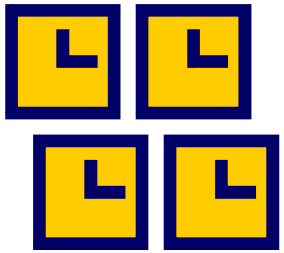


P-SOCRATES

Parallel Software Framework for Time-Critical many-core Systems

A system model and stack for the parallelization of time-critical applications on many-core architectures

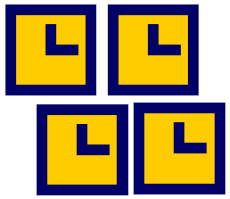
Vincent Nelis, Patrick Meumeu Yomsi, Luís Miguel Pinho, Eduardo Quiñones, Marko Bertogna, Andrea Marongiu, Paolo Gai, and Claudio Scordino

CISTER - Research Center in Real-Time & Embedded Computing Systems

UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich



P-SOCRATES

- **P**arallel **SO**ftware framework for time-**CR**itical m**A**ny-core sys**TE**m**S**
- Three-year FP7 STREP project (Oct-2013, Oct-2016)
- www.p-socrates.eu
- **Partners**

CISTER - Research Center in Real-Time & Embedded Computing Systems

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

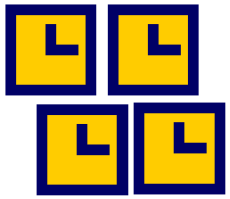
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

EVIDENCE®
EMBEDDING TECHNOLOGY

AT

Atos



P-SOCRATES

Advisory board:

- Airbus Operations, France
- Airbus Defense and Space, France
- Kalray, France
- Honeywell, Czech Republic
- IBM Haifa, Israel
- Saab Avionics, Sweden
- MBDA, Italy
- Expert System, Italy
- Rapita System, UK

Processor Design Evolution

HPC Evolution | **Embedded Evolution**

Single-Core CPU

Single-Core CPU + Hardware Accelerators

+ Performance

+ Ene

+ Time Predictability

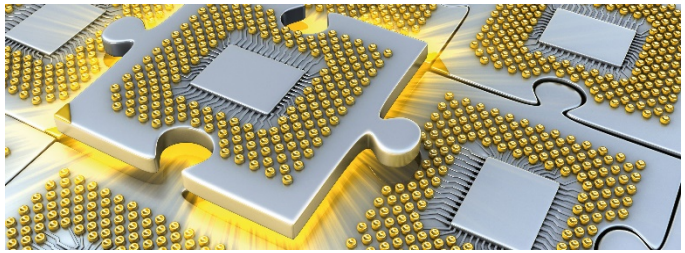
ncy

New systems require a combination of time predictability and high performance

+ Energy-efficiency

+ Time Predictability?

Towards a Real-time Parallel Programming Model



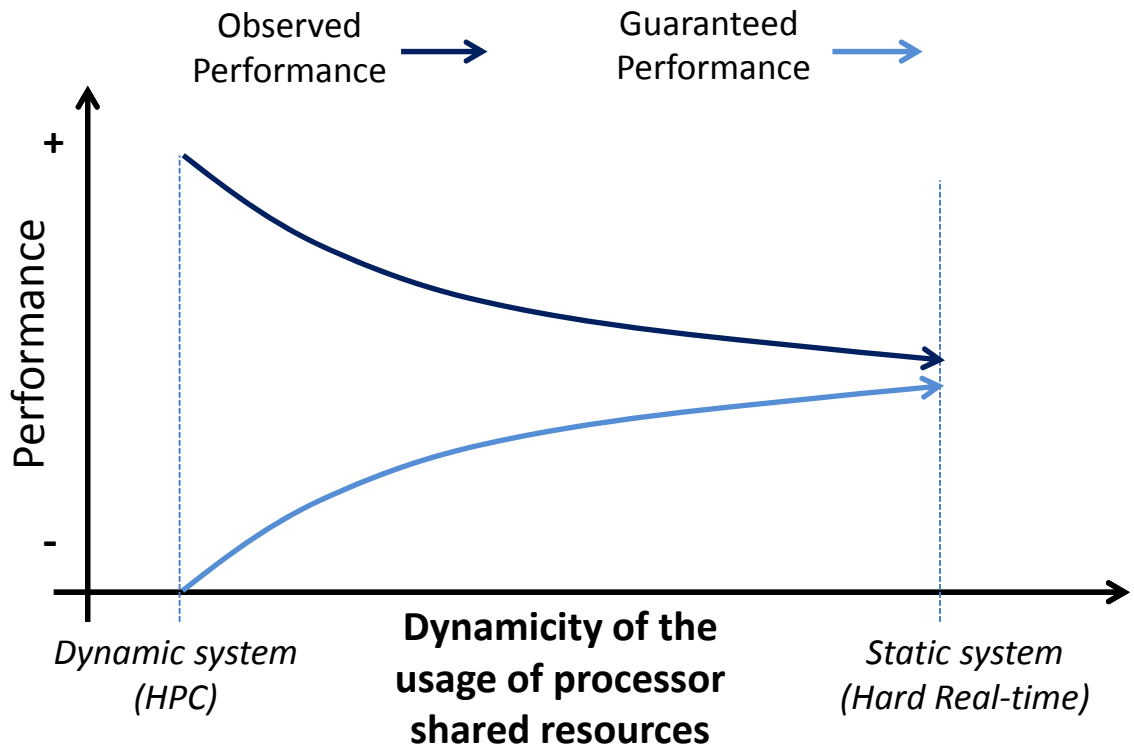
next-generation
many-core accelerators (EC)

real-time
methodologies
to provide **time**
predictability

programmability of
many-core accelerators (HPC)

