Analyzing ARM's MPAM From the Perspective of Time Predictability

M. Zini, D. Casini, A. Biondi,

”Analyzing ARM's MPAM From the Perspective of Time Predictability”,
in *Transactions on Computers*, 2022

ReTiS Lab, Scuola Superiore Sant’Anna, Pisa, Italy

Sant’Anna
Scuola Universitaria Superiore Pisa

Real-Time Systems Laboratory

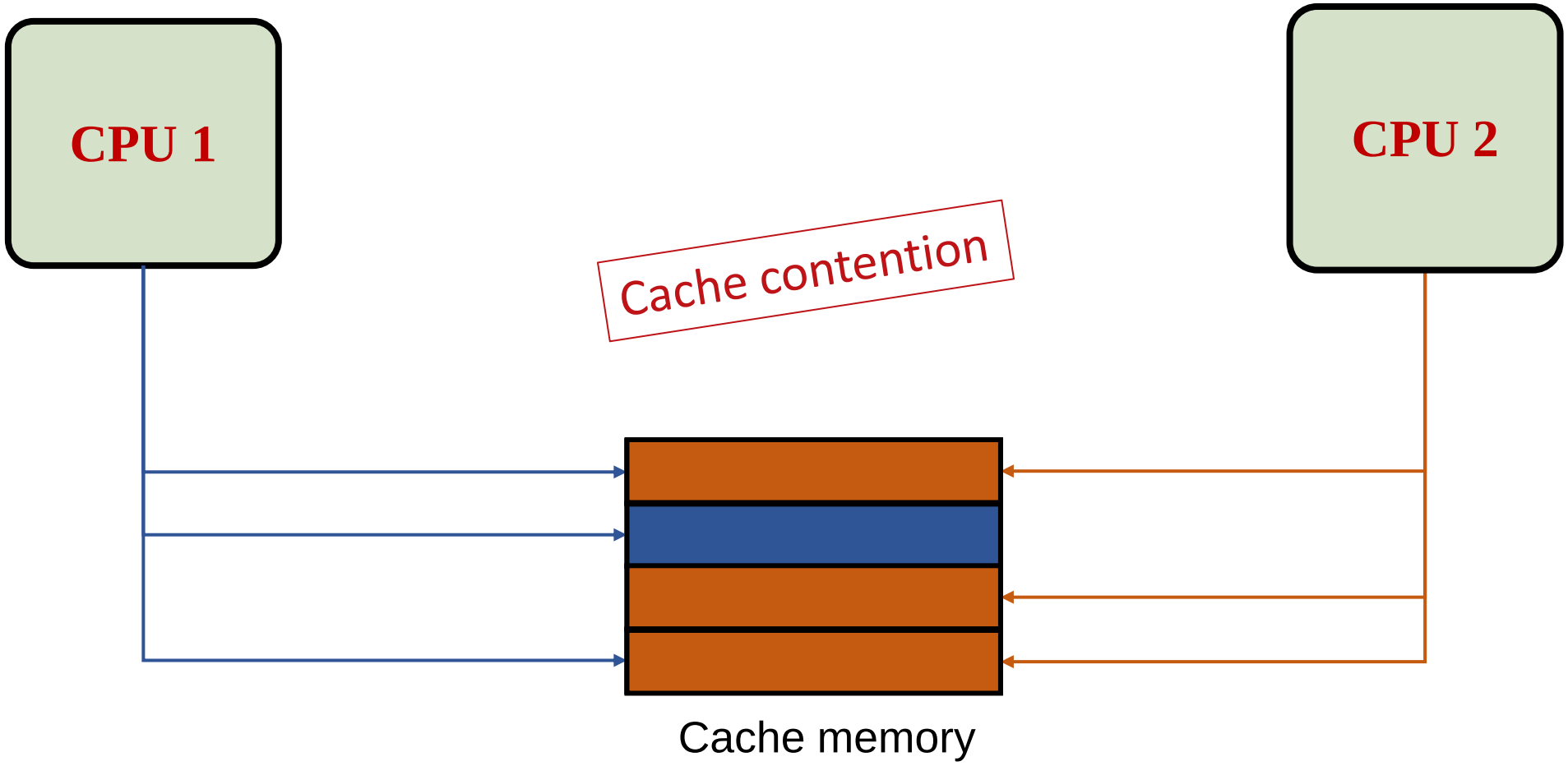
Sources of delay

- In modern computing platform, not only the processor is a shared resource among tasks

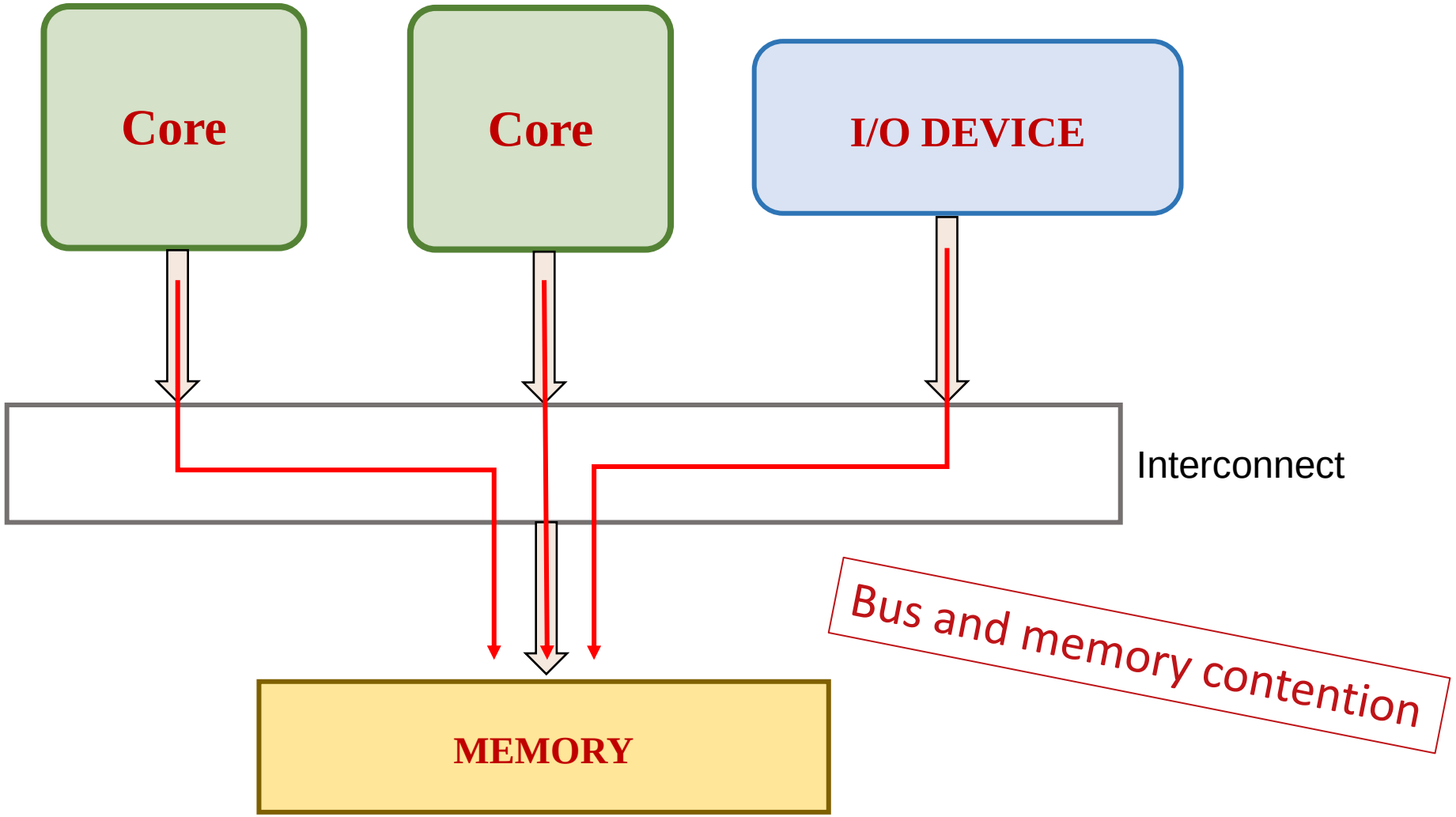
- Systems that require predictability need to take into account other factors: two of them are cache and memory contention

Retis
Real-Time Systems Laboratory

Cache contention



Bus and Memory Contention



Retis
Real-Time Systems Laboratory

ARM MPAM Extension

Memory system resource Partitioning and Monitoring

Extension of ARMv8-A architecture

- **The MPAM specifications describe hardware mechanisms to limit cache and memory access**
- **Such mechanism can be integrated inside CPU cores and Memory System Components**

- **State of the art**
- **MPAM architecture**
- **Modelling the system**
- **Results**

- **State of the art**
- **MPAM architecture**
- **Modelling the system**
- **Results**

State of the art

Software-based techniques:

