Mapping and Integration of Event- and Time-triggered Real-time Tasks on Partitioned Multi-core Systems

Carlo Meroni, Silviu S. Craciunas, Anaïs Finzi, Paul Pop

Principal scientist @ TTTech

28th INTERNATIONAL CONFERENCE ON EMERGING TECHNOLOGIES AND FACTORY AUTOMATION

ETFA 2023

SINAIA, ROMANIA

SEPTEMBER, 12th-15th, 2023

Safe networked computing platforms

TTTech

TTTech Auto safety and robustness in series production

1.5 million
cars with MotionWise on the road

Scalable distributed control system from TTTech Industrial into Vestas 2, 4 and 6 Megawatt turbines

6,000
Vestas turbines

Collins Aerospace Power System and Air Management for the Boeing 787-8, 787-9, 787-10 family

1 billion
flight hours

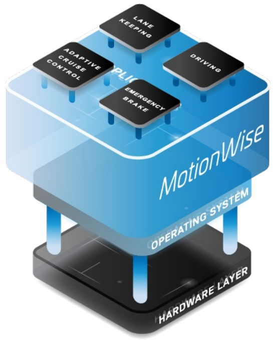
.TTEthernet is the "nervous system" i.e., avionics network platform of the space craft

>2 million
kilometers in deep space

Time-triggered systems

Time-triggered scheduling:

- Tasks are executed based on a static schedule table that is computed at design time (offline)
- The complete scheduling timeline for TT tasks is fixed.
- Simple and complex (e.g., chains) constraints are satisfied by the schedule creation.
- One solution is sufficient, and any solution is a sufficient schedulability test.
- Idle time in the static schedule can be used more dynamically, e.g., for ET tasks.



Time-triggered has many advantages (c.f. [Xu2000] [Locke1992]):

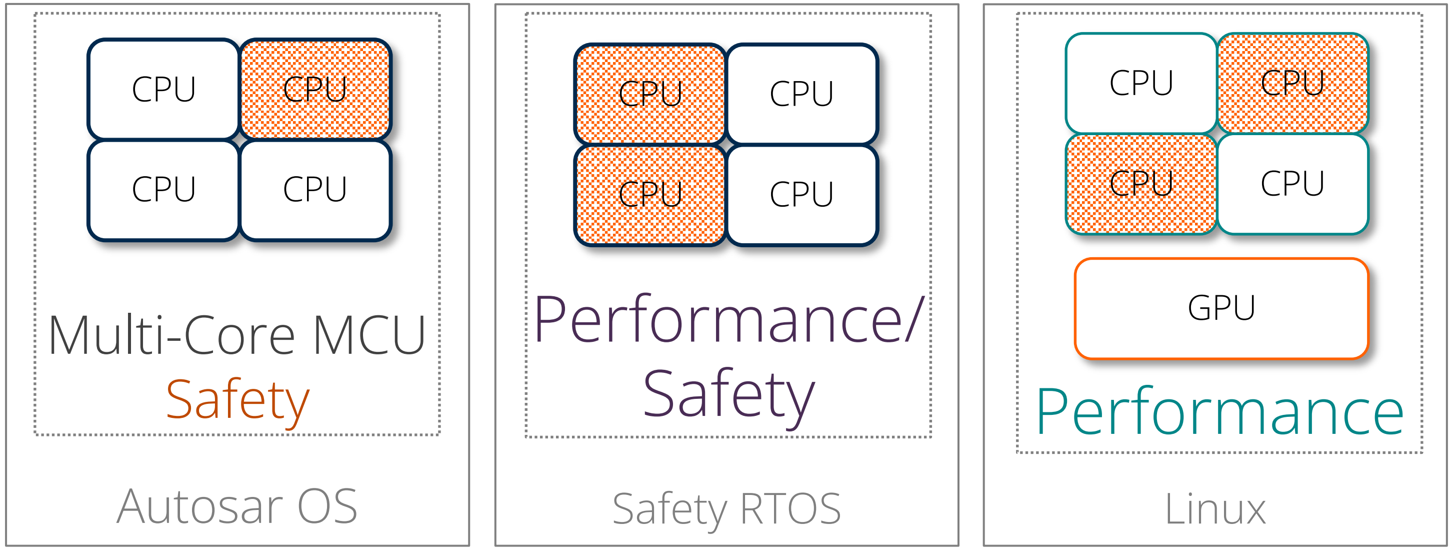
- Complex timing requirements: cause-effect chains, different types of jitter
- Temporal isolation: no starvation possible, temporal isolation between all tasks
- Determinism: increased stability and testability, fewer system states
- Compositionality: adding or modifying tasks can be done incrementally
- Predictability: many system properties become predictable, e.g., locks, task pre-emption
- Schedulability: more correct configurations can be realized

One downside is lack of flexibility wrt. to event-driven execution.

Modern safety-critical systems (e.g., ADAS)

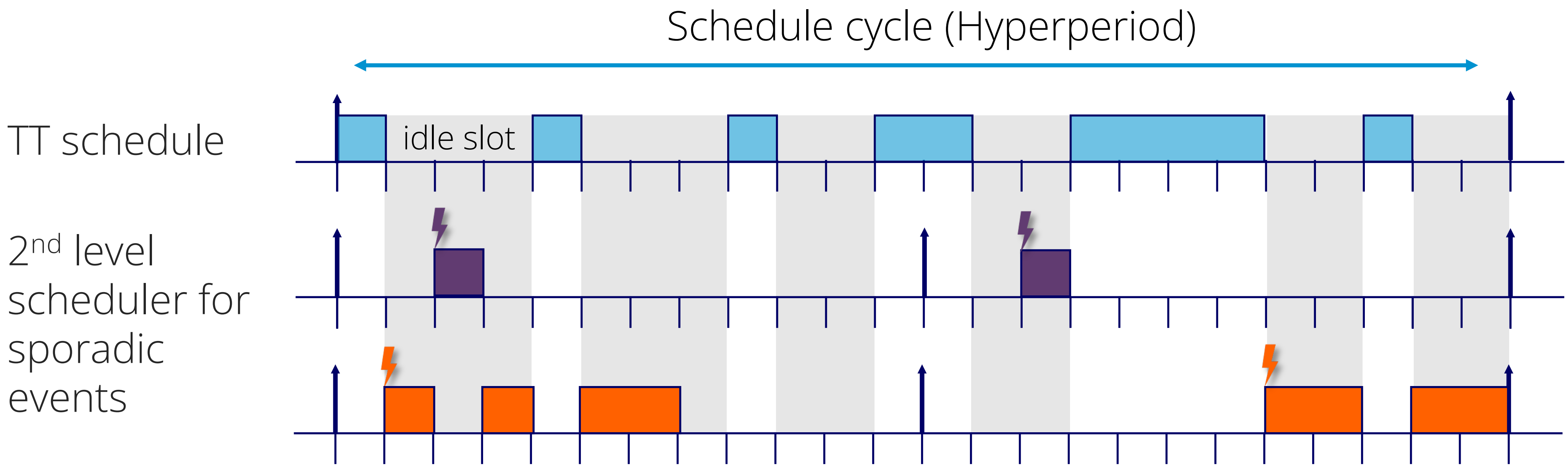
TTTech

- **Heterogeneous** multi-core multi-SoC platforms featuring a variety of CPUs and GPUs
- Time-Triggered (**TT**) tasks are **periodic** and **statically scheduled**
- Event-triggered (**ET**) are **sporadic** but also require **timing guarantees**
- **Partitioned** scheduling approach without runtime migration of tasks



