CISTER - Research Center in Real-Time & Embedded Comp

Methodology for measurement-based timing analysis

Vincent Nelis, Patrick Meumeu Yomsi,
Luís Miguel Pinho

P-SOCRATES

- **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, Set-2016)
- **Website:** www.p-socrates.eu
- Budget: **3.6 M€**
- Coordinator: **Luis Miguel Pinho, CISTER Research Centre, ISEP**
- Partners

CISTER - Rese Real-Time & E

Barcelona Supercomputing Center Centro Nacional de Supercomputación

Università degli Studi di Modena e Reggio Emilia

ETH zürich

EVIDENCE EMBEDDING TECHNOLOGY

AT

Atos

- Industrial Advisory Board

BOSCH

AIRBUS

SAAB

Honeywell

KALRAY

IBM

AIRBUS DEFENCE & SPACE

City of Bratislava

EXPERT SYSTEM SEMANTIC INTELLIGENCE

RAPITA SYSTEMS LTD

MBDA

Use-cases

3 application use-cases

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

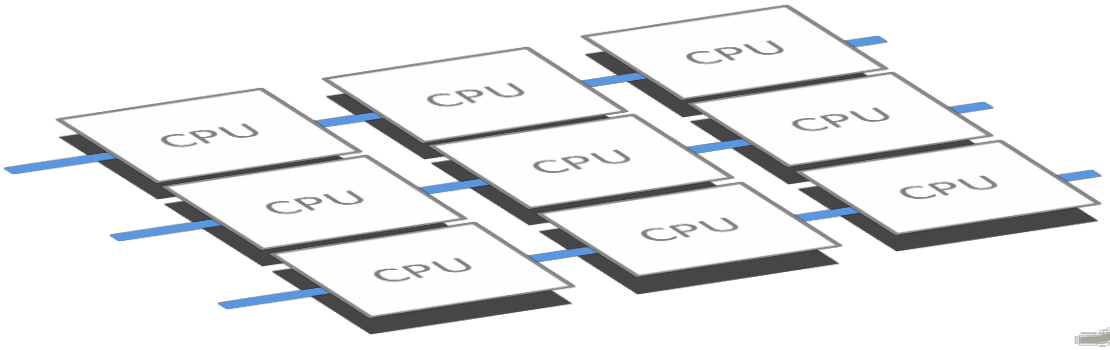
Need High Computing Performance

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

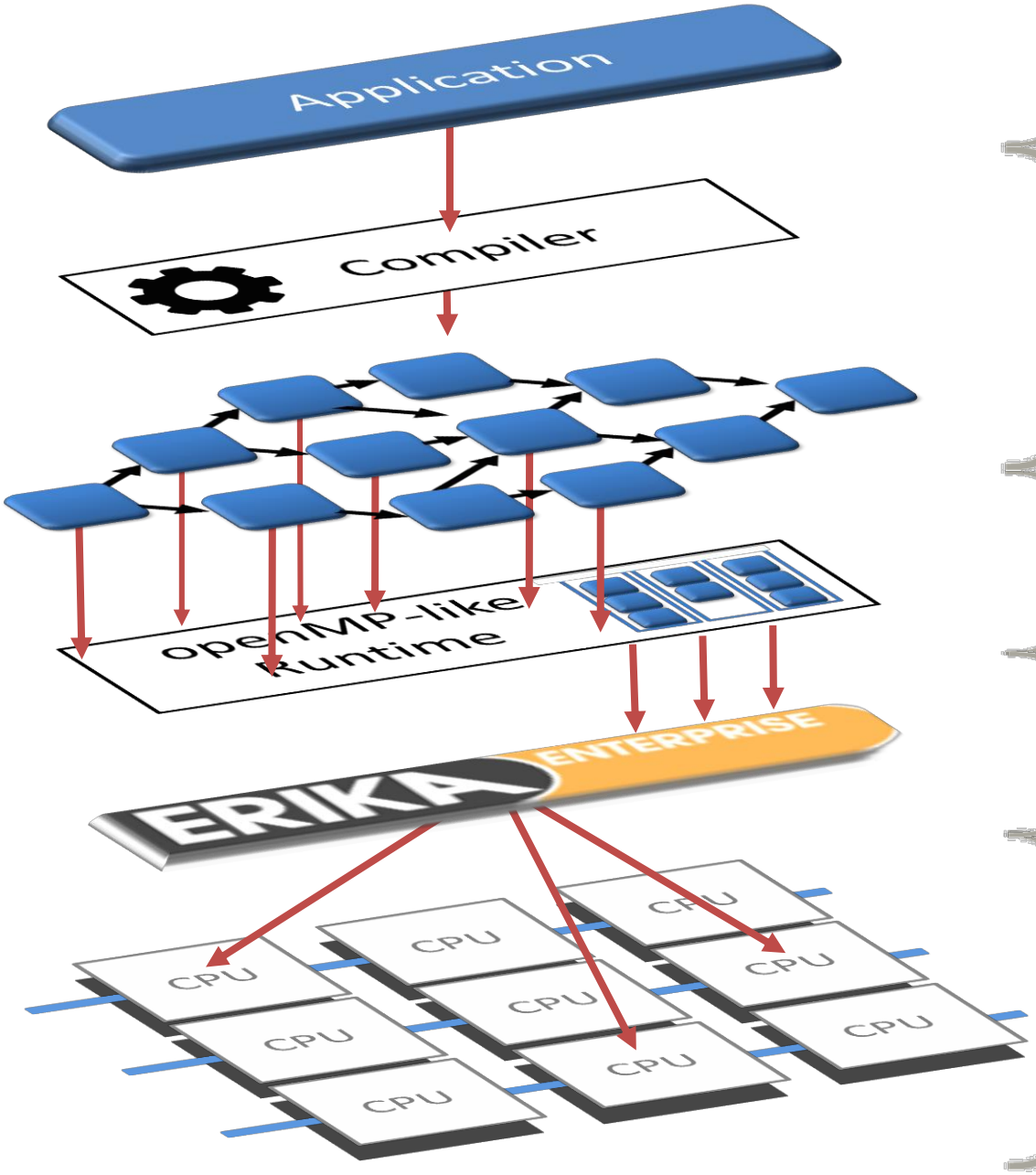
Need for guaranteed performance (not safety-critical)

