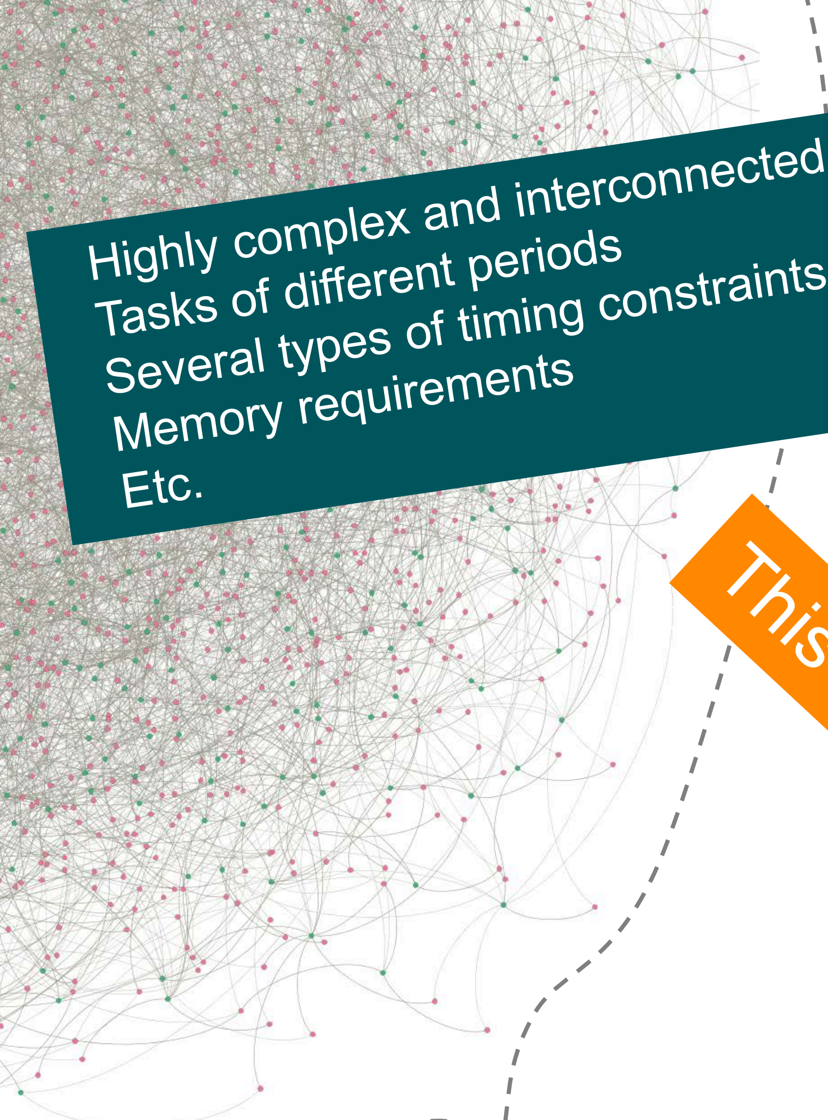
Scheduling Multi-Rate Real-Time Applications on Clustered Many-Core Architectures with Memory Constraints

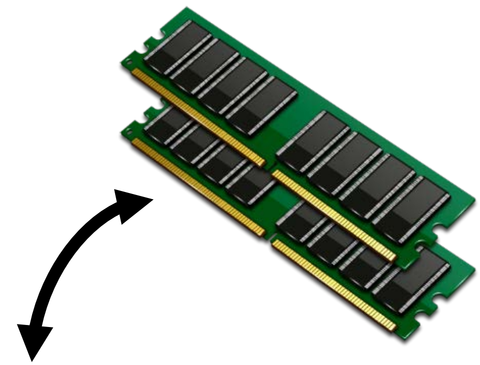
Matthias Becker, Saad Mubeen, Dakshina Dasari, Moris Behnam, Thomas Nolte

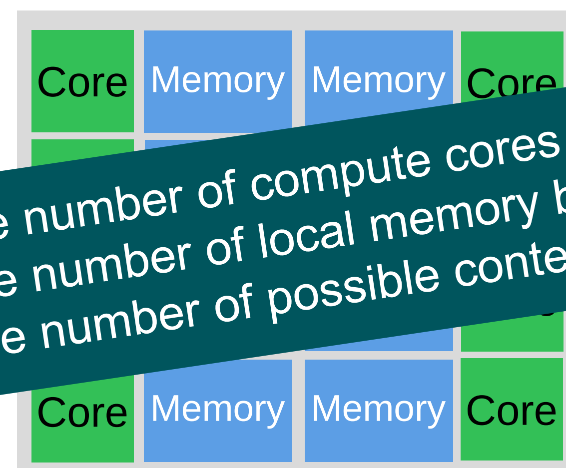
ASP-DAC, Jeju Island, Korea
24. January 2018



Highly complex and interconnected
Tasks of different periods
Several types of timing constraints
Memory requirements
Etc.

This Paper





Large number of compute cores
Large number of local memory banks
Large number of possible contention

Outline

- Background Information
- Contention Free Execution Framework
- Observations and Drawback
- Memory Aware Contention Free Execution Framework
- Evaluation
- Conclusions and Future Work