Modern safety-critical systems benefit most from combining TT and ET, allowing a system to be flexible enough to respond to sporadic events when needed, i.e., **best of both worlds**

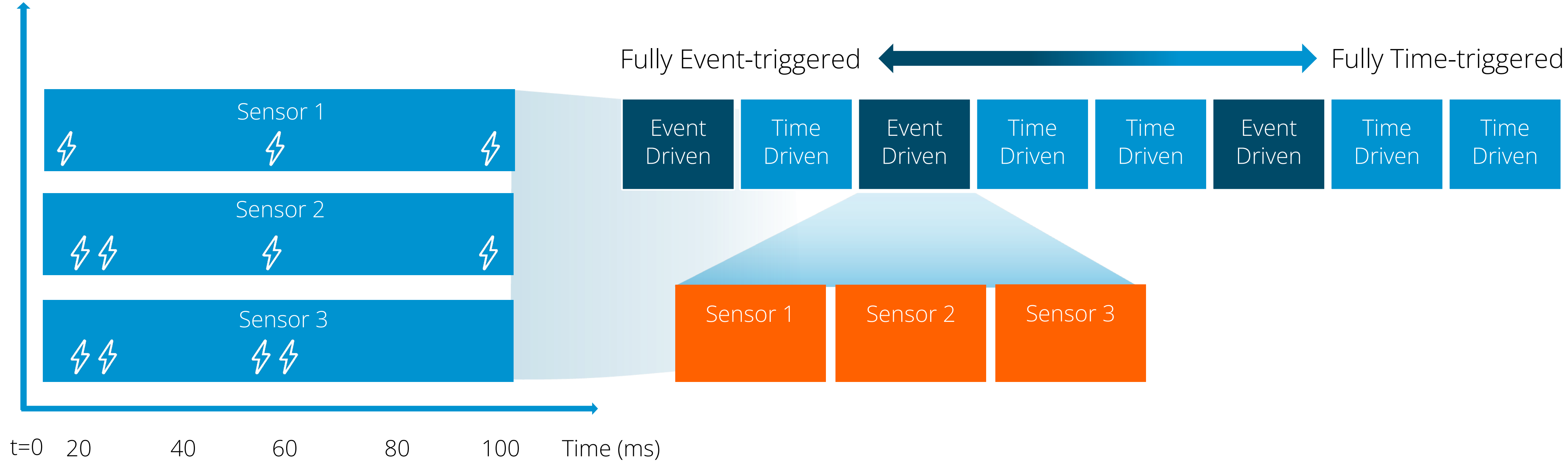
TT and ET integration



- Time-triggered schedule ensures timing behavior of TT tasks and is calculated offline
- Second-level fixed-priority scheduler that services event-triggered tasks in the idle slots at runtime

Problem 1: How can we ensure that both TT and ET tasks respect their deadlines?

TT and ET task allocation and interaction



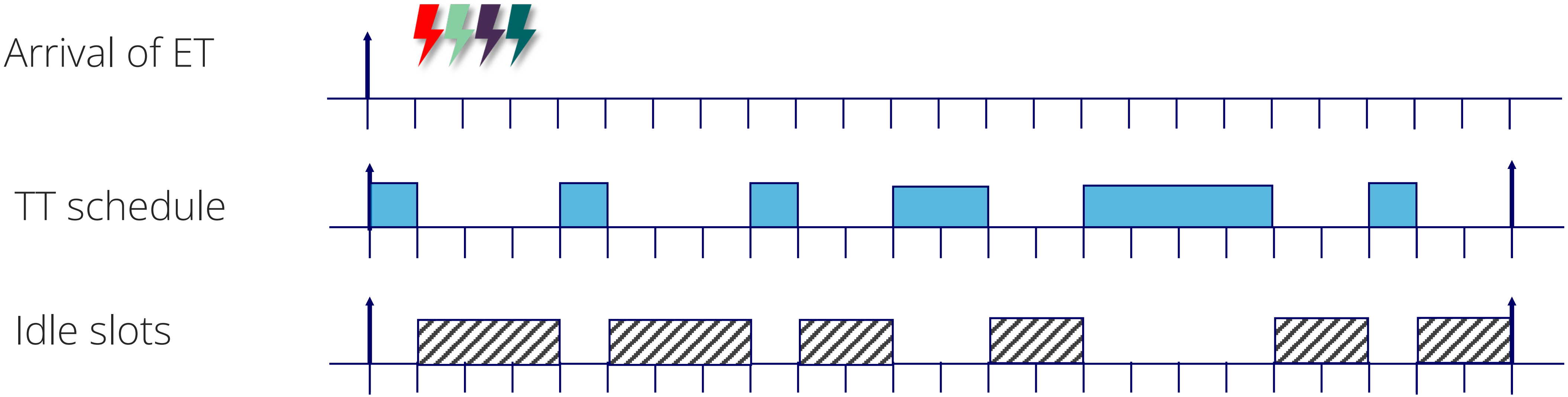
- The system is partitioned, i.e., TT and ET tasks need to be allocated to cores at design-time
- The TT and ET task allocation problem is crucial to increase schedulability of the system
- The problem is NP-complete, i.e., optimal algorithms are infeasible in the general case

Problem 2: Find a task to core allocation such that TT+ET tasks are schedulable (optimization).

How can we check ET task schedulability?

Simulate Fixed-Priority and see if deadlines are met

- We need to do this for every potential correct TT schedule candidate (there are a lot of them)
- ...and for every priority ordering (remember the combinatorial explosion of n!)
- ...and for every arrival pattern of events (also leads to combinatorial explosion)



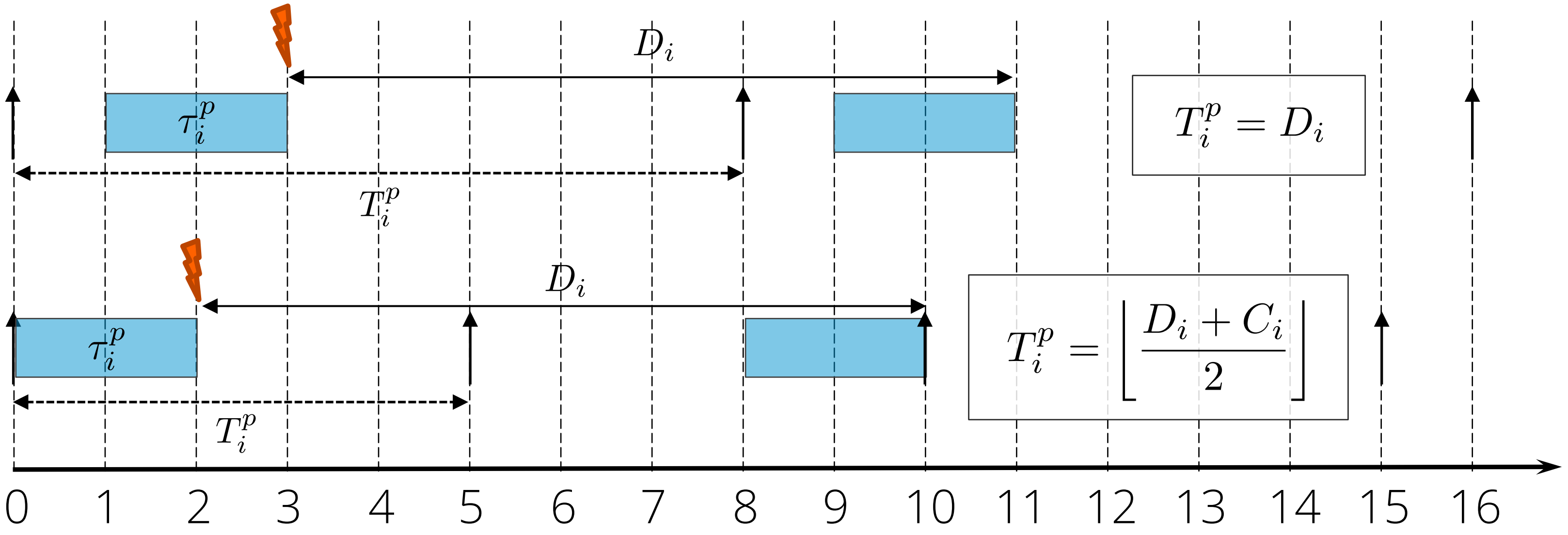
Highly infeasible!