- Cache coloring
- Bandwidth partitioning with usage monitors

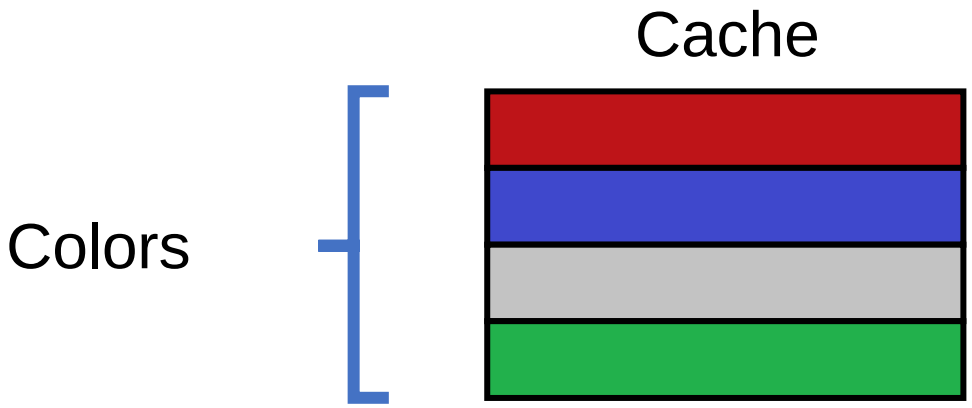
Hardware-based techniques:

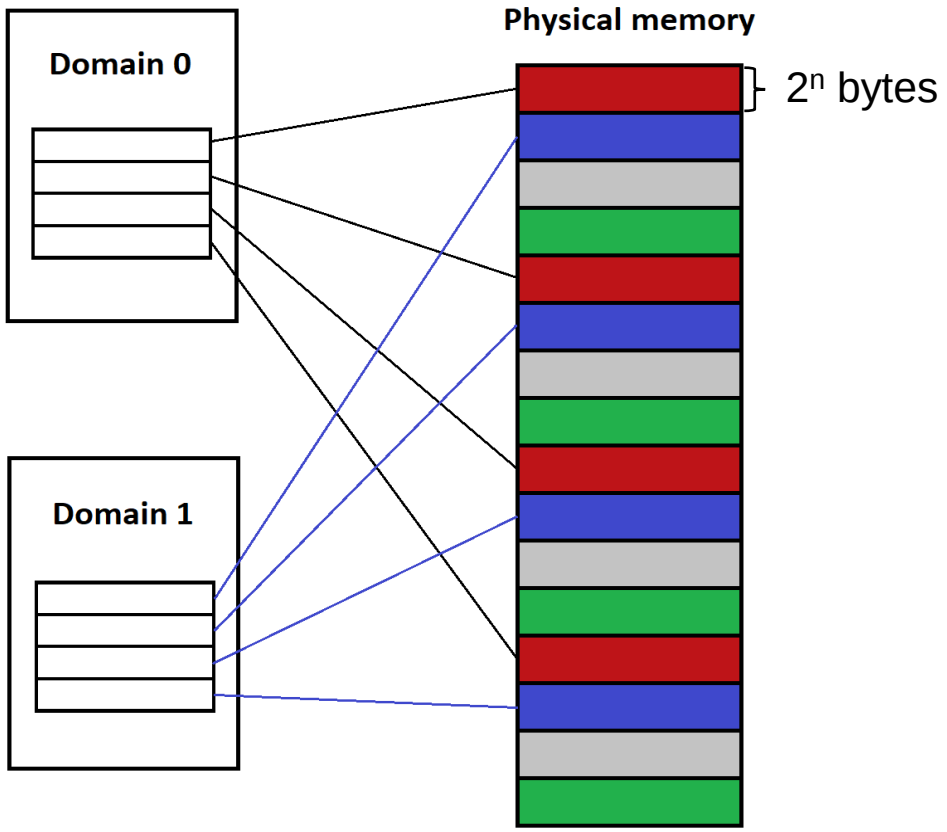
- Intel Resource Director Technology (RDT)

State of the art: cache coloring

Exploit relation between:

- Physical page number
- Cache index

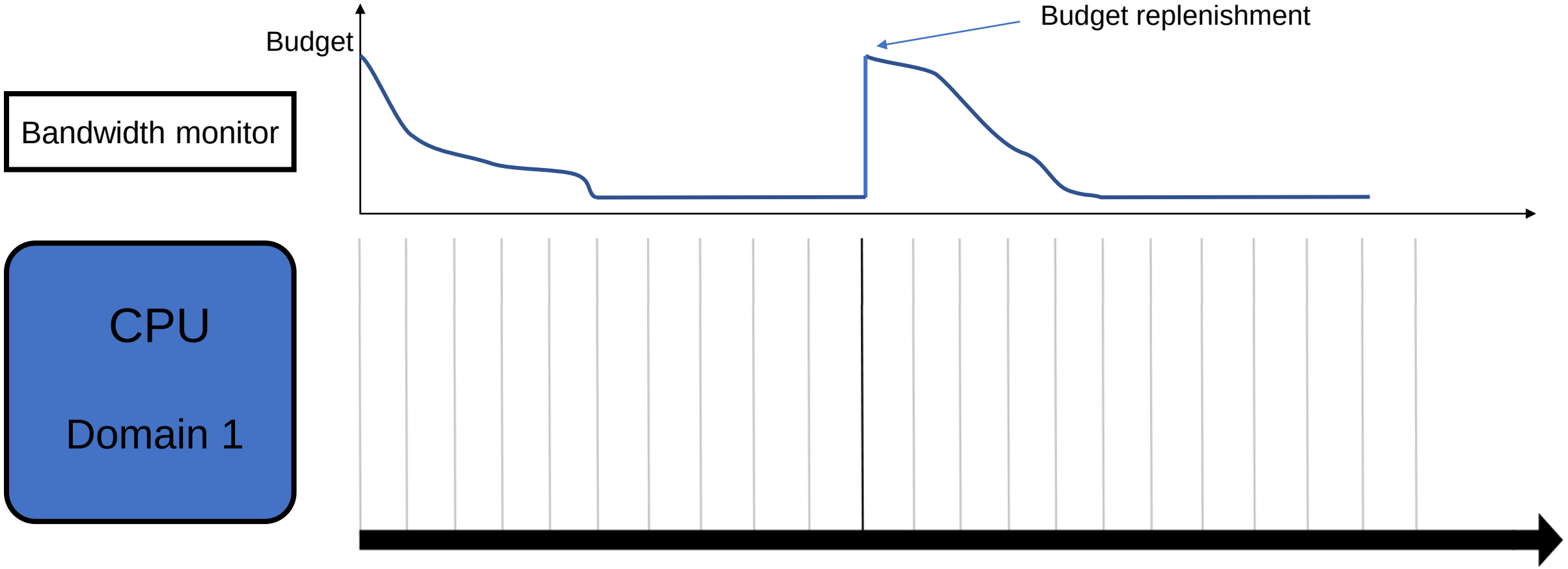




Giovani Gracioli et al. "A Survey on Cache Management Mechanisms for Real-Time Embedded Systems"

State of the art: bandwidth partitioning

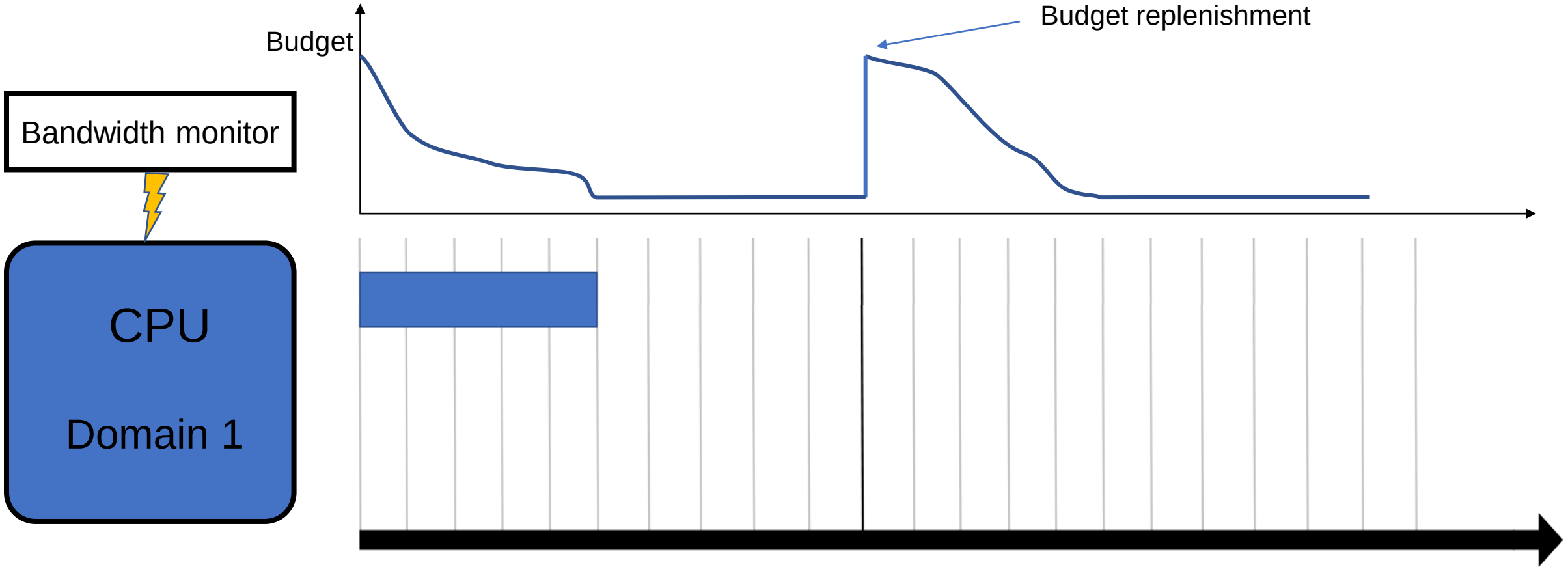
- Memory bandwidth monitors
- Explicit preemption of the tasks