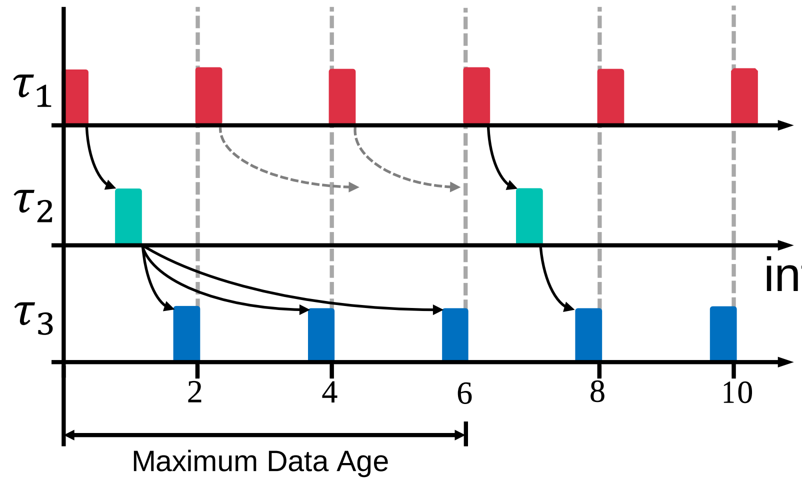
Application Model
Task Set Γ

- Task set Γ, non-preemptive periodic dependent tasks
- Each task has an implicit deadlines
- Tasks communicate over shared variables (register communication)
- Tasks operate based on read-execute-write semantics

- A task is described by
  - Execution times: C_i^{Read}, $C_i^{Execute}$, C_i^{Write}
  - Period: T_i
  - Size in memory: S_i

Application Model
Set of Job-Level Dependencies

- Applications are subject to data propagation constraints
  - Additional timing constraints on **propagation of data** through a chain of independent tasks
  - JLD are a way to **transform** timing constraints on data propagation into **precedence constraints** of tasks jobs [1,2]

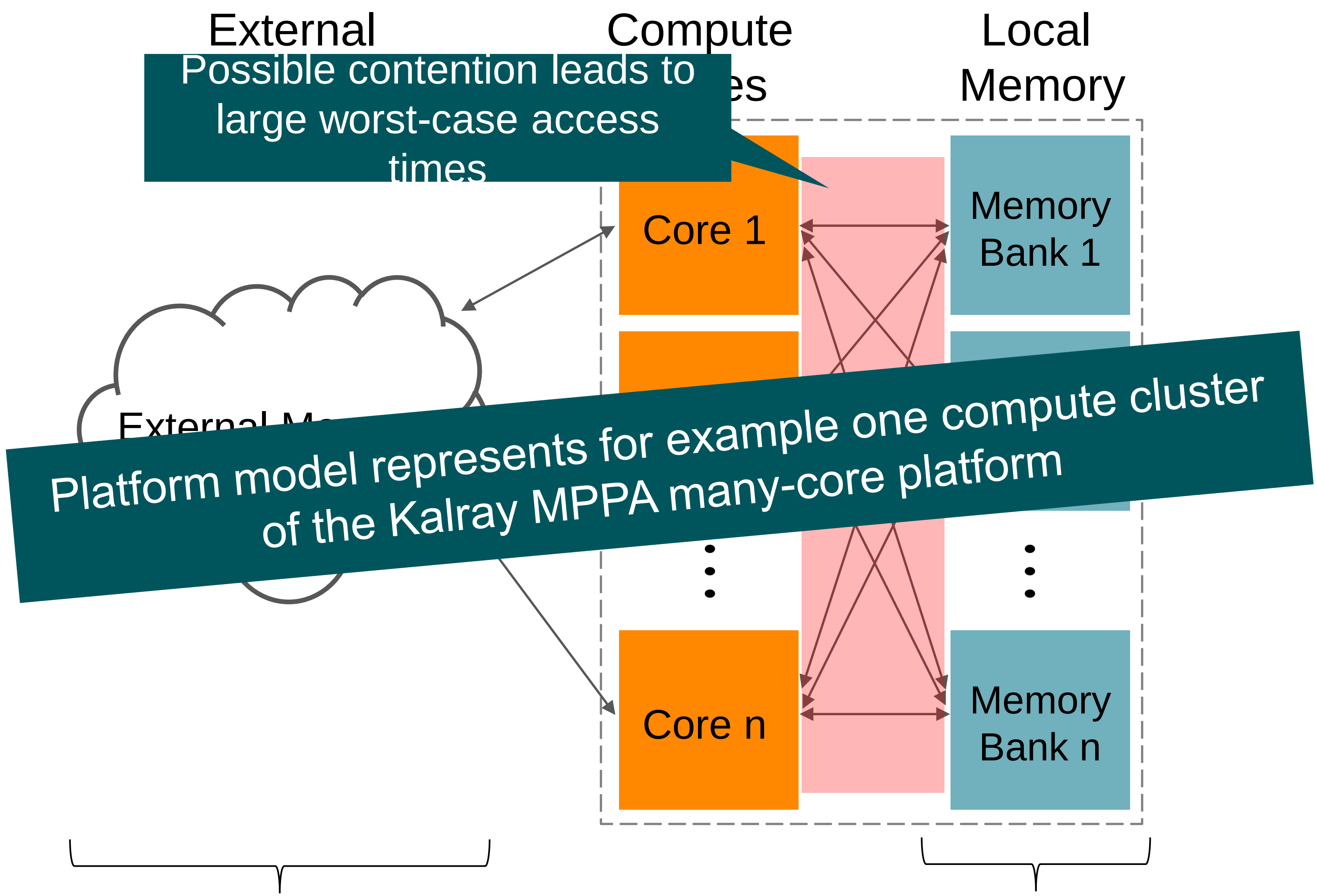


- Set of job-level dependencies Θ
  - $\tau_i \xrightarrow{(k,l)} \tau_j$ meaning: Every k^th job of τ_i must precede every l^th job of τ_j

[1] M. Becker, D. Dasari, S. Mubeen, M. Behnam, and T. Nolte, "Synthesizing job-level dependencies for automotive multi-rate effect chains," in *Proceedings of the 22nd IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2016, pp. 159–169.

[2] M. Becker, S. Mubeen, D. Dasari, M. Behnam, and T. Nolte, "A generic framework facilitating early analysis of data propagation delays in multi-rate systems (invited paper)," in *Proceedings of the 23th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2017, pp. 1–11.