Oversampling – SPoll

Simple polling: Use one periodic polling task per event – oversampling
How long is the sampling period?

a) Strictly periodic

b) Non-strictly-periodic



⚠ Can be quite inefficient!

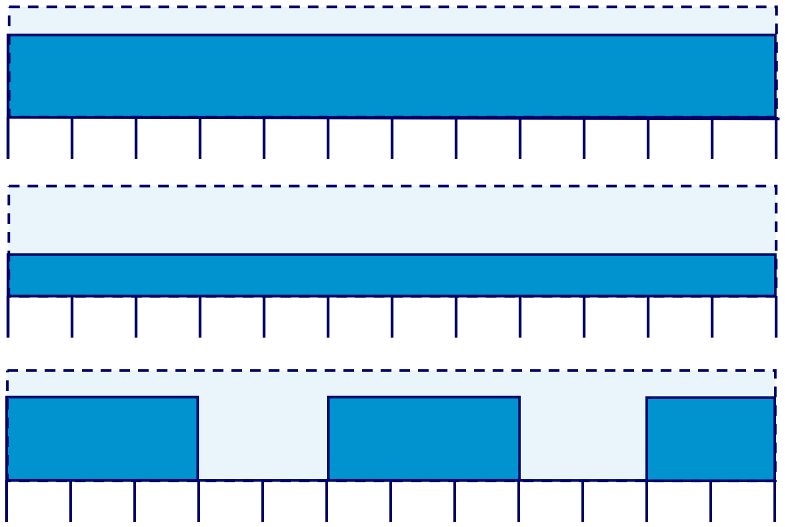
Think of an event with C = 2 ms, D = 20 ms, and T = 100ms. (2% utilization)
We have to reserve a slot every 9 ms, consuming 22,2 % of CPU bandwidth.

Response-time analysis on a partitioned resource

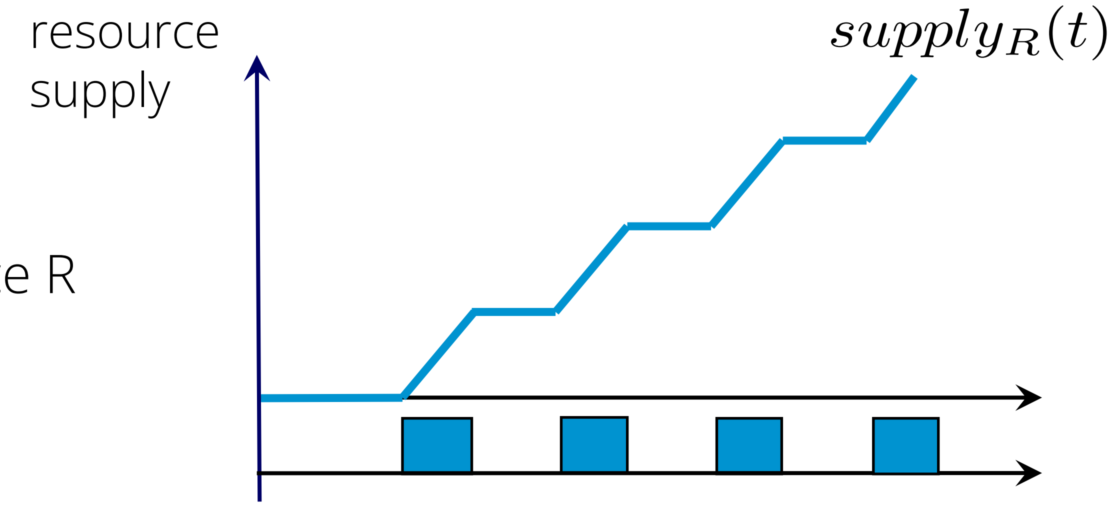
Dedicated resource: available all the time at full capacity

Fractional resource: available all the time at reduced capacity

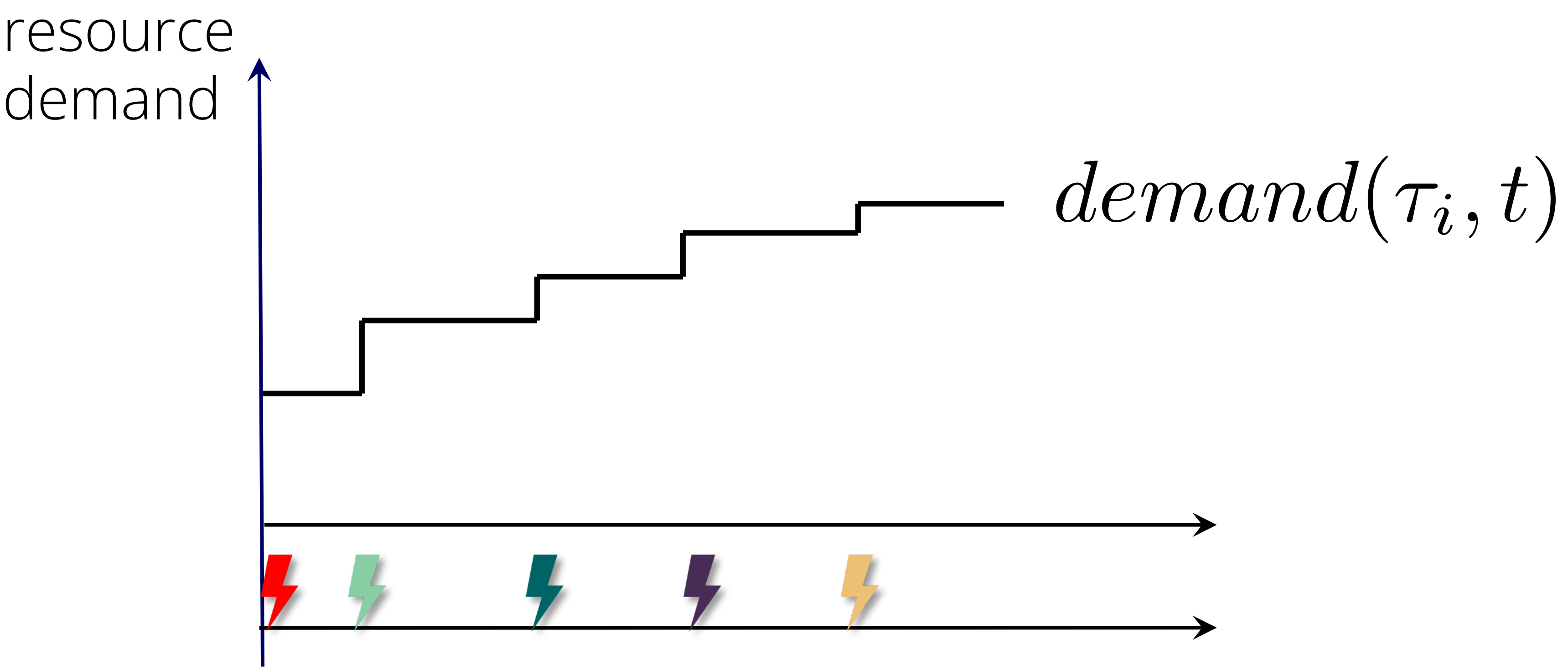
Partitioned resource: available some of the time at full capacity