P. Modica et al. "Supporting temporal and spatial isolation in a hypervisor for ARM multicore platforms"

State of the art: bandwidth partitioning

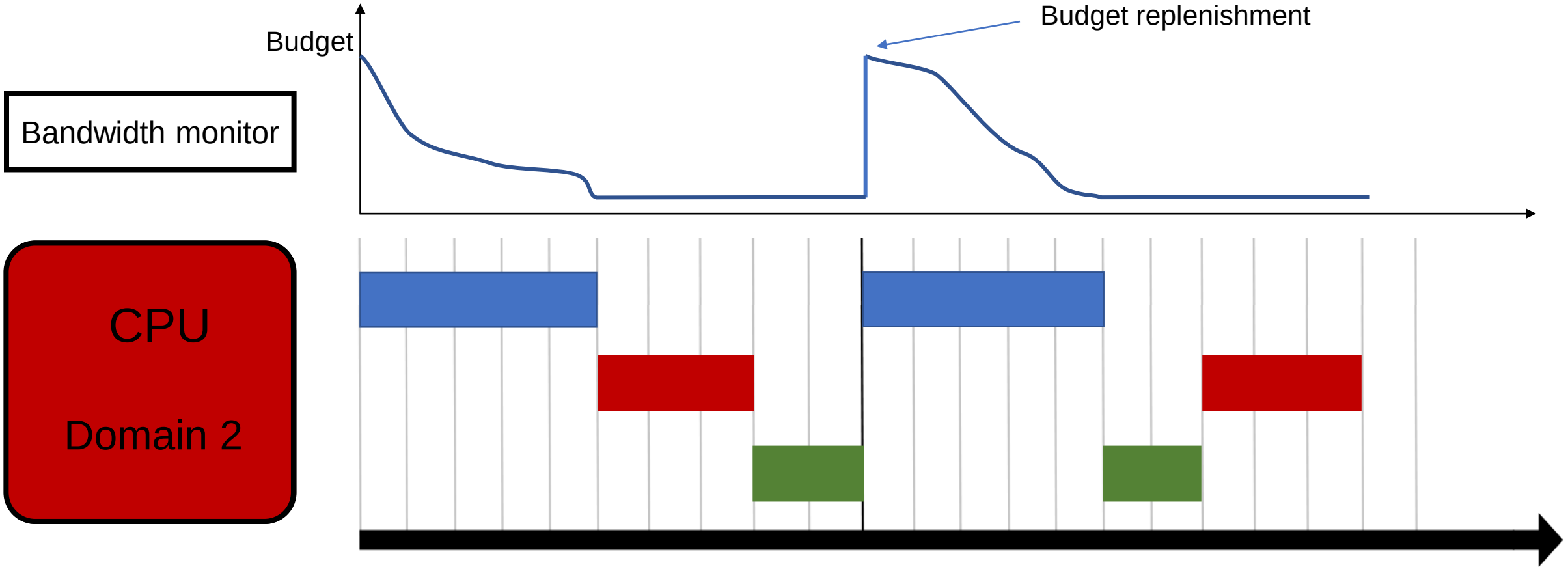
- Memory bandwidth monitors
- Explicit preemption of the tasks



P. Modica et al. "Supporting temporal and spatial isolation in a hypervisor for ARM multicore platforms"

State of the art: bandwidth partitioning

- Memory bandwidth monitors
- Explicit preemption of the tasks



P. Modica et al. "Supporting temporal and spatial isolation in a hypervisor for ARM multicore platforms"

State of the art: Intel RDT

Intel Resource Director Technology functionalities

- Cache Monitoring Technology
- Memory Bandwidth Monitoring
- Cache Allocation Technology
- Code and Data Prioritization
- Memory Bandwidth Allocation

Example of CAT-Only Usage - 16 bit Capacity Masks

COS0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
COS1	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
COS2	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
COS3	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Traditional CAT

Example of Code/Data Prioritization Usage - 16 bit Capacity Masks

COS0.Data	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
COS0.Code	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
COS1.Data	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
COS1.Code	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0
Other COS.Data	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
Other COS.Code	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

CAT with CDP

- State of the art
- **MPAM architecture**
- Modelling the system
- Results

ARM MPAM Extension

Memory system resource Partitioning and Monitoring

Extension of ARMv8-A architecture

- **Resource Monitoring**
- **Resource Partitioning**

} Cache and memory bandwidth

Retis
Real-Time Systems Laboratory

MPAM monitoring

Resource monitoring

- **Cache storage usage monitors**

- **Memory-bandwidth usage monitors**

MPAM partitioning

Resource partitioning

- **Cache partitioning**
  - Portion partitioning
  - Minimum-Maximum capacity partitioning
  - Associativity partitioning

- **Memory bandwidth partitioning**
  - Portion partitioning
  - Proportional stride partitioning
  - Minimum-maximum partitioning
  - Priority partitioning

