



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

Methodology for measurement-based timing analysis

Vincent Nelis, Patrick Meumeu Yomsi,
Luís Miguel Pinho

P-SOCRATES

- **P-SOCRATES: Parallel Software framework for time-CRITICAL mANy-core systemS**
- Three-year FP7 STREP project (Oct-2013, Set-2016)
- **Website:** www.p-socrates.eu
- Budget: 3.6 M€
- Coordinator: Luis Miguel Pinho, CISTER Research Centre, ISEP
- Partners



- **Industrial Advisory Board**



City of Bratislava



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

Use-cases

3 application use-cases

- Online ads selection system
- Data extraction from images
- Complex Event Processing Engine

Need High Computing Performance

- Software Parallelization → openMP 4.0
- Parallel processing (multi-cores) → Kalray MPPA-256
- Optimized hardware configuration

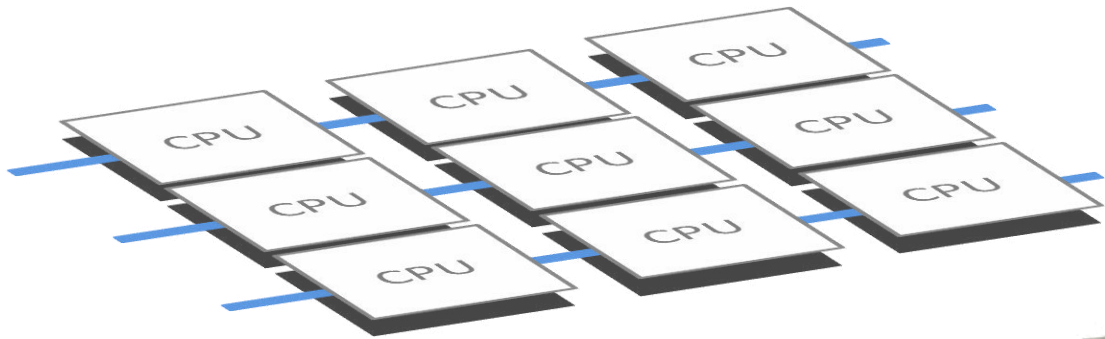
Need for guaranteed performance (not safety-critical)



