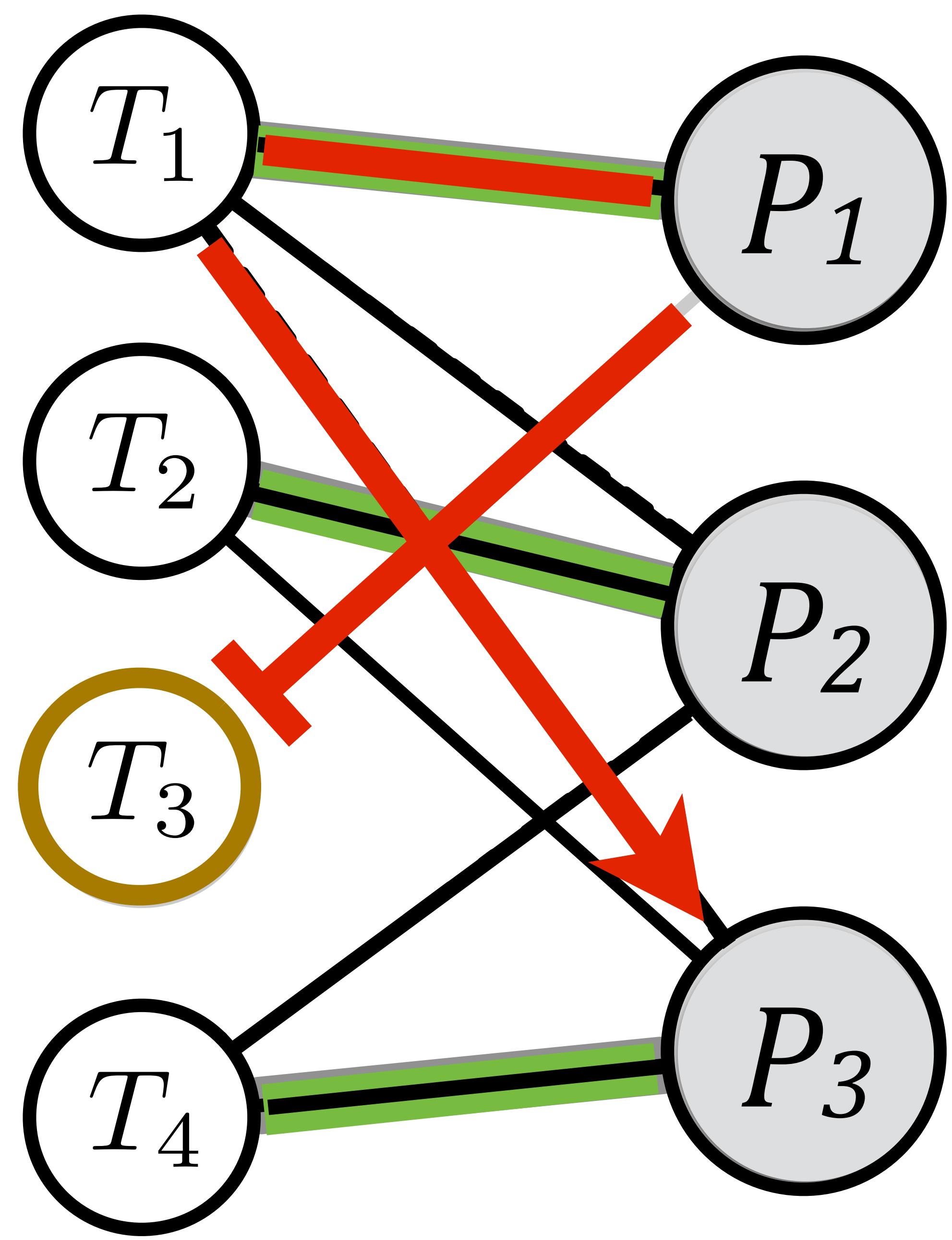
Shifting Tasks with Graph Search



Shifting Tasks with Graph Search



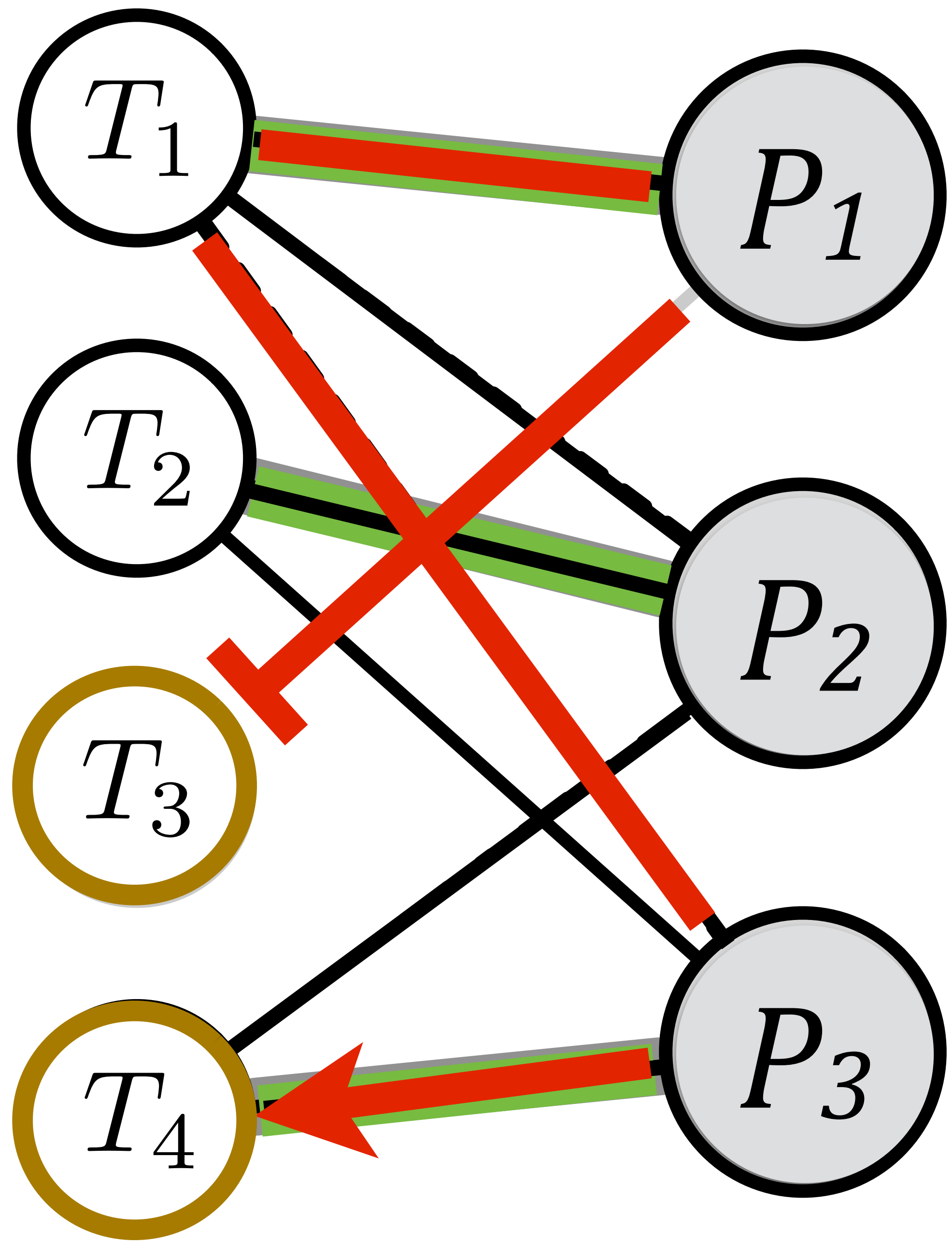
Shifting Tasks with Graph Search



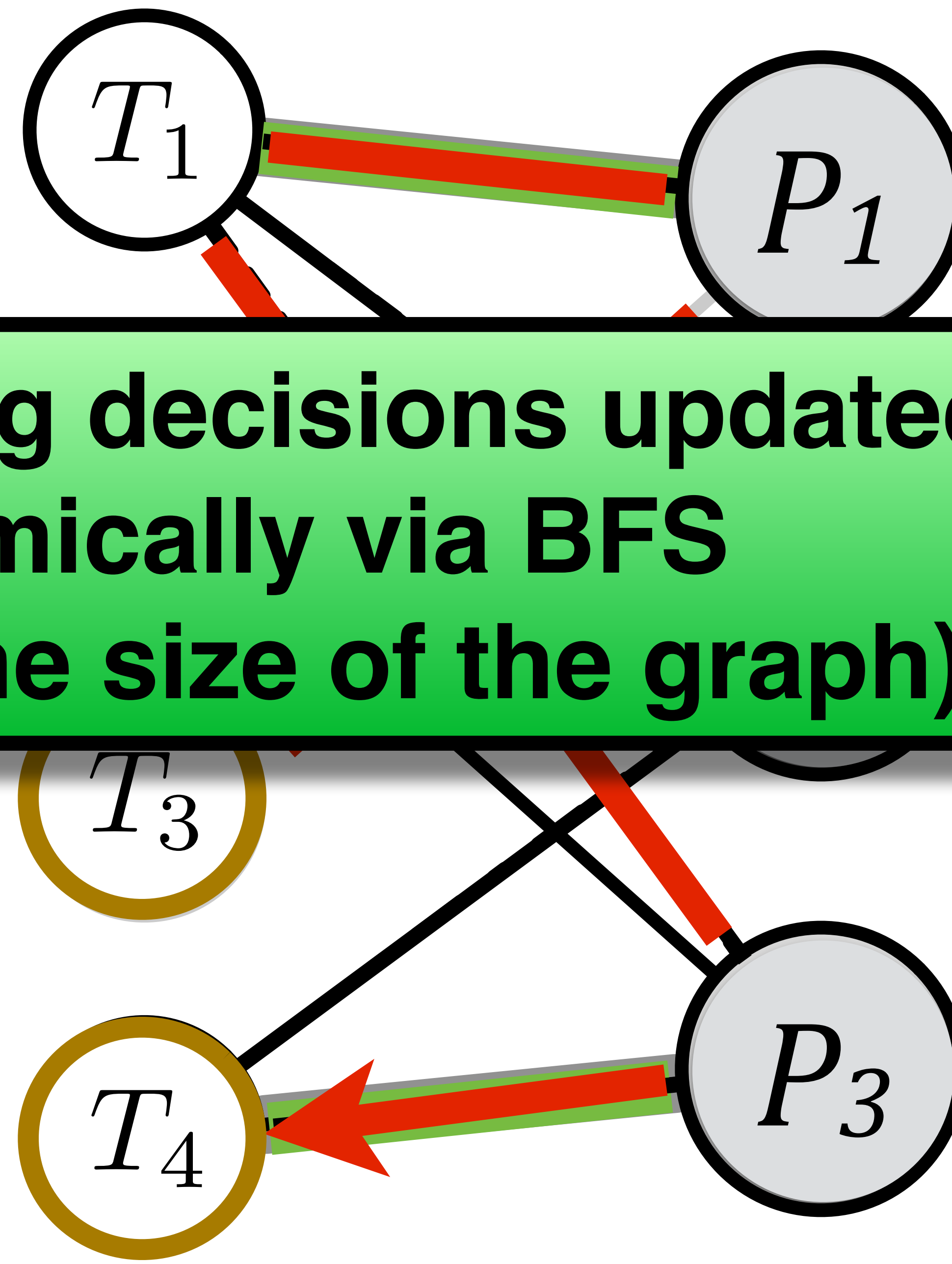
Shifting Tasks with Graph Search

Preempting the lowest-priority task produces an MVM!

Migrations determined via backtracking



Shifting Tasks with Graph Search



Scheduling decisions updated dynamically via BFS (linear in the size of the graph).