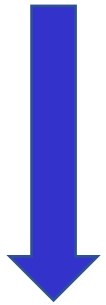
The MPAM specifications are vague

Many details are implementation defined

This is needed because ARM has to take into account many different needs

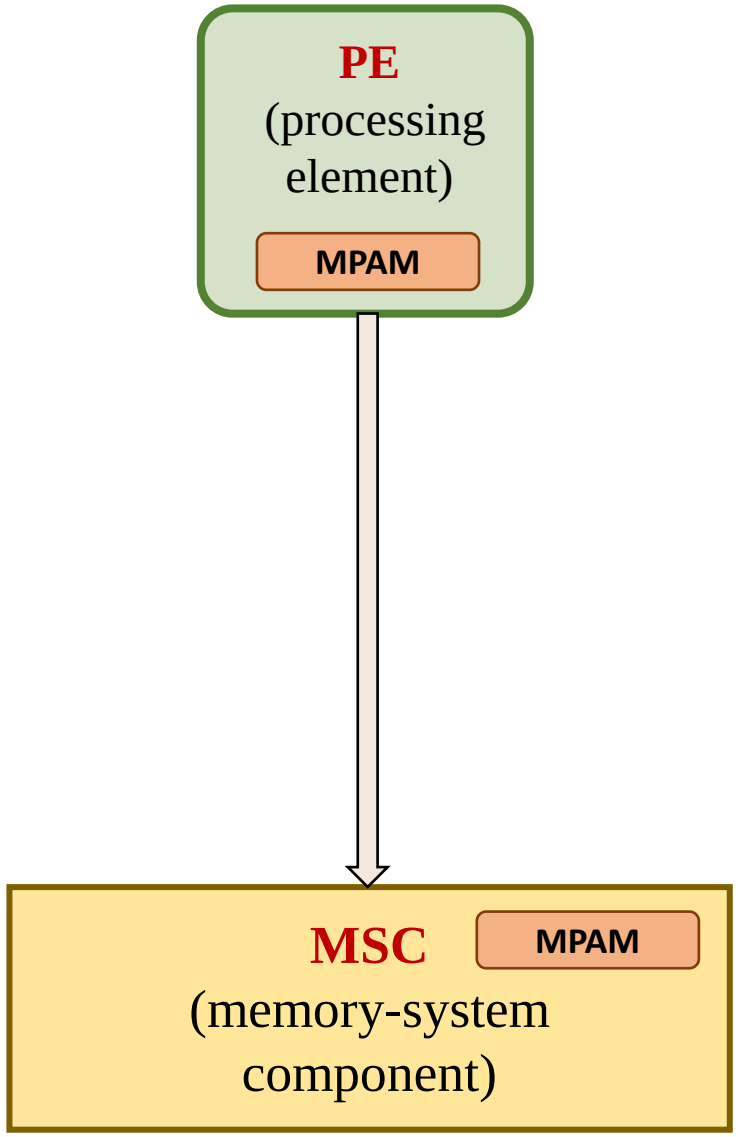
MPAM implementations

No practical implementation to date for real-time systems



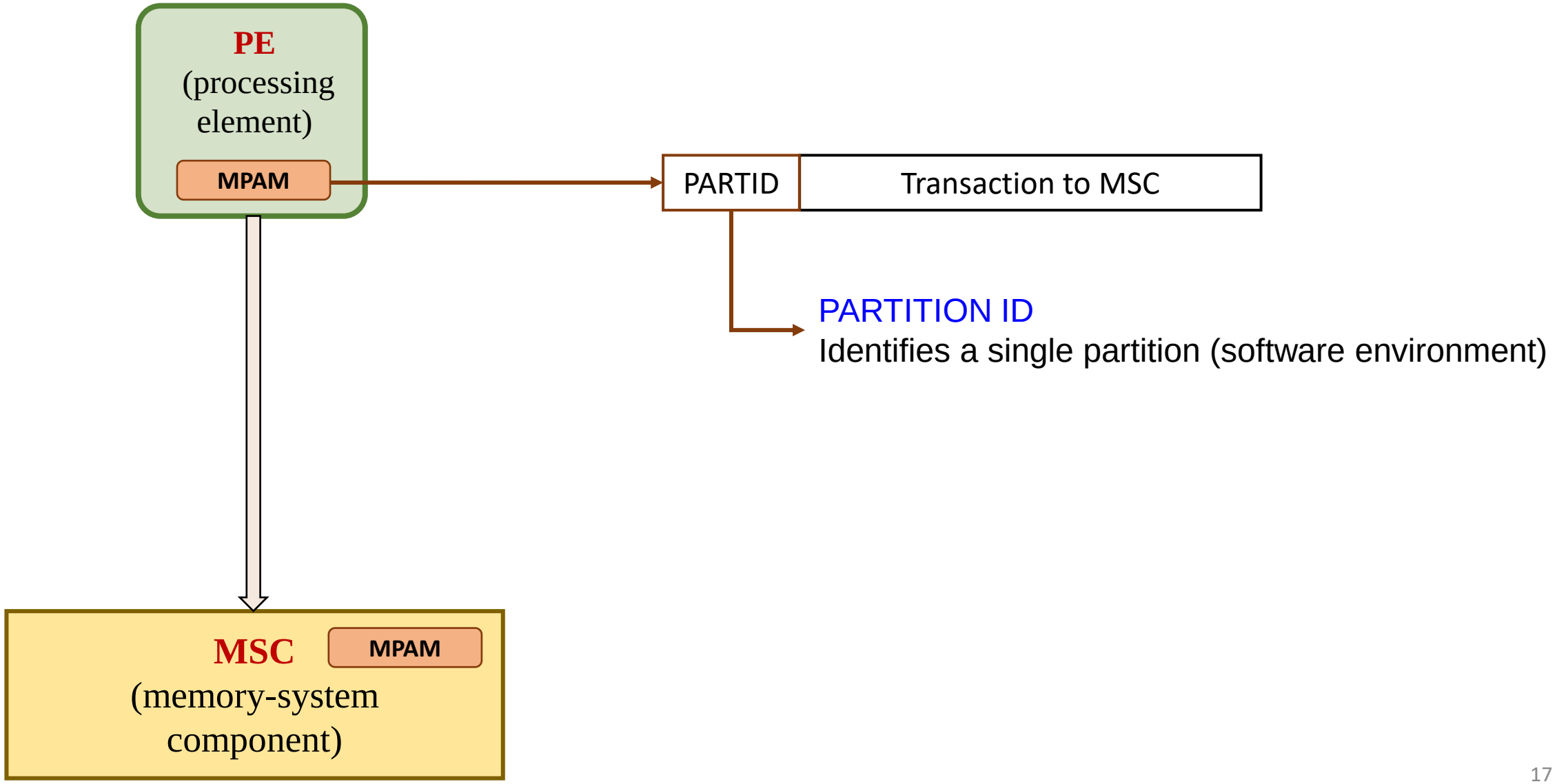
Possibility to influence the future design choices of hardware vendors

MPAM System Architecture

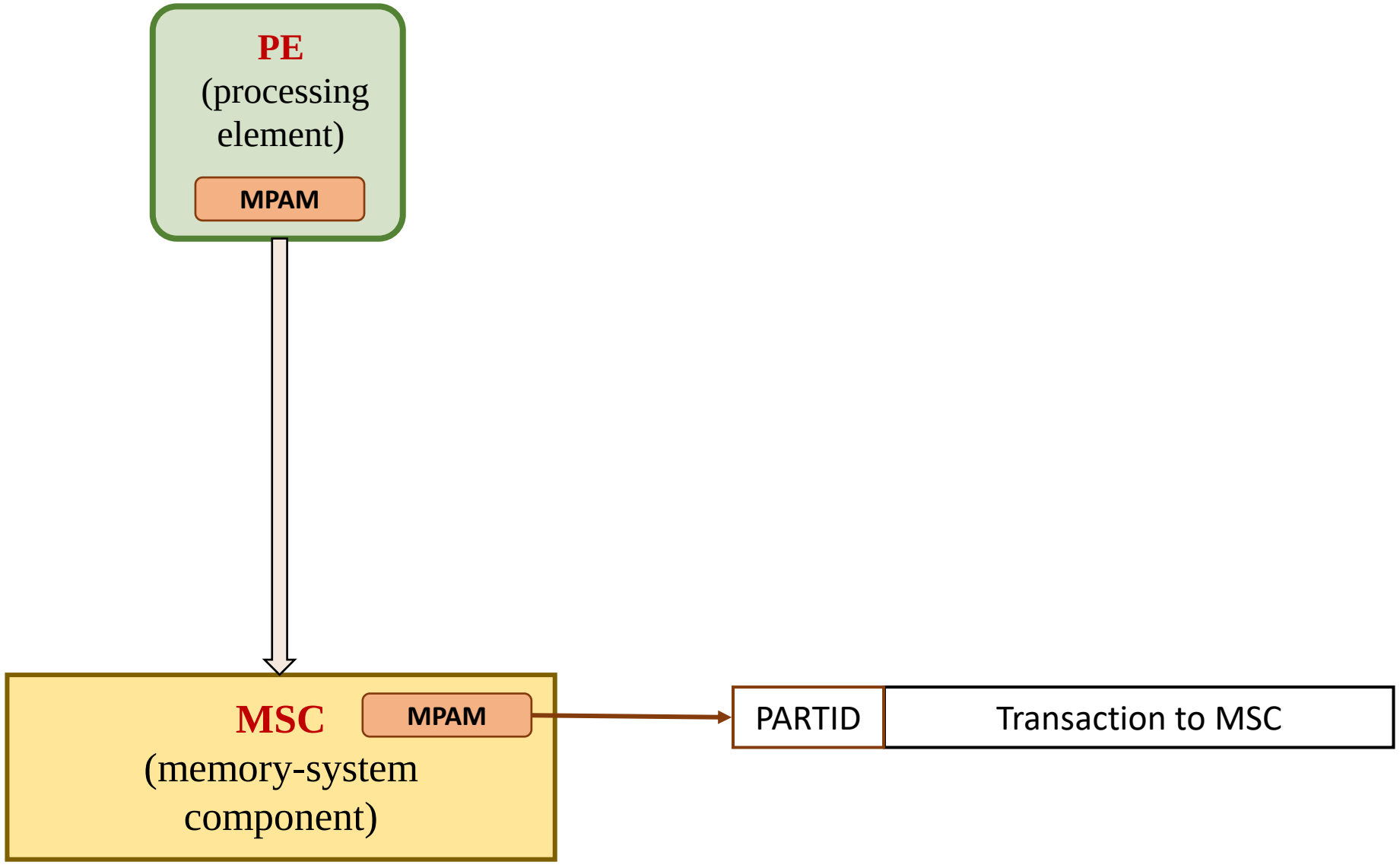


Transaction to MSC

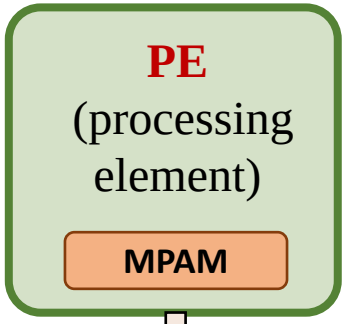
MPAM System Architecture

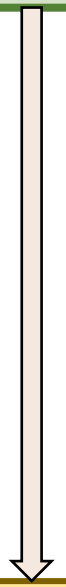


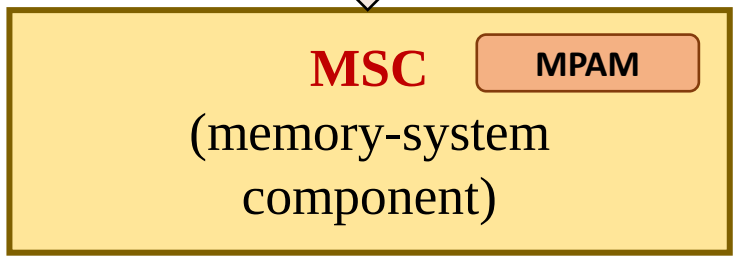
MPAM System Architecture



MPAM System Architecture







- **PARTID** → PARTITION ID
  Identifies a single resource partition
- **PMG** → PARTITION MONITORING GROUP
  Identifies a specific group within a partition
- **MPAM_NS** → MPAM NON-SECURE BIT
  Secure or non-secure execution environment