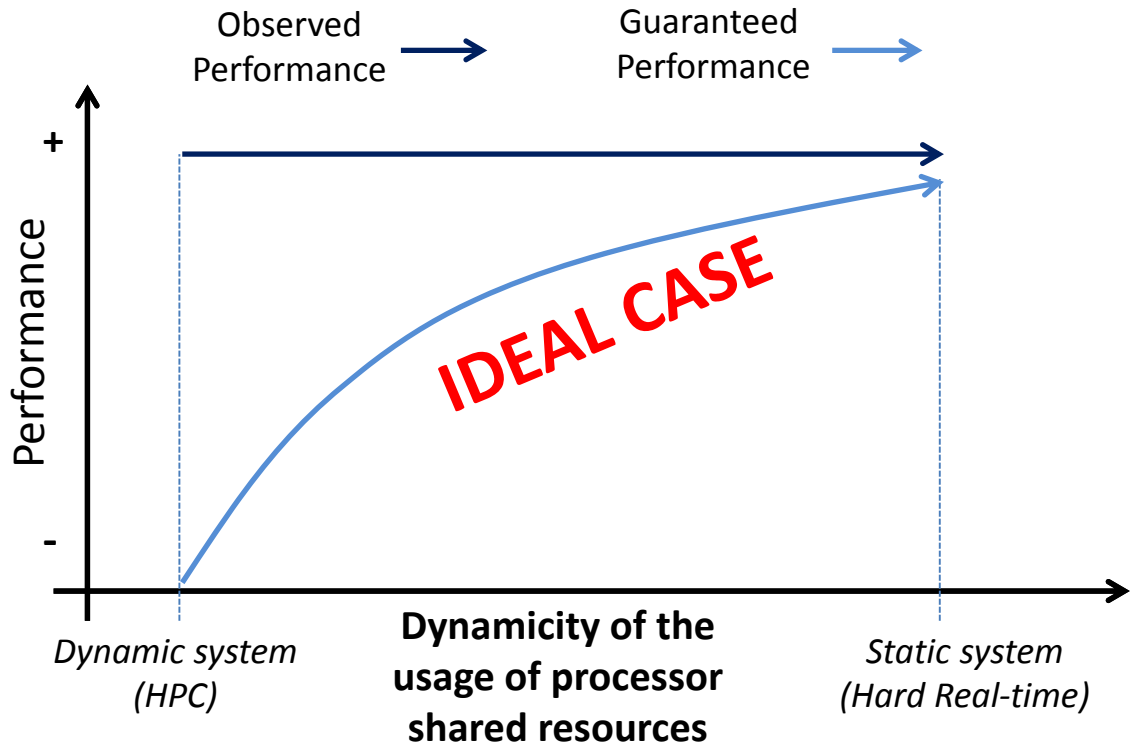
Research Challenges

- Minimizing performance lost, maximizing guaranteed performance



Research Challenges

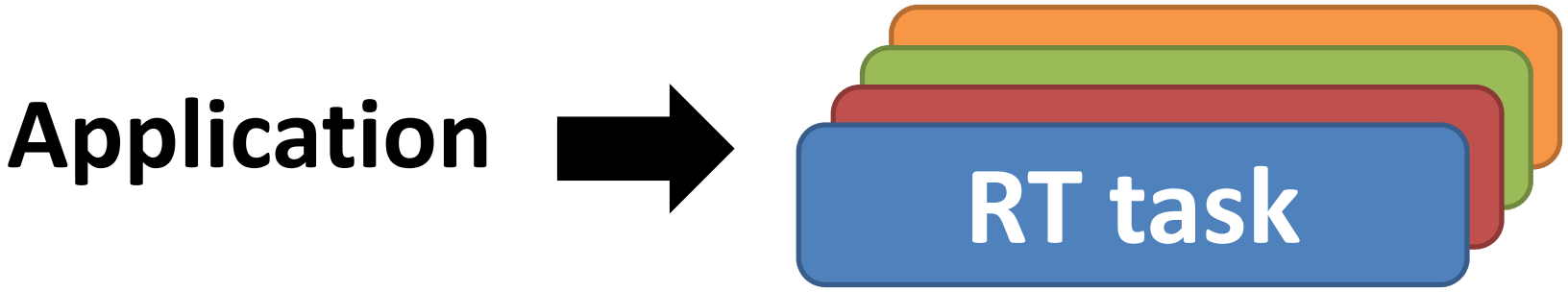
- Minimizing performance lost, maximizing guaranteed performance

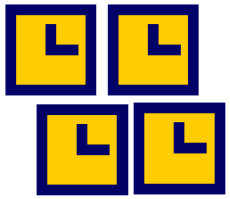


Application model

The **application** is the software implementation (i.e., the code) of the **functionality that the system must deliver to the end-user**. It comprises all the software parts of the systems that operate at the user-level and that have been explicitly defined by the user.

The **application** is organized as a collection of **real-time tasks**.



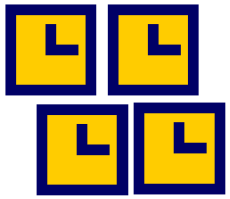


Application model

A **real-time (RT) task** is a recurrent activity that is a part of the overall system functionality to be delivered to the end-user.

An **RT task** is characterized by a few parameters related to its timing behavior, such as the frequency of its activation (aka its period), the time frame in which it must complete (aka its deadline), etc.

Every **RT task** is implemented and rendered parallelizable using OpenMP 4.0.