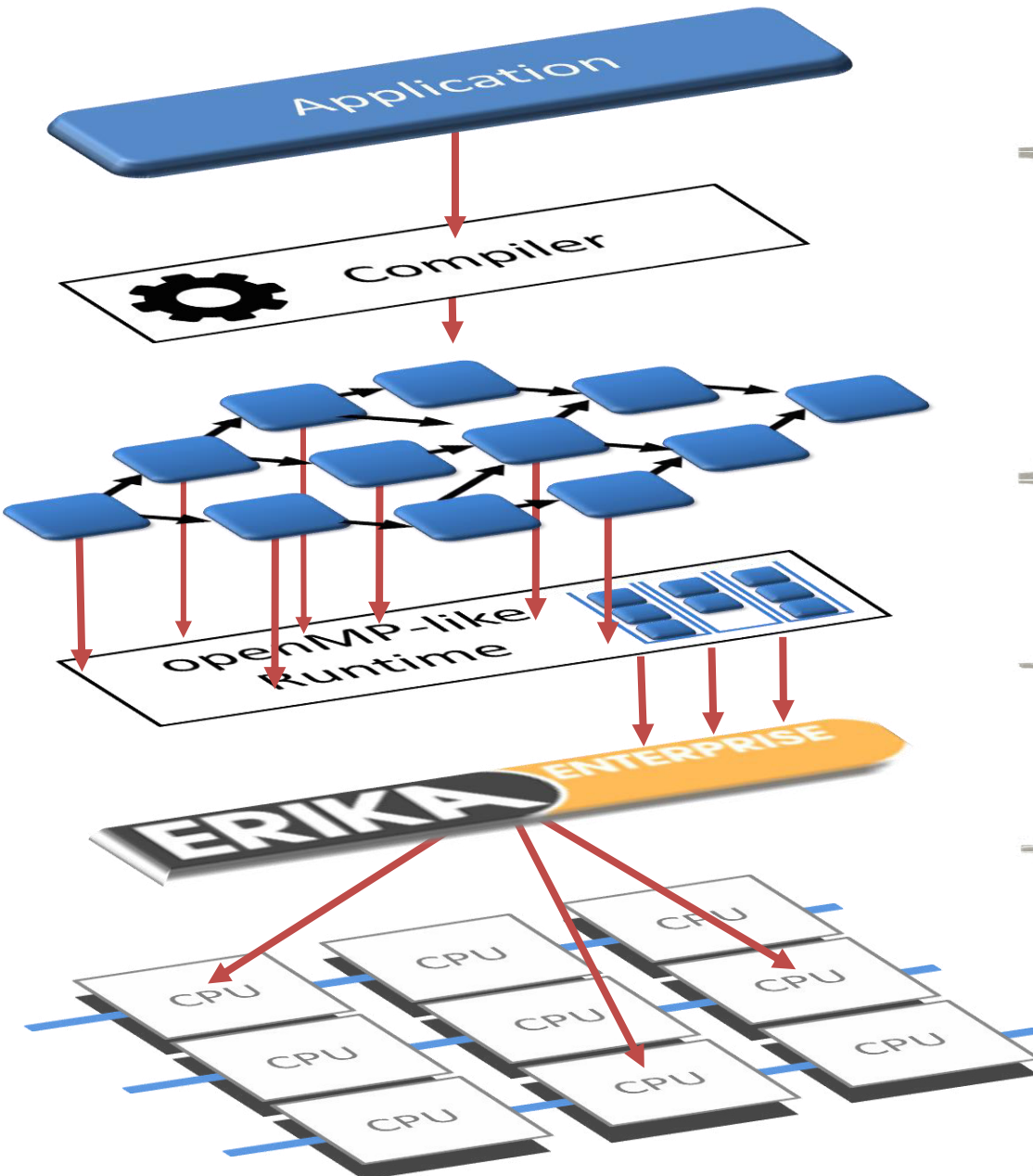


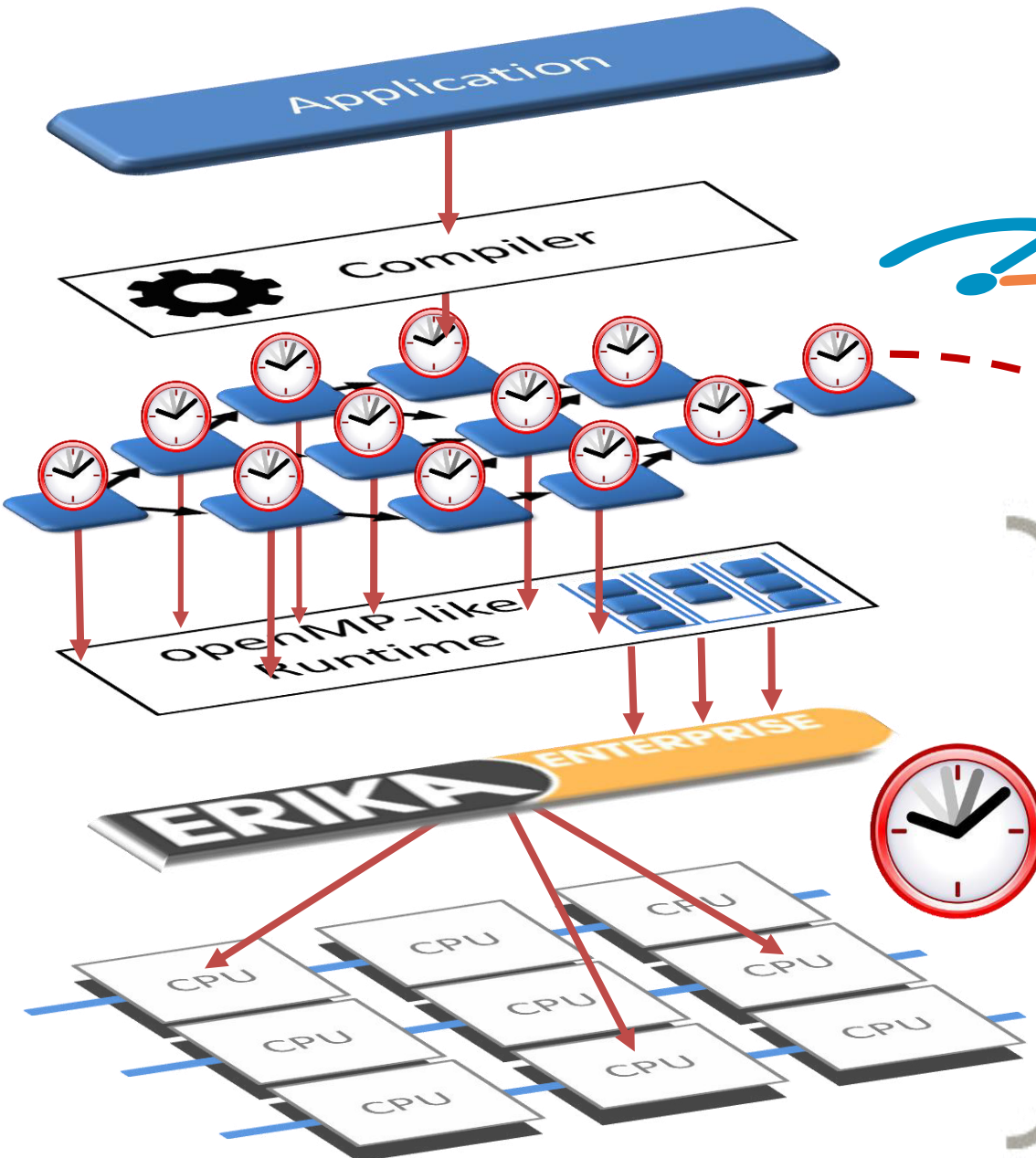
Timing objectives

Performance requirements



KALRAY





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



UNIVERSITÀ DEGLI STUDI
DI MODENA E REGGIO EMILIA

Our approach

Worst input data sets
+
Worst resource availability
=
Worst execution conditions



Our approach

Worst input data sets

+

Best resource availability

→ Understand the importance of the impact of the shared resources



New approach

Not one but two WCET estimates

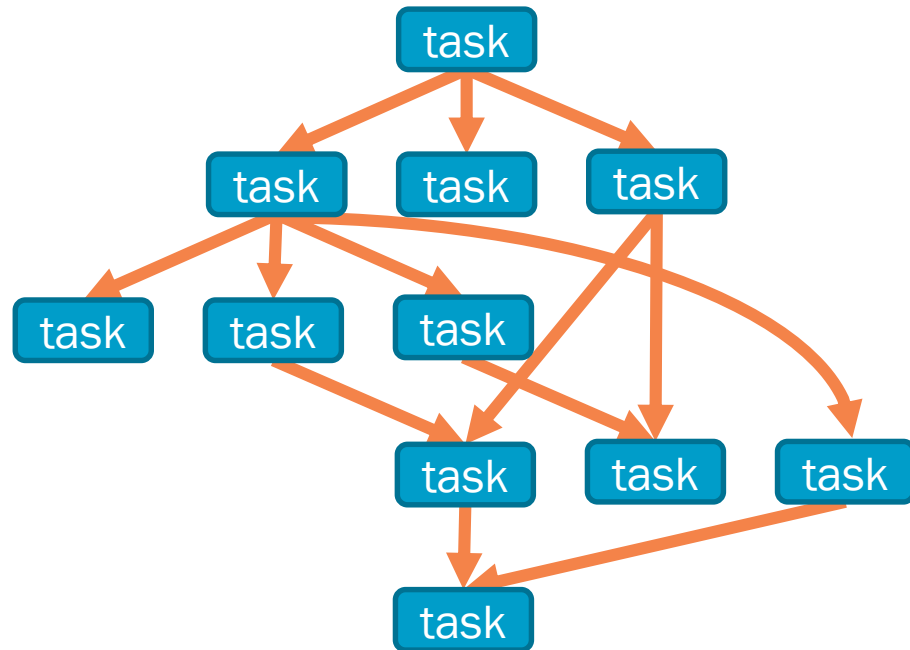
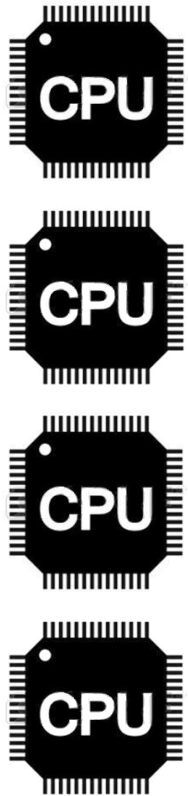
1