- State of the art
- MPAM architecture
- **Modelling the system**
- Results

ARM MPAM Extension

Memory system resource Partitioning and Monitoring

Extension of ARMv8-A architecture

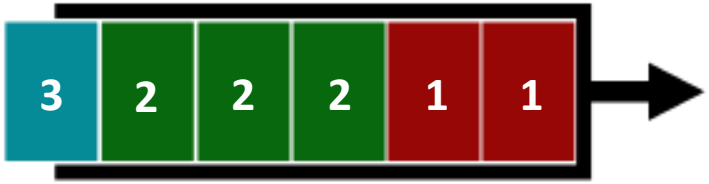
- **Resource Monitoring**
- **Resource Partitioning**

Cache and memory bandwidth

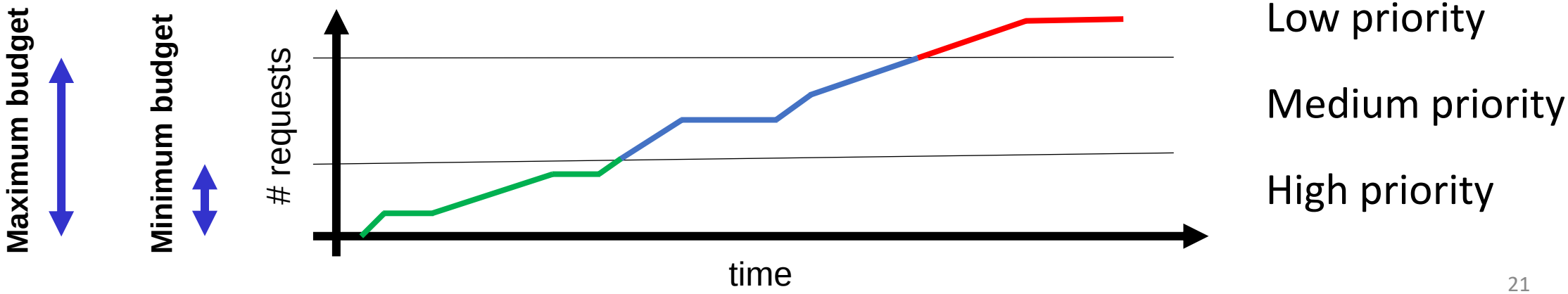
- Priority partitioning
- Minimum-maximum partitioning

MSC Bandwidth Control

- **Priority partitioning:** priority-based requests scheduling

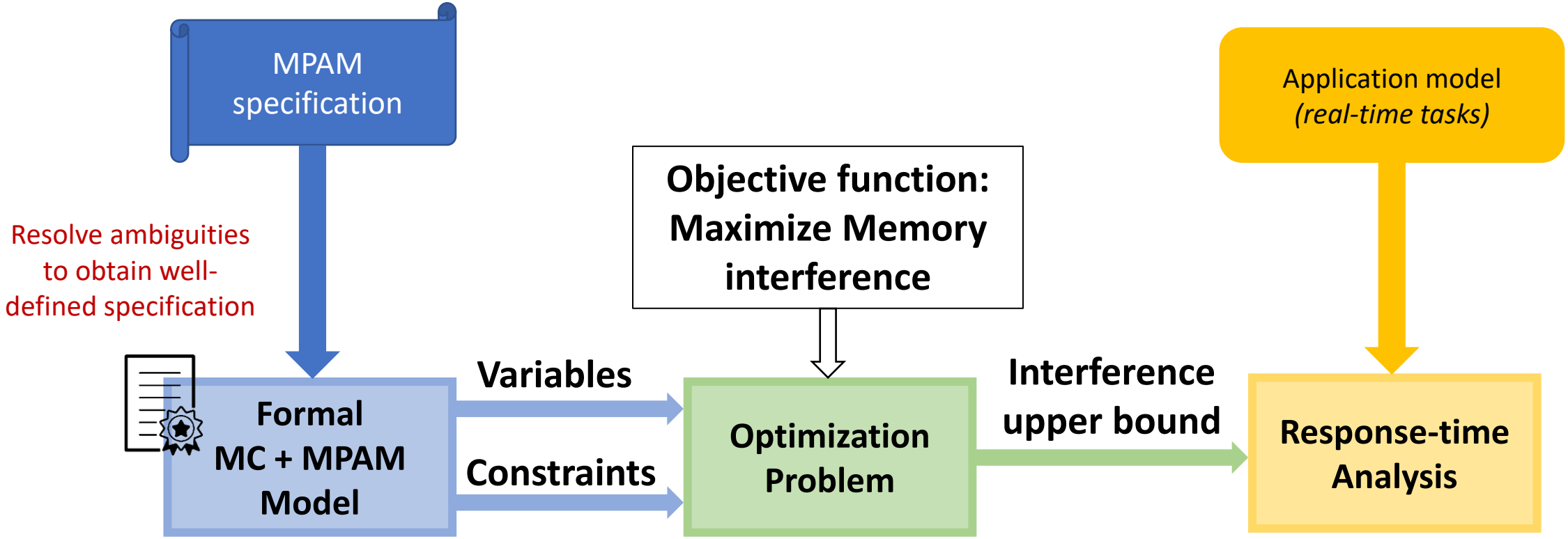


- **Memory-bandwidth minimum and maximum partitioning:** a minimum and maximum budget can be specified for every partition



Modelling the system

The system has been modelled with an optimization problem, in order to compute the worst-case memory interference with and without the MPAM's functionalities.

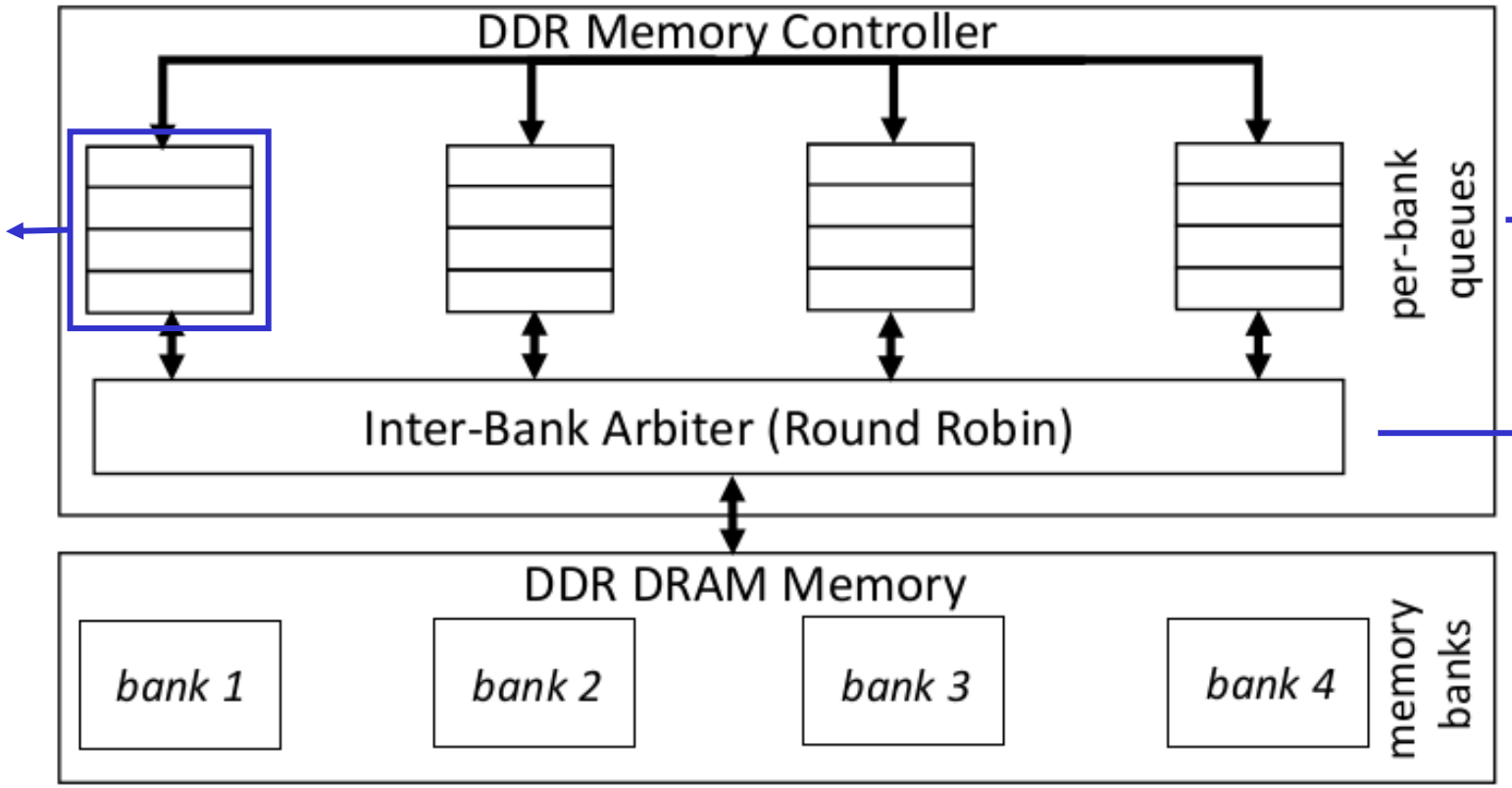


Memory controller model

The memory controller model is based on a generalization of models already used in literature. The assumptions of the existing models have been relaxed.