EXPERT SYSTEM
SEMANTIC INTELLIGENCE
Atos
esa

- Timing objectives
- Performance requirements



KALRAY



EXPERT SYSTEM
SEMANTIC INTELLIGENCE
Atos
esa

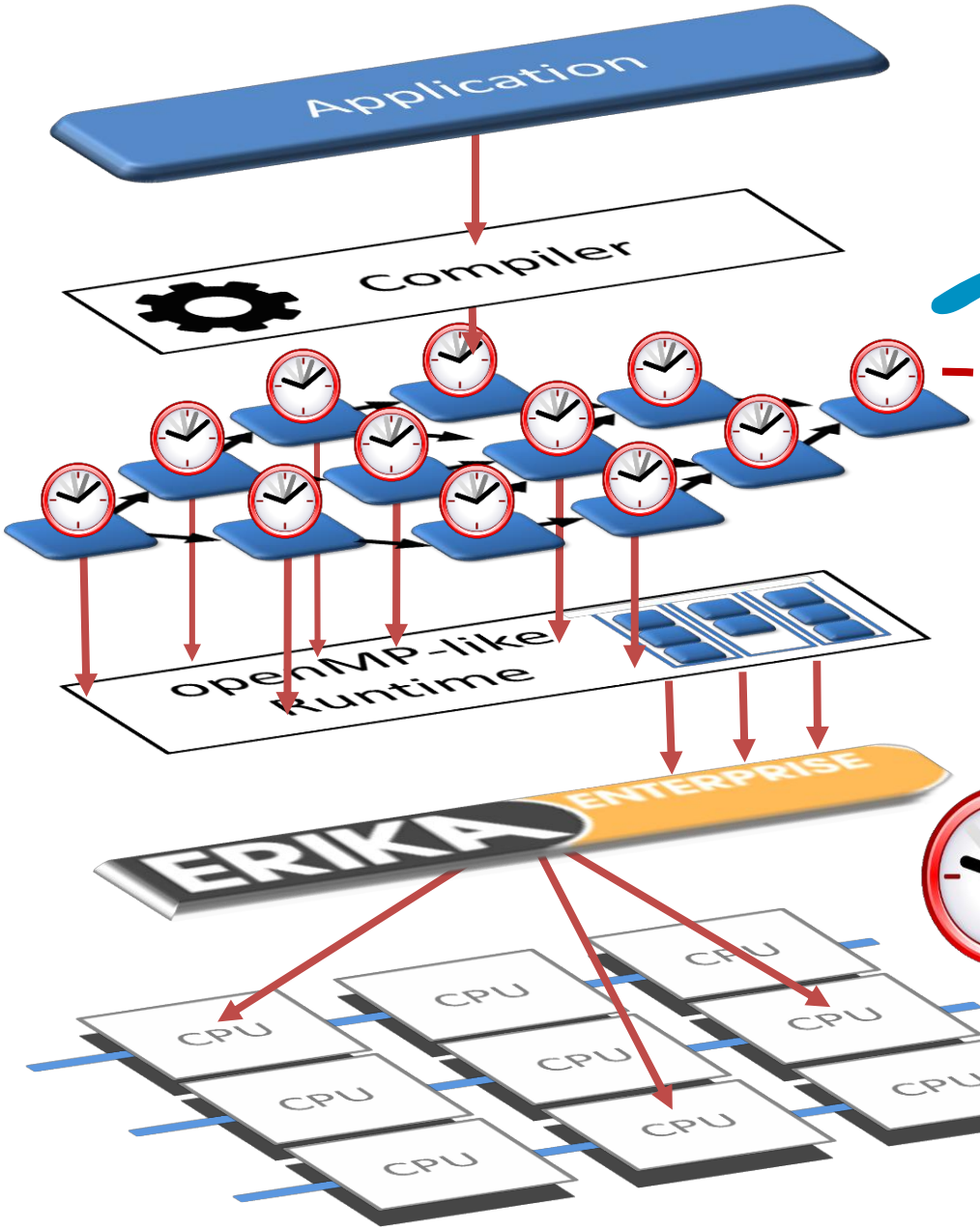
BSC
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