Platform Model



Slow access times

To small for complete application footprint

Predictable Execution of Real-Time Applications on Clustered Many-Core Platforms

- Contention-Free Execution Framework [2]

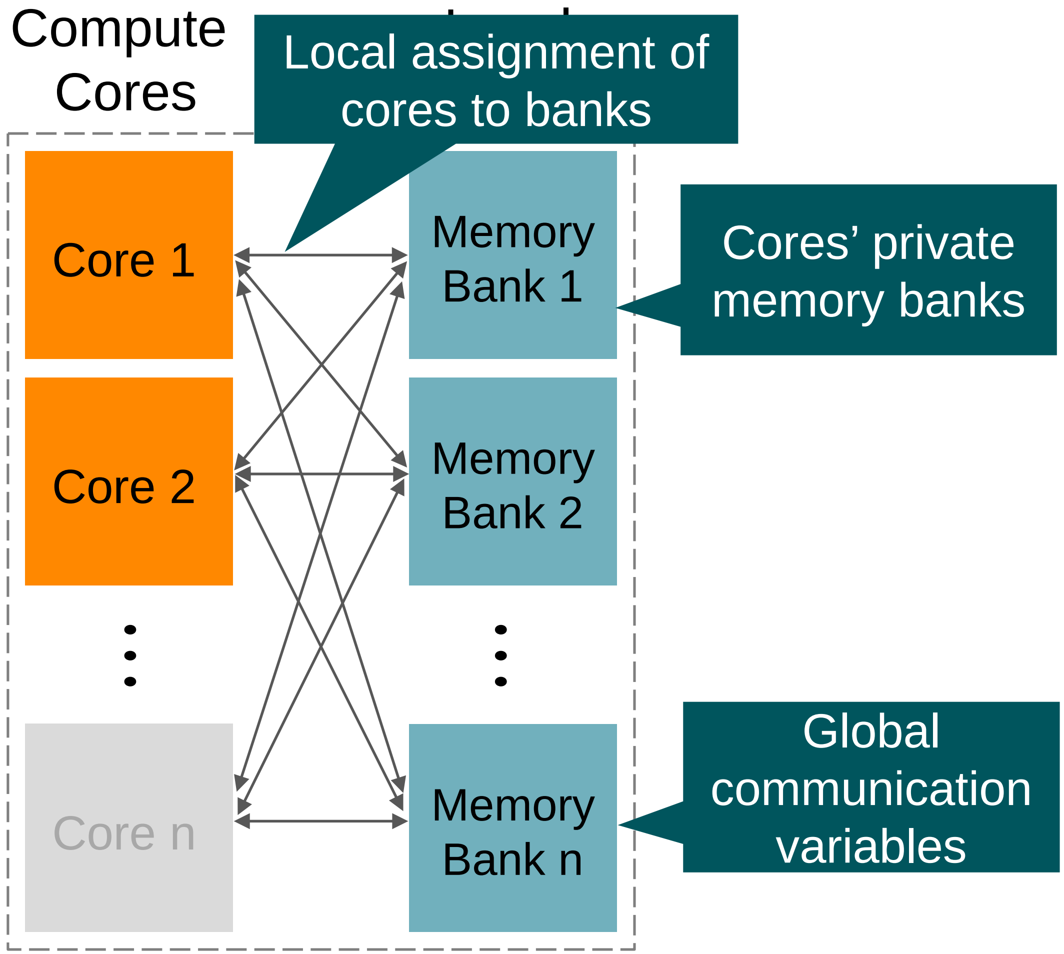
Memory Bank Privatization

Read-Execute-Write Semantic

Time-Triggered Scheduling

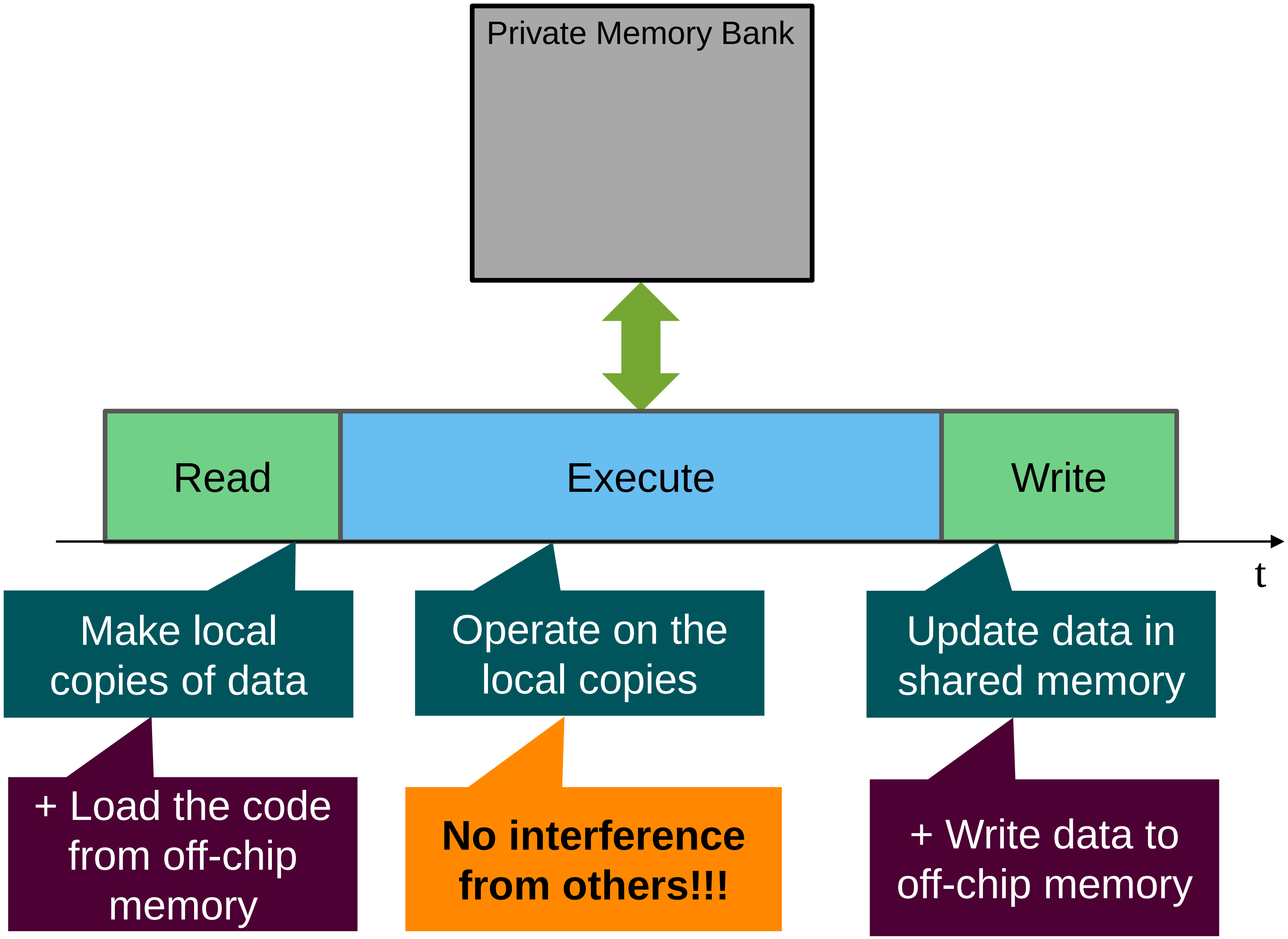
[2] M. Becker, D. Dasari,B. Nicolic, B. Åkesson, V. Nélis, and T. Nolte, "Contention-free execution of automotive applications on a clustered many-core platform," in *Proceedings of the 28th Euromicro Conference on Real-Time Systems (ECRTS)*, 2016, pp. 14–24.