FR-FCFS intra-bank scheduling:

- FR (first-ready)
- FCFS (first come, first served)



One queue per memory bank

Round robin inter-bank scheduling

D. Casini et al., "A holistic memory contention analysis for parallel real-time tasks under partitioned scheduling"

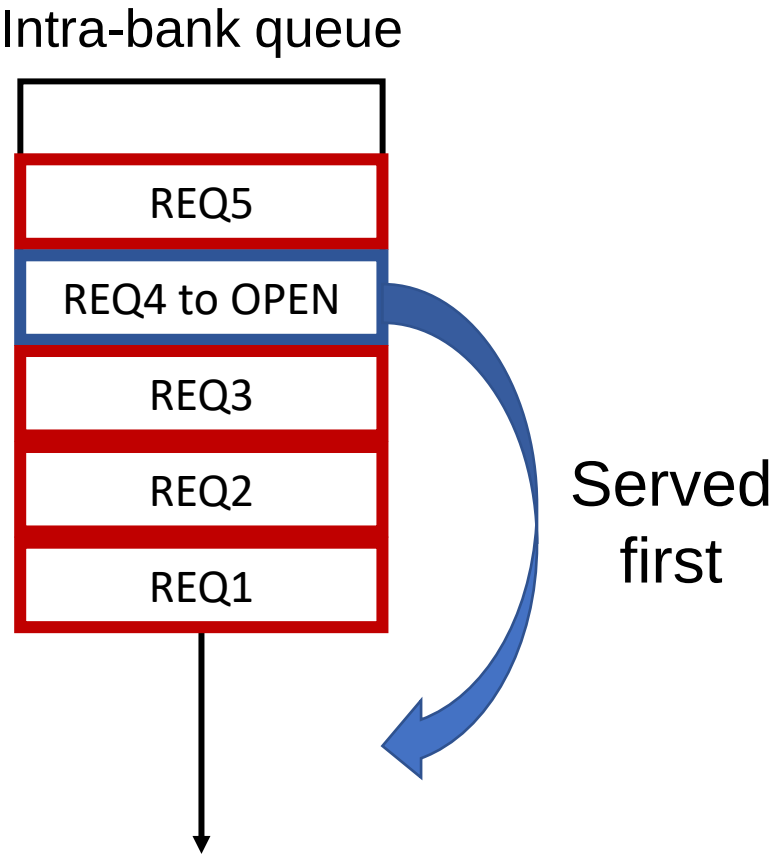
First-ready strategy

The intra-bank queues are scheduled according to the FR-FCFS policy (with thresholding)

- FCFS (first come, first served)
- FR (first-ready):

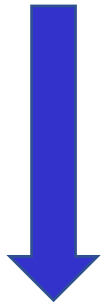
  If a request targets an *open* memory row, it is prioritized

The previously served request targeted the same memory row



MC + MPAM

Ho do we combine the rules of the MC and the rules of MPAM?



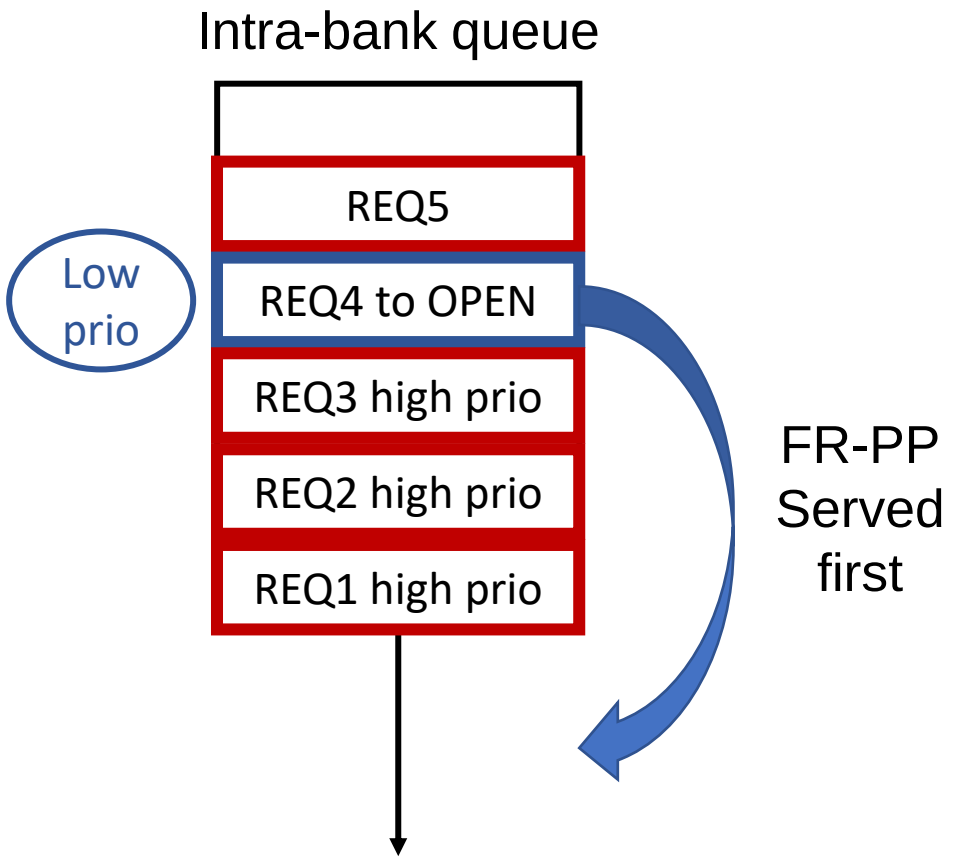
We need to find a global solution that preserves the properties of the two components

Modelling the system

Two possible implementations of priority partitioning were compared:

- **FR-PP (first-ready - priority partitioning):**

  In case a request targeting an open row is pending, it has precedence over other requests with any priority



Modelling the system

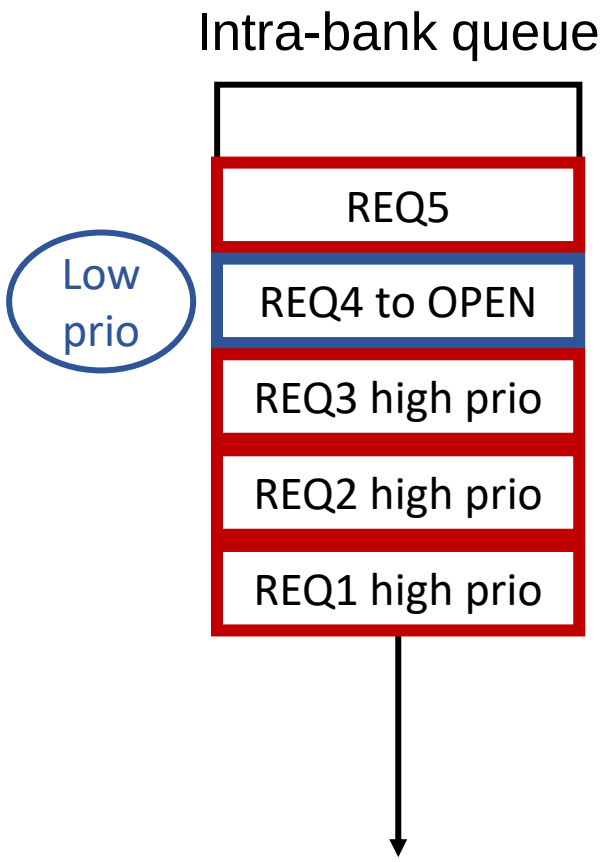
Two possible implementations of priority partitioning were compared:

- **FR-PP (first-ready - priority partitioning):**

  In case a request targeting an open row is pending, it has precedence over other requests with any priority

- **PP-FR (priority partitioning - first-ready):**

  In case a request targeting an open row is pending, it DOES NOT have precedence over other requests with higher priority