ETH zürich

EVIDENCE®
EMBEDDING TECHNOLOGY

KALRAY

AT



CISTER - Rese
Real-Time & E

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

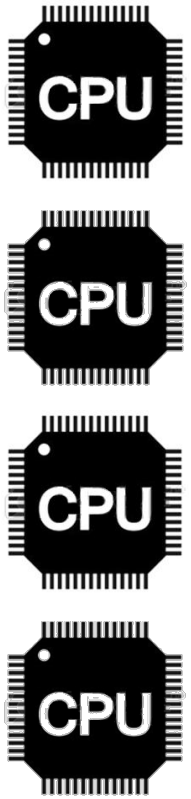
New approach

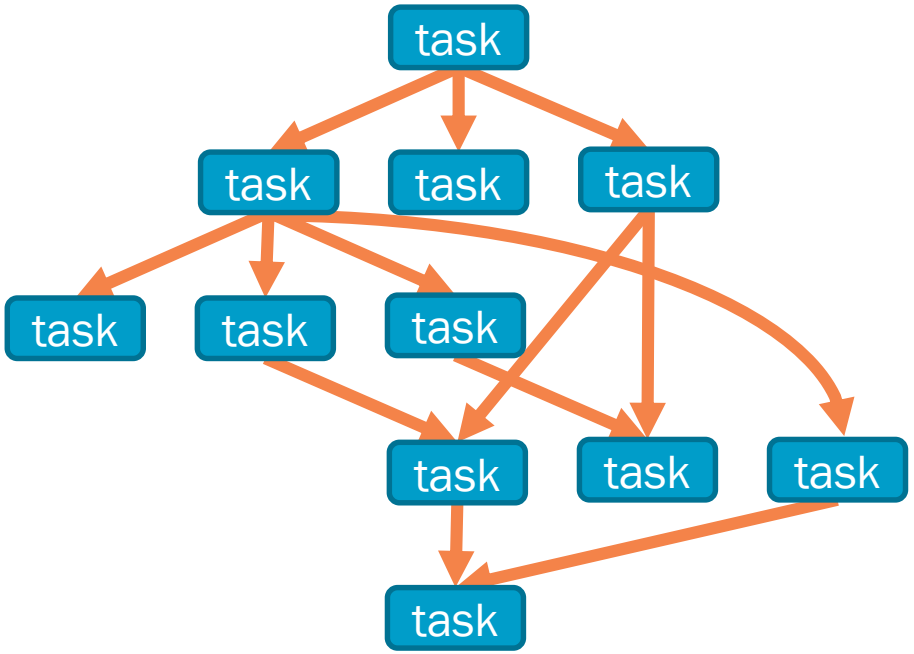
Not one but two WCET estimates

Isolation mode

Contention mode

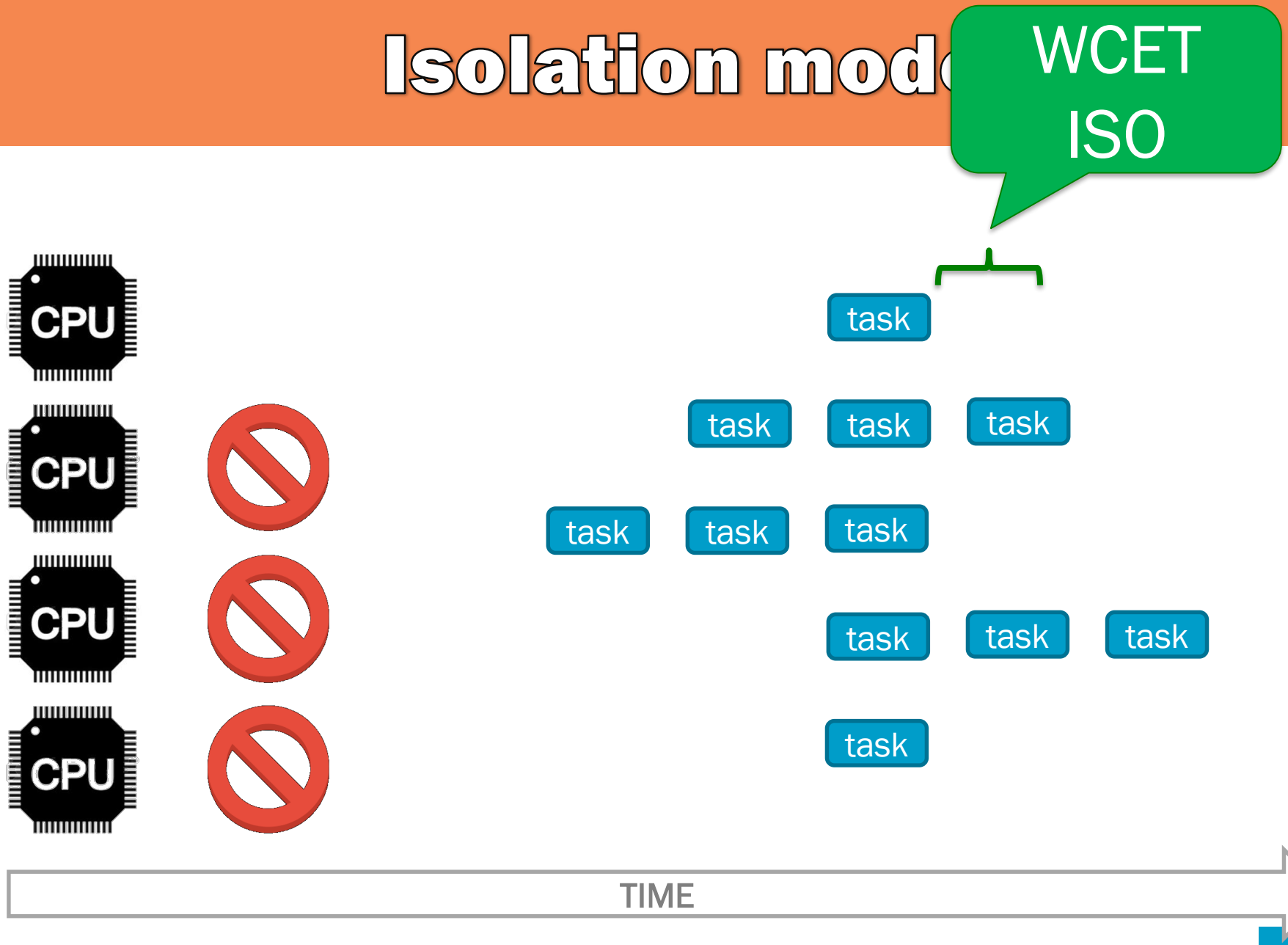
