An Exact Schedulability Test for Non-Preemptive Self-Suspending Real-Time Tasks

Beyazit Yalcinkaya **Mitra Nasri** Björn Brandenburg

DATE'19

The paper in a nutshell

The paper provides the

first <u>exact</u> schedulability test

for the following open research problems:

Global <u>multiprocessor</u>

fixed-priority scheduling of

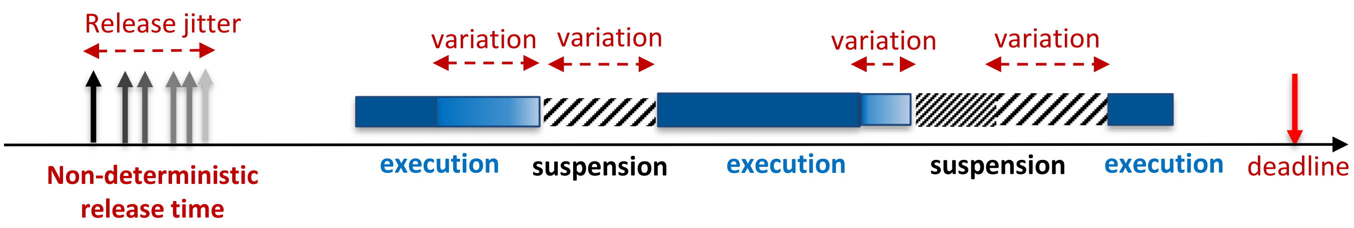
Uniprocessor

fixed-priority scheduling of

- **non-preemptive tasks**
- **limited-preemptive tasks**
- **limited-preemptive segmented self-suspending tasks**

For tasks with **bounded** yet **non-deterministic**

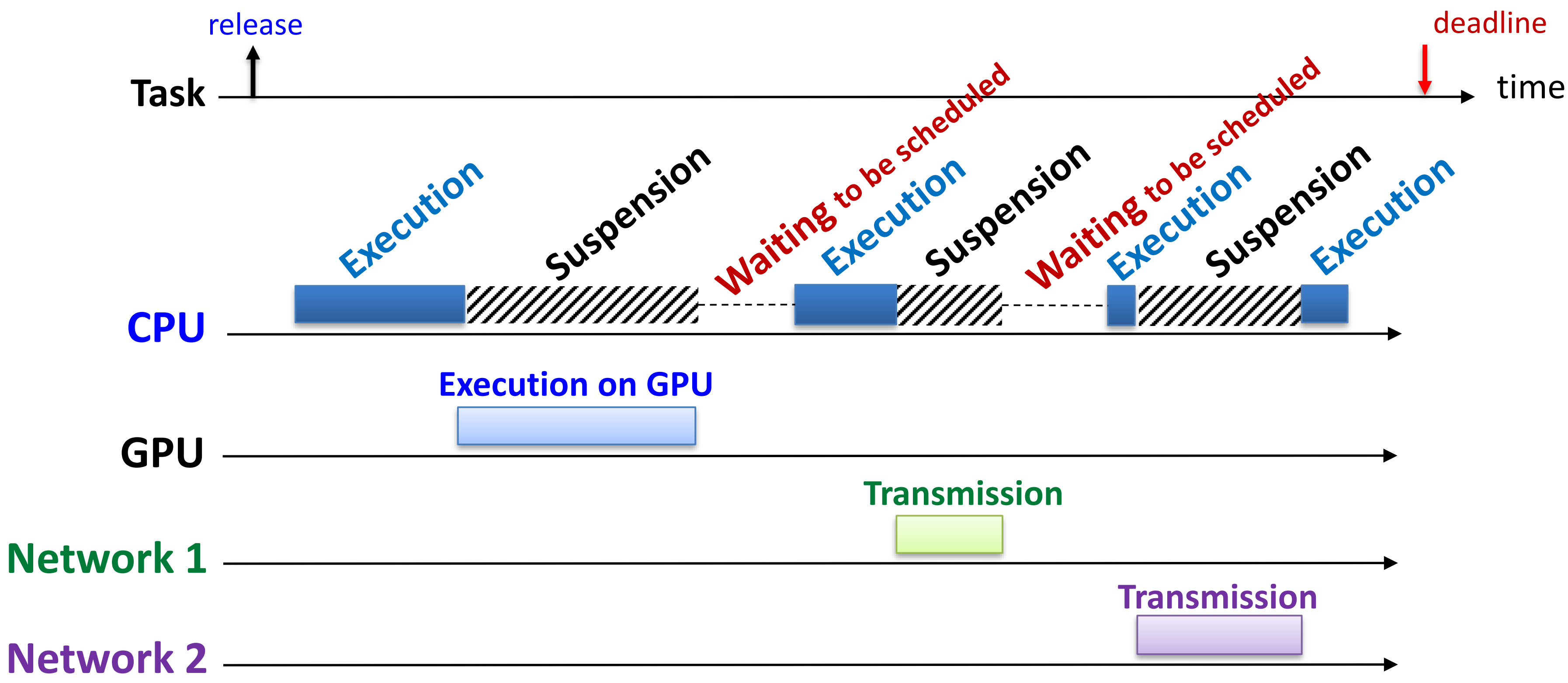
- Execution time
- Suspension time
- Release jitter