Modelling the system

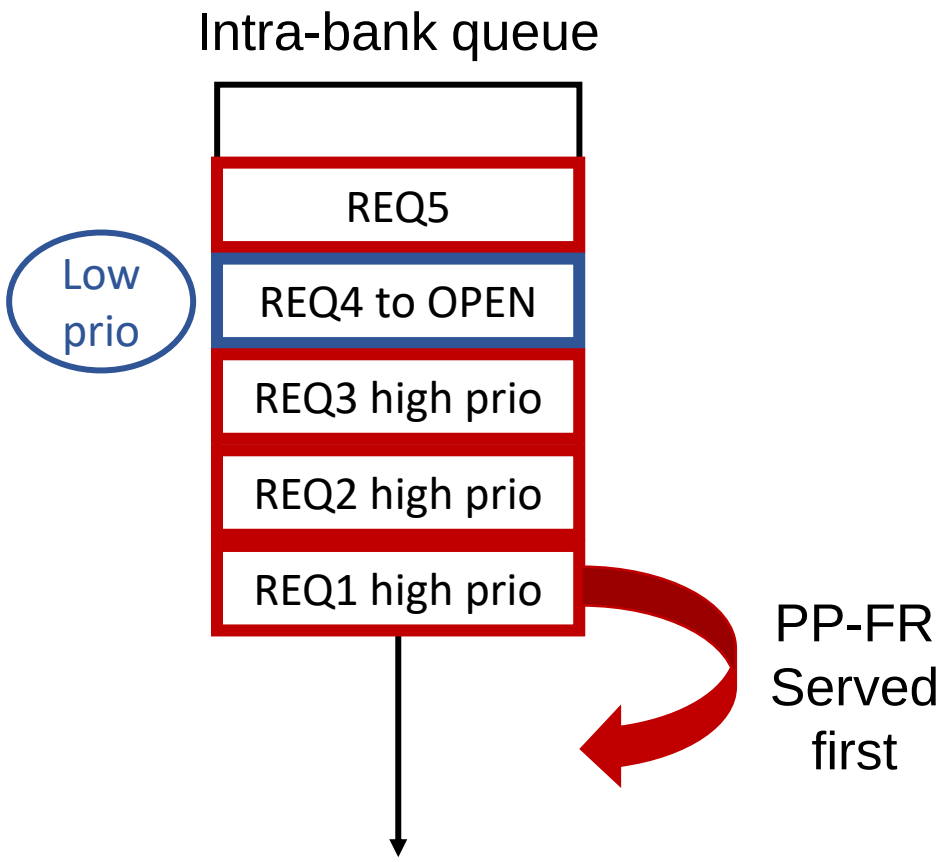
Two possible implementations of priority partitioning were compared:

- **FR-PP (first-ready - priority partitioning):**

  In case a request targeting an open row is pending, it has precedence over other requests with any priority

- **PP-FR (priority partitioning - first-ready):**

  In case a request targeting an open row is pending, it DOES NOT have precedence over other requests with higher priority

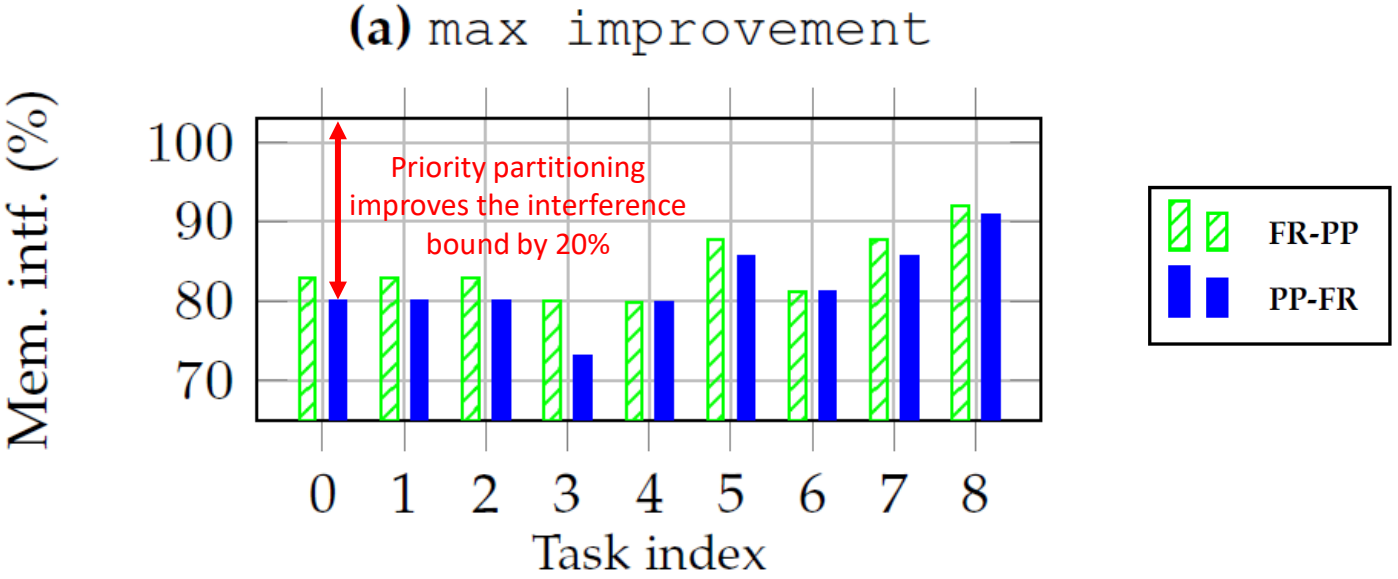


- State of the art
- MPAM architecture
- Modelling the system
- **Results**

Priority Partitioning

The model has been tested with a task set derived from the WATERS 2019 industrial challenge by Bosch

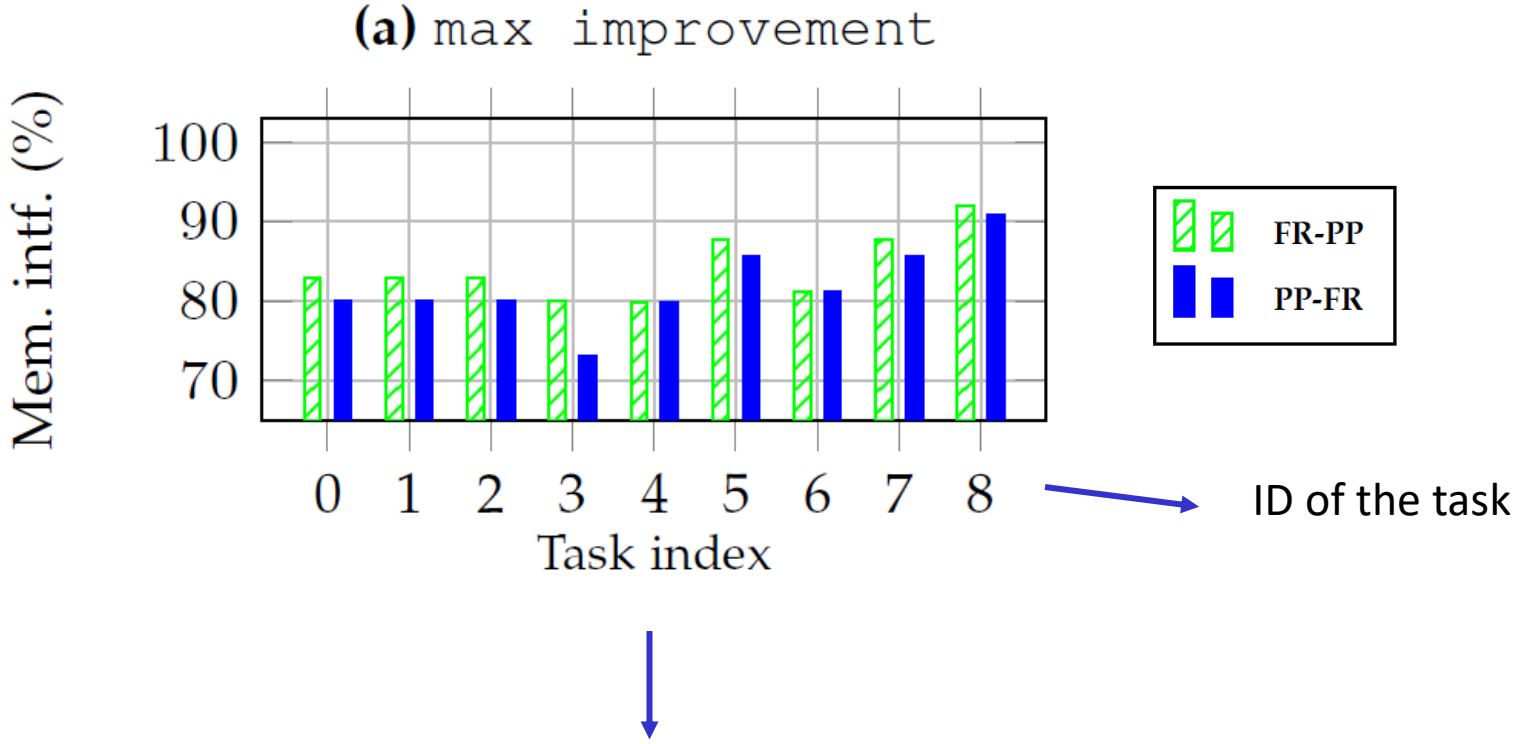
% of memory interference, w.r.t. the case w/o MPAM