Isolation mode



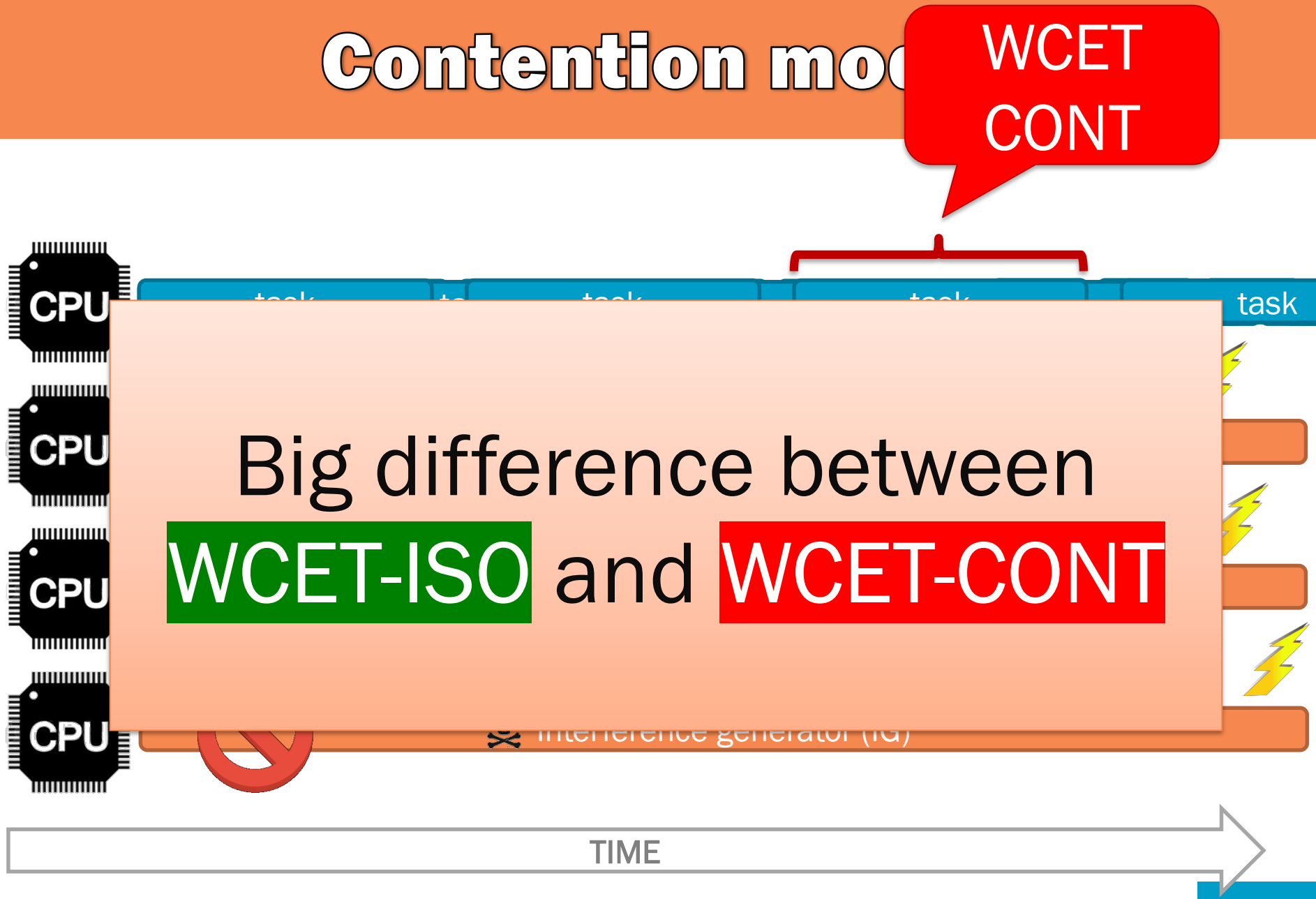


TIME

Isolation mod



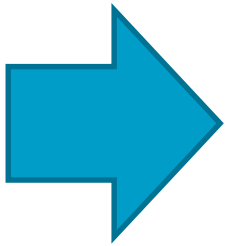
Contention mo



Big difference between WCET-ISO and WCET-CONT

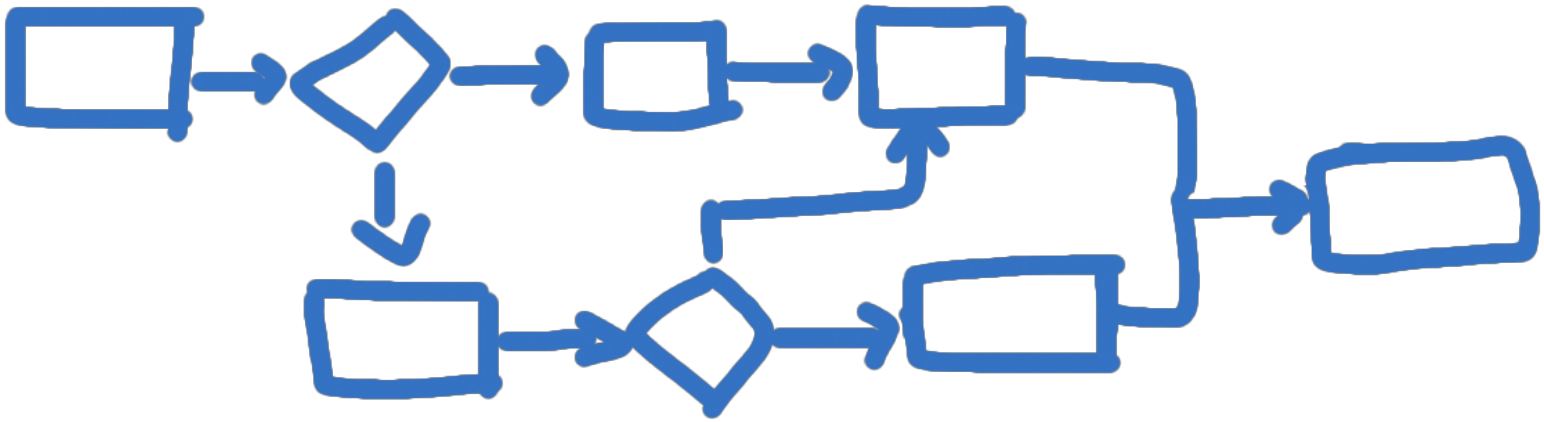
Very easy to use!

```
#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)

    }

}
```