Memory Bank Privatization

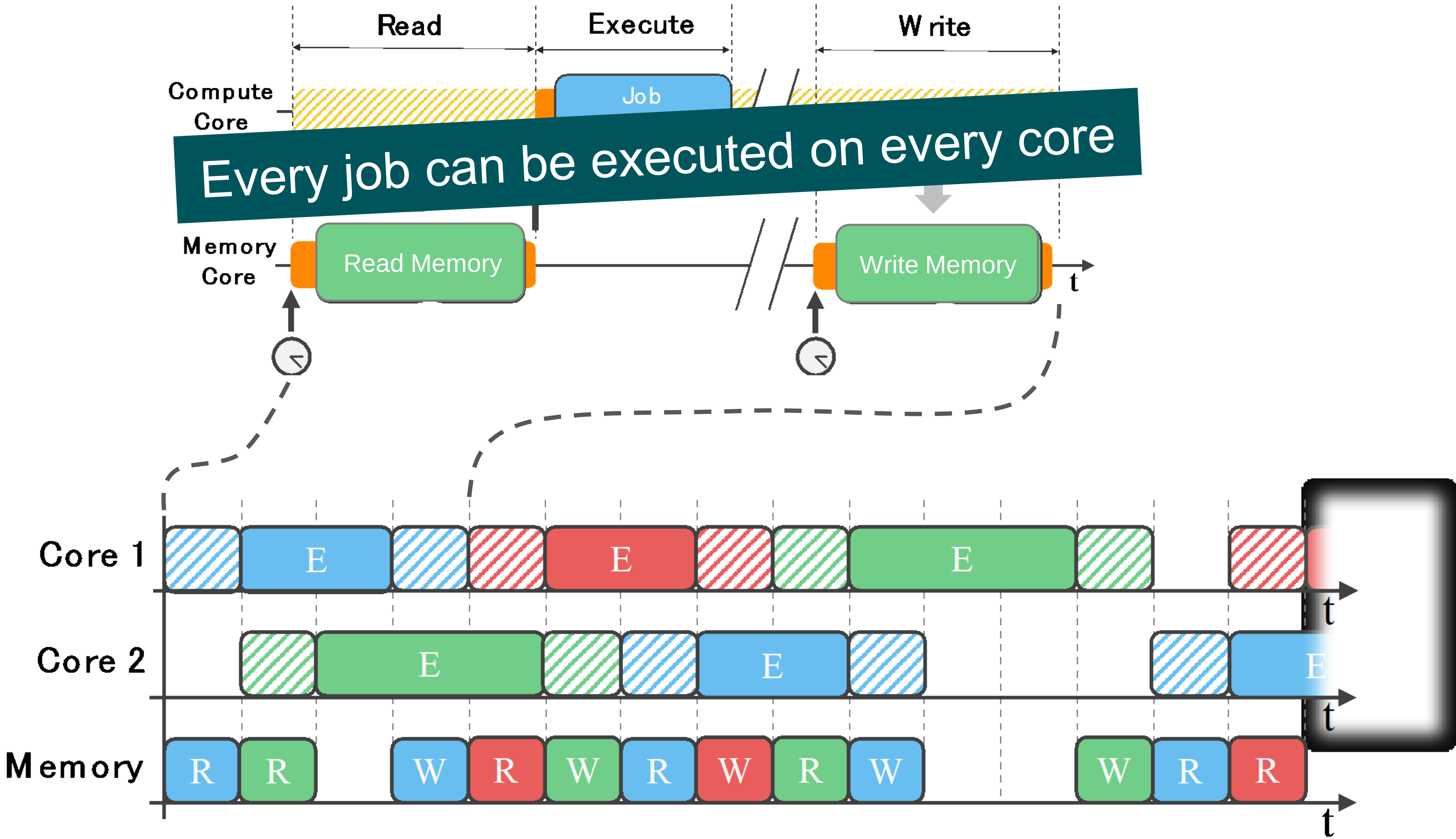


No contention when compute cores access local memory banks during execution phase!