Migrations determined via backtracking

This Talk

Limitations of current APA schedulers

How to perform task shifting

Schedulability Analysis

Evaluation

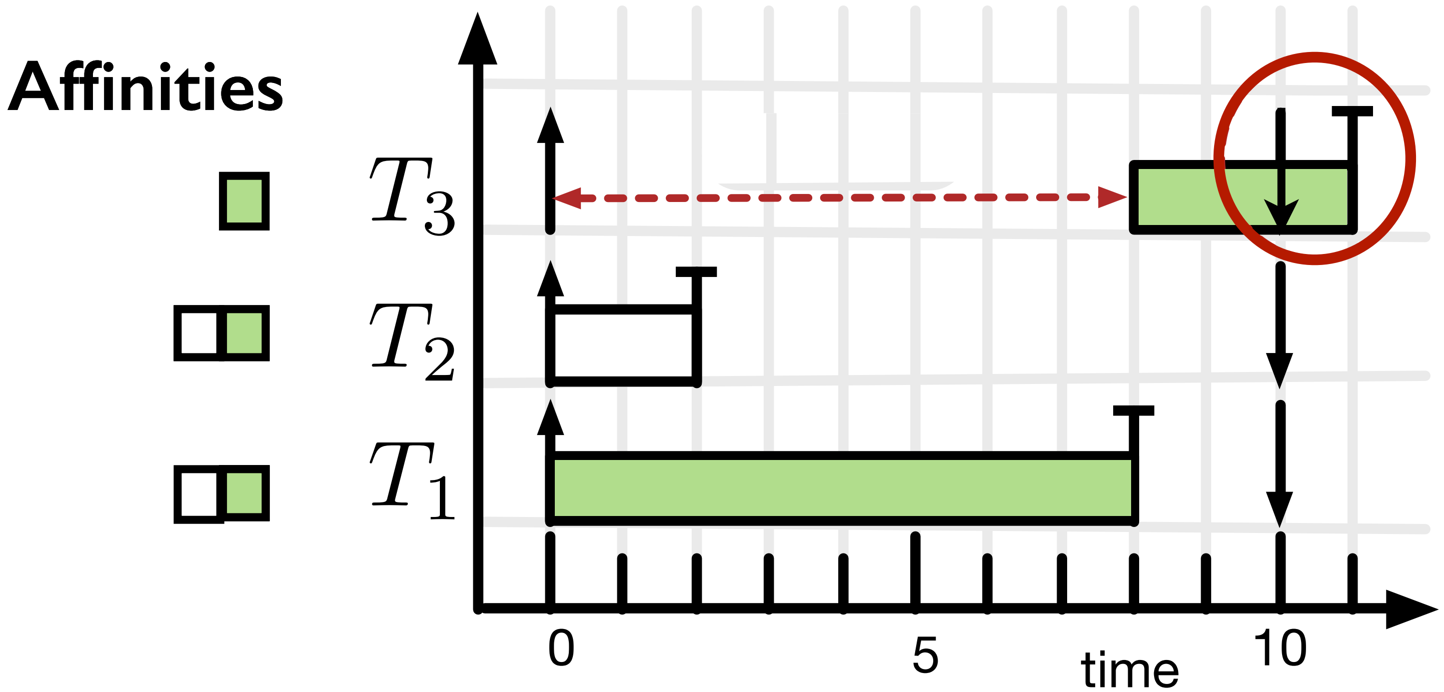
Analyzing Strong APA Scheduling

- Previous work: Schedulability analysis for APA scheduling [1]
  - Works only with Linux's migration semantics
- Recently: Linear-programming-based response-time analysis [2]
  - Faster in practice

We extend the LP-based RTA to consider task shifting!