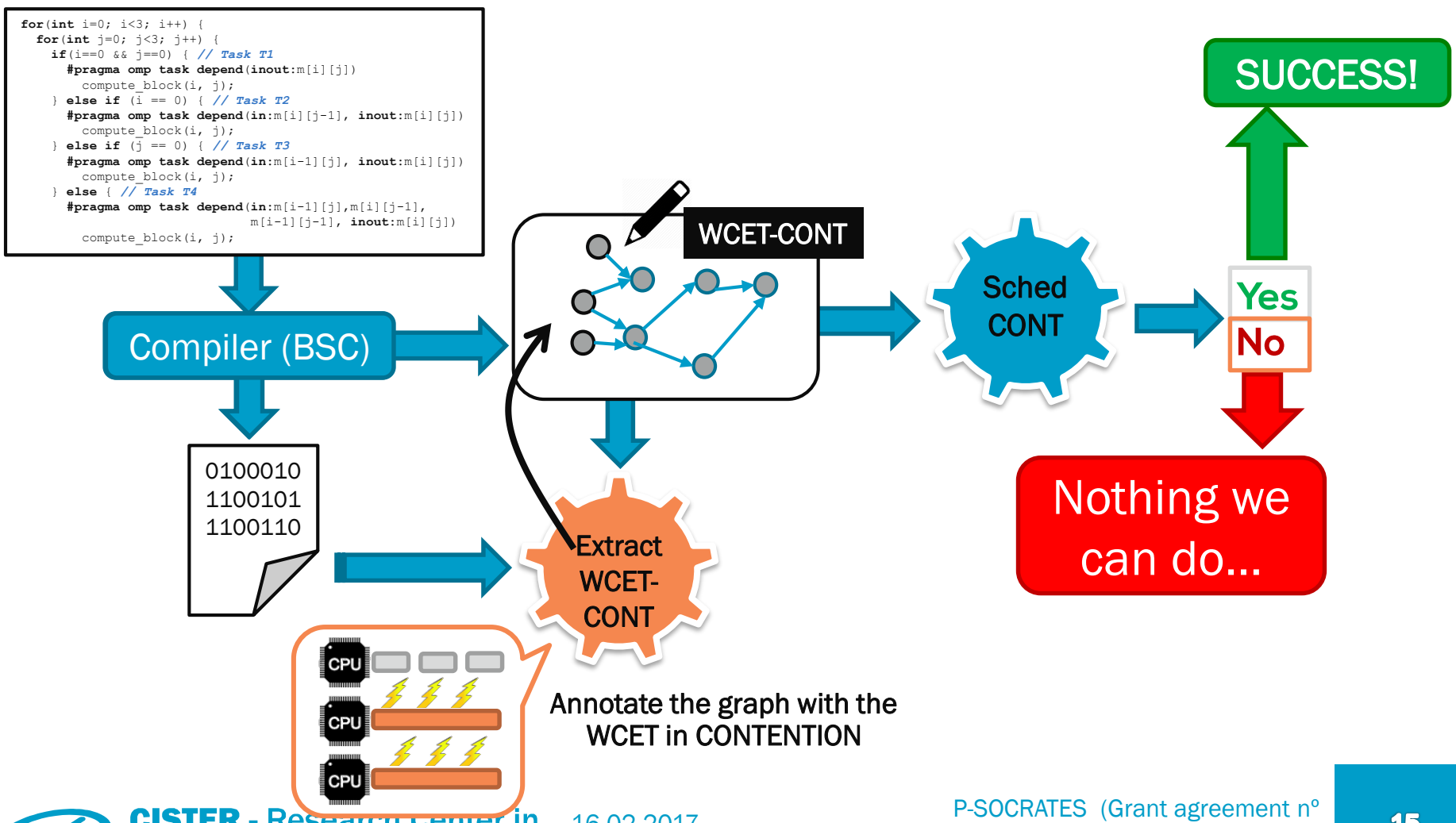
```
#include "timing.h"
TIMING_ANALYSIS_INIT();
#pragma omp parallel
{
    #pragma omp master
    {

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



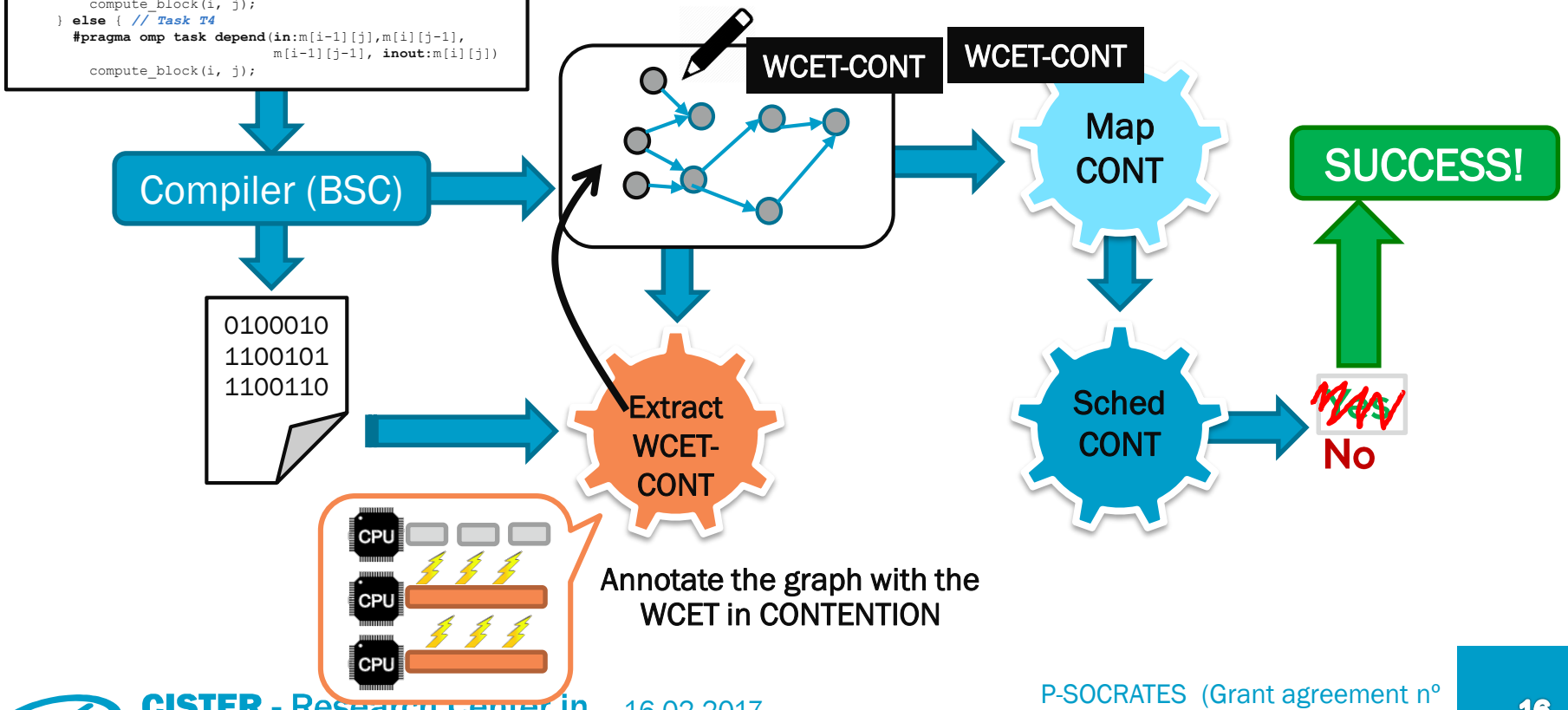
Dynamic scheduling