Task Execution – Read-Execute-Write

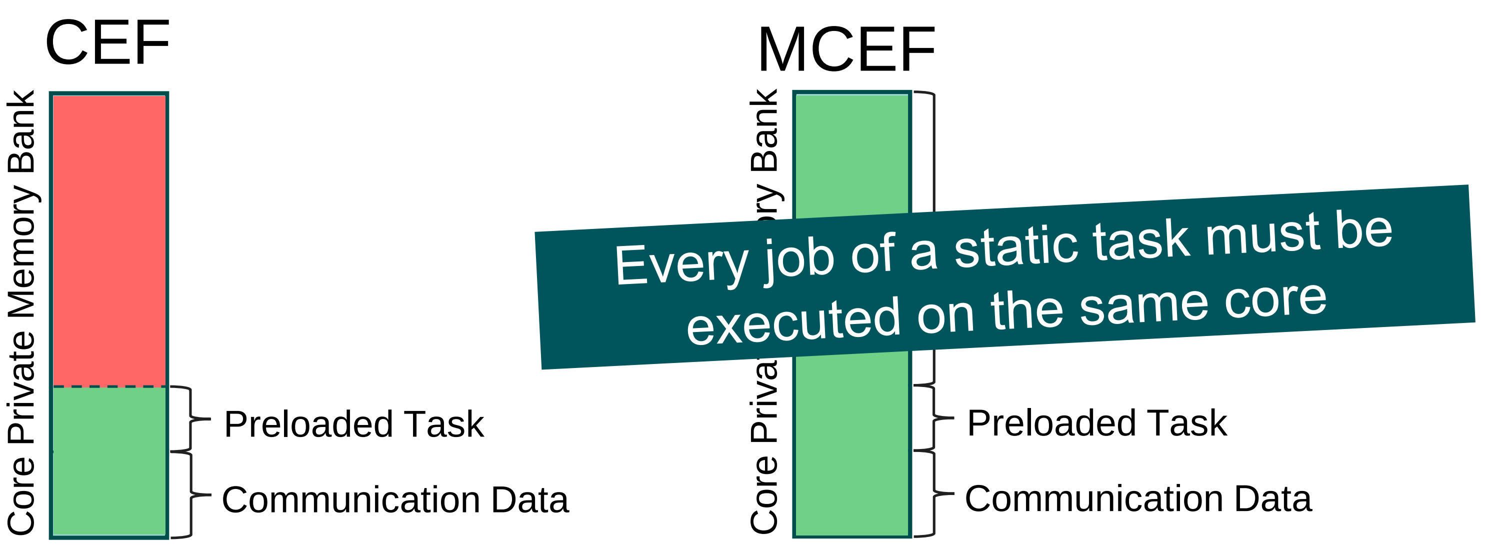


Time Triggered Scheduling



Memory Aware Contention Free Execution Framework

- Local memory banks >> tasks footprint
  - CEF only uses a fraction of the available memory



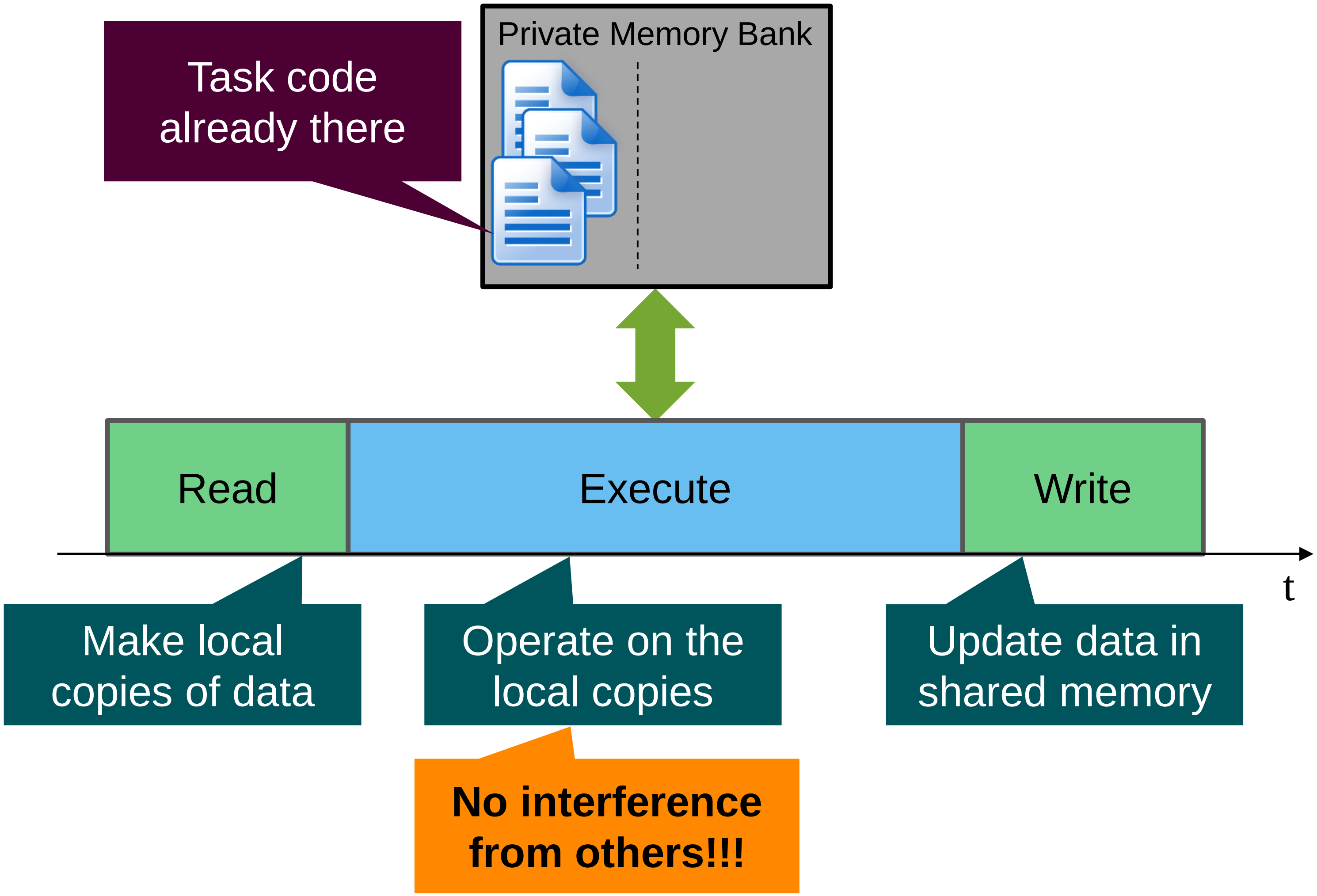
Differentiate tasks based on execution behaviour