Segmented self-suspending task model

A rich model to express systems that use/have

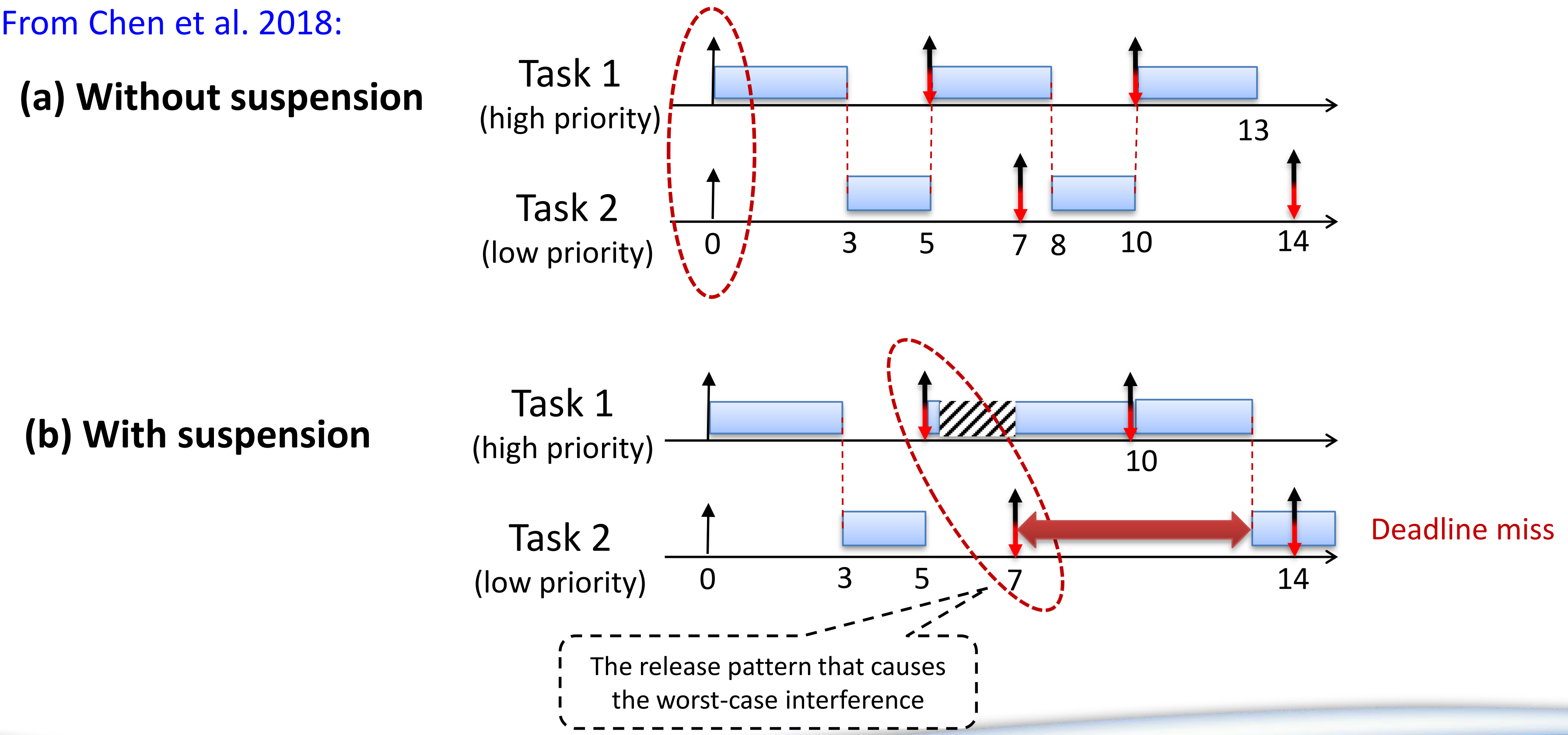
hardware accelerators (GPUs, co-processors, etc.)	intensive I/O accesses	computation offloading (to the cloud, edge, etc.)



Why is analyzing self-suspending tasks hard?

Classic "worst-case release" scenarios
cannot be used for self-suspending tasks

From Chen et al. 2018:

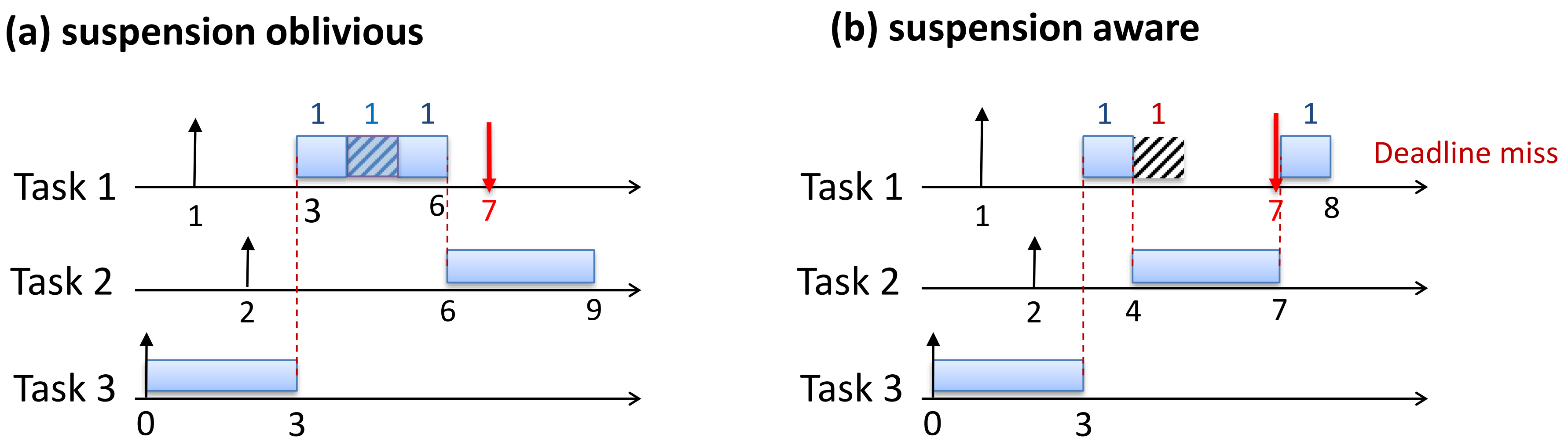


Why is analyzing self-suspending tasks hard?

Suspension-oblivious analysis is unsafe

(i.e., under limited-preemptive scheduling, treating suspension segments as if they were execution segments is unsound)

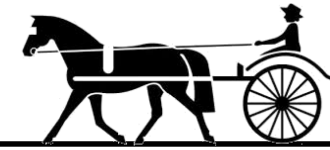
From this paper:



This counter example is valid for both periodic and sporadic limited-preemptive tasks.

Current challenges

Industry is rapidly moving towards more complex execution models
(including **self-suspending tasks**)



State of the art on self-suspending tasks is not advancing fast enough

Prior work is focused on **sufficient (pessimistic) schedulability tests**

Even without self-suspensions, there is **no exact analysis** for **global limited-preemptive scheduling**

Given the lack of an exact test, **there is no way** to know **how good or bad the existing tests** are

Designing an exact test: where to start?