```
for(int i=0; i<3; i++) {
  for(int j=0; j<3; j++) {
    if(i==0 && j==0) { // Task T1
      #pragma omp task depend(inout:m[i][j])
        compute_block(i, j);
    } else if (i == 0) { // Task T2
      #pragma omp task depend(in:m[i][j-1], inout:m[i][j])
        compute_block(i, j);
    } else if (j == 0) { // Task T3
      #pragma omp task depend(in:m[i-1][j], inout:m[i][j])
        compute_block(i, j);
    } else { // Task T4
      #pragma omp task depend(in:m[i-1][j],m[i][j-1],
                              m[i-1][j-1], inout:m[i][j])
        compute_block(i, j);
```



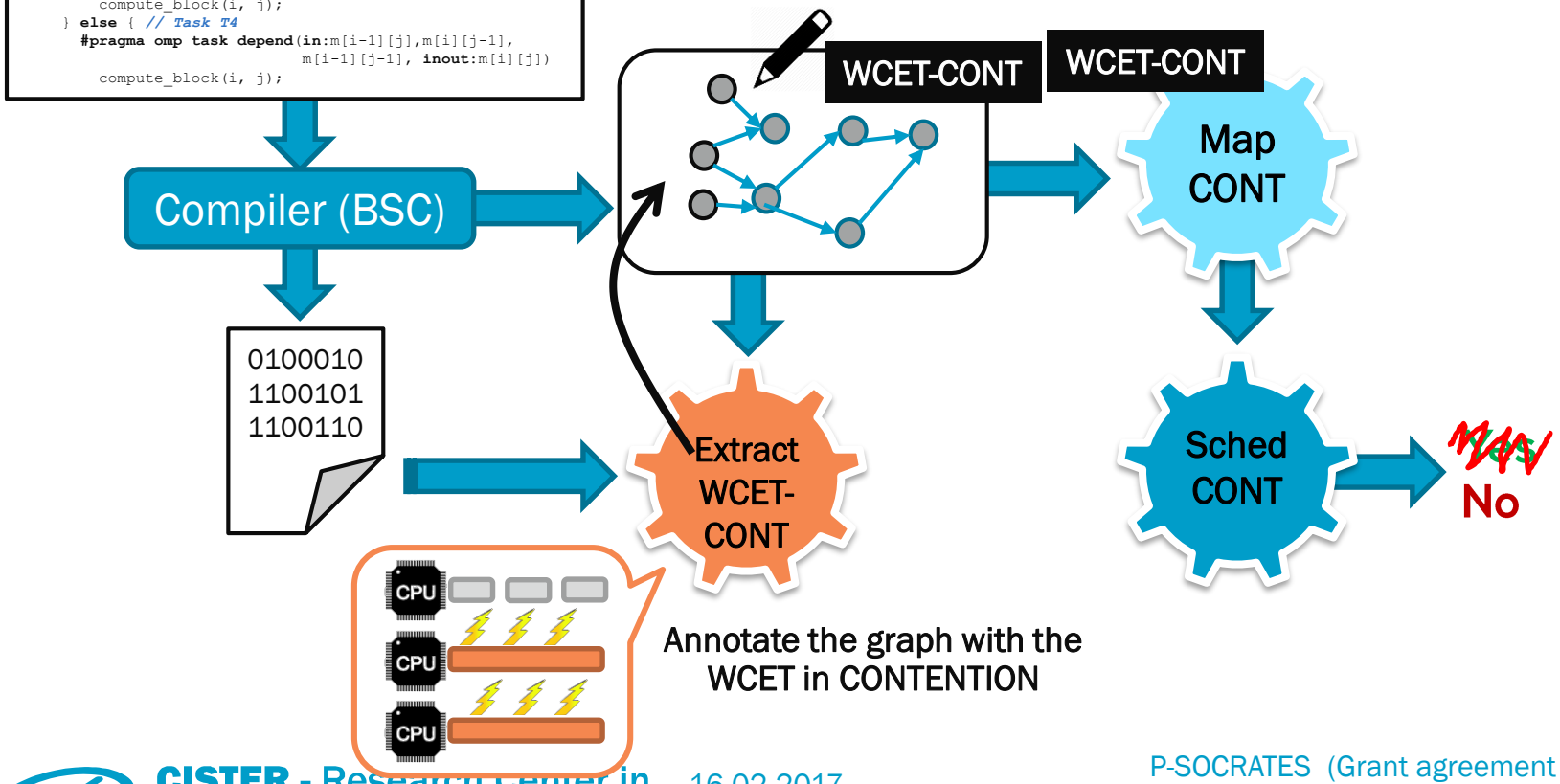
Static scheduling