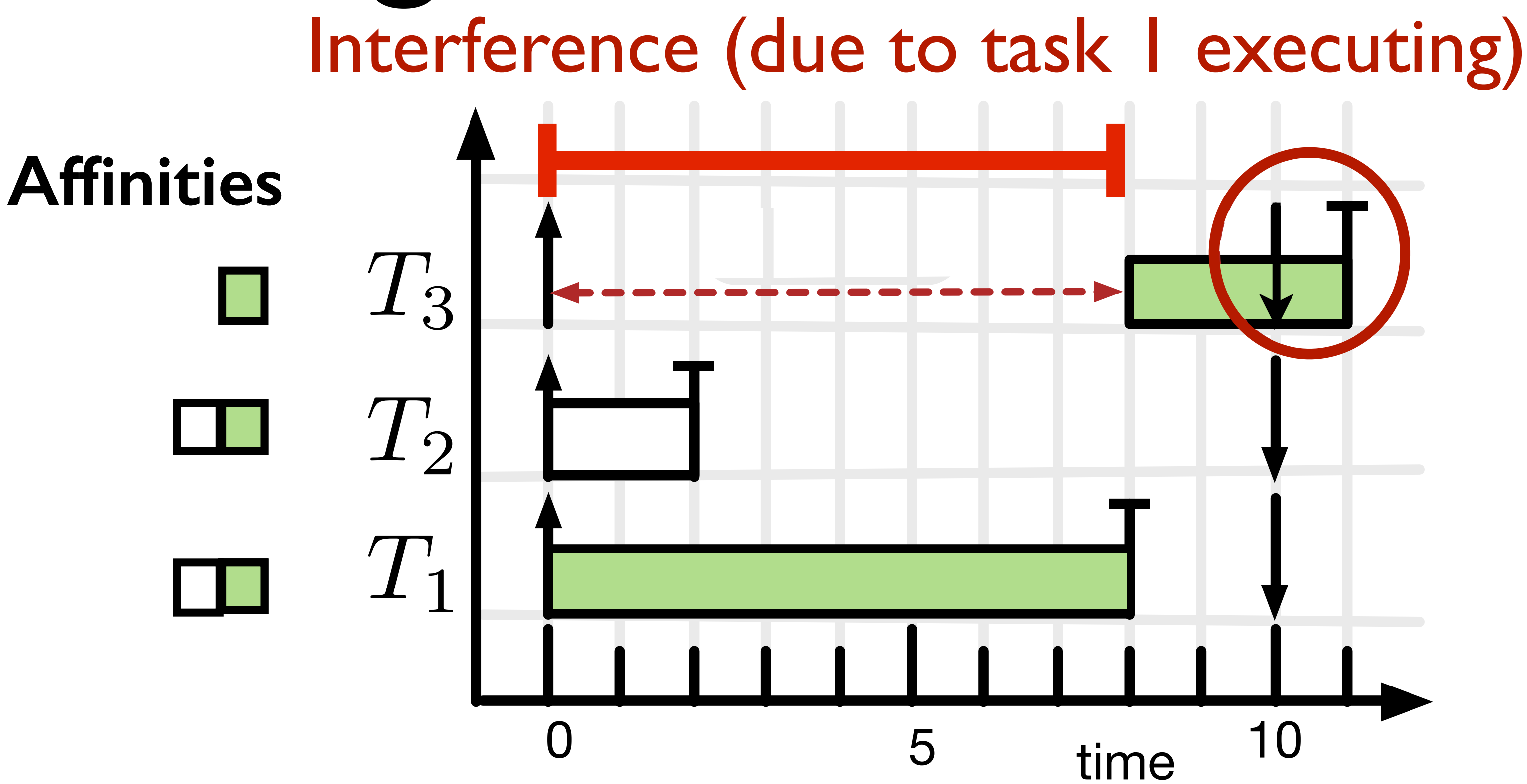
[1] A. Gujarati, F. Cerqueira, and B. Brandenburg, "Schedulability Analysis of the Linux Push and Pull Scheduler with Arbitrary Processor Affinities", ECRTS'13, 2013.
[2] A. Gujarati, F. Cerqueira, and B. Brandenburg, "Multiprocessor Real-Time Scheduling with Arbitrary Processor Affinities: From Practice to Theory", Real-Time Systems, Springer, July 2014.

Shifting Reduces Task Interference



Weak APA (Linux)

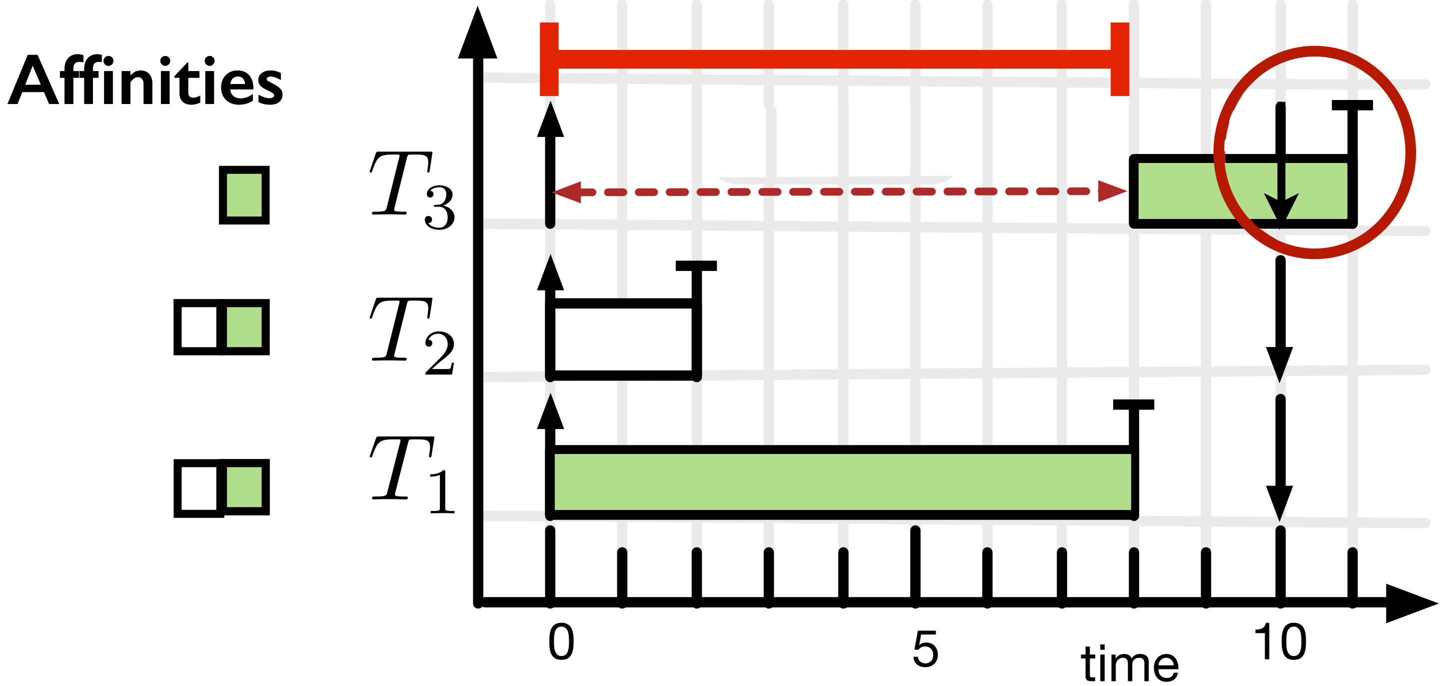
Shifting Reduces Task Interference



Weak APA (Linux)

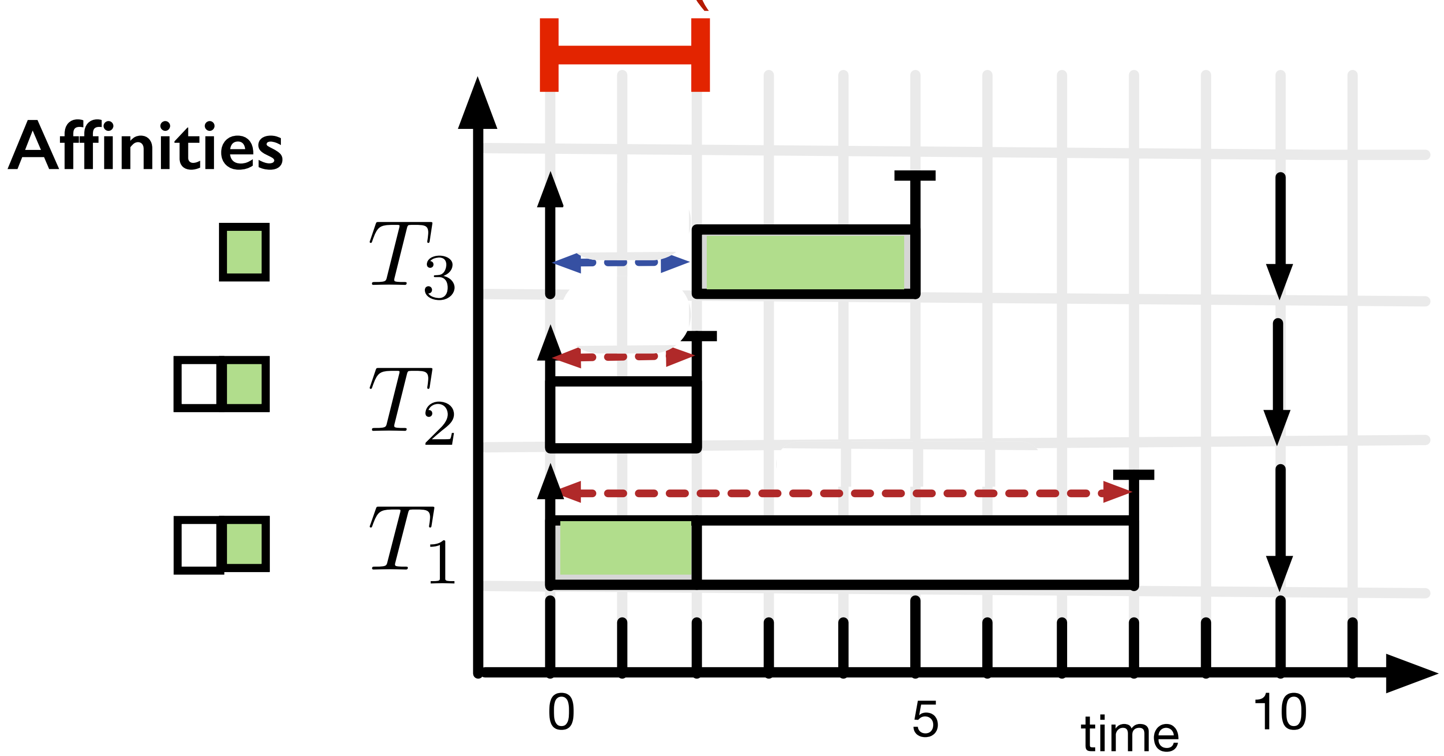
Shifting Reduces Task Interference

Interference (due to task 1 executing)