Available supply over time of a partitioned resource R

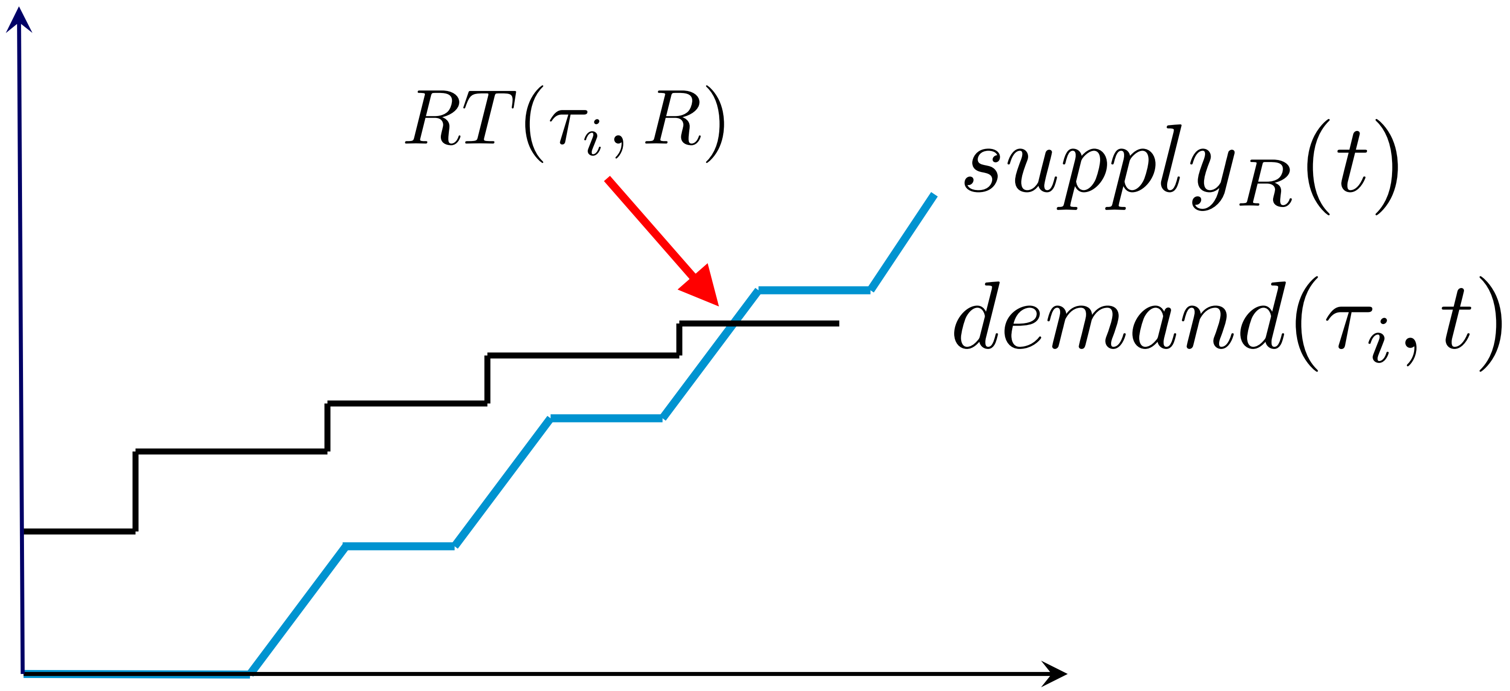


Response-time analysis on a partitioned resource



$$demand(\tau_i, t) = C_i + \sum_{\forall \tau_j \in HP(\tau_i)} \left\lceil \frac{t}{T_j} \right\rceil \cdot C_j.$$

Response-time analysis on a partitioned resource



$$RT(\tau_i, R) = earliest\ t : supply_R(t) \geq demand(\tau_i, t)$$

Solution using response-time analysis

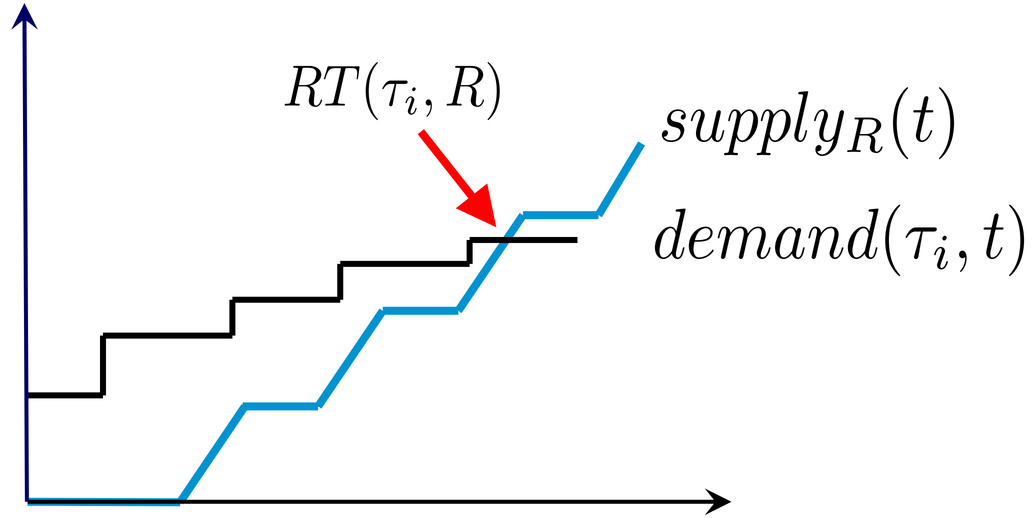
Holistic scheduling [Pop2003]

1. Generate TT schedule such that TT tasks are fulfilled.
2. Check schedulability over resulting idle slots using the response-time method over partitioned resources for every event-based task.
3. Recompute TT schedule if ET tasks not feasible

Slot-shifting [Isovic2009] is a similar method with a different schedulability test that does not assume FP scheduling.

Can be very time-consuming!
No guidance on how to place idle slots.