Schedulability analysis problem in real-time systems → Map to → **Reachability problem in timed automata**

Of course, we are not the first to observe this!

(Guan et al. 2007 and 2008, David et al. 2009, Sheng et al. 2010, Cordovilla et al. 2011, David et al. 2011, Cicirelli et al. 2012, Gu et al. 2014, …)

Some of the existing analyses based on timed automata use "stop watches"
(e.g., David et al 2009)

↓

This makes the reachability problem undecidable
(in practice, these tests are only sufficient and very inaccurate)

Other analyses use models that allow for **impossible priority inversions** and hence are pessimistic (for periodic tasks)

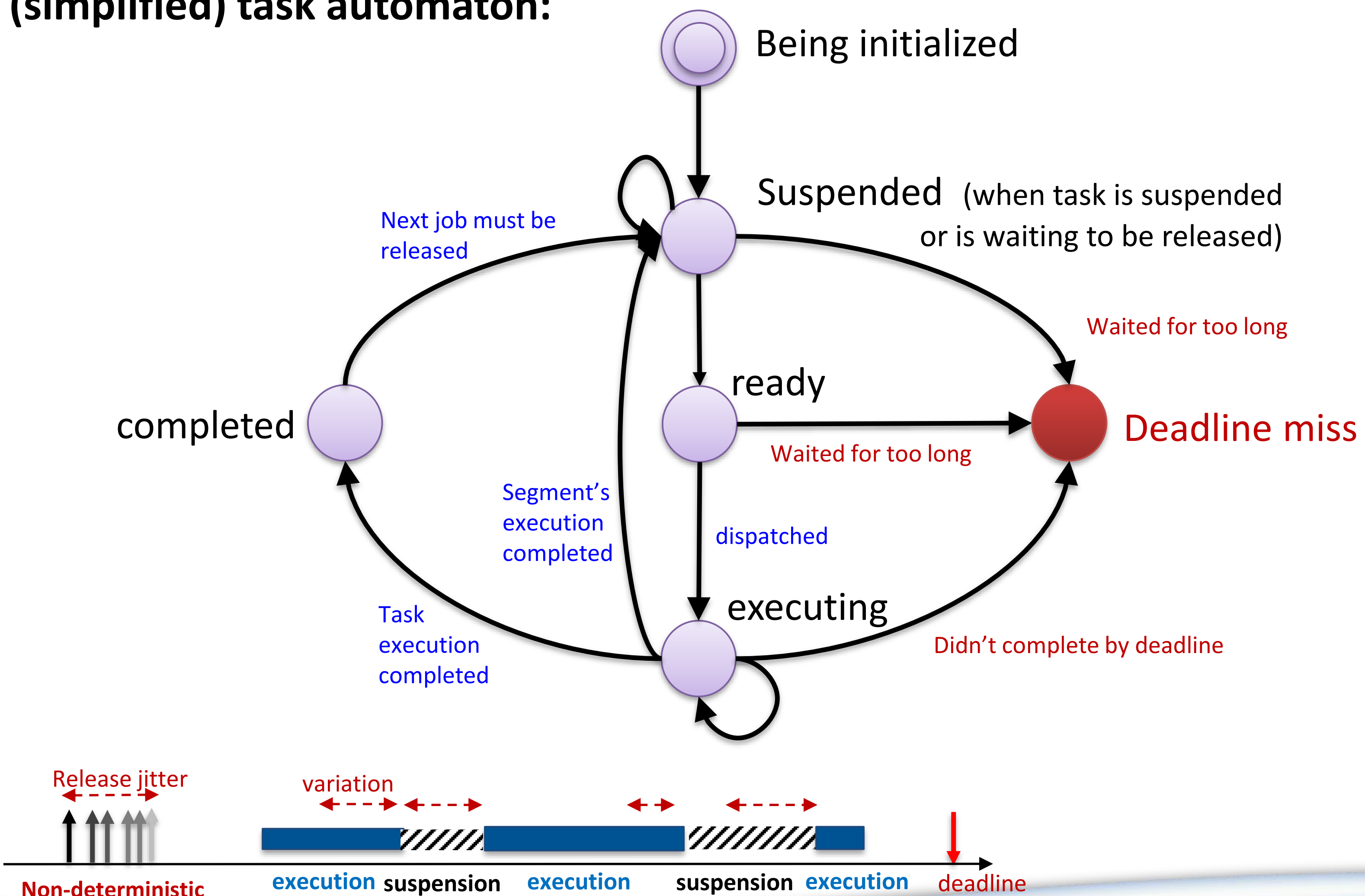