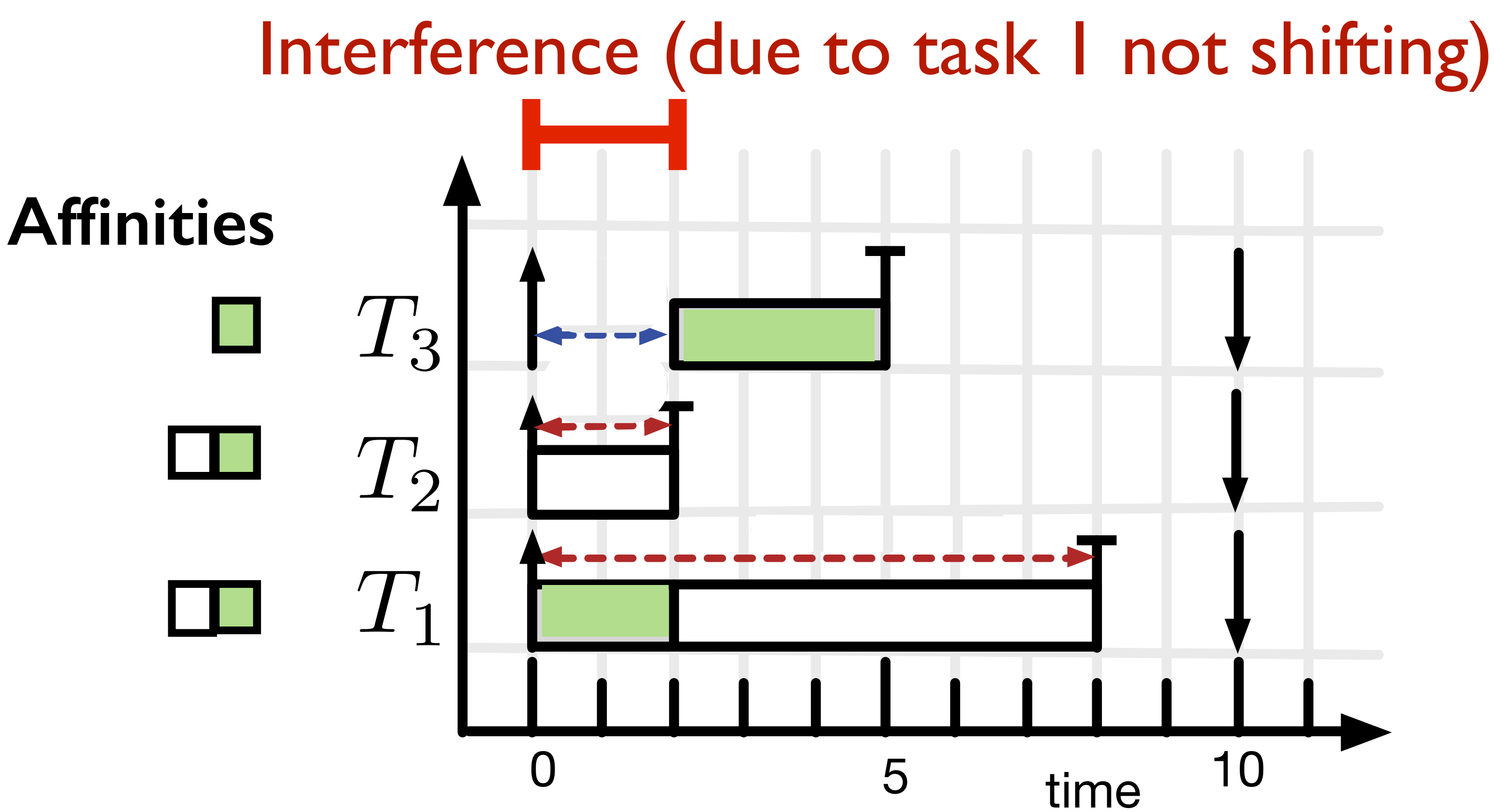
Weak APA (Linux)

Interference (due to task 1 not shifting)



Strong APA

Shifting Reduces Task Interference



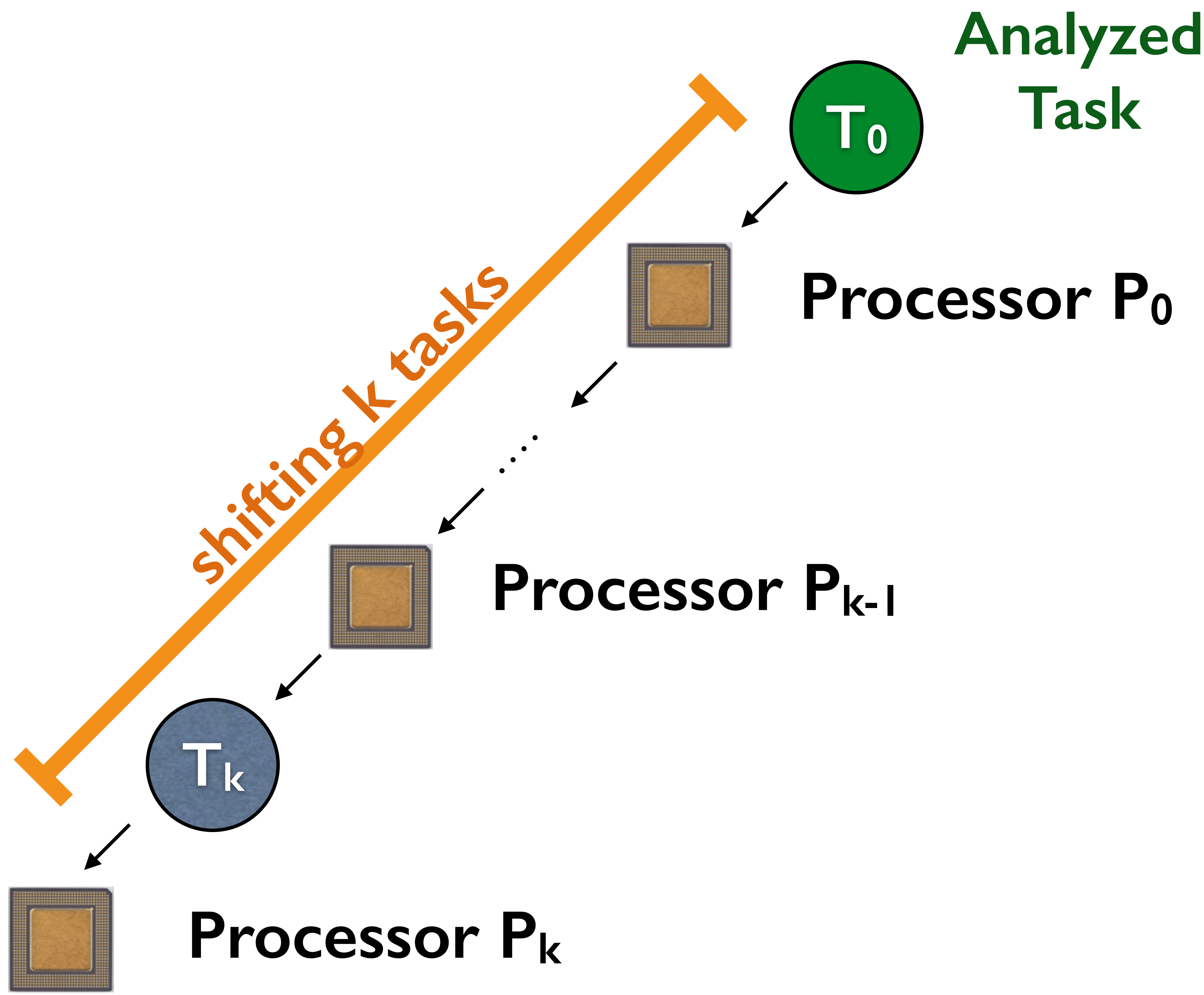
Strong APA

The interference incurred by T_3 is bounded by the time that high priority tasks cannot shift outside T_3's affinity.