Priority Partitioning

The model has been tested with a task set derived from the WATERS 2019 industrial challenge by Bosch

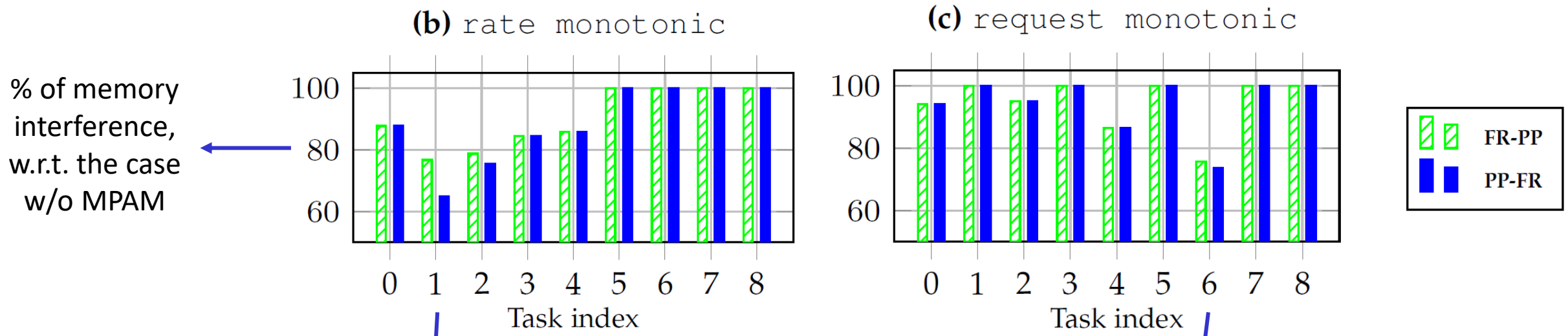
% of memory interference, w.r.t. the case w/o MPAM



ID of the task

For each task, the plot shows the improvement when that task is assigned the highest priority

Priority Partitioning

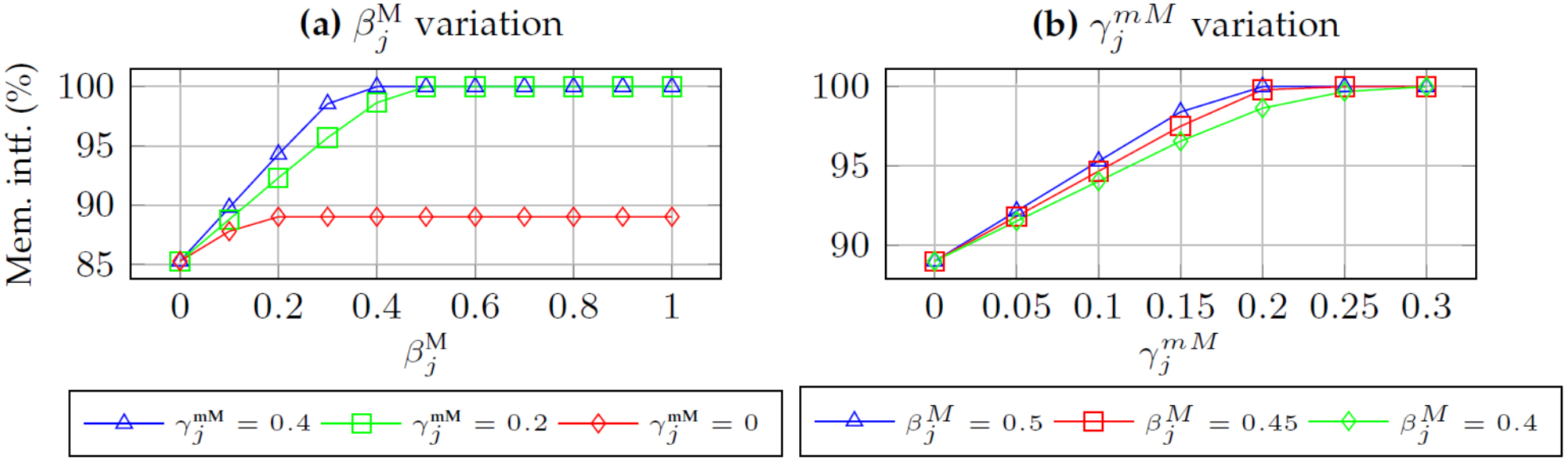


Task 1 has the smallest period → Task 1 has the biggest improvement

Task 6 has the highest number of requests → Task 6 has the biggest improvement

Min-max Partitioning

% of memory interference, w.r.t. the case w/o MPAM



$$\gamma = \frac{\#req\ high\ prio}{\#req\ medium\ prio}$$

Increase the number of medium priority requests of interfering tasks

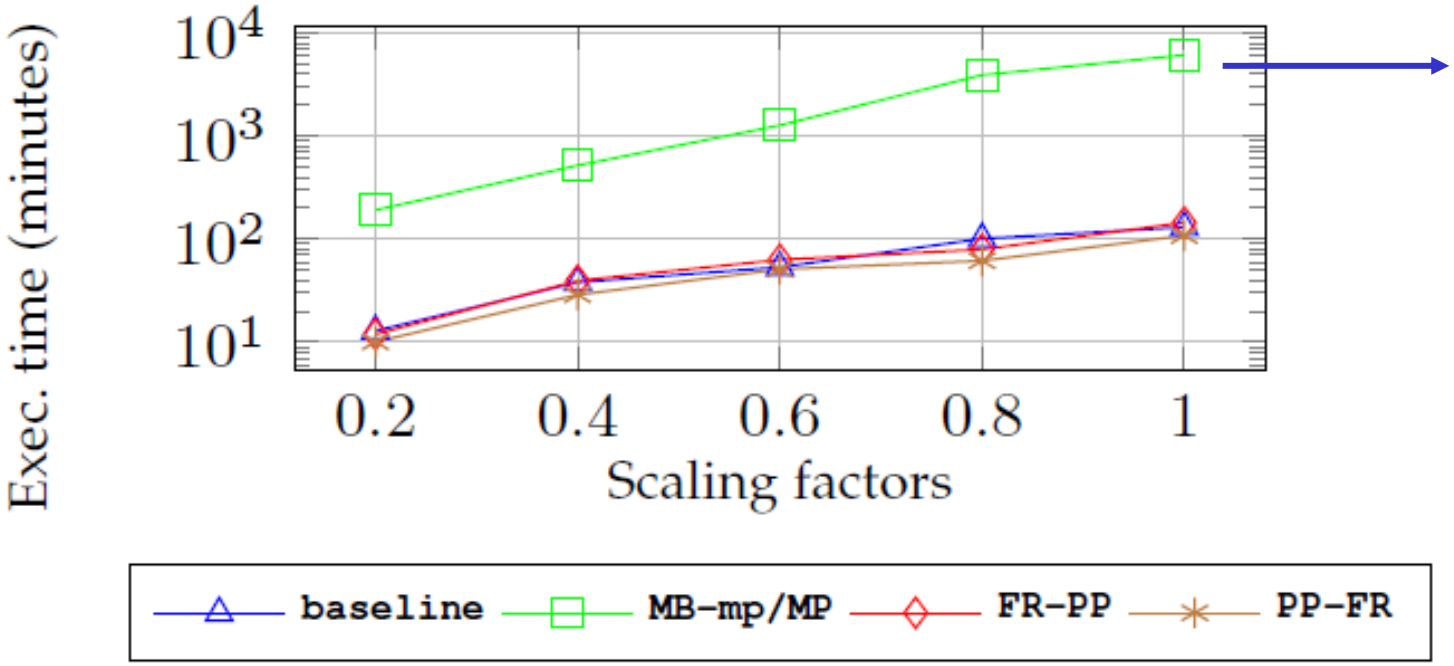
$$\beta = \frac{\#req\ medium\ prio}{\#req\ total}$$

Increase the number of high priority requests of interfering tasks

Scalability

We also assessed how the number of requests affects the execution time of the optimization problem

Time increase



The min-max partitioning is significantly more time consuming

Increase the overall number of requests in the system

Retis
Real-Time Systems Laboratory

Conclusions

- MPAM offers interesting tools to reduce memory contention
- The priority partitioning strategy can bring good results
- The min-max strategy offers very reduced benefits in the worst case

Often, simpler is better for predictability

Retis
Real-Time Systems Laboratory

Future work

- **Analysis of other MPAM strategies**
- **Use more aggressive configuration of min-max strategy**
- **Evaluation of the strategies' effectiveness in the average-case scenario (in a simulated environment)**

Thank you!

Matteo Zini
matteo.zini@santannapisa.it