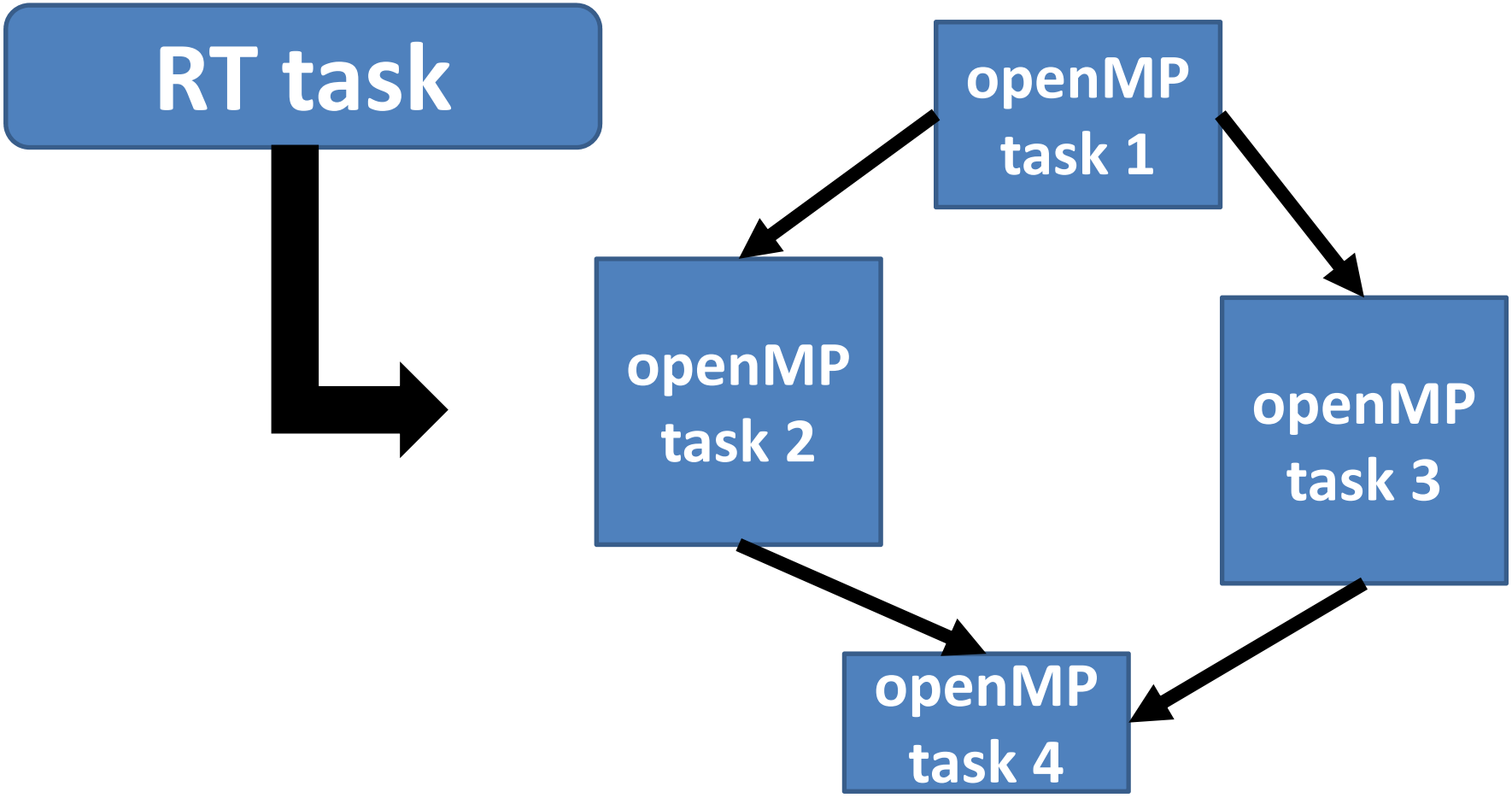
Application model

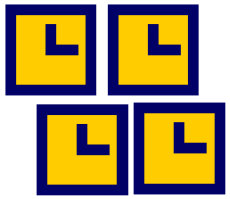
```
#pragma omp task
{
// The brackets identify the boundaries of the task region
// The code goes here
}
```

Every **RT task** comprises a **collection of openMP tasks** whose inter-dependencies are captured and modeled by a graph called the extended task dependency graph (eTDG).

An **openMP task** is defined at run-time by the syntactic boundaries of an openMP task construct.