This bound is valid only for a single migration!

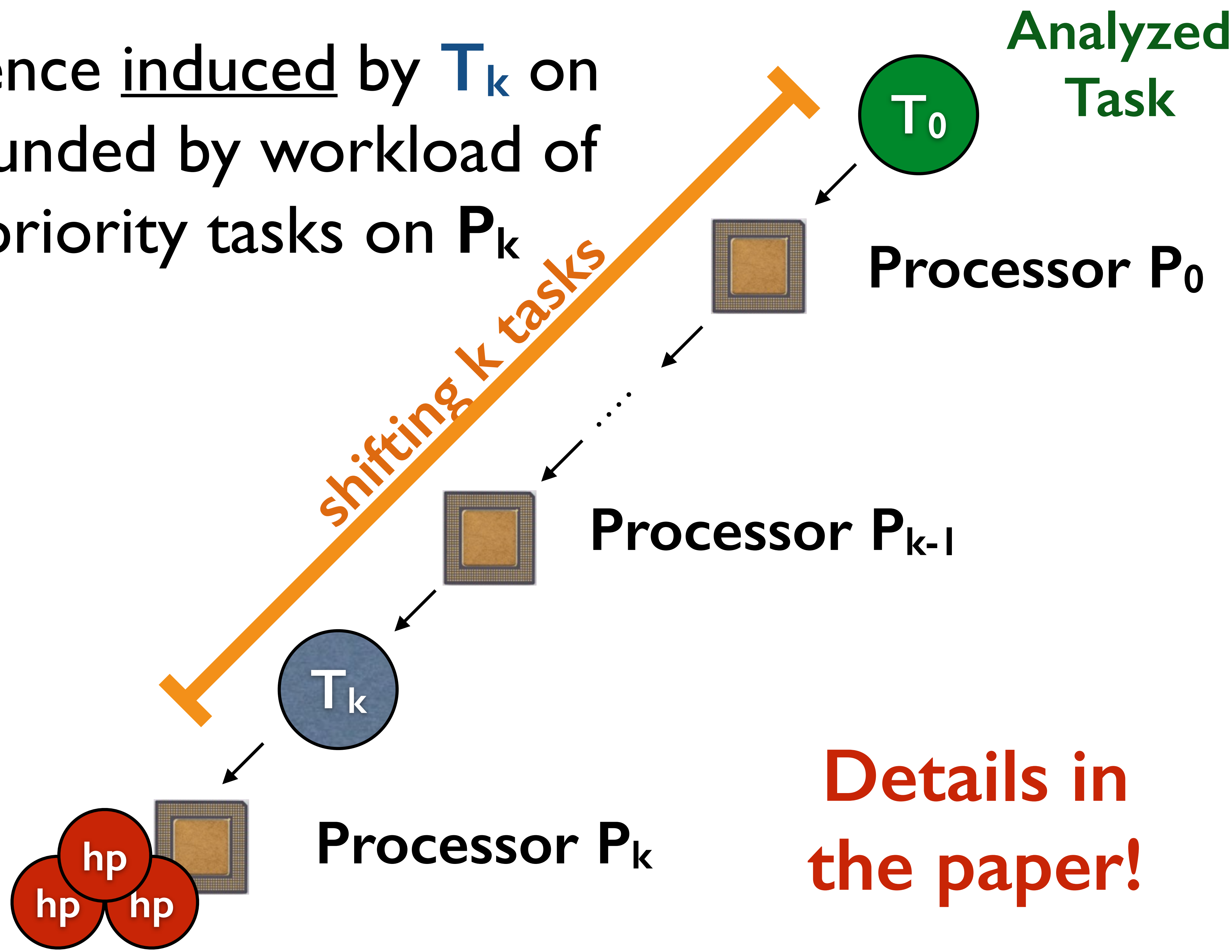
Accounting for K-hop Shifting

Accounting for K-hop Shifting



Accounting for K-hop Shifting

Interference induced by T_k on T_0 is bounded by workload of high-priority tasks on P_k



Details in the paper!

This Talk

Limitations of current APA schedulers

How to perform task shifting

Schedulability Analysis

Evaluation

Two Questions about Strong APA Scheduling

- To which extent does enabling task shifting prevent deadline misses?
- Assuming non-zero migration overheads, do the additional task migrations penalize the benefits of shifting?

Phase 1: Task Set Generation

1) For each point, 800 randomly generated task sets (Emberson et al.'s method [1])

2) Fixed-Priority tasks: DkC order [2]

3) Random generation of affinity assignments

- Try to emulate application requirements
- More details in the paper

[1] P. Emberson, R. Stafford, and R. Davis, "Techniques for the synthesis of multiprocessor tasksets," 1st Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems, 2010

[2] R. Davis and A. Burns, "Improved priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems," Real-Time Systems, vol. 47, no. 1, pp. 1–40, 2011

Phase 2: Schedulability Tests

Weak APA

Sim-Weak: Simulation of APA scheduling **without** shifting

RTA-Weak: Previous response-time analysis for Linux

Strong APA

Sim-Strong: Simulation of APA scheduling **with** shifting

RTA-Strong: New LP-based response-time analysis

Analysis vs. Simulation

Simulation

Upper Bound

Failure ⇒ not schedulable
(necessary condition)

Analysis

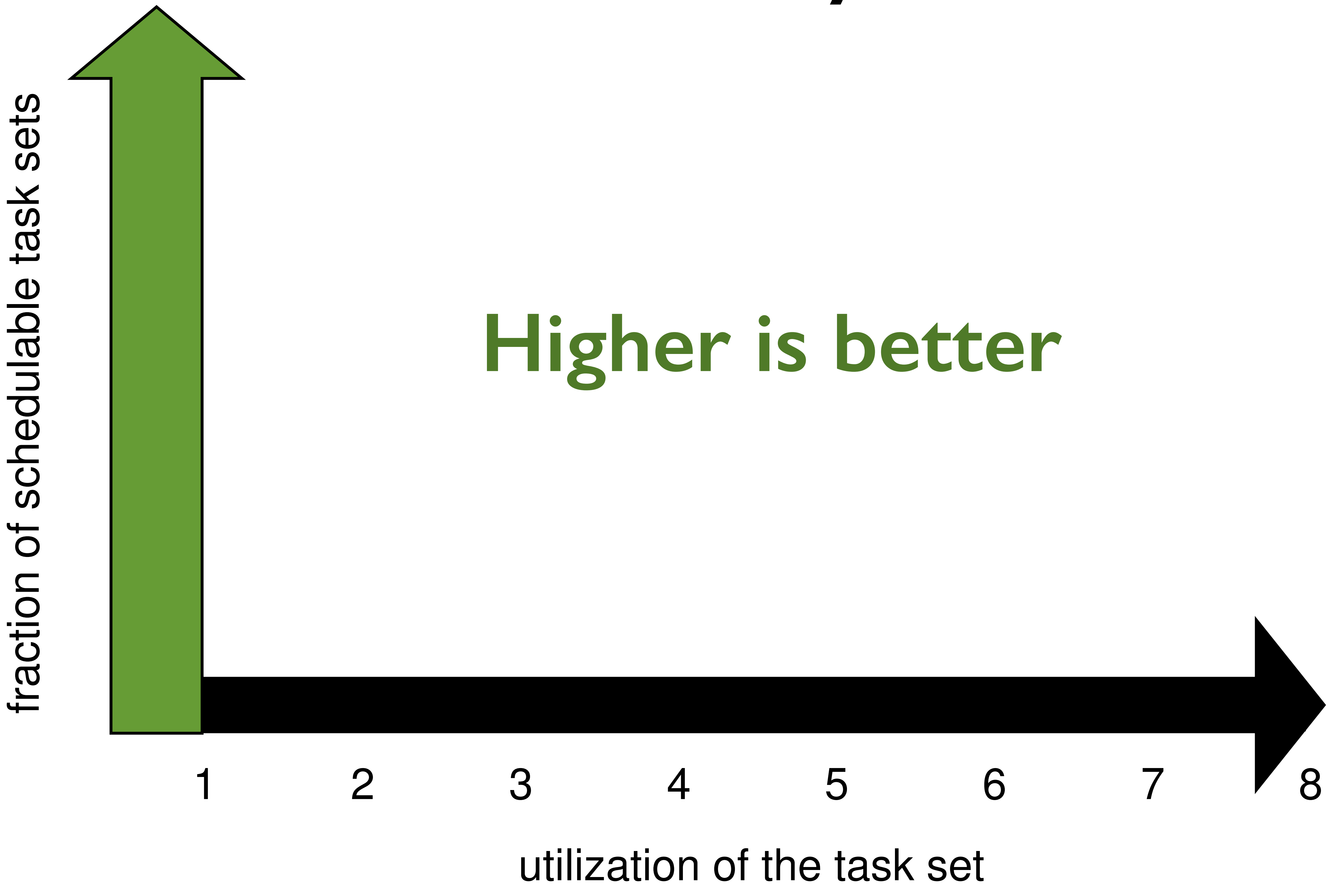
Lower Bound

Success ⇒ schedulable
(sufficient condition)