↓

Examples in the paper

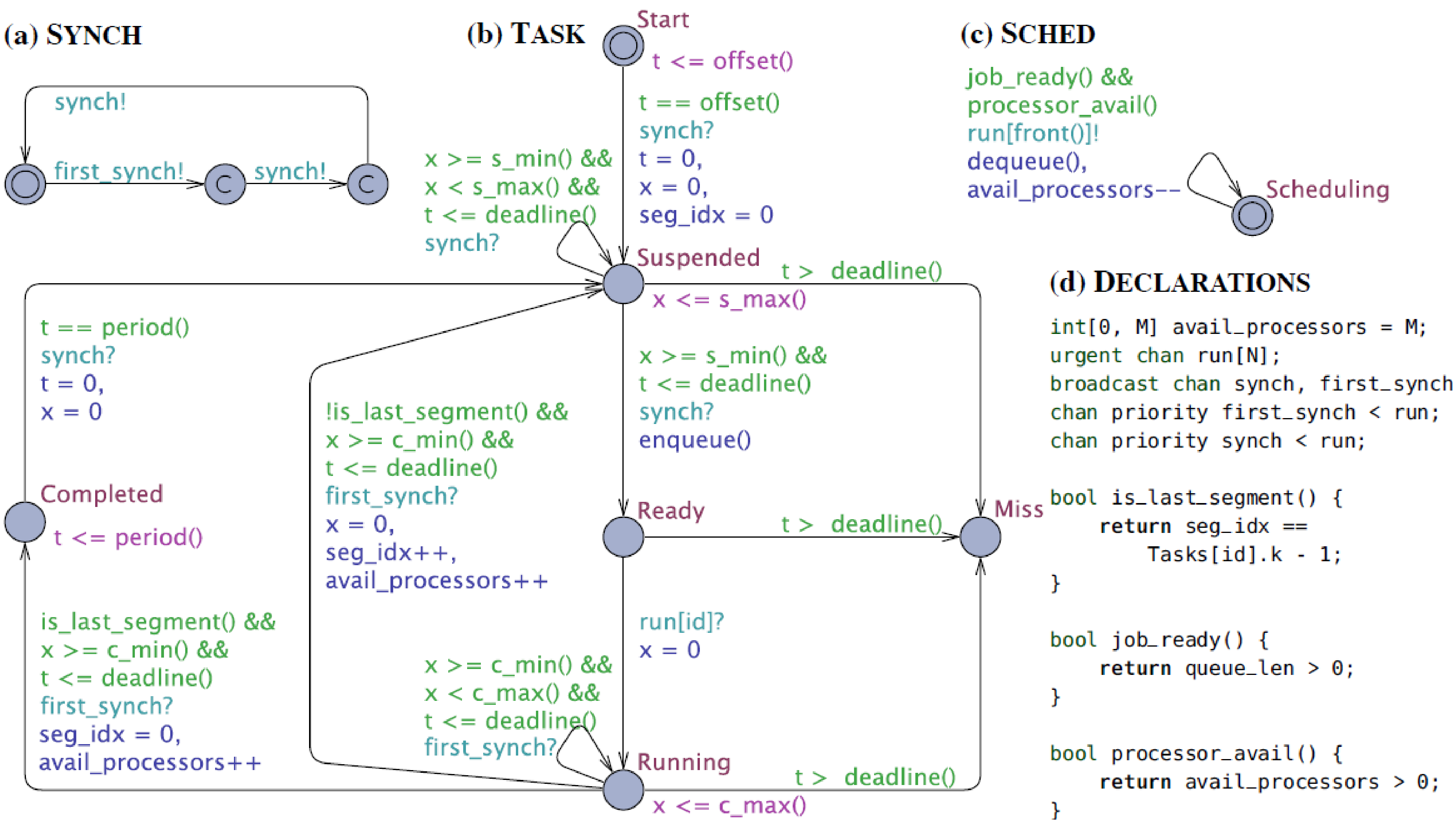
Designing an exact test: high-level idea

Model **Task**, **Scheduler**, and the **Event Synchronizer** as timed automata.

(simplified) task automaton:



Designing an exact test: high-level idea



More details in the paper

Experiments

Evaluation

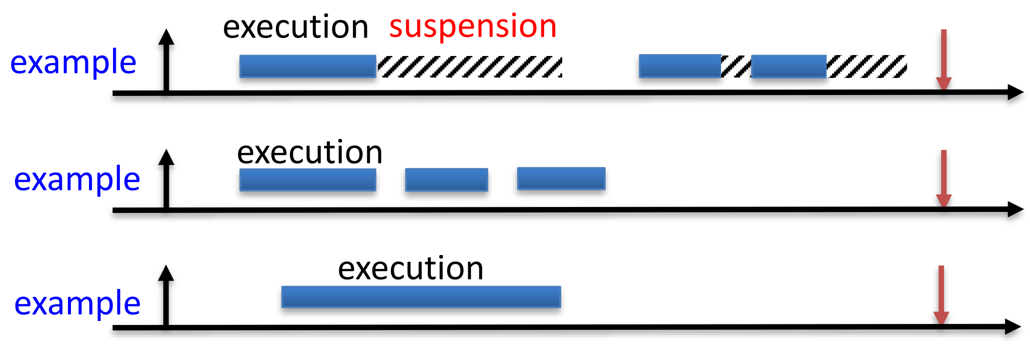
Questions:

- **How much schedulability gain is achieved using our exact analysis?**

- **How far does the analysis scale w.r.t.**
  - Number of tasks
  - Number of processors
  - Number of code segments
  - Length of self-suspensions

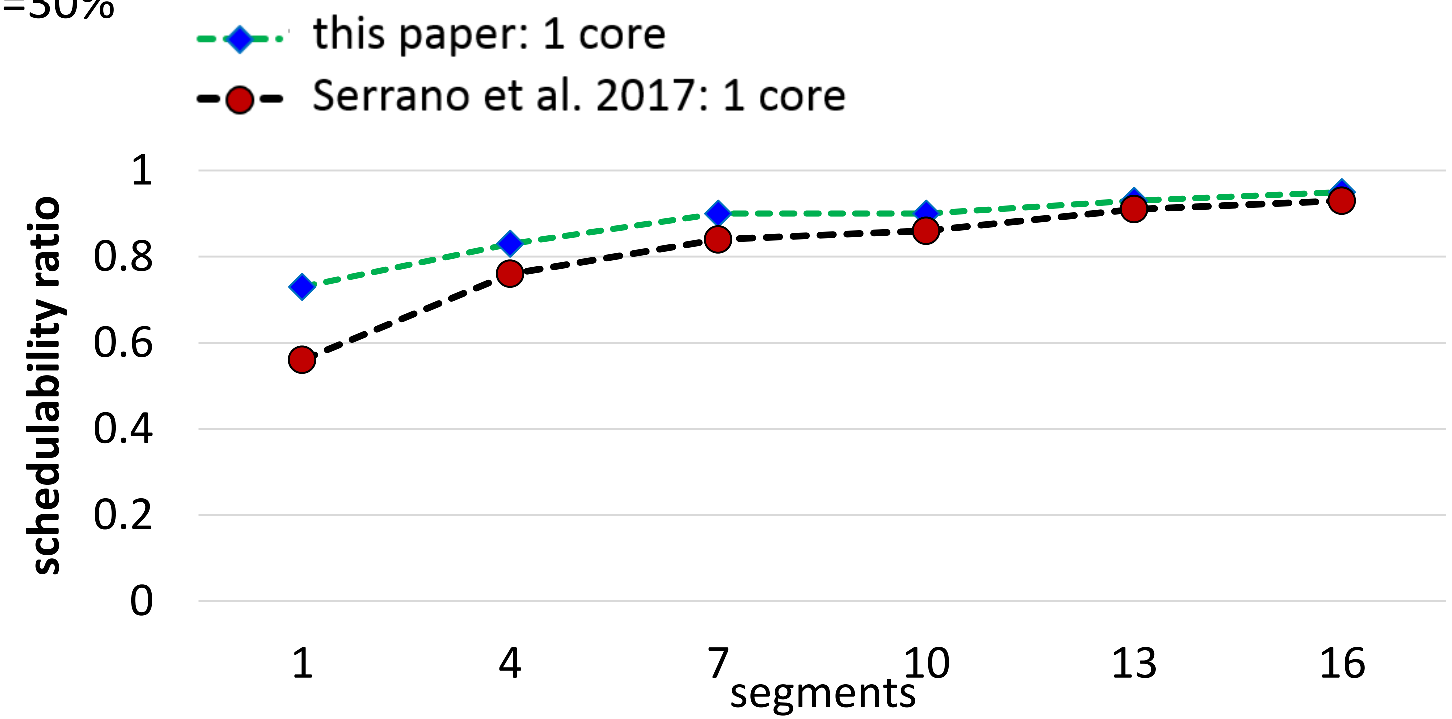
Considered task models:

- Segmented self-suspending limited-preemptive tasks
- Limited-preemptive tasks
- Non-preemptive tasks



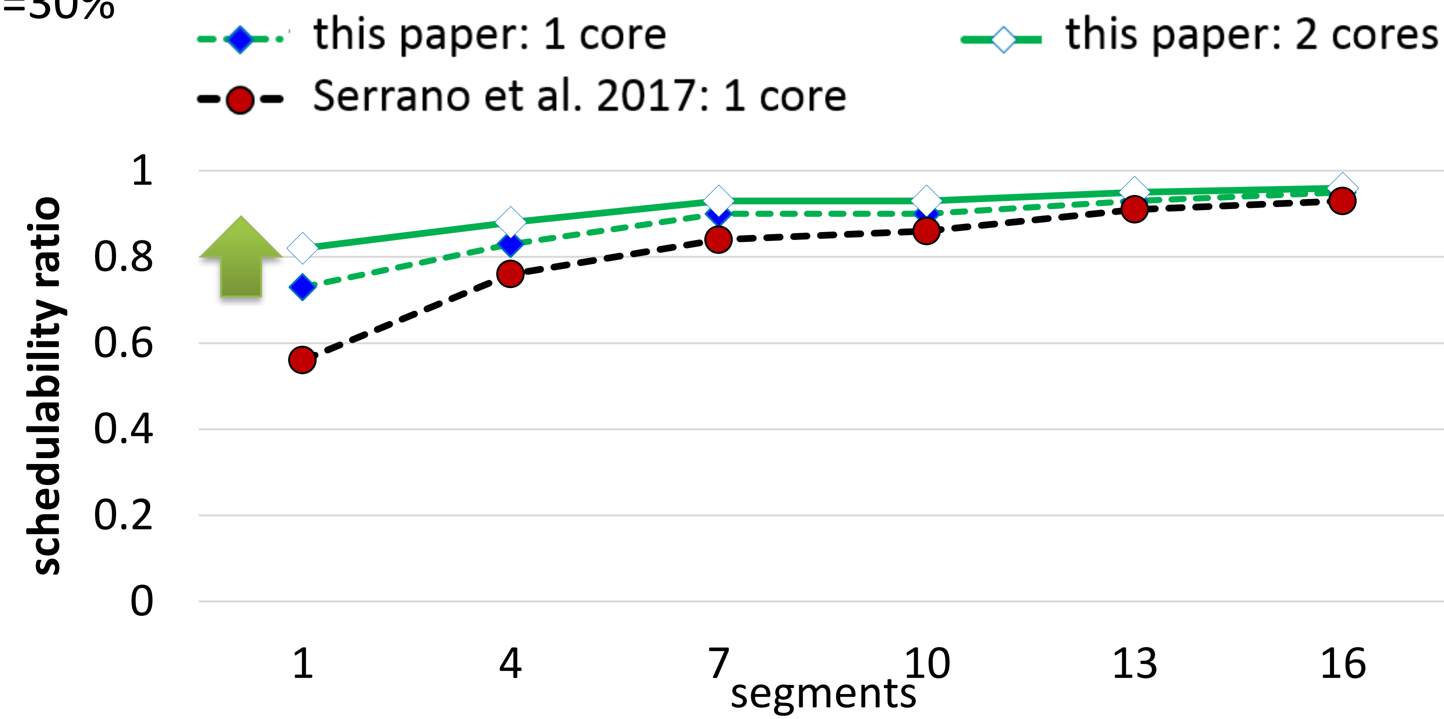
Limited-preemptive tasks

Utilization=30%
10 tasks



Limited-preemptive tasks

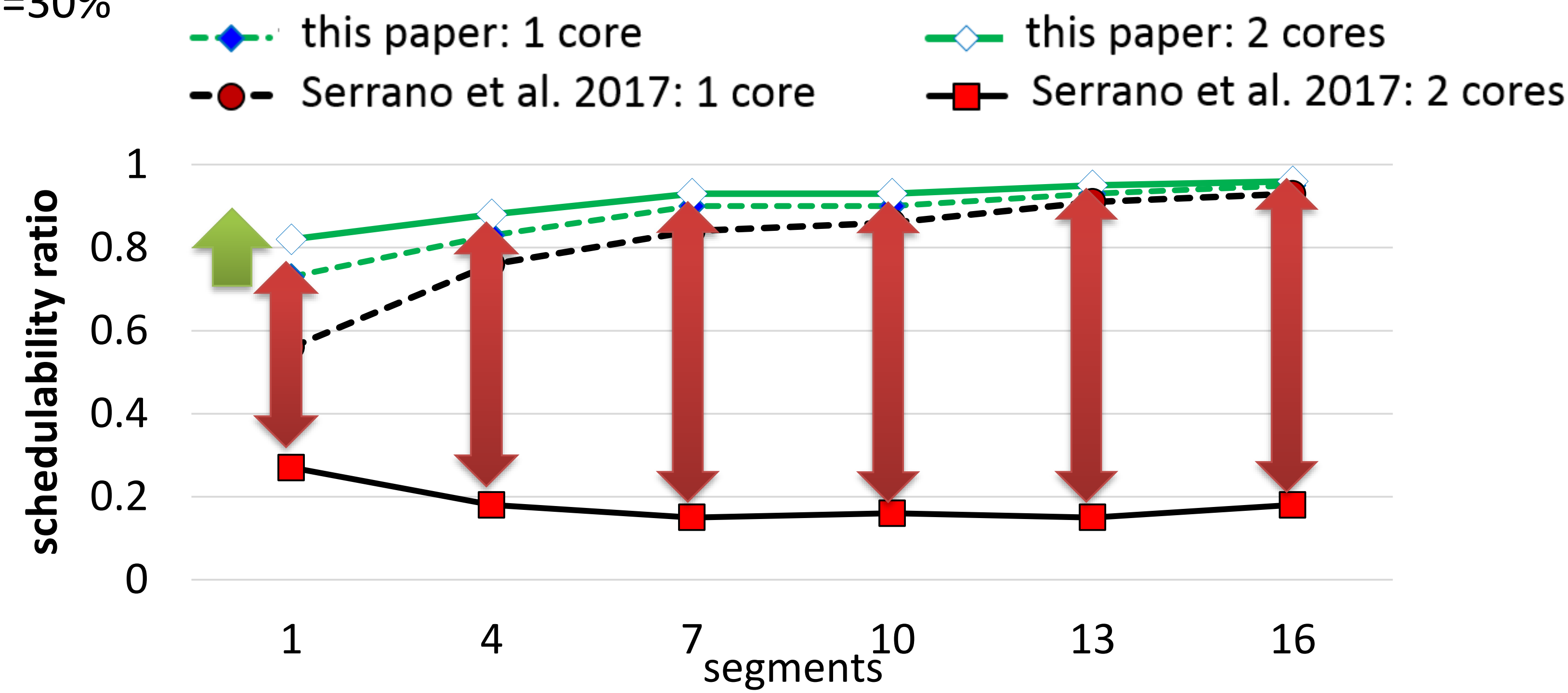
Utilization=30%
10 tasks