- Distinguish task types, **static** and **dynamic**

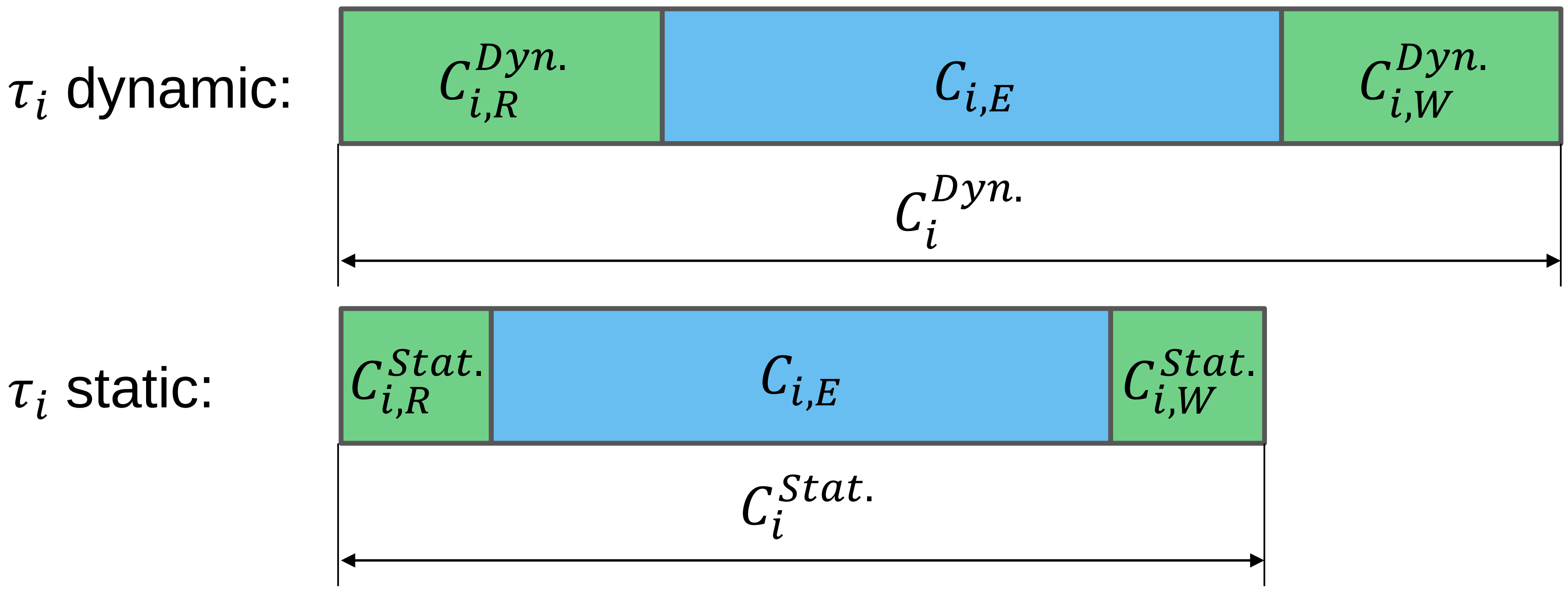
- Dynamic task:
  - **Not all** jobs of the task are executed on the same core
  - Required memory is allocated in **external memory**

- Static task:
  - **All** jobs of the task are executed on the same core
  - Required memory is allocated on the cores **private memory** bank

Static Task


Task code
already there
Private Memory Bank
Read
Execute
Write
t
Make local
copies of data
Operate on the
local copies
Update data in
shared memory
No interference
from others!!!