Isolation mode

2

Contention mode

Isolation mode

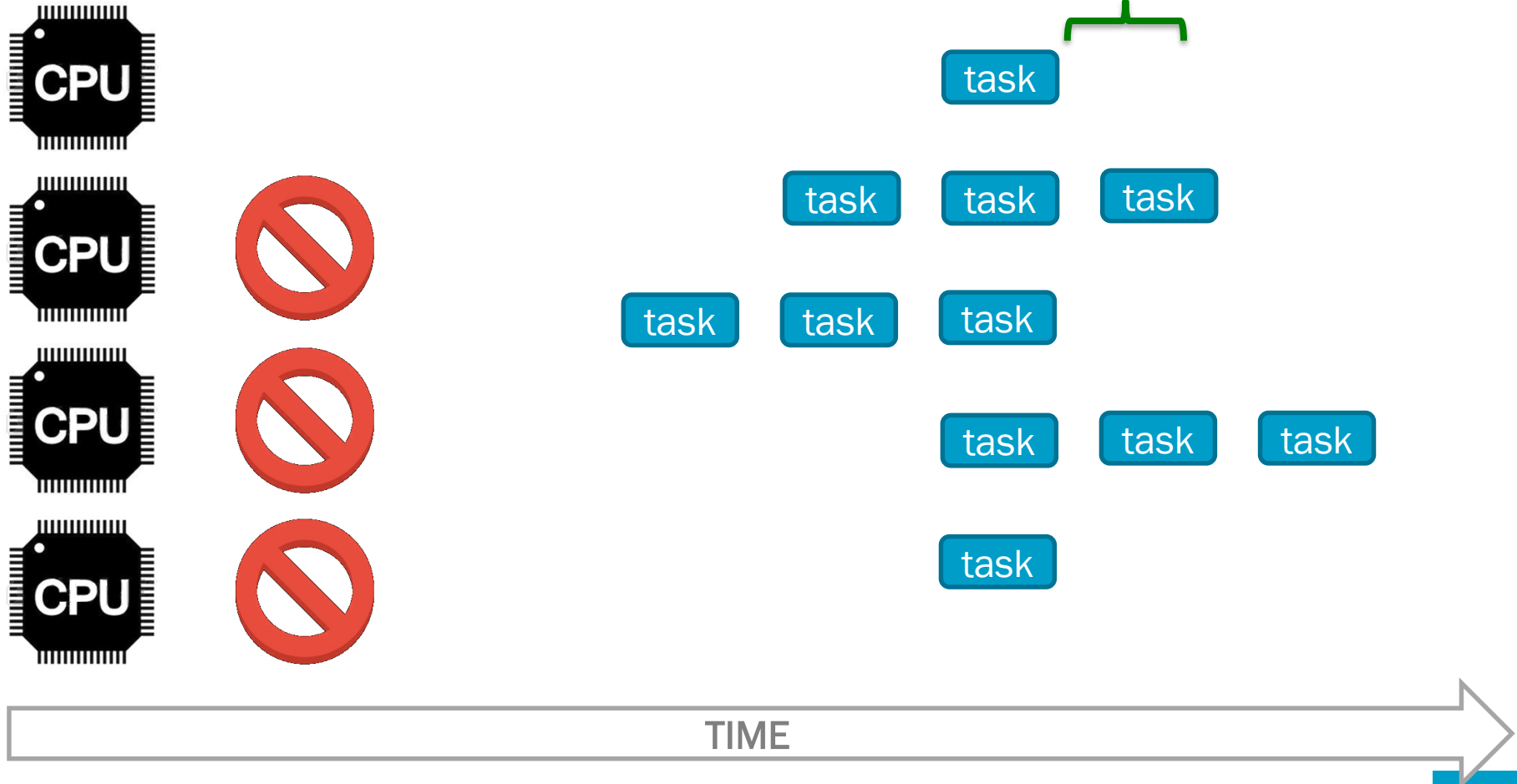


TIME



Isolation mode

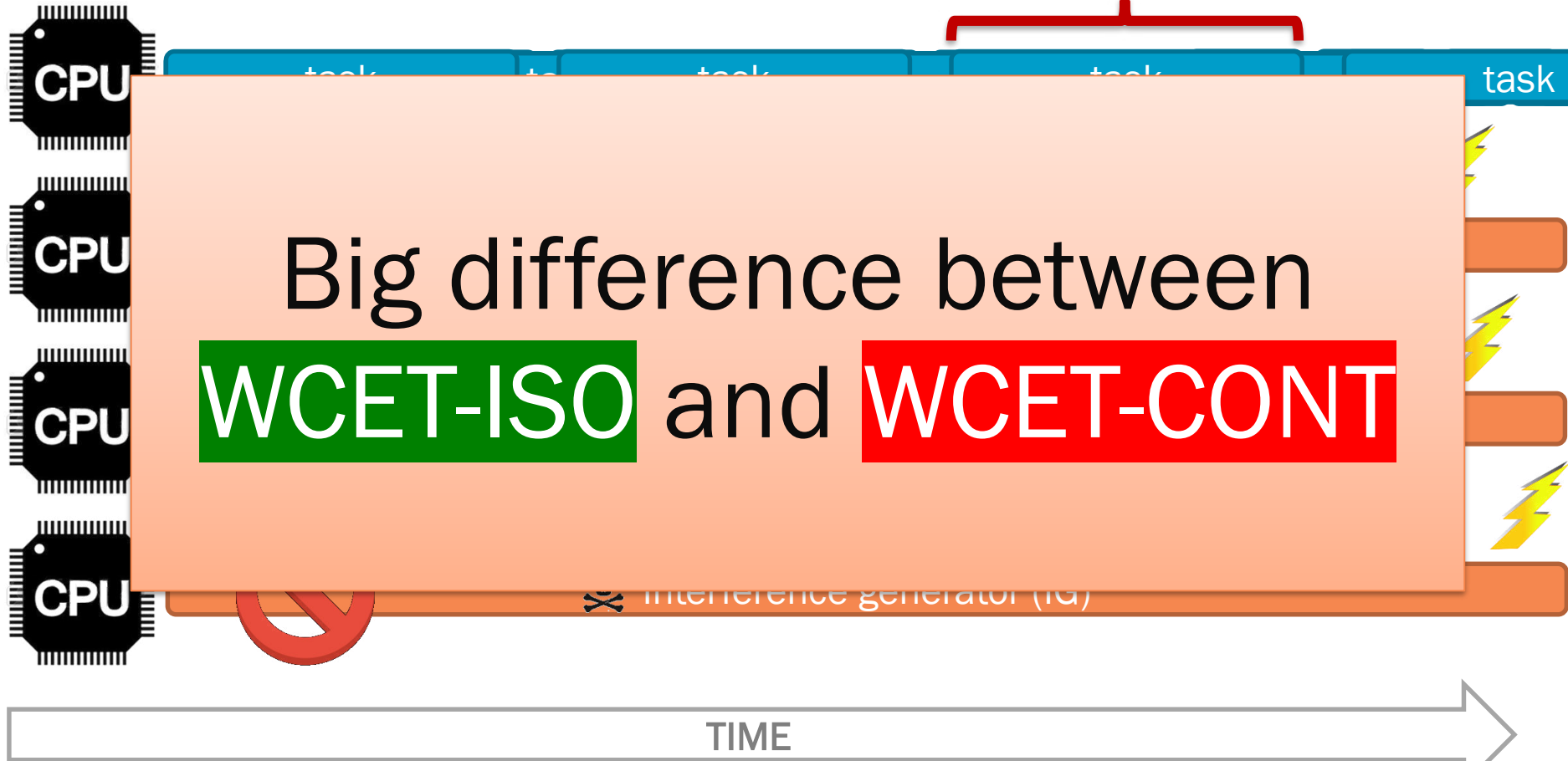
WCET
ISO



Contention mode

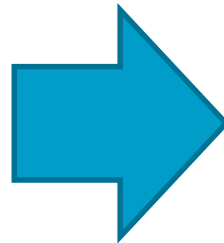
WCET
CONT

Big difference between
WCET-ISO and **WCET-CONT**



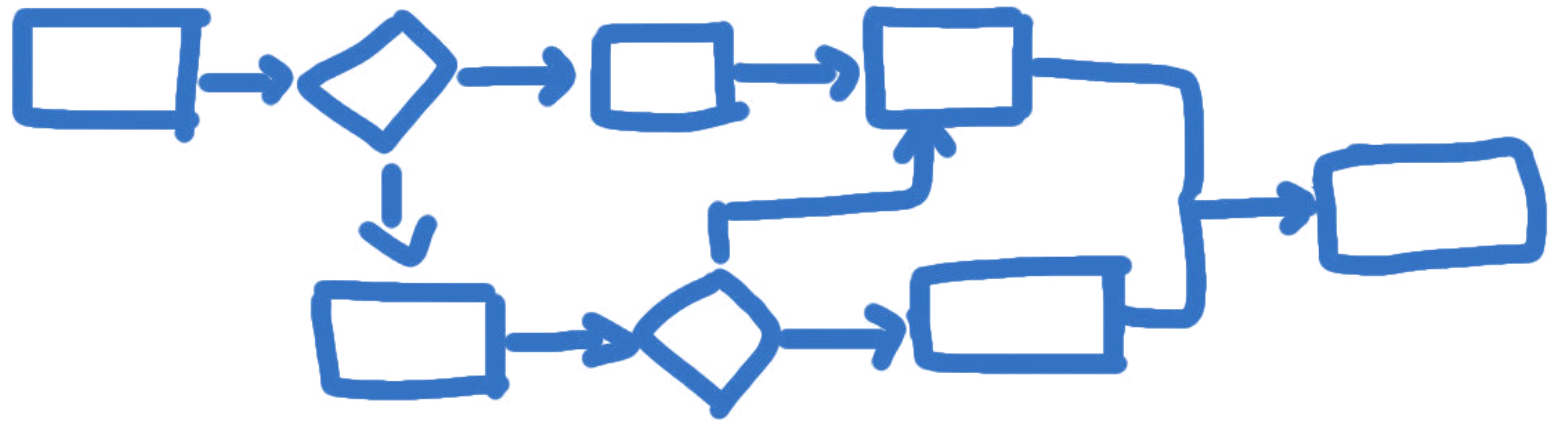
Very easy to use!

```
#pragma omp parallel
{
    #pragma omp master
    {
        (Some initialization code)
        #pragma omp task
        {Some code for the task}
        #pragma omp task