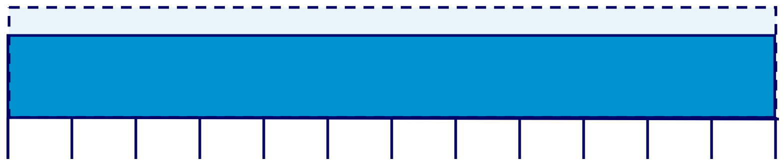
? Can we do better?



Periodic resource abstraction

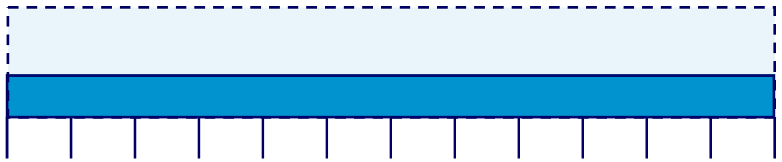
Dedicated resource:

- available all the time at full capacity



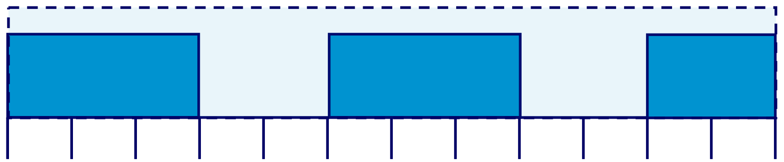
Fractional resource:

- available all the time at reduced capacity



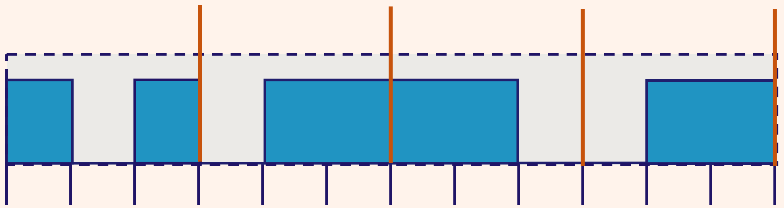
Partitioned resource:

- available some of the time at full capacity



Periodic resource:

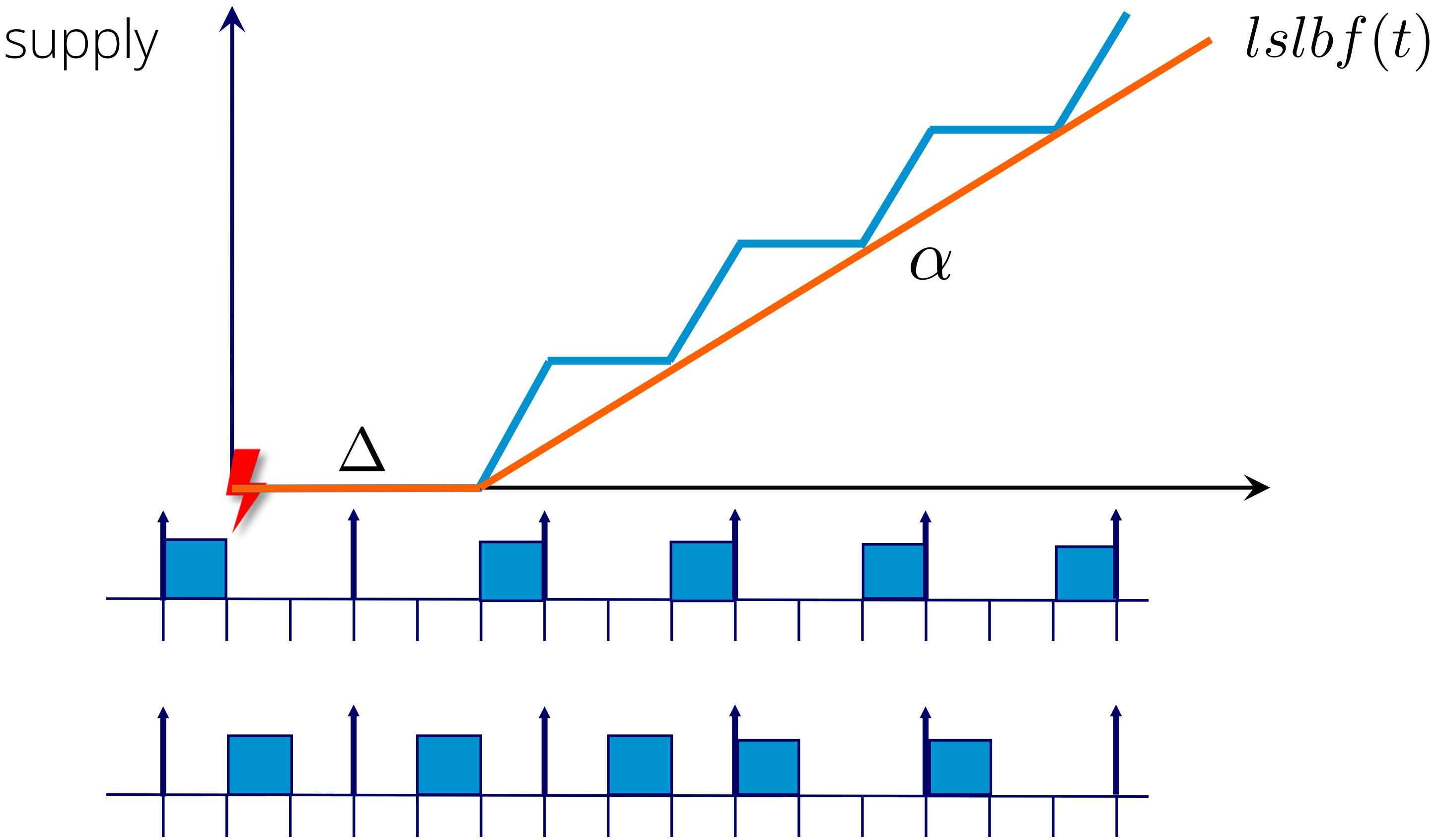
- available periodically at full capacity (e.g., R(2,3,3))



Periodic resource abstraction - **AdvPoll**

Explicit Deadline Periodic: R(C,D,T) [Shin2008]

- we do not care when the supply C is given, as long as it is given in every period T until the deadline D
- we can use the worst-case linear supply bound and schedulability test defined below [Almeida2004]



$$\alpha = \frac{C}{T}$$

$$\Delta = T + D - 2C$$

$$lslbf(t) = max\{0, (t - \Delta) \cdot \alpha\}$$

$$RT(\tau_i, R) = earliest\ t :$$
$$t \geq \Delta + demand(\tau_i, t)/\alpha$$

$$RT(\tau_i, R) = earliest\ t : t \geq \Delta +$$
$$\frac{1}{\alpha}\left(C_i + \sum_{\forall \tau_j \in HP(\tau_i)} \left\lceil \frac{t}{T_j} \right\rceil \cdot C_j\right)$$