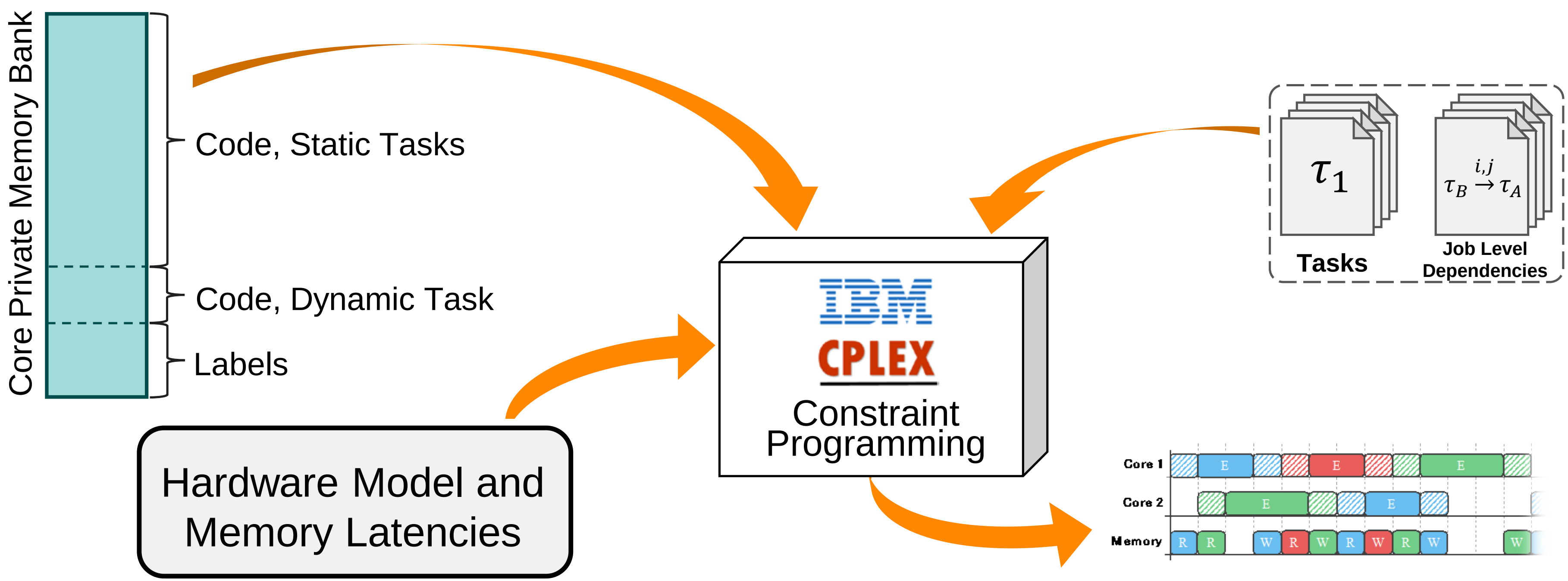
Static vs. Dynamic Tasks



- Static Tasks
  - \+ Less utilization due to shorter memory phases
  - \+ Efficient usage of internal memory
  - \- Scheduling problem becomes harder
- Dynamic Tasks
  - \+ Flexibility during schedule generation
  - \- Larger memory phases that negatively impact scheduling
  - \- Add traffic to the access path to external memory

Generation of System Configuration and Time Triggered Schedule

- Constraint programming (Conditional Time-Intervals)
  - Find the assignment of tasks to static/dynamic types
  - Allocate static tasks to cores considering local memory constraints
  - Generate time triggered schedule to consider execution times depending on task types
  - Consider job-level dependencies between the tasks jobs