The **true schedulability increases** with the increase in the number of cores

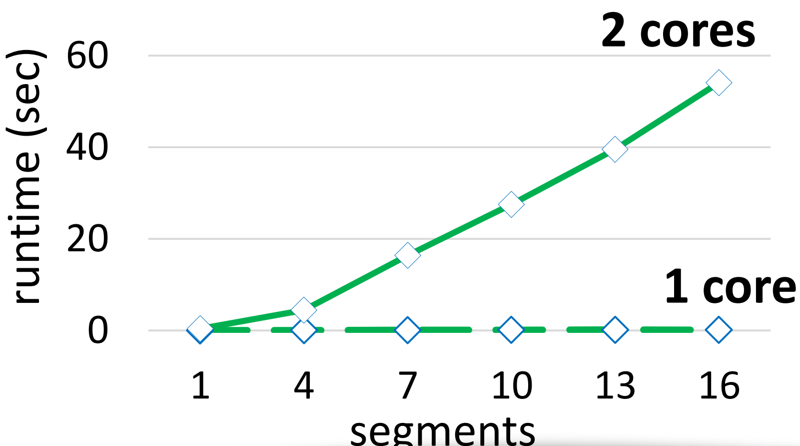
Limited-preemptive tasks

Utilization=30%
10 tasks



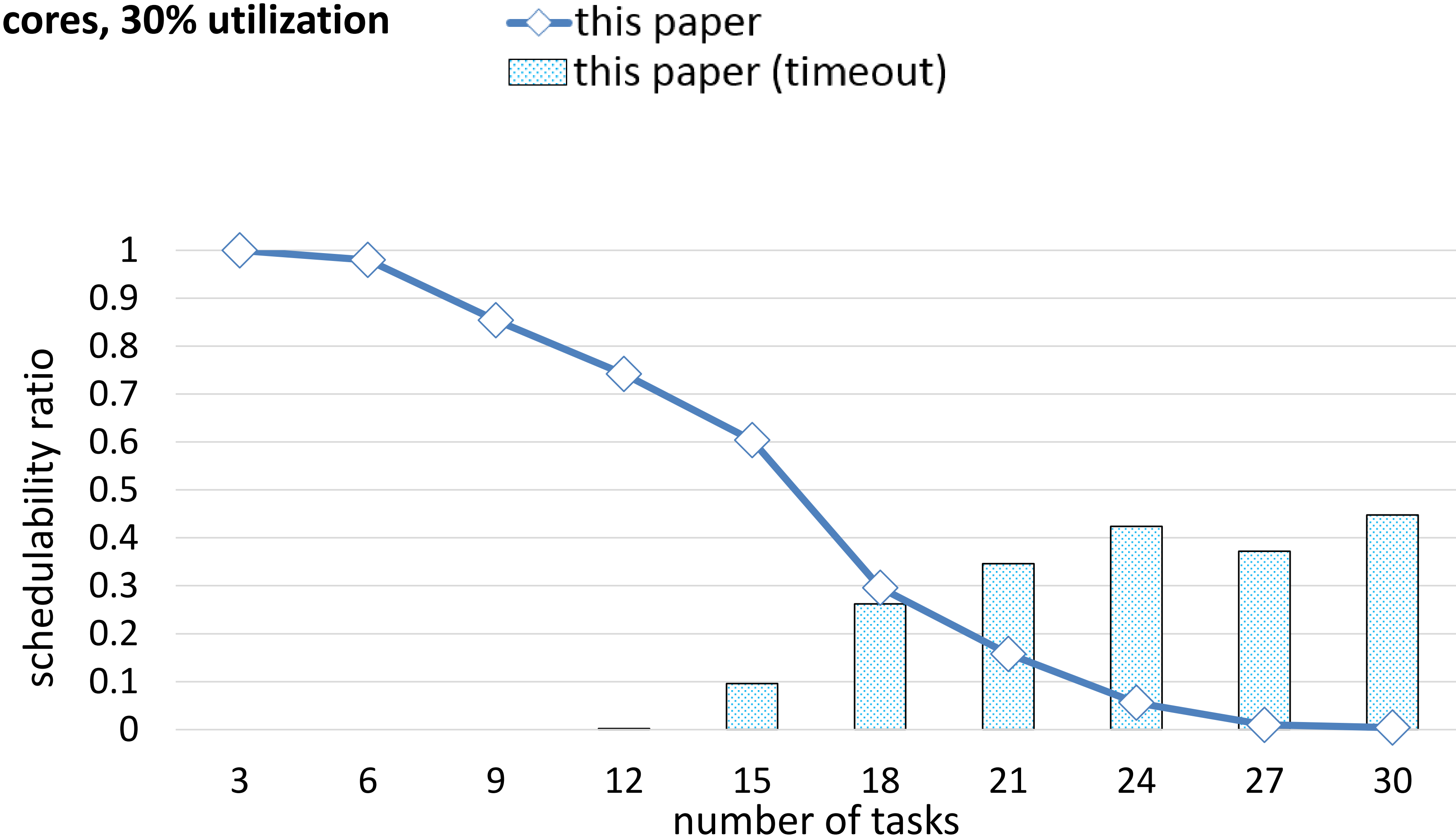
Serrano's test becomes **very pessimistic** when there are multiple cores.