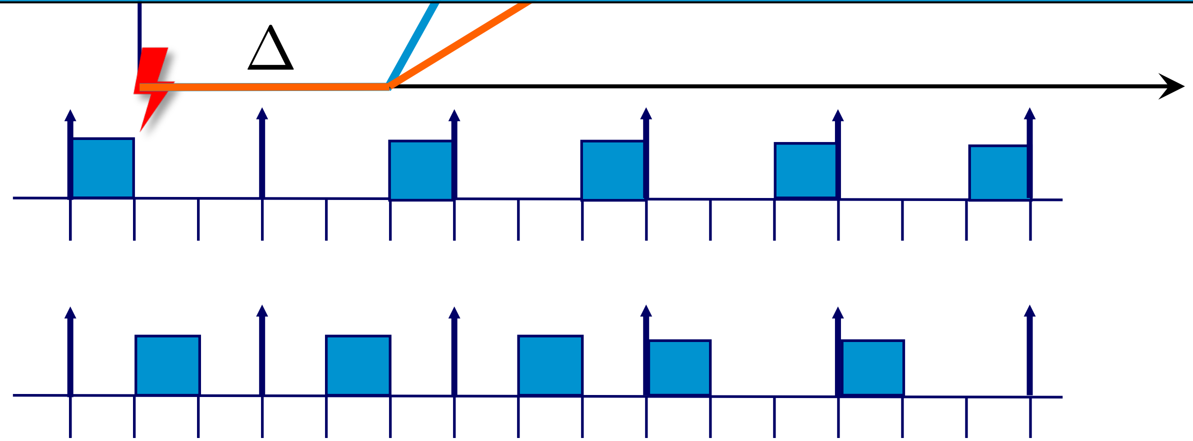
Periodic resource abstraction - **AdvPoll**

Explicit Deadline Periodic: R(C,D,T) [Shin2008]

- we do not care when the supply C is given, as long as it is given in every period T until the deadline D
- we can use the worst-case linear supply bound and schedulability test defined below [Almeida2004]

- Schedulability analysis for ET is independent of the TT schedule
- The polling tasks (periodic resources) can be scheduled as normal TT tasks

Tradeoff – schedulabiliy for runtime



$$\alpha = \frac{C}{T}$$

$$\Delta = T + D - 2C$$

$$lslbf(t) = max\{0, (t - \Delta) \cdot \alpha\}$$

$$RT(\tau_i, R) = earliest\ t : t \geq \Delta + demand(\tau_i, t)/\alpha$$

$$RT(\tau_i, R) = earliest\ t : t \geq \Delta + \frac{1}{\alpha}\left(C_i + \sum_{\forall \tau_j \in HP(\tau_i)} \left\lceil \frac{t}{T_j} \right\rceil \cdot C_j\right)$$

September 14, 2023

Task allocation problem

NP-complete and heavily influences TT and ET schedulability and optimality of e.g., response times

Our solution:

- **Genetic algorithm** with k-tournament selection, good mix between exploration and exploitation, good scalability, and a suitable degree of parallelization [Goldberg1991]
- Algorithm begins with an initial task-to-core mapping, we introduce 4 different methods: **Random, Laxity round-robin, Load balancing, Delay minimizing**
- Chromosomes include **task allocation & parameters for AdvPoll** (budget, period, deadline)
- Each solution is evaluated based on a fitness function that tries to minimize response times and maximize schedulability

$$f_1(x) = \frac{\omega^{TT}}{|\mathcal{T}^{TT}|} \times \sum_{i=1}^{|\mathcal{T}^{TT}|} \frac{R_i^{TT}(x)}{D_i^{TT}(x)} + \frac{\omega^{ET}}{|\mathcal{T}^{ET}|} \times \sum_{i=1}^{|\mathcal{T}^{ET}|} \frac{R_i^{ET}(x)}{D_i^{ET}(x)} \qquad f_2(x) = \lambda^{TT} \times \sum_{j=1}^{K} u_j(x) + \lambda^{ET} \times \sum_{k=1}^{K} v_k(x) \qquad f(x) = -\left[f_1(x) + f_2(x)\right]$$

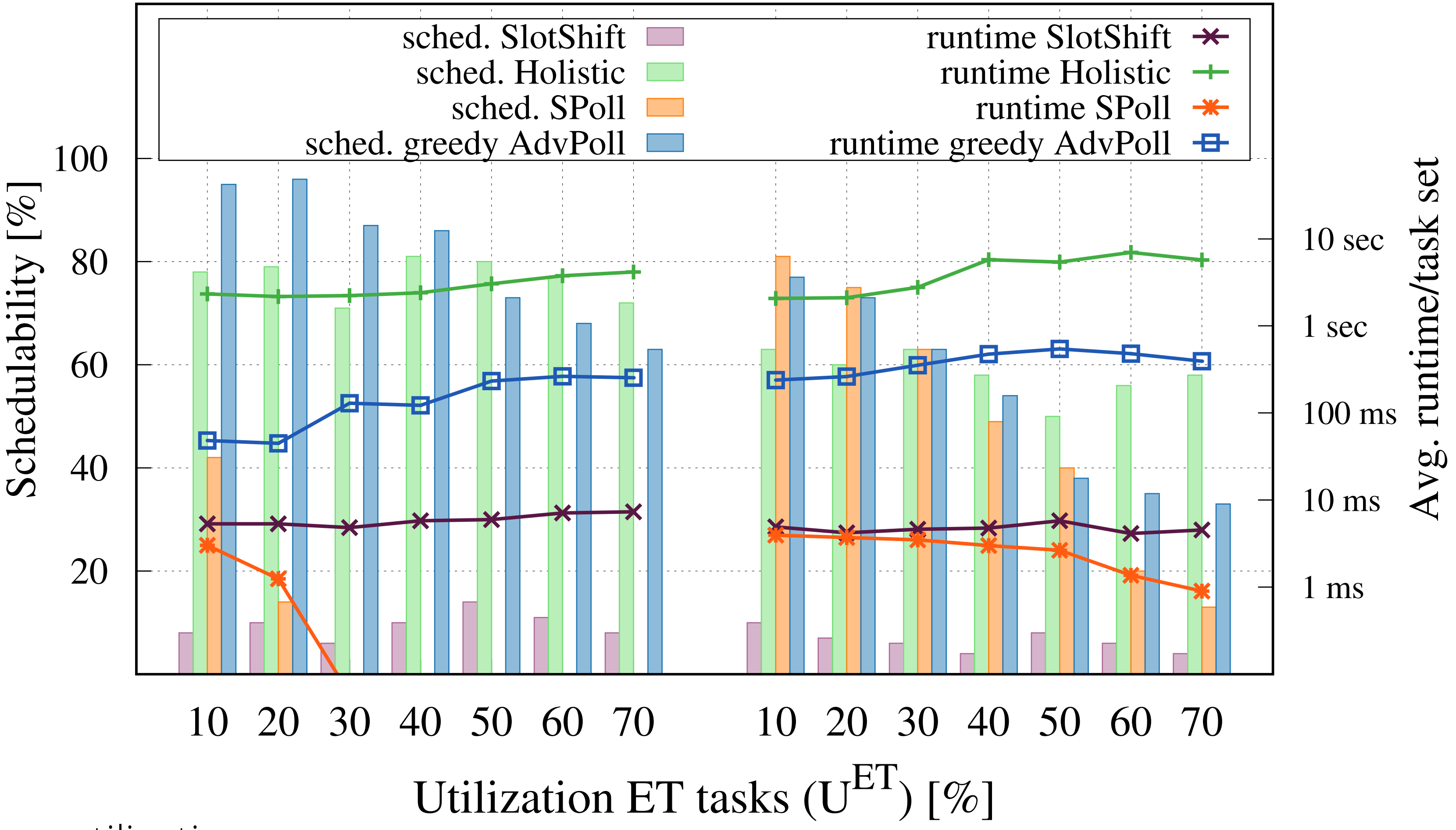
- **Mutation step** to avoid local optima and encourage search space exploration
- Terminate when a solution meets a **fitness threshold** or after predefined **iteration count**