        {Some code for the task}
        (Some more code)
    }
}
```

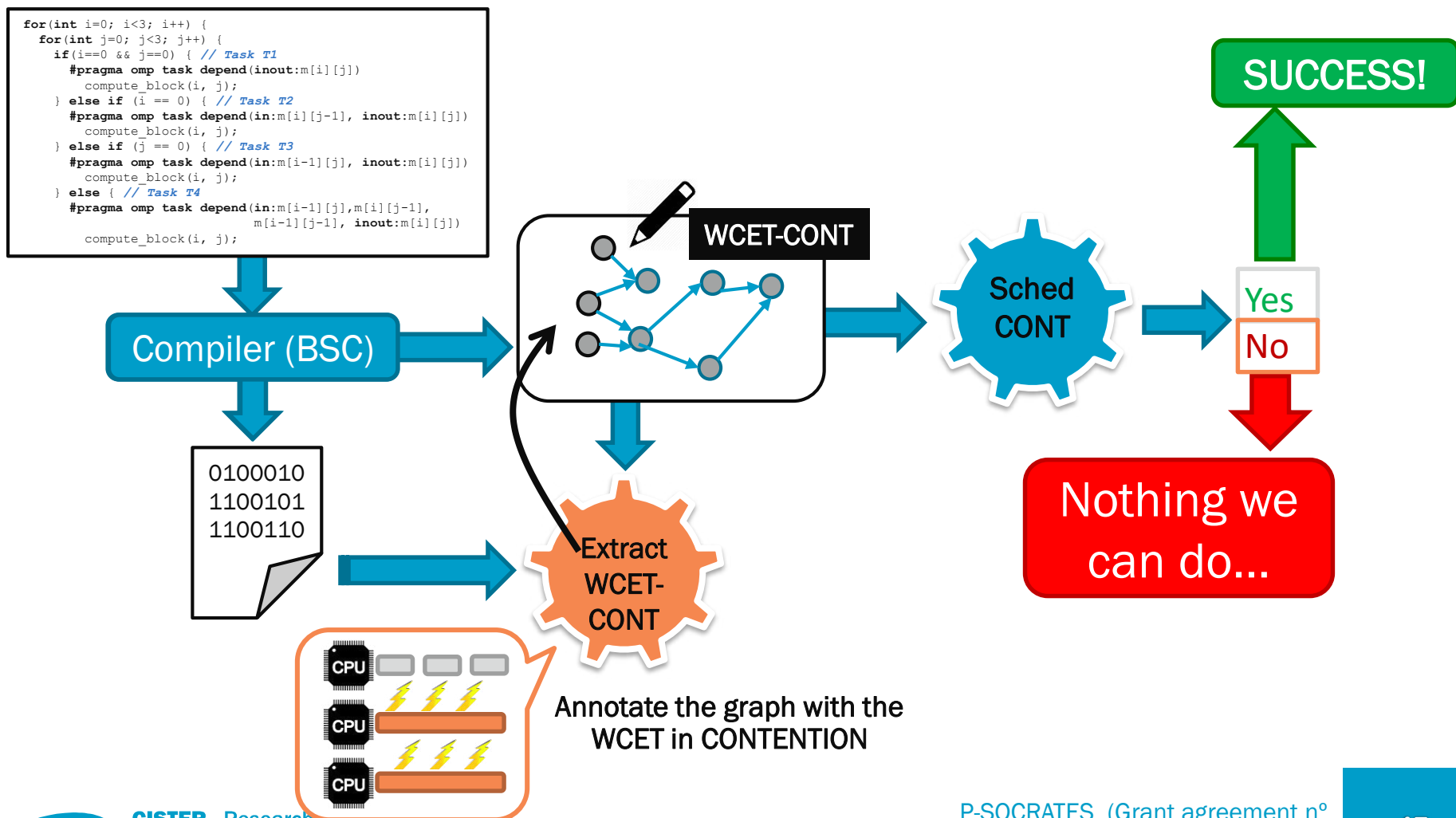


```
#include "timing.h"
TIMING_ANALYSIS_INIT();
#pragma omp parallel
{
    #pragma omp master
    {
        TIMING_BEGIN_OF_OMP_MASTER();
        (Some initialization code)
        #pragma omp task
        {Some code for the task}
        #pragma omp task
        {Some code for the task}
        (Some more code)
        TIMING_END_OF_OMP_MASTER();
    }
    TIMING_END_OF_OMP_PARALLEL();
}
TIMING_ANALYSIS_END();
```