The **true schedulability increases** with the increase in the number of cores, while Serrano's test shows the **opposite**!

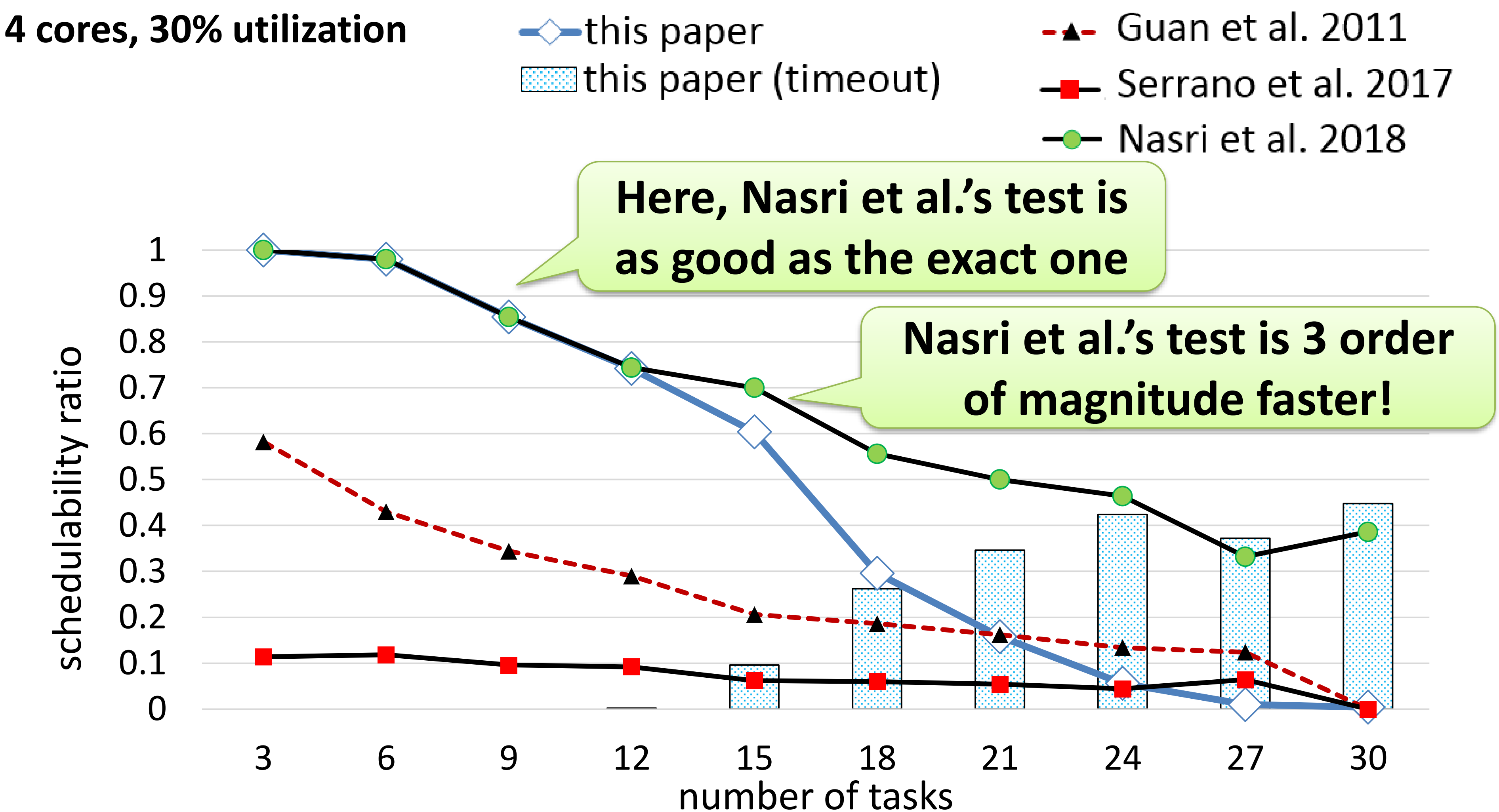


Non-preemptive scheduling

4 cores, 30% utilization



Non-preemptive scheduling



Nasri et al.'s test explores the space of possible schedules efficiently (with the help of schedule abstraction and effective path merging techniques).

M. Nasri, G. Nelissen, and B. B. Brandenburg, "A Response-Time Analysis for Non-Preemptive Job Sets under Global Scheduling," in ECRTS, 2018.