Application model



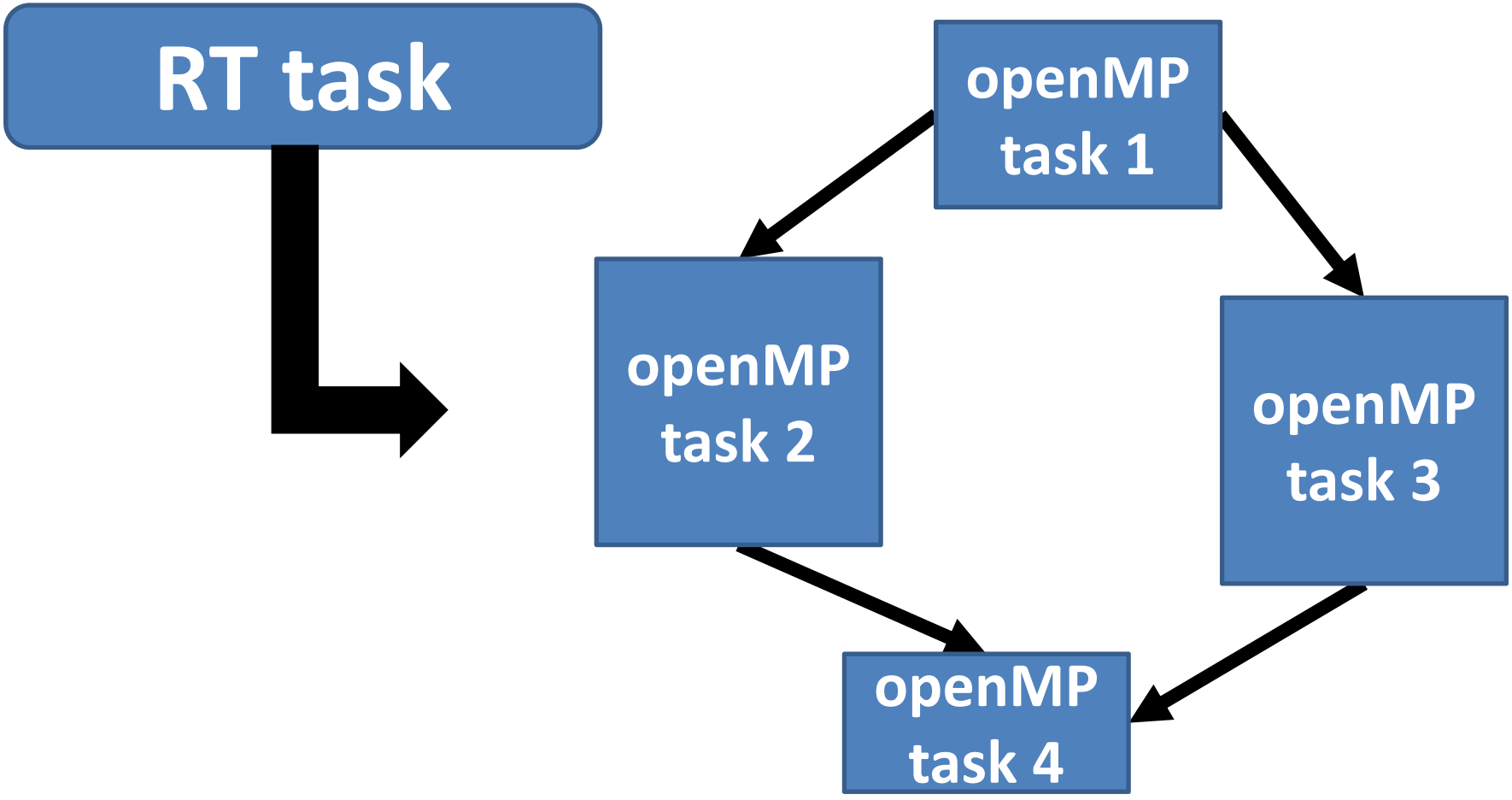


Application model

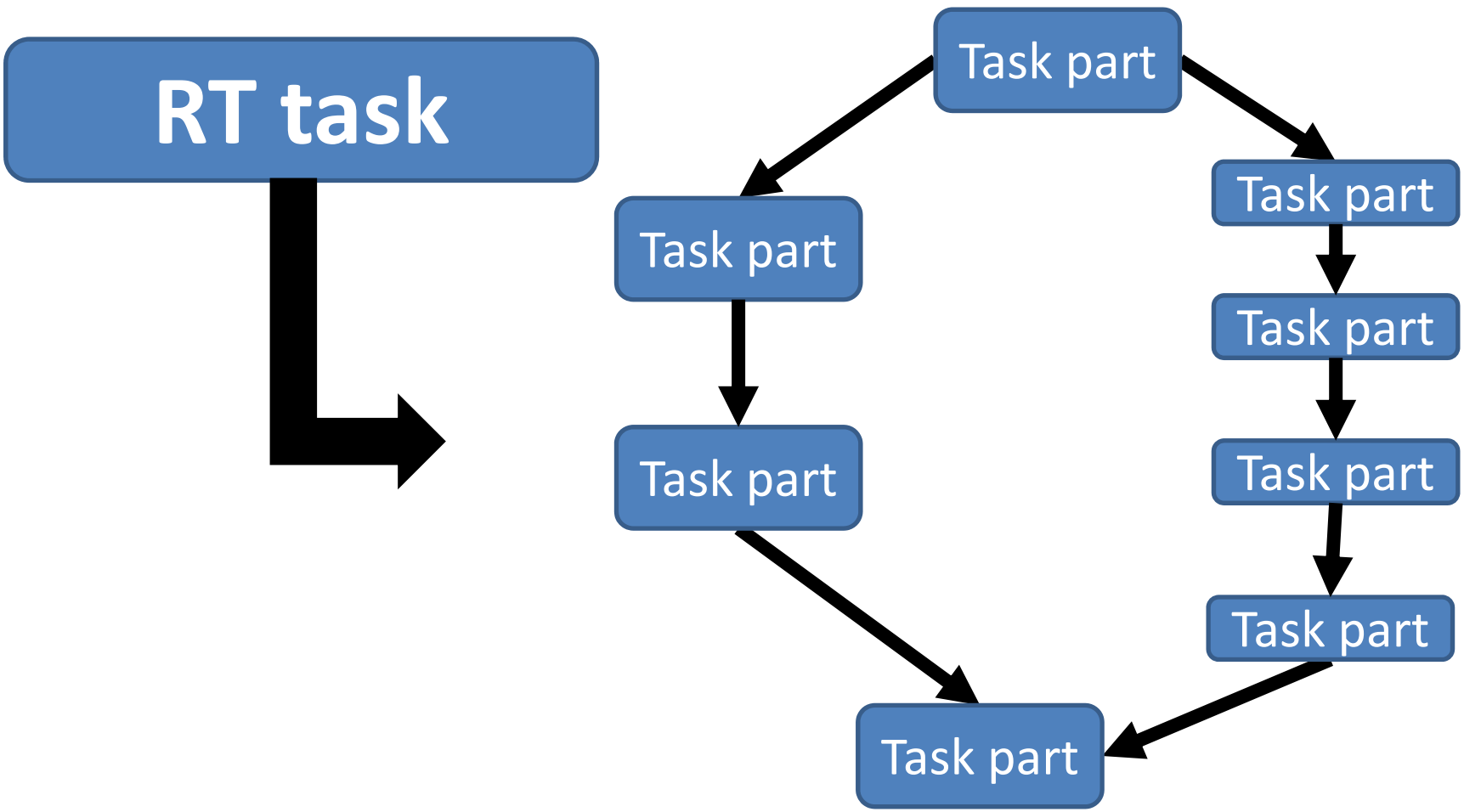
An **openMP task part** (or simply, a **task part**) is a non-pre-emptible (at least from the OpenMP view of the world) portion of an openMP task.

```
#pragma omp task
{
// some code (first task part)
barrier();
// some code (second task part)
}
```