Single-core experiments

- 30 TT and 20 ET tasks per task set
- Periods ∊ {5, 10, 20, 40, 80} ms
- Microtick of 250μs

- Constrained Deadline: Di is uniformly selected in upper half of [Ci,Ti]
- Arbitrary Deadline: Di ∊ [Ci,5 · Ti]

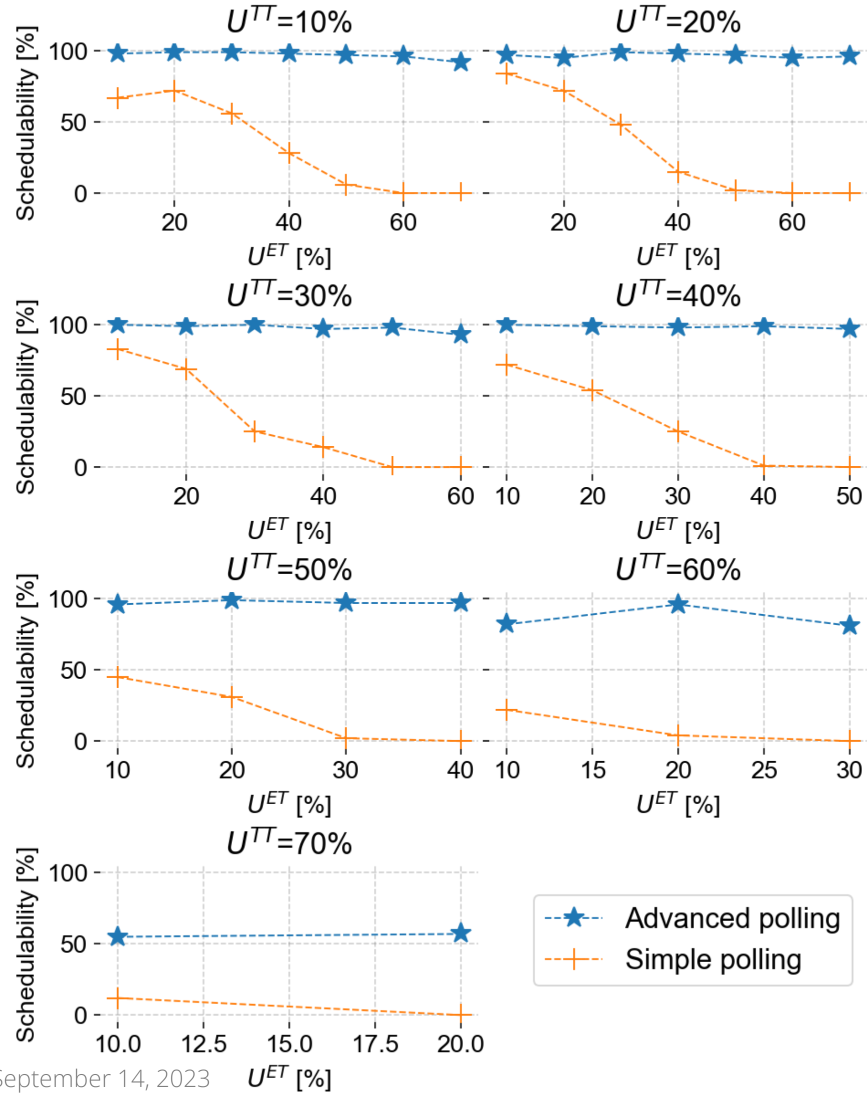
- 20% TT task utilization
- Increasing ET utilization
- 100 task sets per configuration



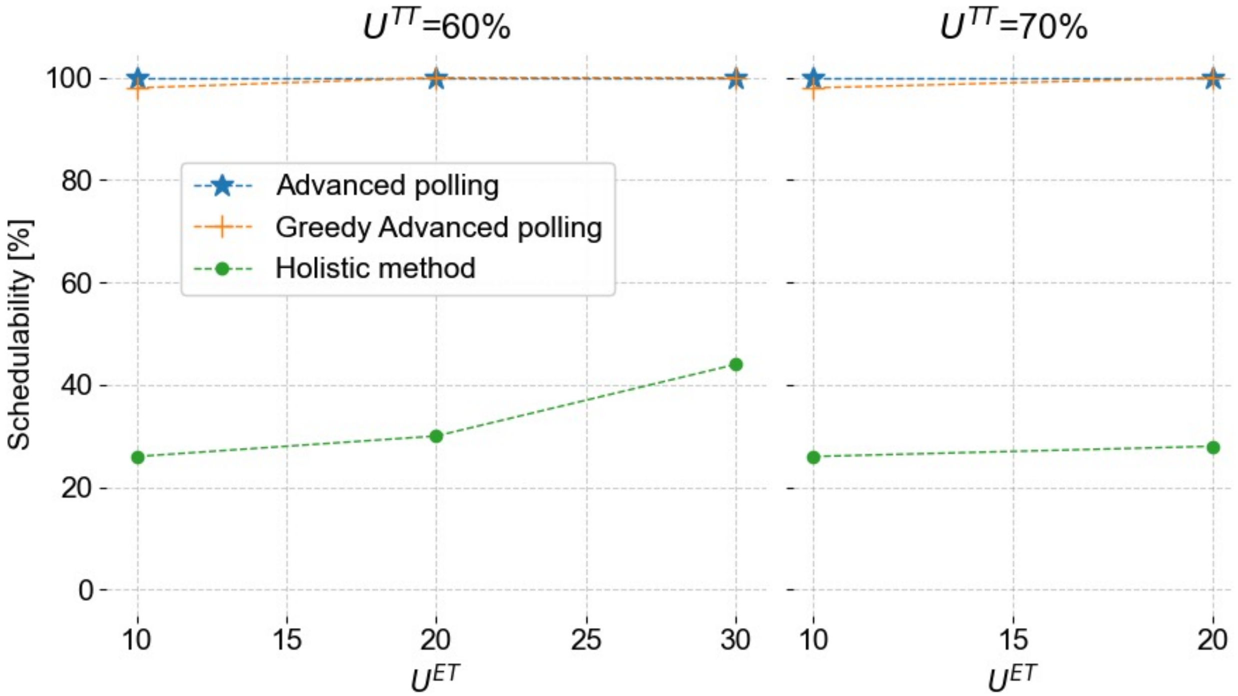
SPoll is quickest and can be used for low utilization
Holistic has better schedulability at high utilization but takes much longer time
AdvPoll has the best overall trade-off between runtime and schedulability

Task to core allocation experiments

Schedulability after 2500 iterations of optimization on different task sets with 8 cores