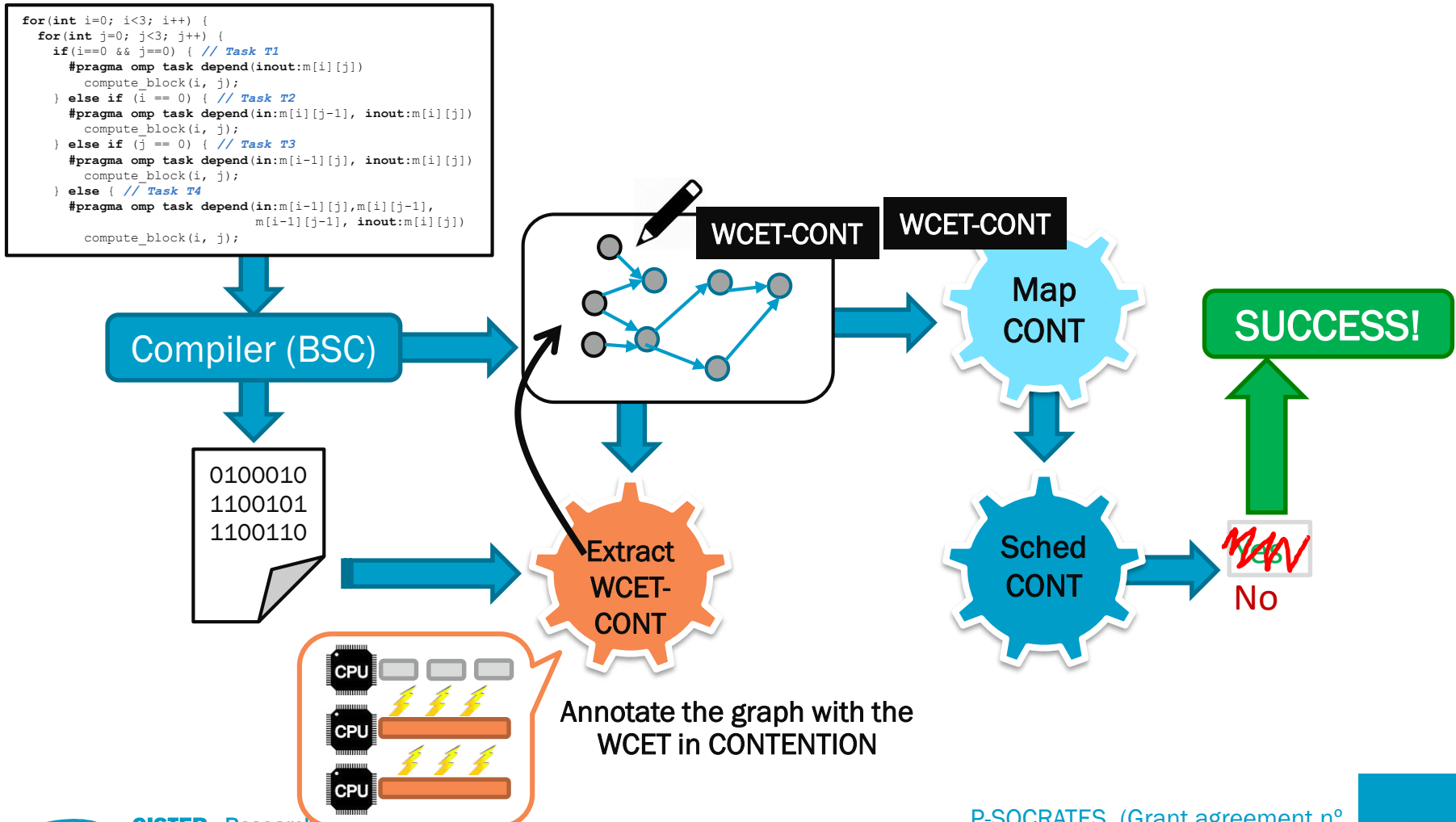
The schedulability analysis flow



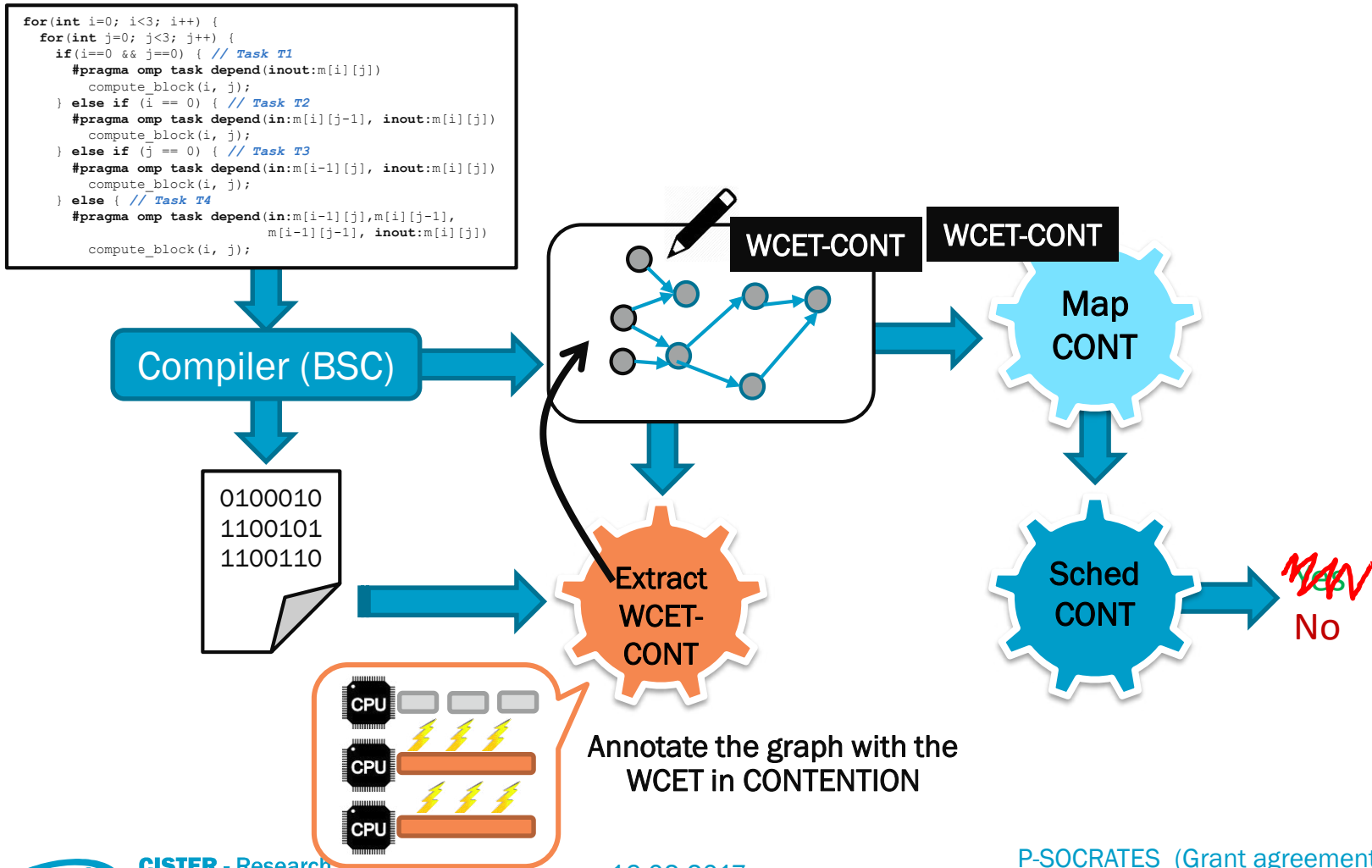
Dynamic scheduling



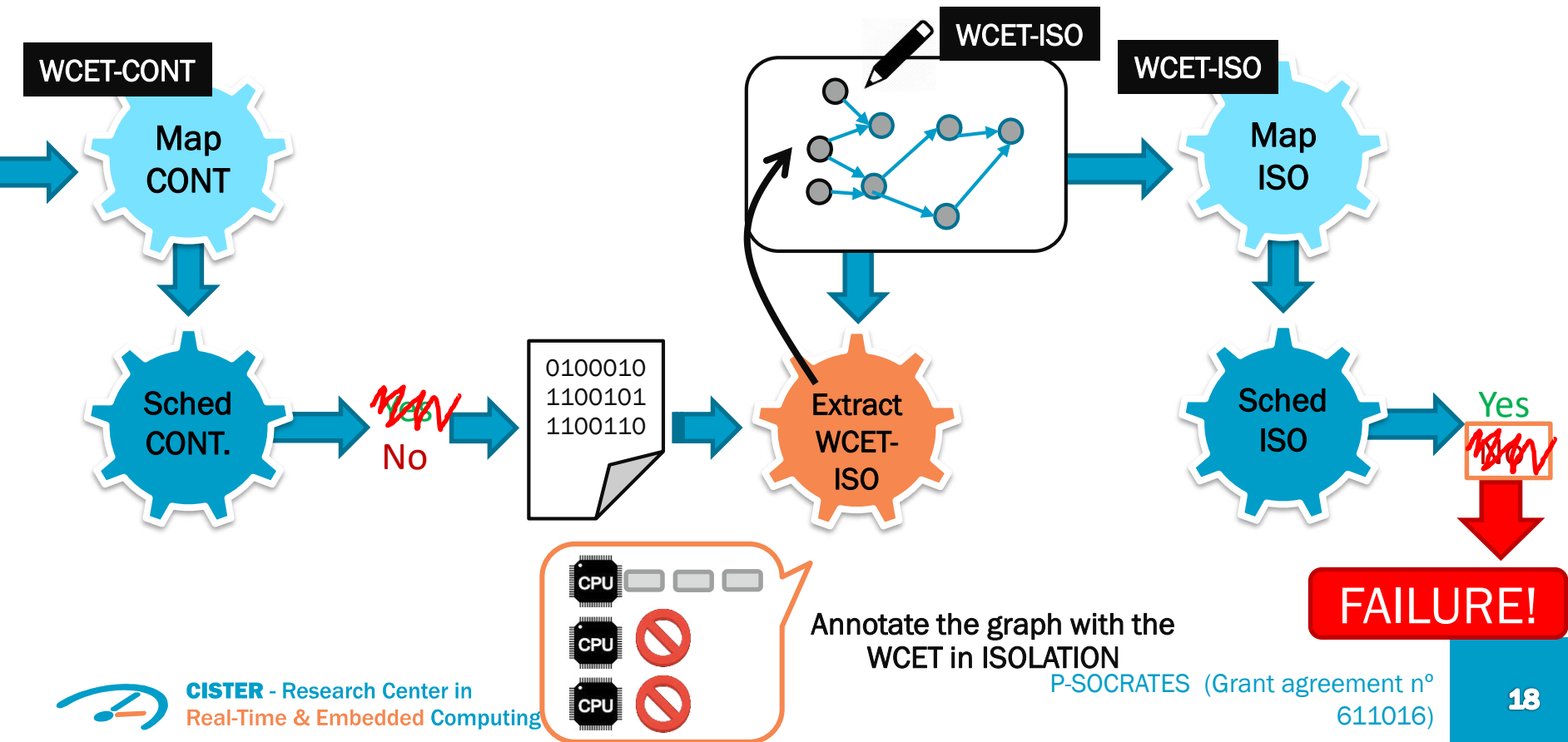
Static scheduling



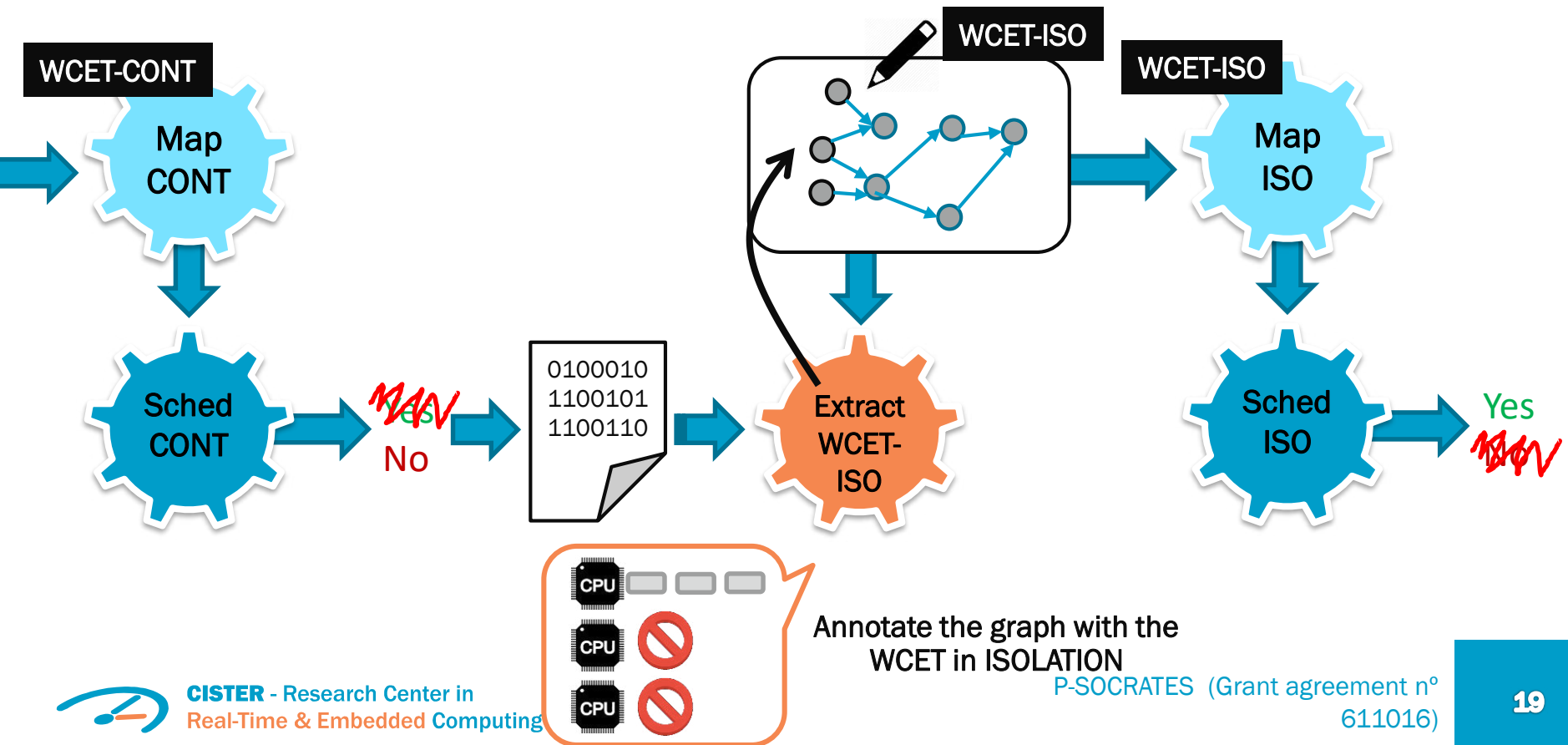
Static scheduling



Static scheduling



Static scheduling



Results

- Two sets of results:
 - On the MPPA Andey
 - One the MPPA Bostan

MPPA Andey

PLATFORM SETTINGS	High-perf	Low-perf
Activate the D-cache	Yes	No
Invalidate the D-cache before running each task	No	-
D-cache: Stall on access	No	Yes
Activate the I-cache	Yes	No
Invalidate the I-cache before running each task	No	-