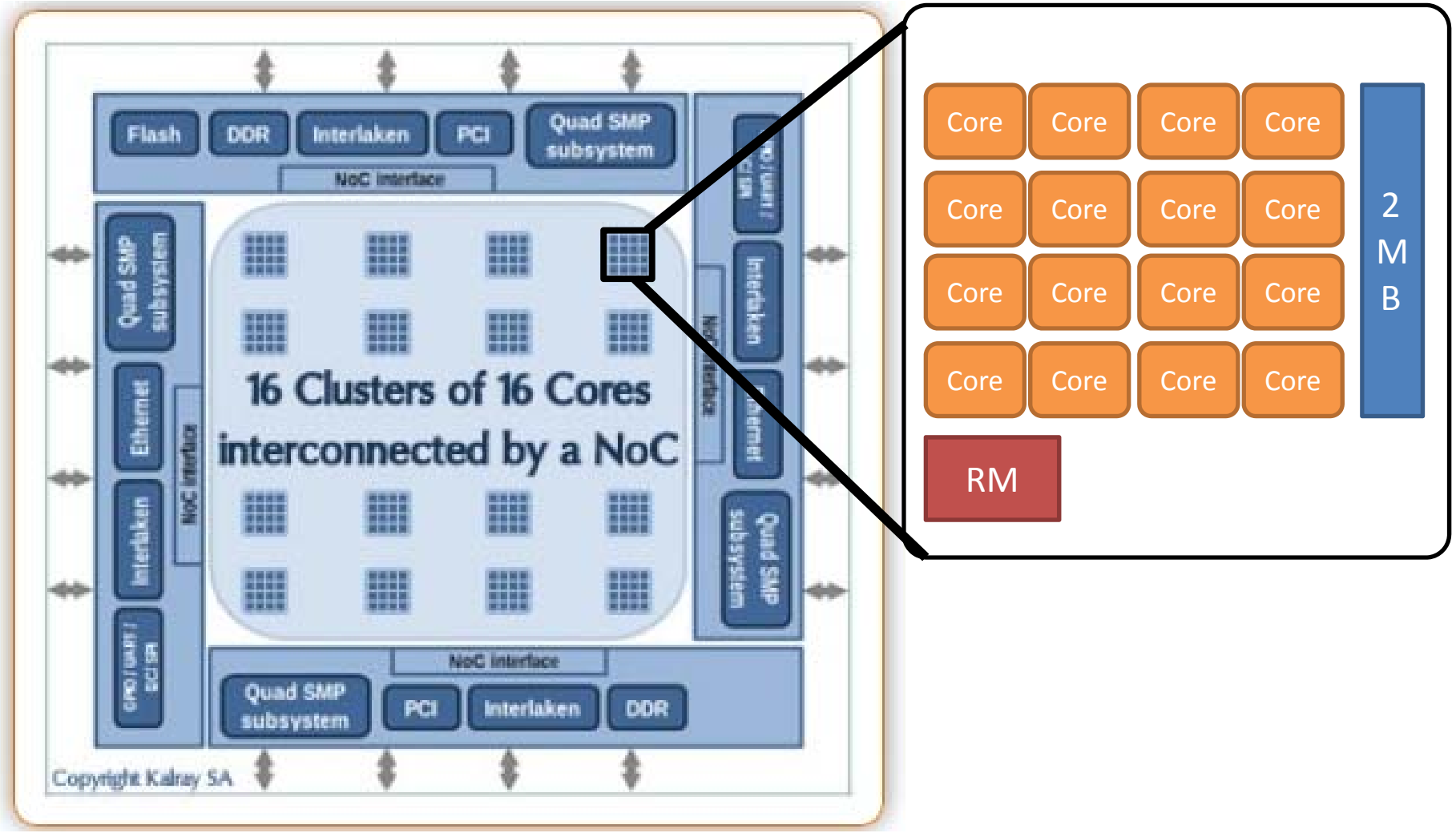
Application model



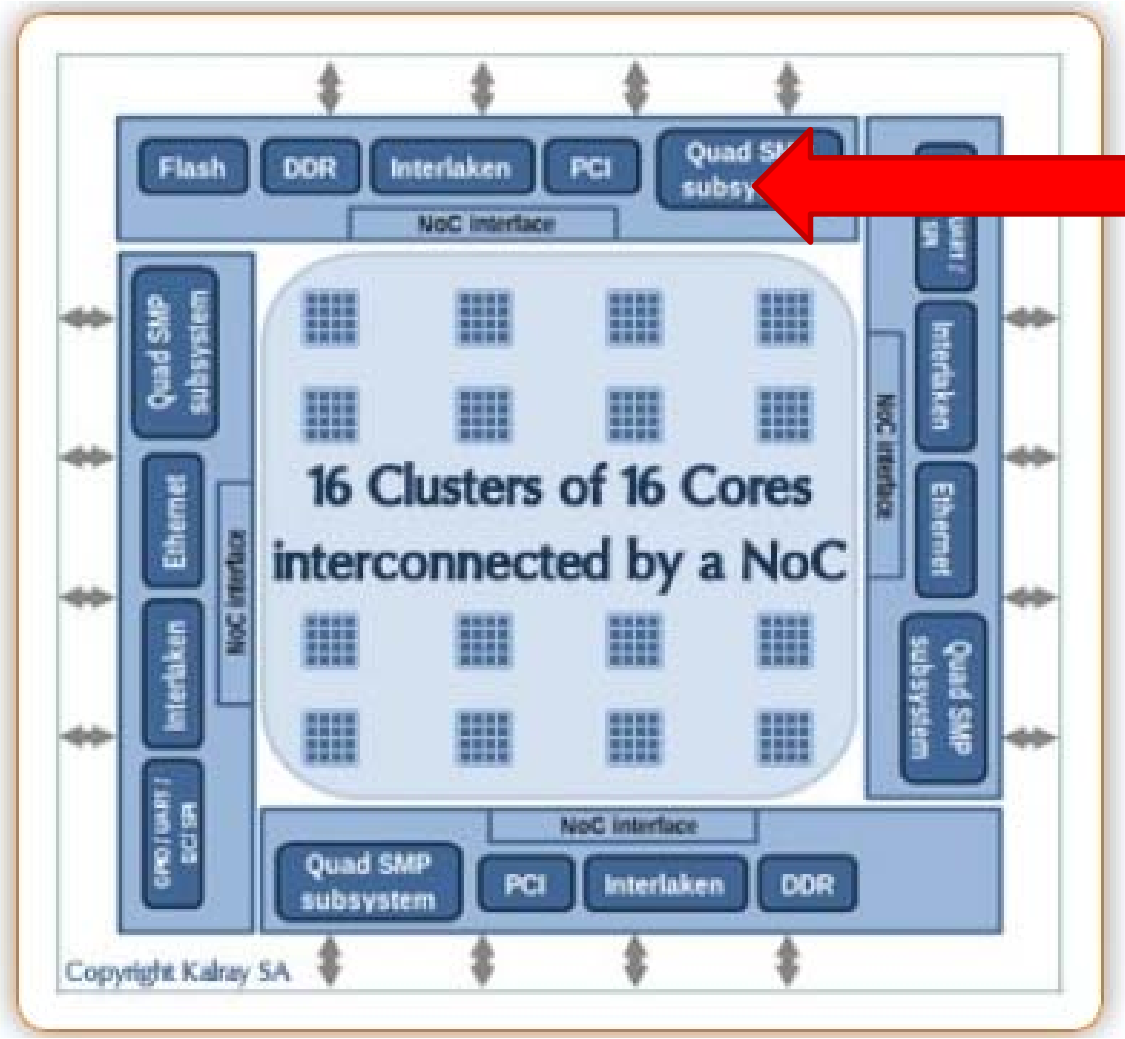
Application model

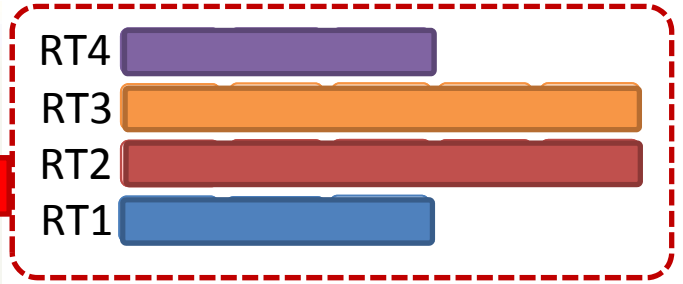


Kalray MPPA architecture



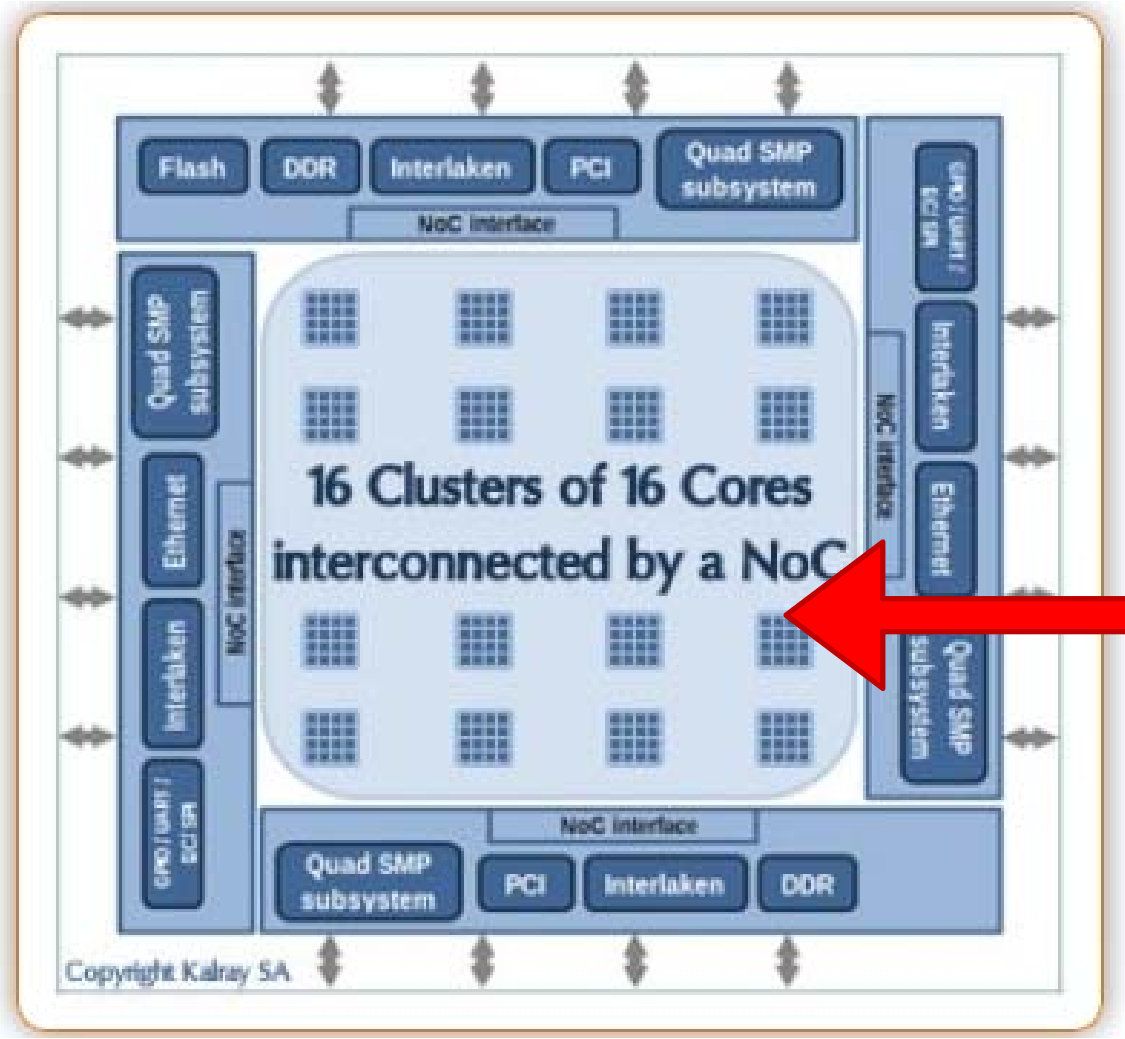
System model

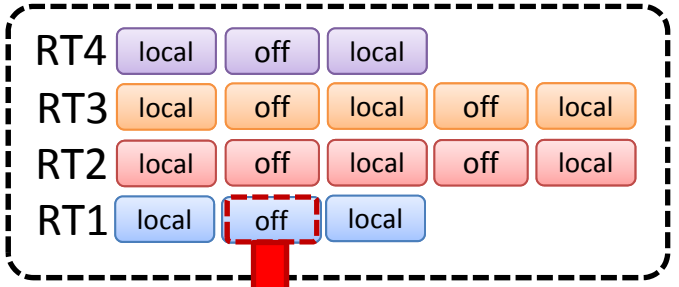