Conclusions

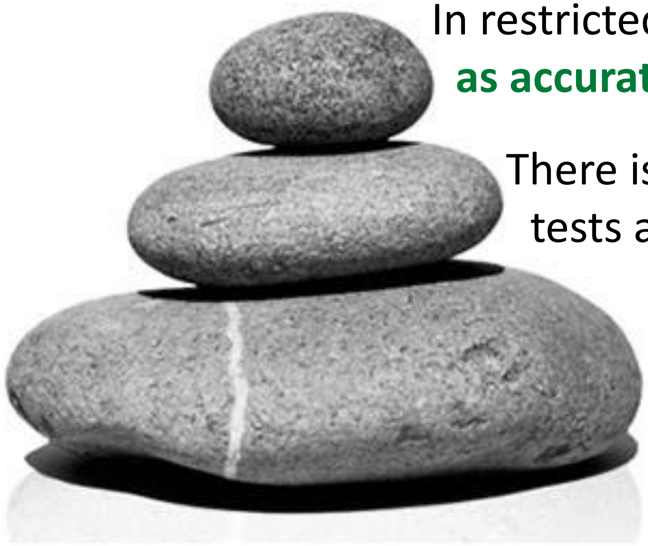
This paper:

An extensible timed automata model in UPPAAL that provides **the first exact schedulability tests for**

Global multiprocessor fixed-priority scheduling of

Uniprocessor fixed-priority scheduling of

- **non-preemptive tasks**
- **limited-preemptive tasks**
- **limited-preemptive segmented self-suspending tasks**



In restricted settings, some of the existing tests are **almost as accurate as the exact test** while being much faster

There is **a large gap** between the accuracy of various sufficient tests and the new exact baseline

Exact tests can **quantify the pessimism** of the existing sufficient (but faster) tests

Beyazit Yalcinkaya, **Mitra Nasri**, Björn Brandenburg

ODTÜ

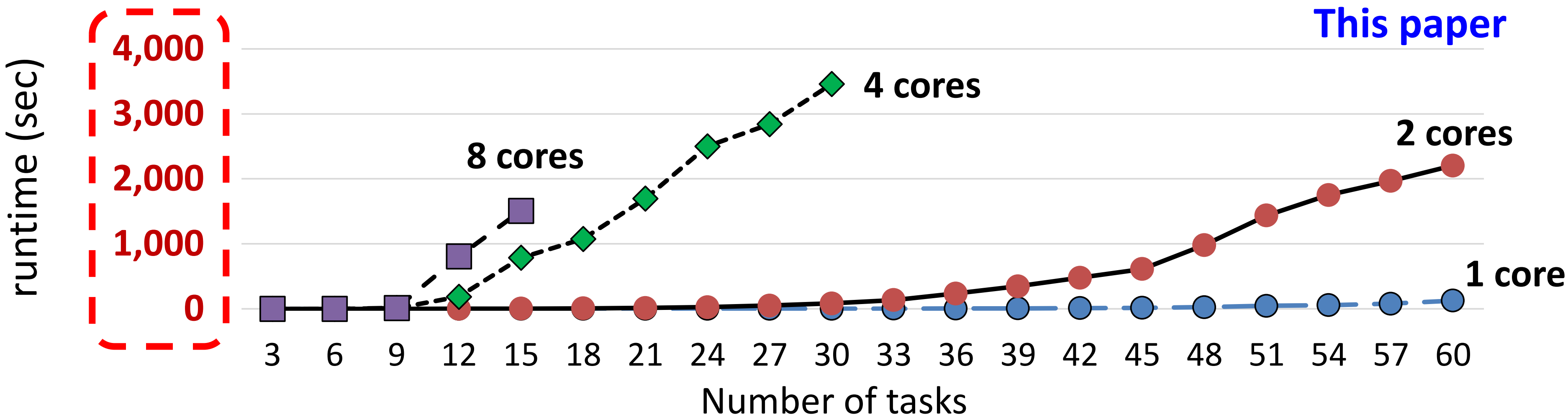
TU Delft

MAX PLANCK INSTITUTE FOR SOFTWARE SYSTEMS

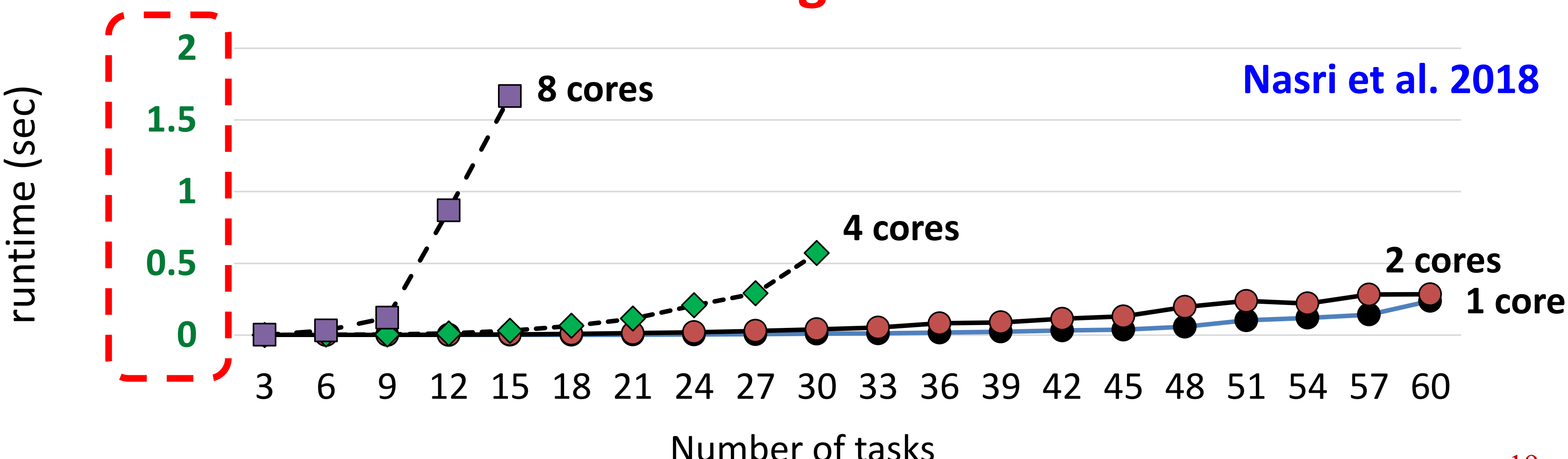
Thank you

Scalability w.r.t. the number of tasks and cores

(non-preemptive tasks)



3 orders-of-magnitude difference!



Timeout limit was set to 1 hour.