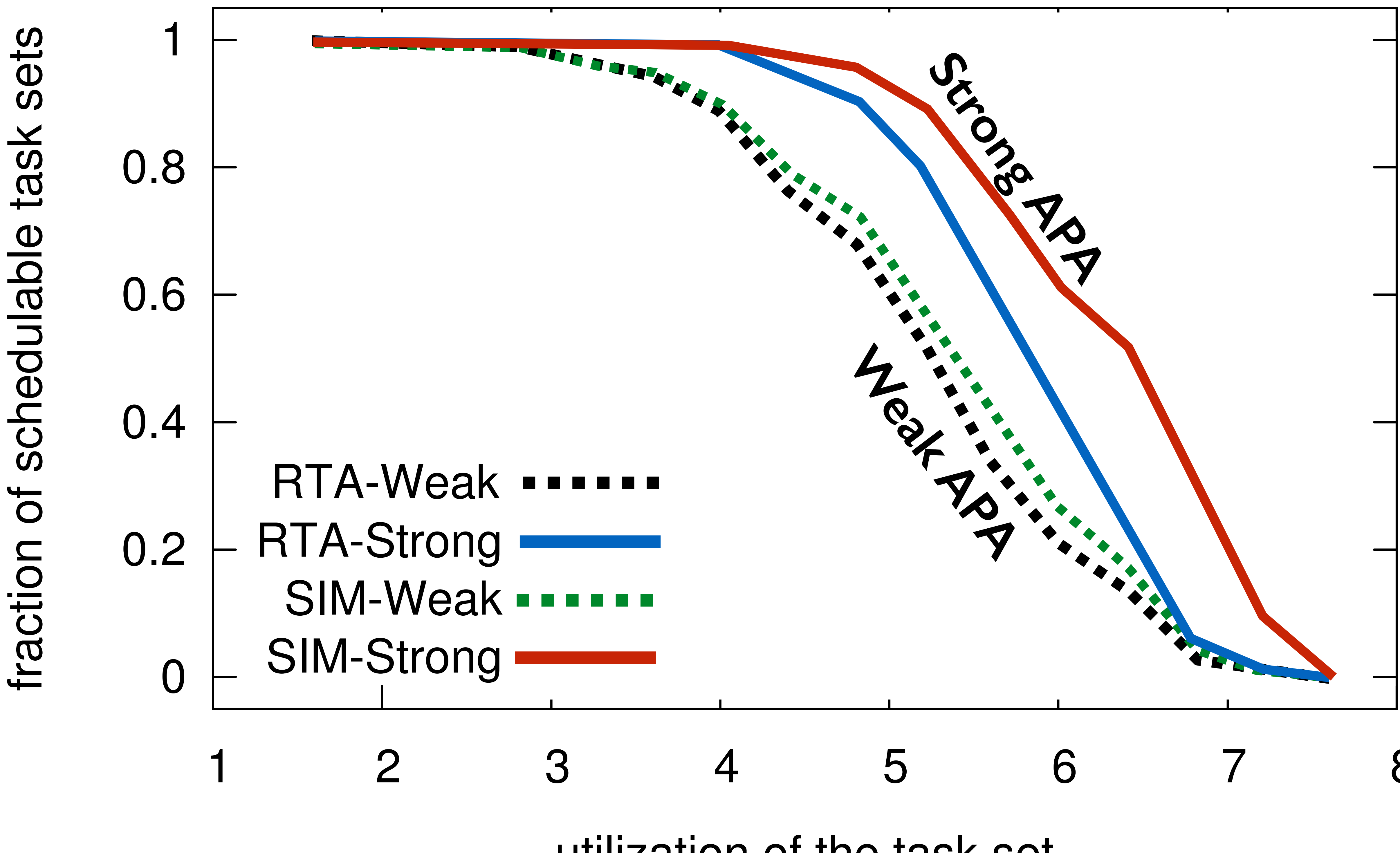
Question 1

- To which extent does enabling task shifting prevent deadline misses?