- Static assignment of the RT tasks to the IOS
- No migration between IOS

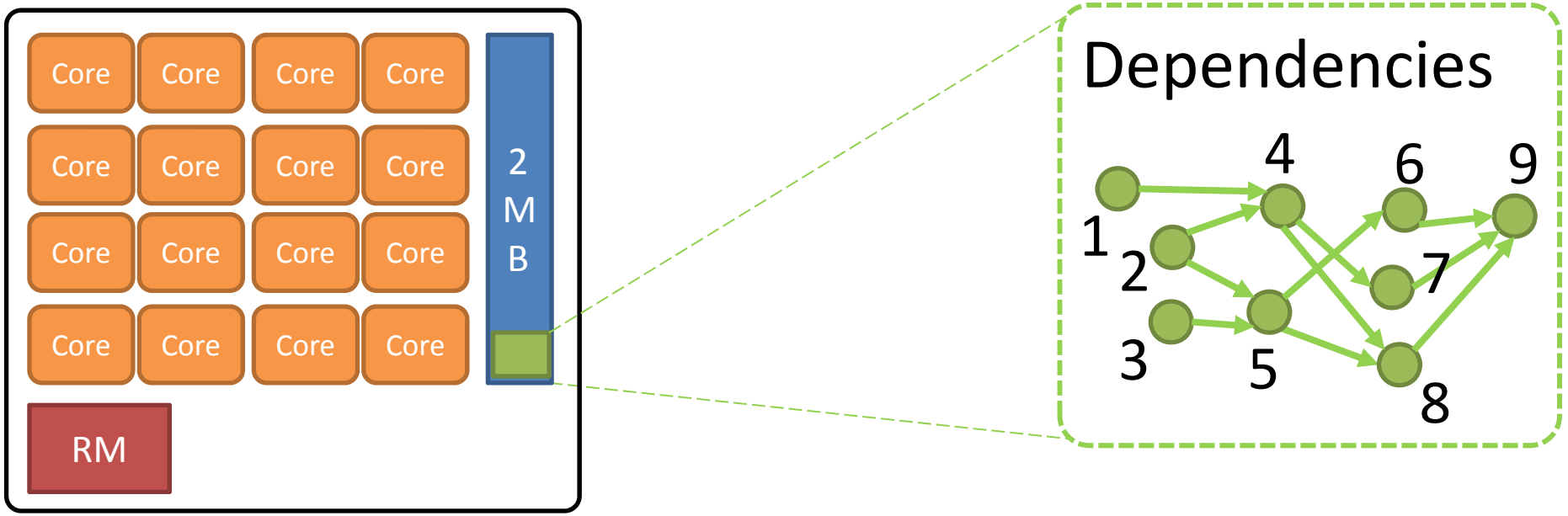
System model





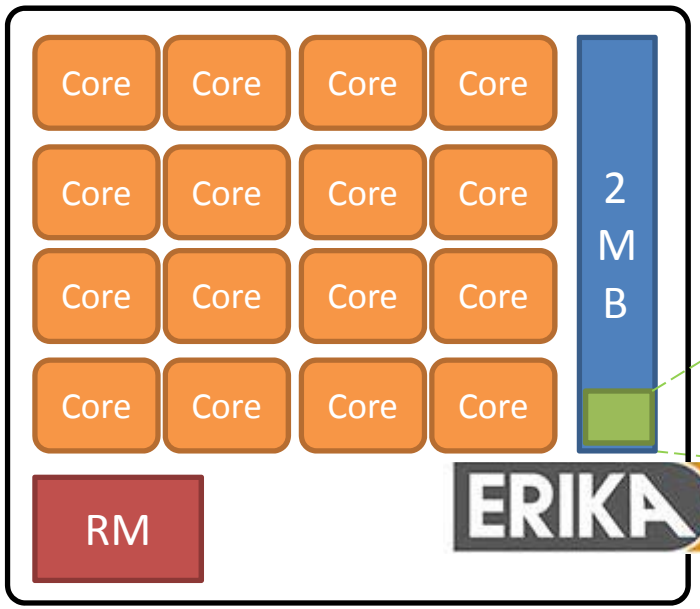
- one task to one cluster so that there is no need for inter-cluster synchronization and it generates less traffic
- Decide on which cluster it executes (first-fit, next-fit, etc.)