```
for(int i=0; i<3; i++) {
  for(int j=0; j<3; j++) {
    if(i==0 && j==0) { // Task T1
      #pragma omp task depend(inout:m[i][j])
        compute_block(i, j);
    } else if (i == 0) { // Task T2
      #pragma omp task depend(in:m[i][j-1], inout:m[i][j])
        compute_block(i, j);
    } else if (j == 0) { // Task T3
      #pragma omp task depend(in:m[i-1][j], inout:m[i][j])
        compute_block(i, j);
    } else { // Task T4
      #pragma omp task depend(in:m[i-1][j],m[i][j-1],
                              m[i-1][j-1], inout:m[i][j])
        compute_block(i, j);
```

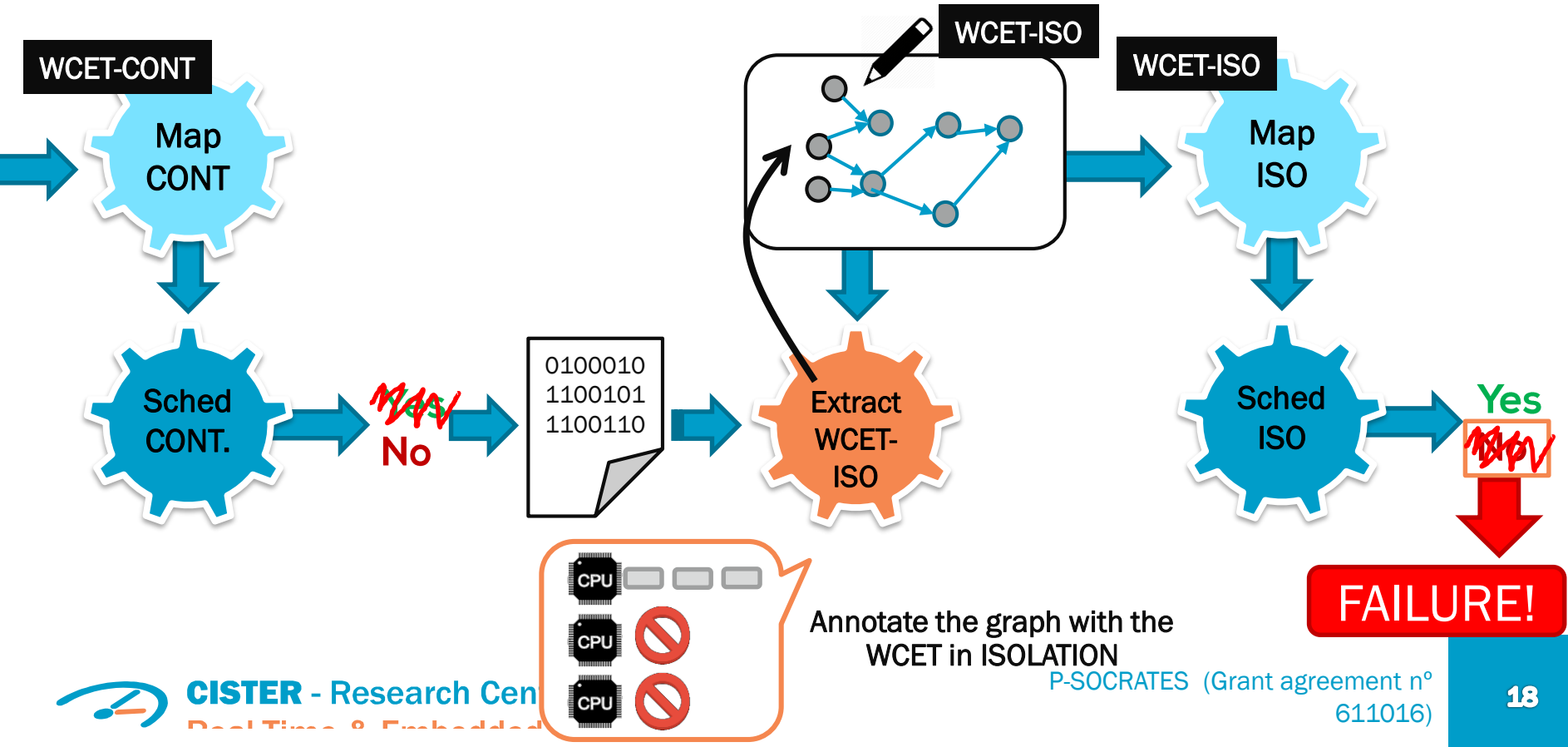


Static scheduling

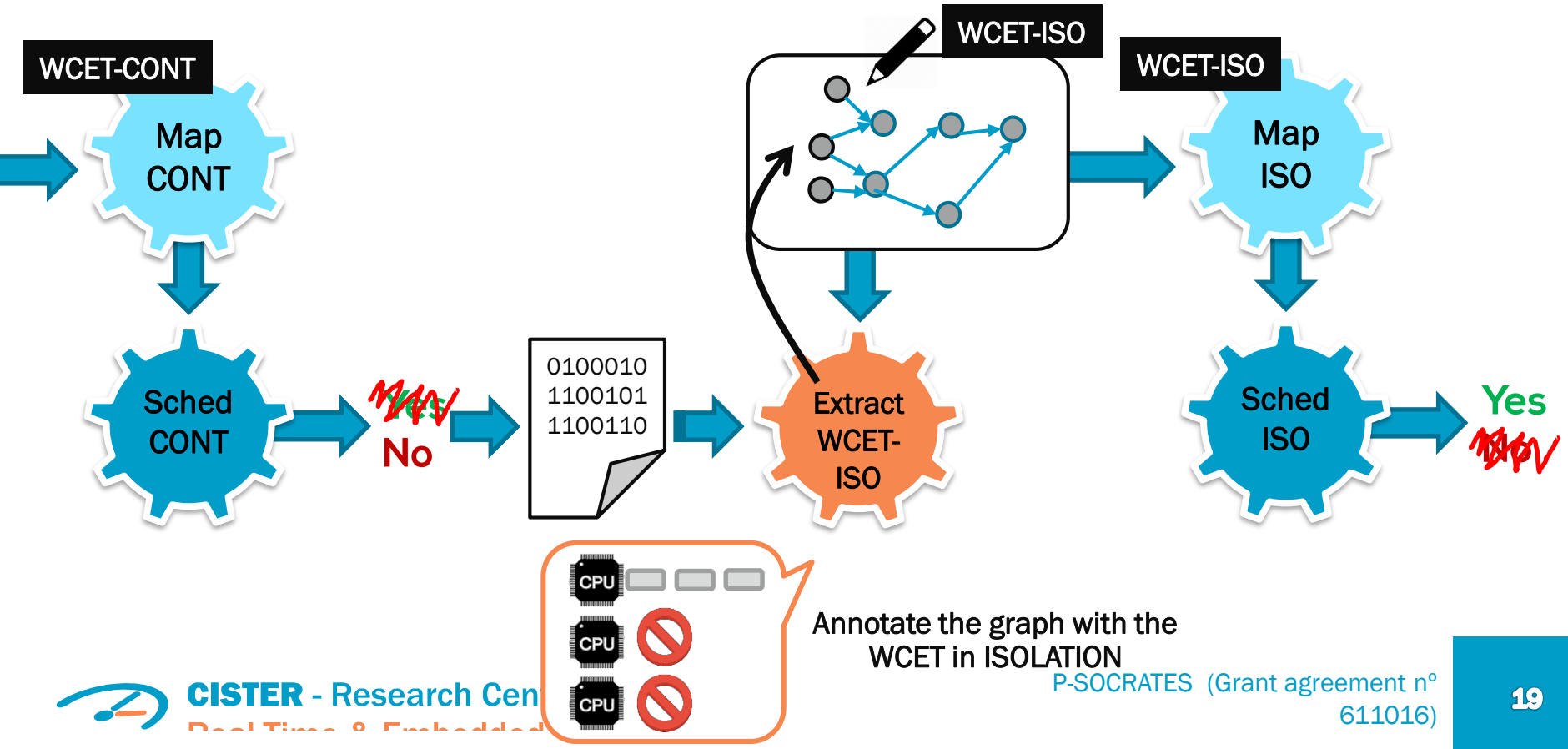
```
for(int i=0; i<3; i++) {
  for(int j=0; j<3; j++) {
    if(i==0 && j==0) { // Task T1
      #pragma omp task depend(inout:m[i][j])
        compute_block(i, j);
    } else if (i == 0) { // Task T2