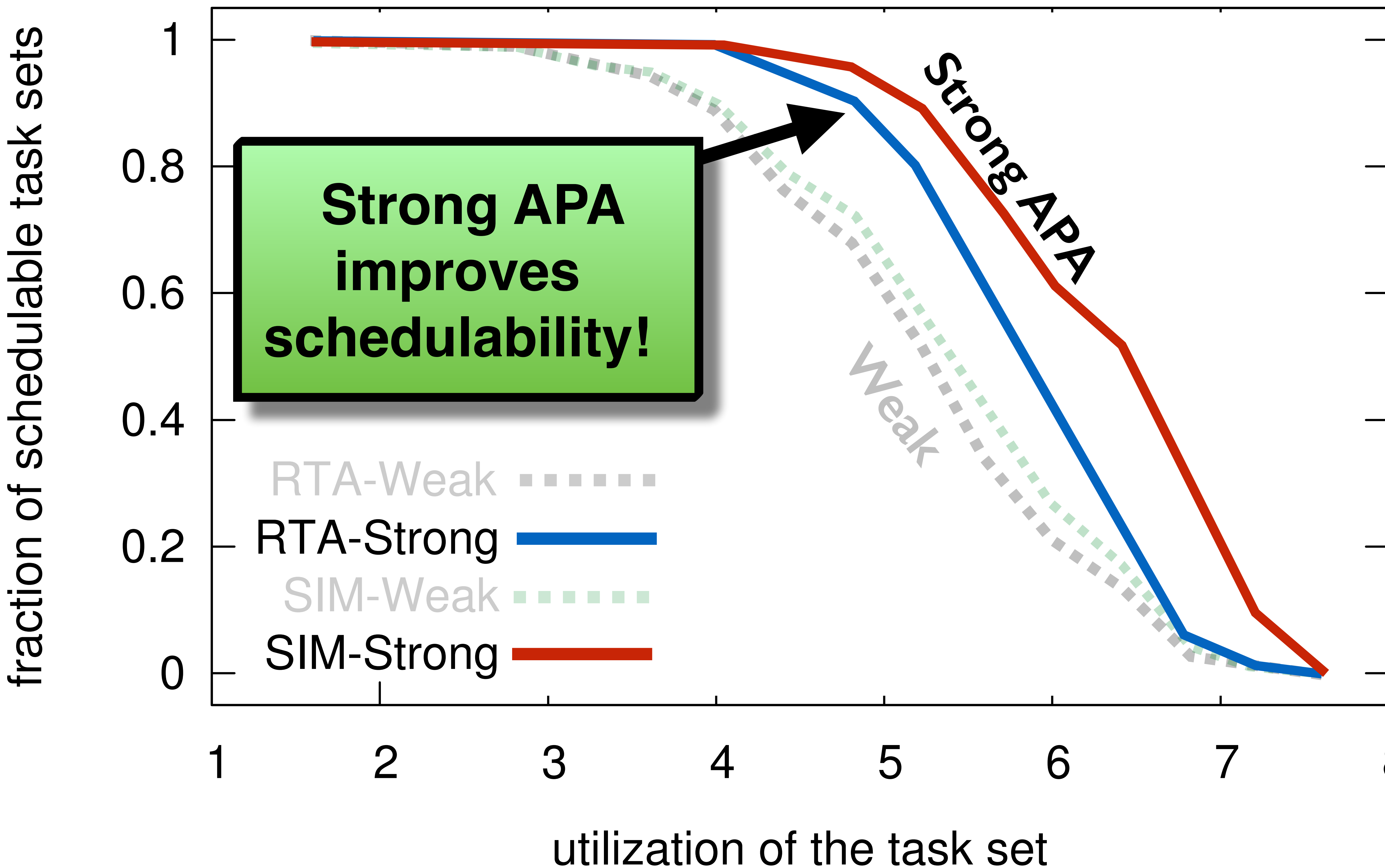
Schedulability Curve



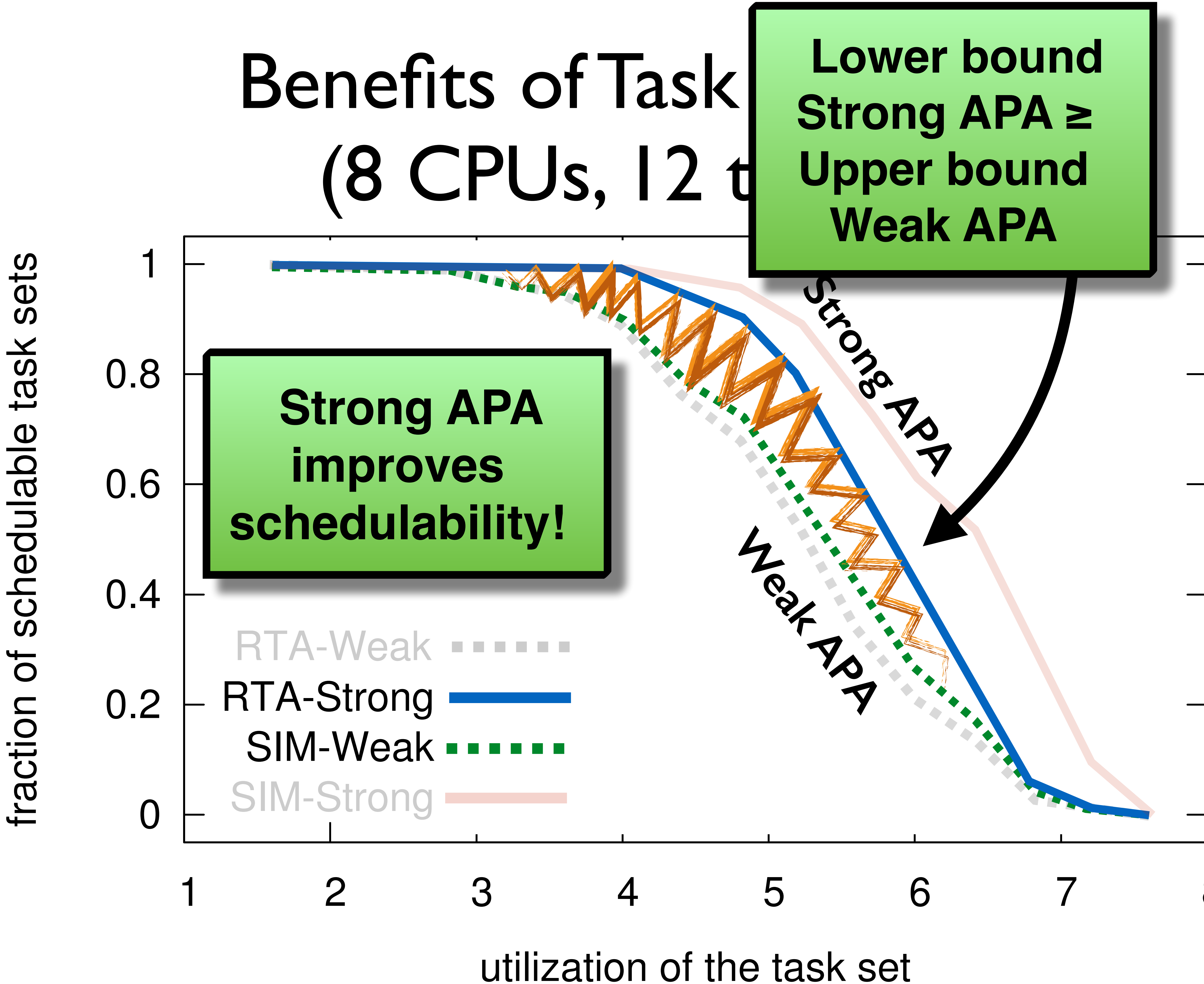
Benefits of Task Shifting (8 CPUs, 12 tasks)



Benefits of Task Shifting (8 CPUs, 12 tasks)