Memory mapping	Interleaved	Sequential



ESA application: Isolation vs Contention

Isolation	
High-performance	Low-performance
Max = 523 M	Max = 3.071 M
Variation = 377 k (0,07%)	Variation = 1 M (0,03%)

Contention	
High-performance	Low-performance
Max = 527 M	Max = 27.925 M
Variation = 710 k (0,13%)	Variation = 12 M (0,04%)

MPPA Bostan

- We couldn't reproduce the same experiments
- Use of printf() for generating interference
- Disastrous results: slow down of up to 1000x



MPPA Bostan

- The 3 application use-cases have been tested
- Overall, results seemed to confirm that static mapping and scheduling leads to **less performance at runtime** but **more accurate analysis** (the blame is on the schedulability analysis)
 - Over-estimation of carry-in and carry-out
 - New results on the way (RTSS?)



Go for pWCETs

Investigate statistical approaches (Already fully compatible with DiagXTrem)

ISO or **CONT**

5435 ms
5678 ms
4876 ms
3874 ms
7624 ms
...

Stationarity
Stationarity of the extreme
Independence
Independence of the extreme
...

Apply the EVT

WCET - ISO
or **WCET - CONT**

Thank you



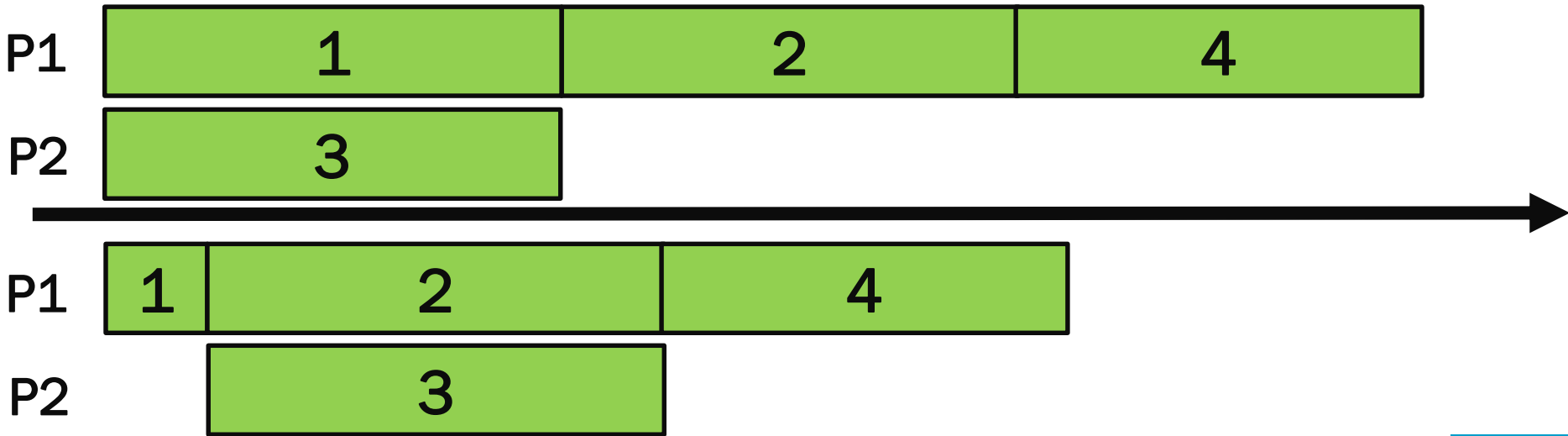
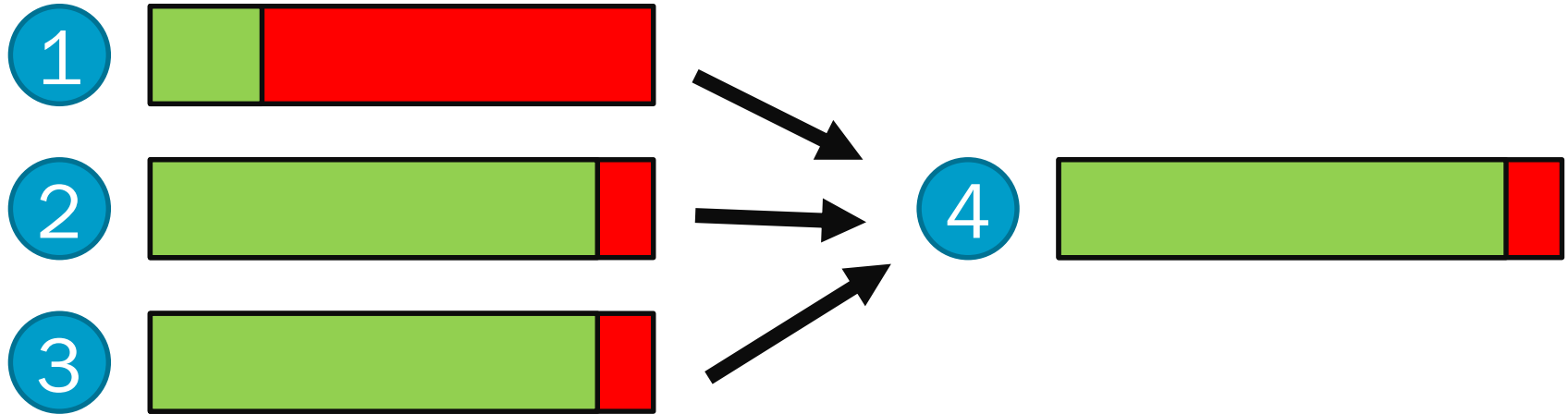
Open research problems