      #pragma omp task depend(in:m[i][j-1], inout:m[i][j])
        compute_block(i, j);
    } else if (j == 0) { // Task T3
      #pragma omp task depend(in:m[i-1][j], inout:m[i][j])
        compute_block(i, j);
    } else { // Task T4
      #pragma omp task depend(in:m[i-1][j],m[i][j-1],
                              m[i-1][j-1], inout:m[i][j])
        compute_block(i, j);
```



Static scheduling

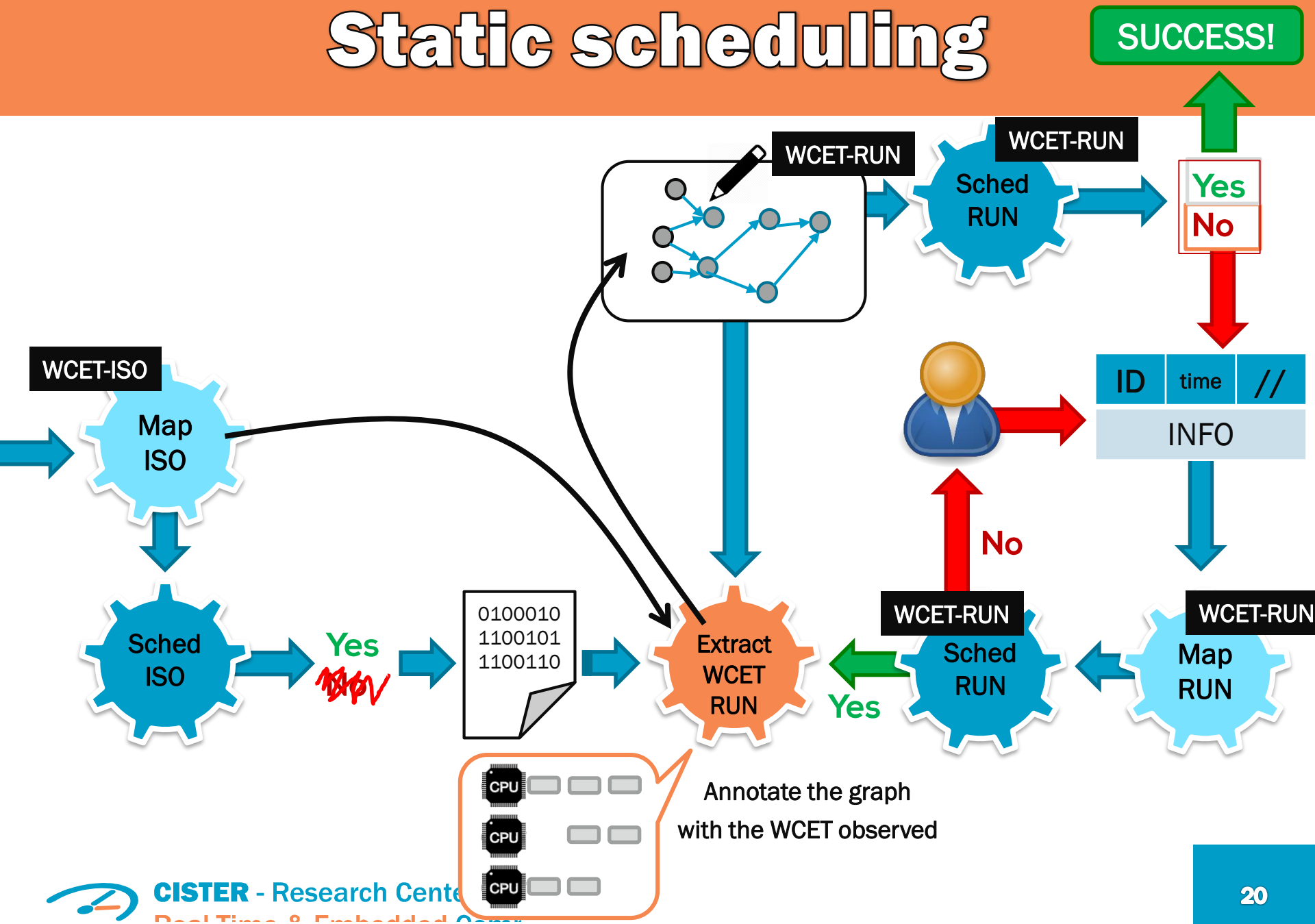


Static scheduling



Annotate the graph with the WCET in ISOLATION

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

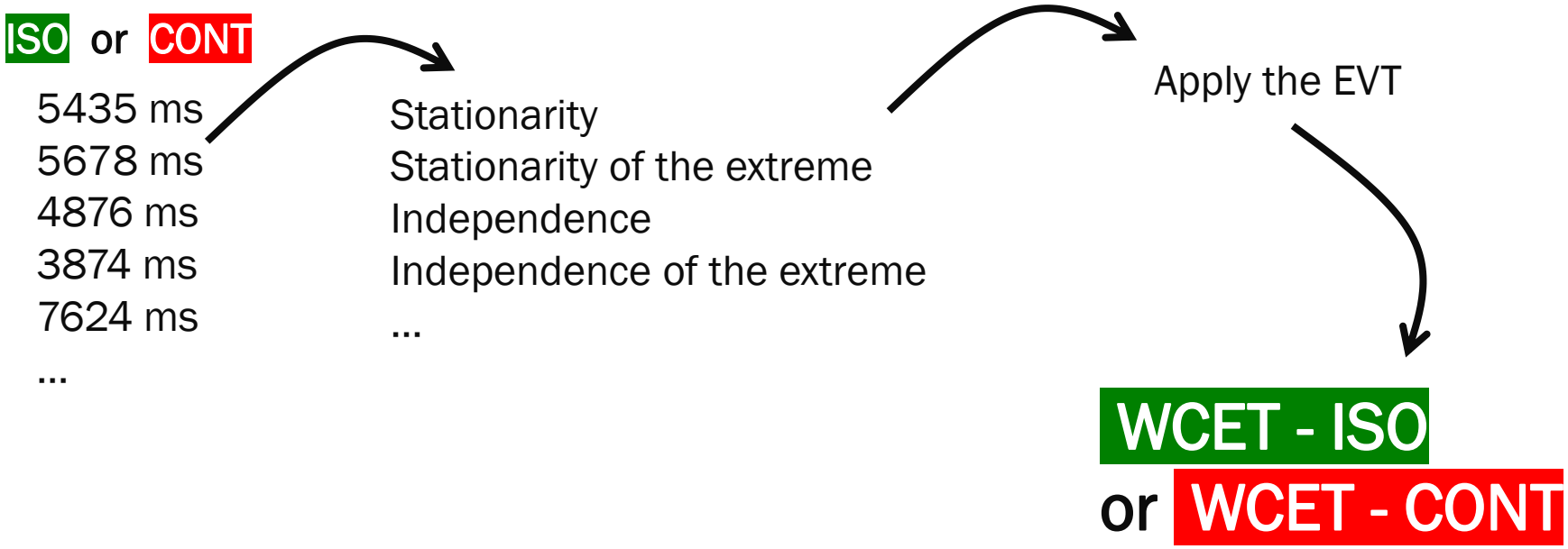
- The 3 application use-cases have been tested
- Overall, results seemed to confirm that static mapping and scheduling leads to **less performance at runtime** but **more accurate analysis** (the blame is on the schedulability analysis)

→ 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)



Thank you