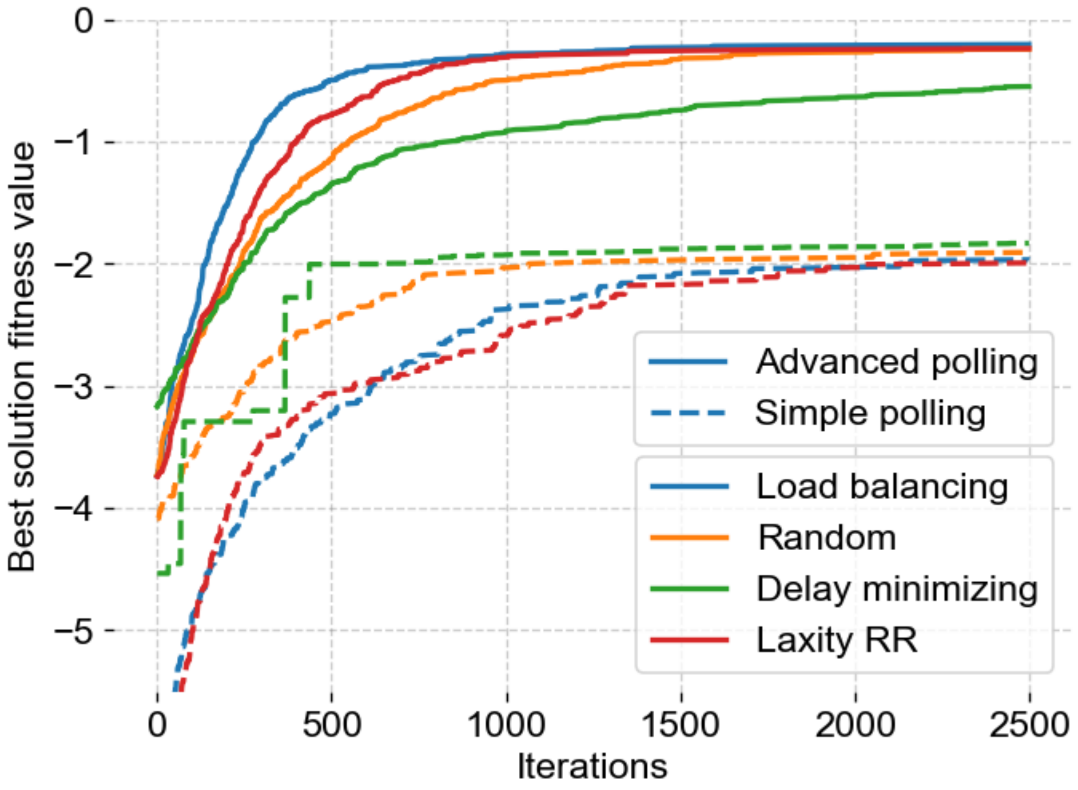


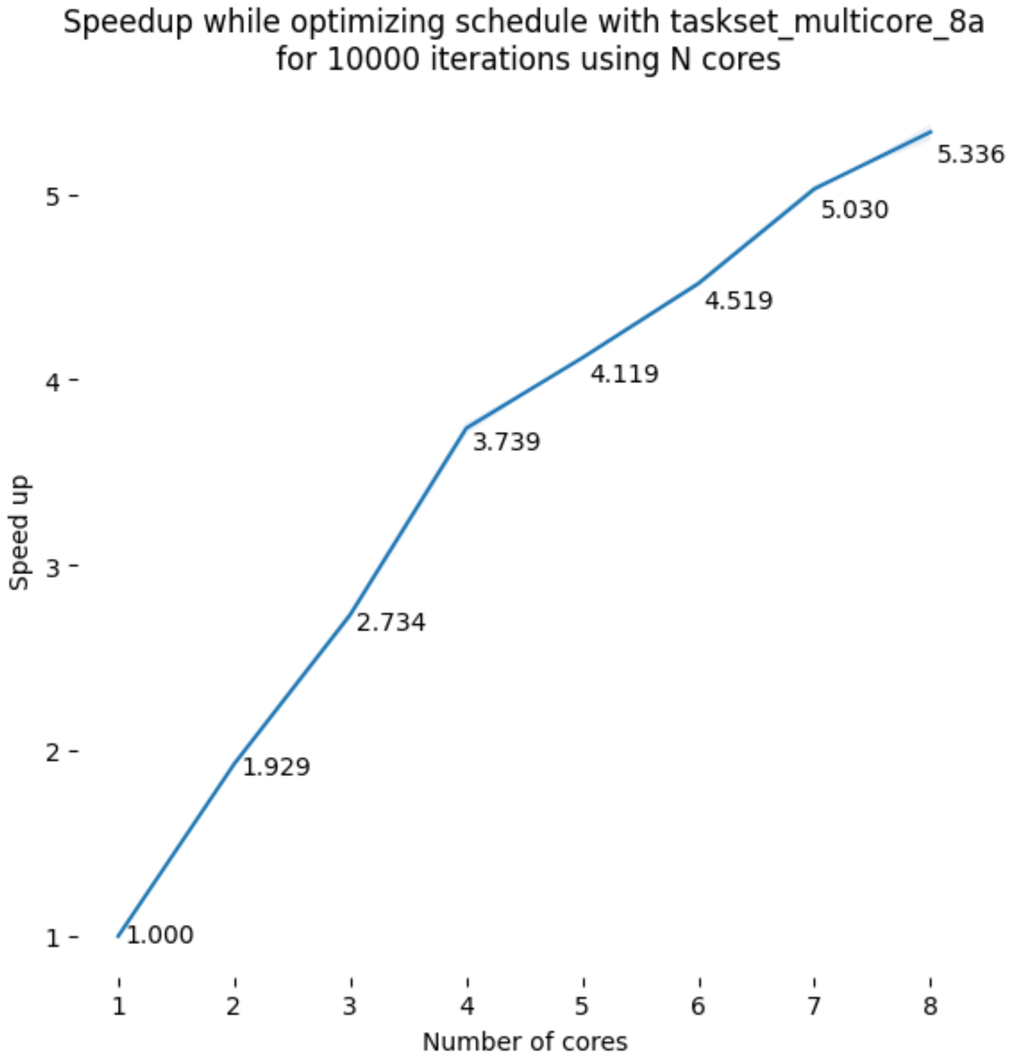
Comparison of greedy AdvPoll vs. optimized AdvPoll vs. Holistic scheduling.



Initial mapping and scalability experiments

Fitness value while optimizing solutions using different mapping initializations (higher is better).





Thank you!

[Xu2000] Xu, Parnas - Priority Scheduling Versus Pre-Run-Time Scheduling. Real-Time Syst, 2000
[Locke1992] C. D. Locke - Software architecture for hard real-time applications: Cyclic executives vs. fixed priority. Real-Time Syst., 1992
[Pop2003] T. Pop, P. Eles, Z. Peng, Schedulability analysis for distributed heterogeneous time/ event triggered real-time systems, ECRTS 2003.
[Isovic2009] D. Isovic, G. Fohler, Handling mixed sets of tasks in combined offline and online scheduled real-time systems, Real-Time Syst, 2009.
[Shin2008] I. Shin et al.- Hierarchical Scheduling Framework for Virtual Clustering of Multiprocessors. ECRTS, 2008
[Almeida2004] L. Almeida, P. Pedreiras, Scheduling within temporal partitions: Response-time analysis and server design. EMSOFT, 2004.

We are hiring

https://www.tttech.com/jobs-career/