New scheduling problems

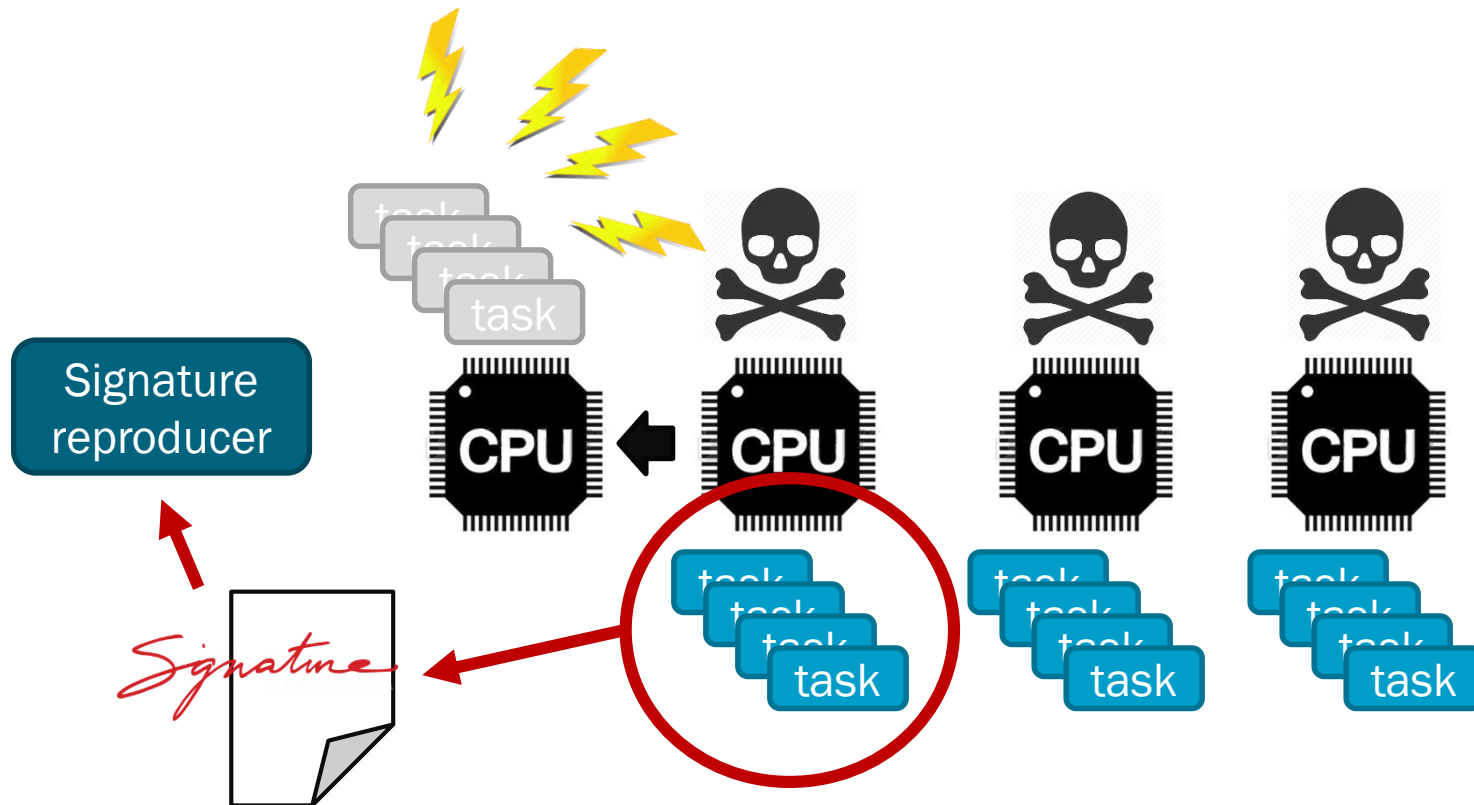
- Tasks have two WCET estimates:
 - One estimate when they execute on one core and all the other cores are kept idle
 - One estimate when they execute concurrently with other tasks
- Adding scheduling restrictions at runtime may increase the observed response time but decrease its estimates, hence offering better guarantees.

New scheduling problems



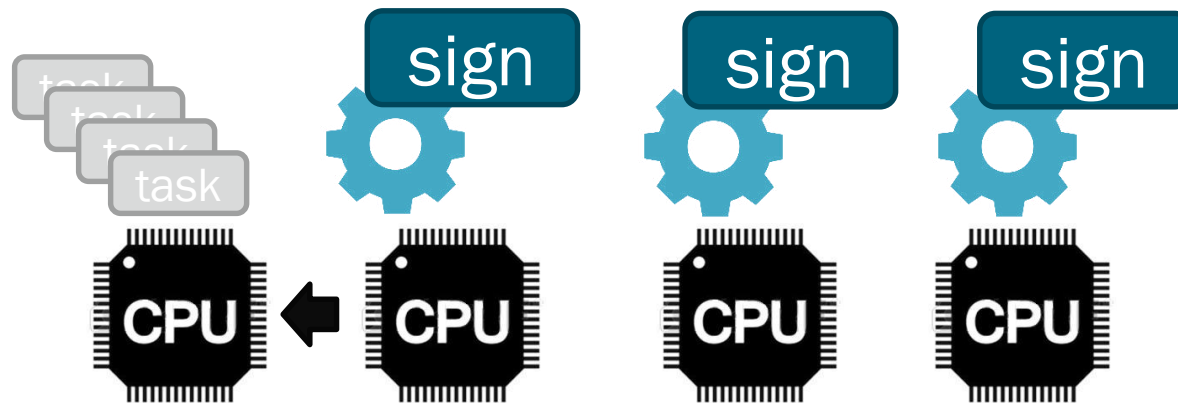
Reduce pessimism of WCET-CONT

Reduce the pessimism of the CONTENTION mode



Reduce pessimism of WCET-CONT

Reduce the pessimism of the CONTENTION mode



Use the signature instead of the  to simulate the environment post-integration

Intrusiveness of the code

The code used for timing the application cannot simply be removed (change the memory layout and thus the cache behavior)

- Option 1: Just leave it there
- Option 2: Replace the code with “NOP”