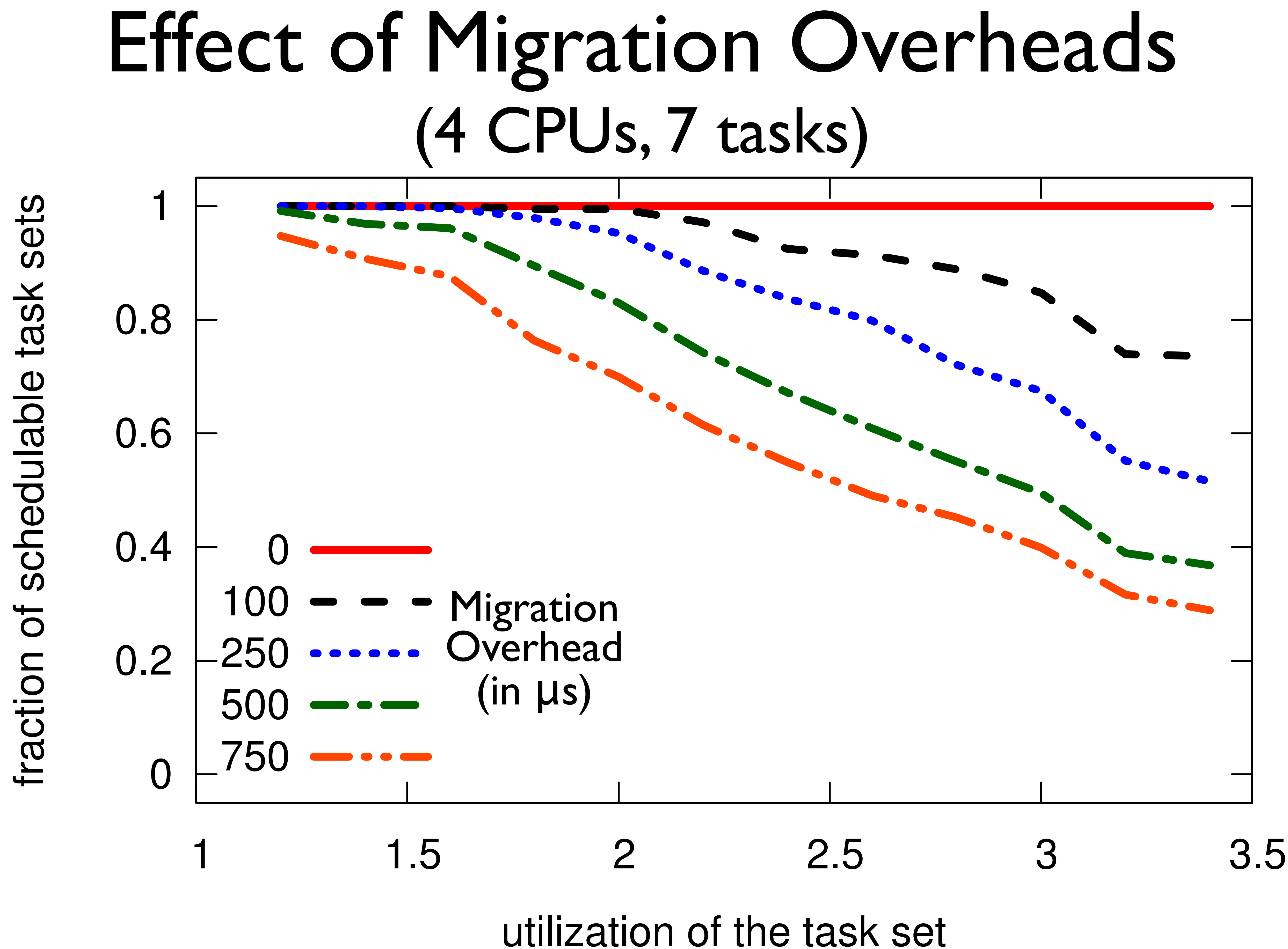


Benefits of Task
(8 CPUs, 12 t

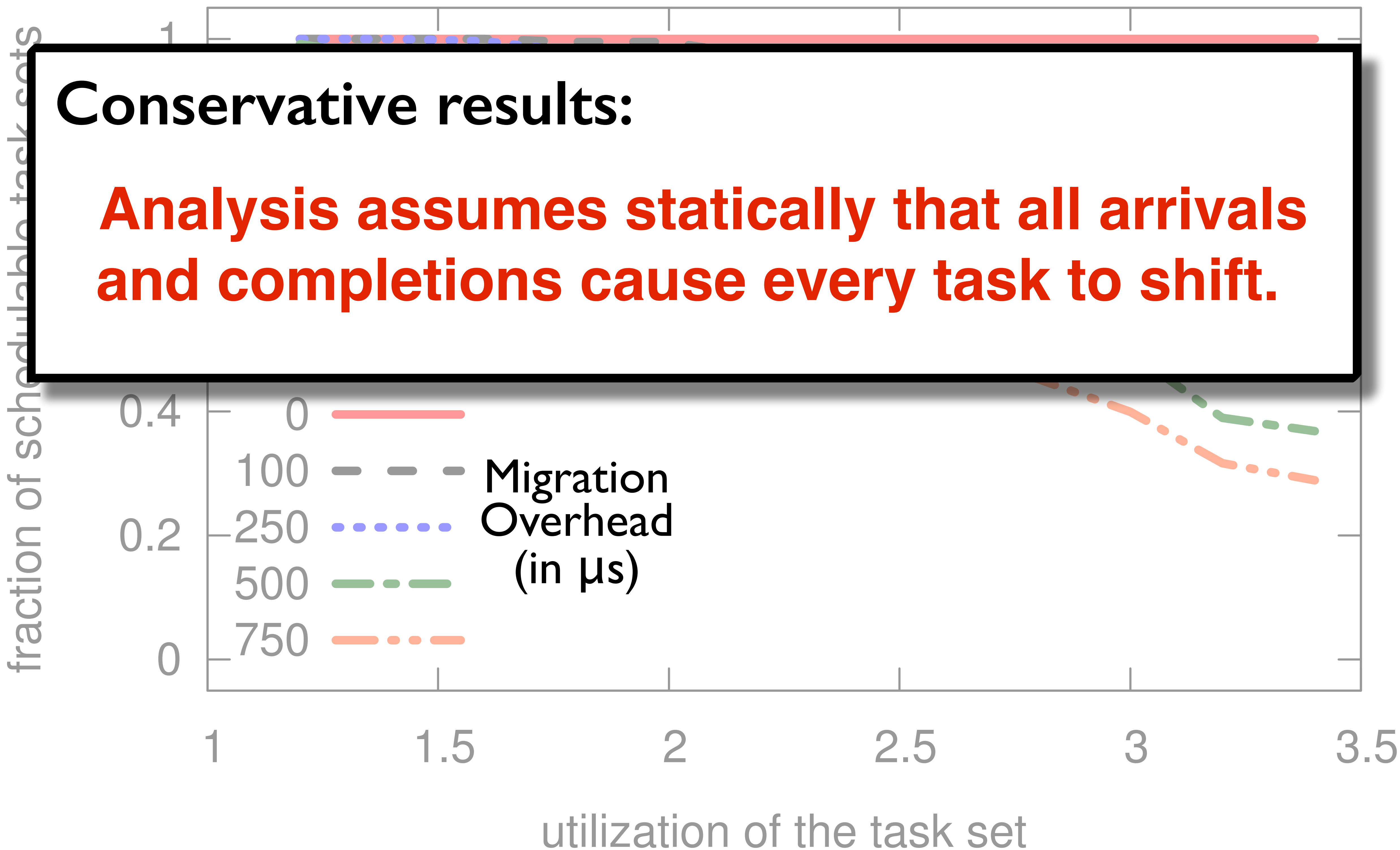


Question 2

- Assuming non-zero migration overheads, do the additional task migrations penalize the benefits of shifting?


Effect of Migration Overheads
(4 CPUs, 7 tasks)
fraction of schedulable task sets
utilization of the task set
1
0.8
0.6
0.4
0.2
0
1
1.5
2
2.5
3
3.5
0
100
250
500
750
Migration
Overhead
(in μs)

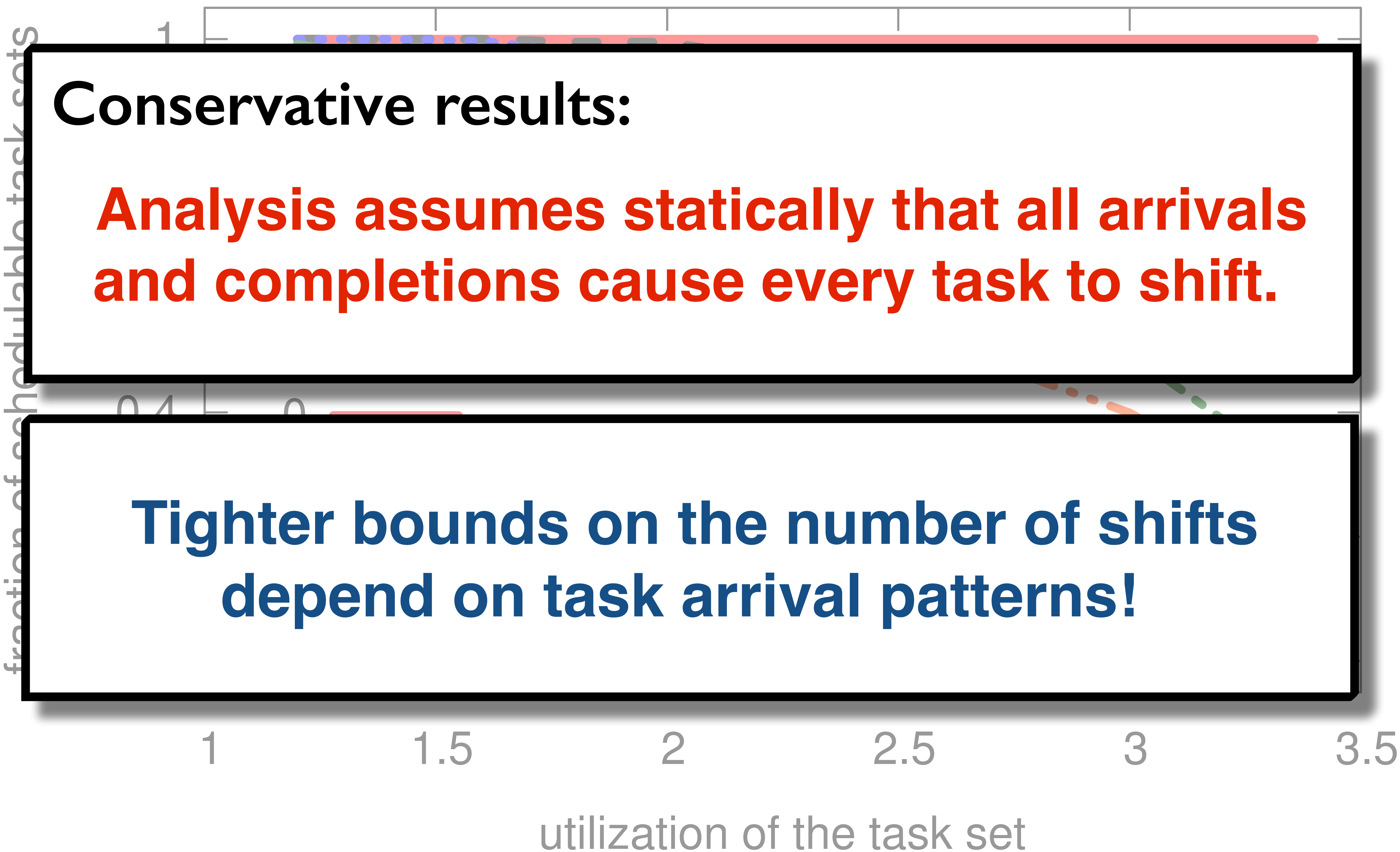
Pessimism in Overhead Analysis



Conservative results:

Analysis assumes statically that all arrivals and completions cause every task to shift.

Pessimism in Overhead Analysis



Conservative results:

Analysis assumes statically that all arrivals and completions cause every task to shift.

Tighter bounds on the number of shifts depend on task arrival patterns!

Conclusion

- We proposed new migration semantics called **strong APA scheduling**, with better temporal guarantees and maintaining API compatibility with major OSs.
- We presented a dynamic algorithm for scheduling decisions based on task shifting.
- Strong APA scheduling significantly improves schedulability (assuming negligible overheads). Migration overheads can still be analyzed (with pessimism).