System model



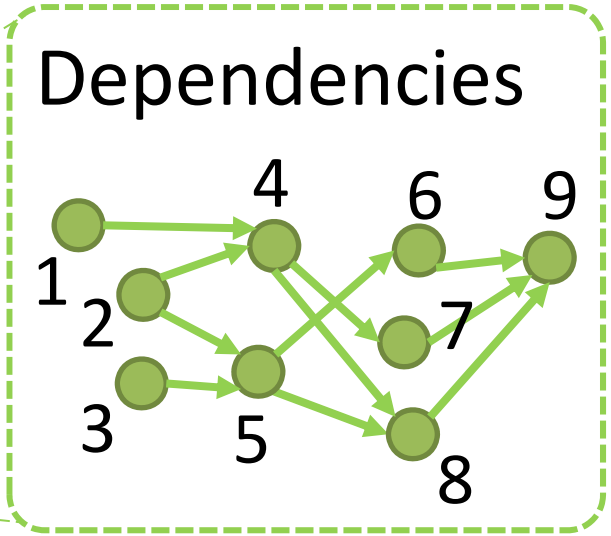
- The essential openMP task dependency information is captured within a streamlined data structure hosted in the on-cluster shared memory

System model

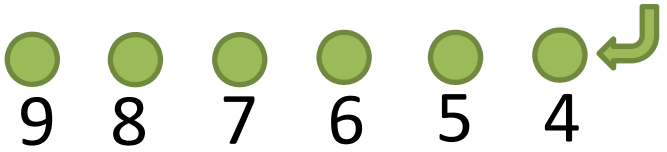


openMP runtime environment:

Libgomp 3.0 extended



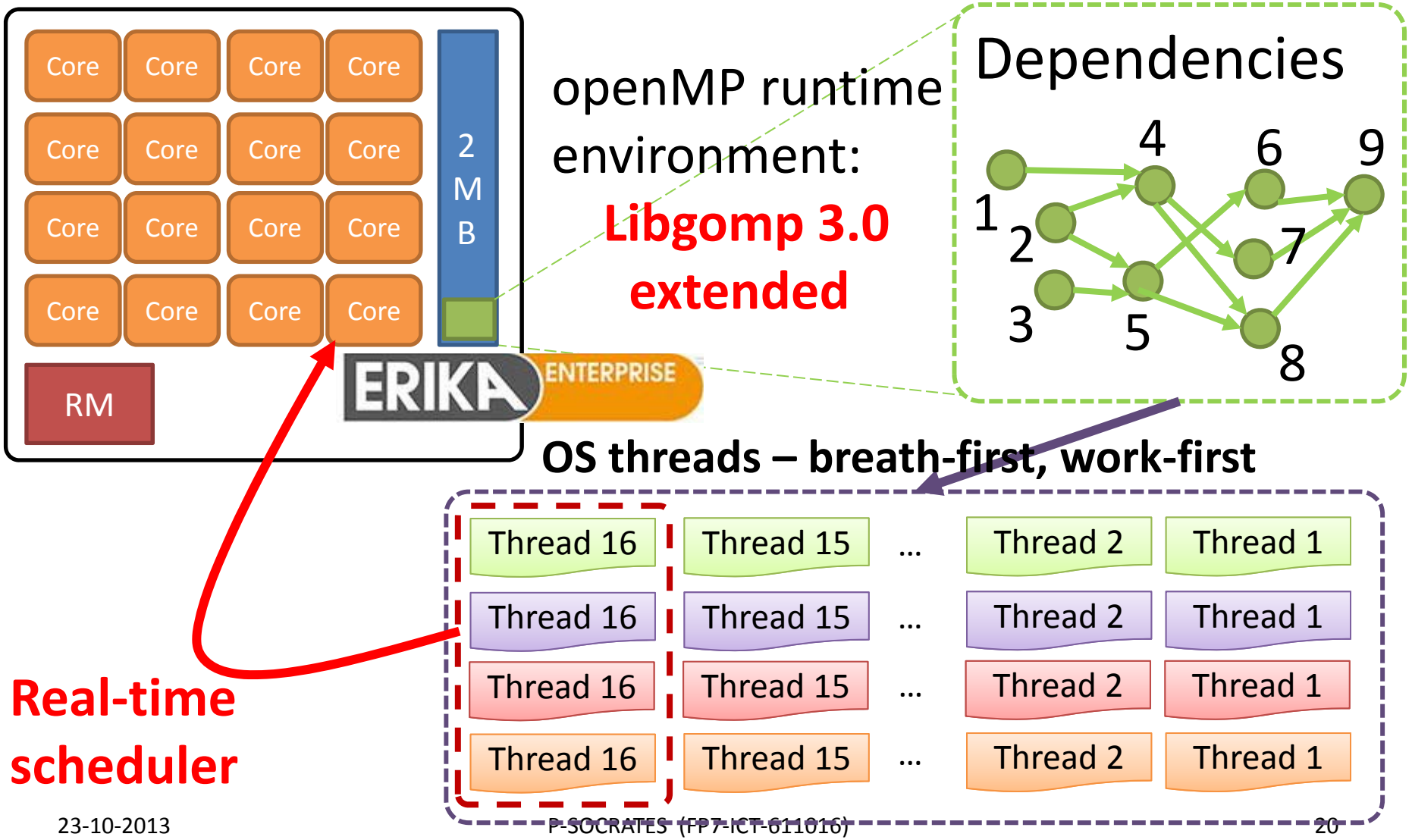
Waiting queue

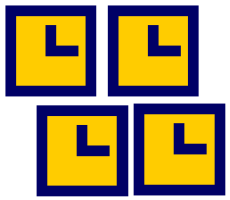


Ready queue

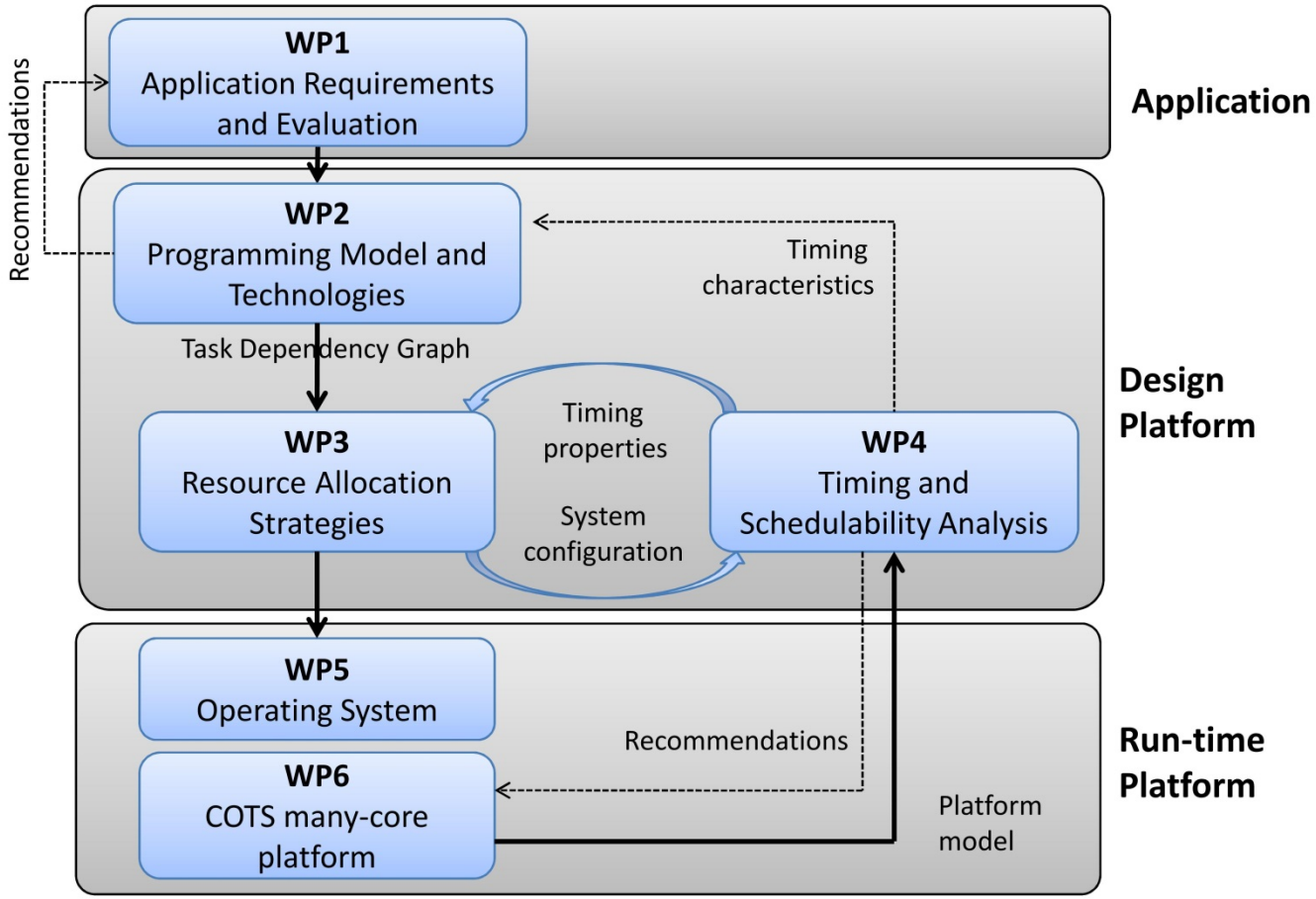
- The cluster also defines and maintains a ready-queue and a waiting-queue for that real-time task.

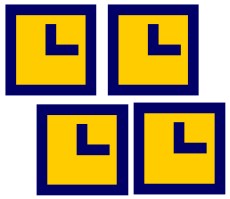
System model





P-SOCRATES


WP1
Application Requirements and Evaluation
Application
Recommendations
WP2
Programming Model and Technologies
Timing characteristics
Task Dependency Graph
Timing properties
System configuration
WP3
Resource Allocation Strategies
WP4
Timing and Schedulability Analysis
Design Platform
WP5
Operating System
Recommendations
WP6
COTS many-core platform
Platform model
Run-time Platform



Discussions – Questions & Answers

Thank you very much for your attention!