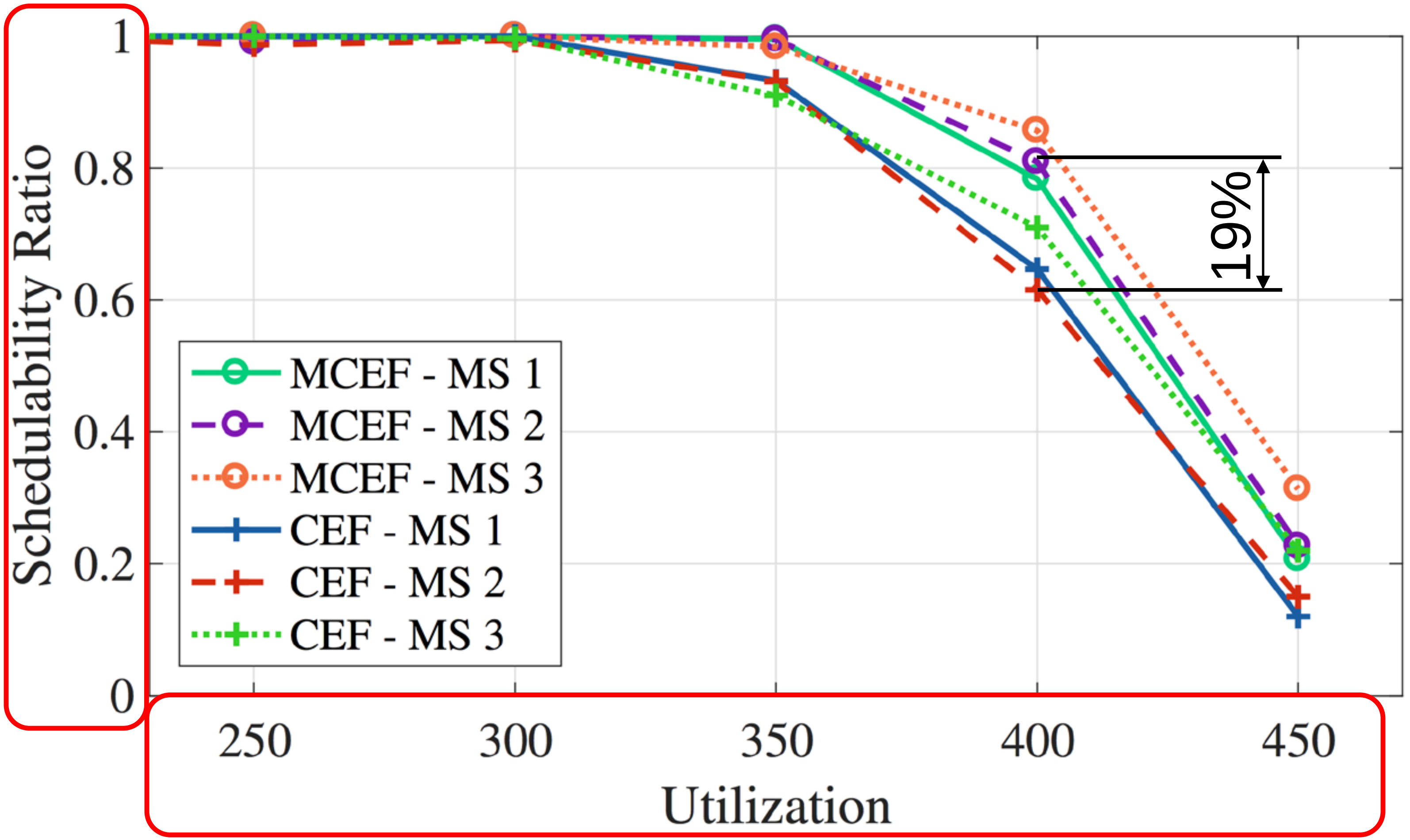


Evaluation – Synthetic Experiments

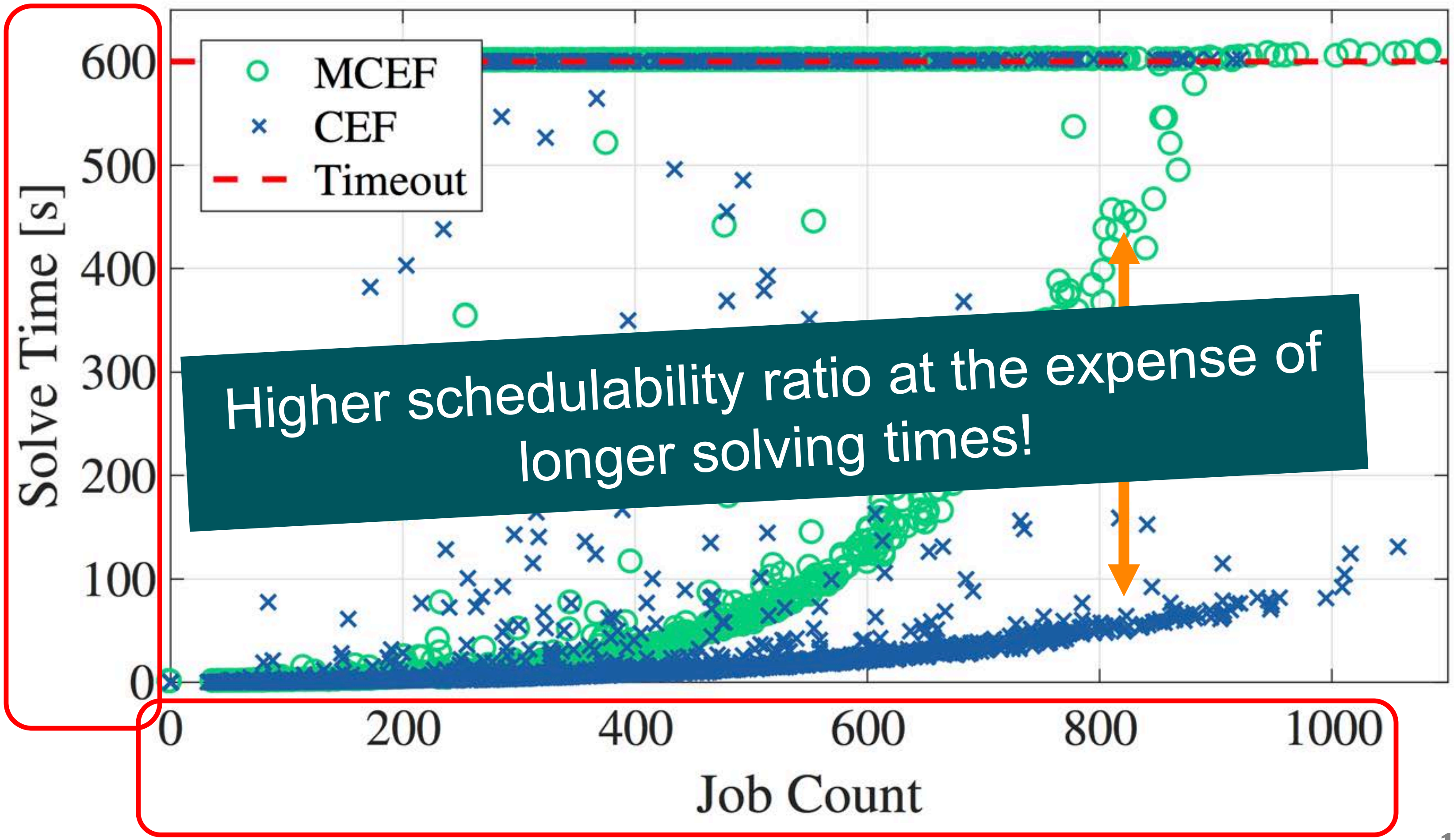
- 5 compute cores @ 400 MHz
- Access to local memory takes 1 cycle to fetch 8 bytes
- Access to external memory 3x slower
- Local memory banks have a size of 64 KB
- Task set contains 10 tasks
  - Periods [1, 2, 5, 10, 20, 50, 100, 200] ms
  - Utilization generated by UUniFast
  - Memory region in 3 Memory Settings (MS)
    - MS1 Footprint: [6, 30] KB, Local Data: [64, 512] bytes
    - MS2 Footprint: [6, 60] KB, Local Data: [64, 1024] bytes
    - MS3 Footprint: [6, 90] KB, Local Data: [64, 2048] bytes

Each data point represents 300 random task sets

Evaluation – Schedulability Ratio



Evaluation – Solving Times



Conclusions and Future Work

- Many-core platforms become increasingly relevant for embedded real-time systems
- CEF provides a way to deterministically execute real-time applications on a clustered many-core platform
- Drawback exists in the inefficient usage of local memory which leads to high NoC utilization
- MCEF overcomes this limitation by assigning tasks statically to local memory
- Constraint Programming formulation to efficiently find a schedule and system configuration that outperforms CEF

- Heuristic solutions to scale to industrial sized applications
- Task grouping to reduce the number of schedulable entities

Thank you for the attention!

Questions?

MÄLARDALEN UNIVERSITY SWEDEN