High-performance parallelization of real-time applications

Luís Miguel Pinho, Vincent Nélis
School of Engineering of the Polytechnic of Porto (ISEP)
Portugal
{lmp,nelis}@isep.ipp.pt

Eduardo Quinoñes
Barcelona Supercomputing Centre
Barcelona, Spain
eduardo.quinones@bsc.es

Paolo Burgio
University of Modena
Italy
paolo.burgio@unimore.it

Andrea Marongiu
ETH Zurich
Switzerland
a.marongiu@iis.ee.ethz.ch

Paolo Gai
Evidence Srl
Italy
pj@evidence.eu.com

Juan Sancho
ATOS
Spain
juan.sancho@atos.net

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement nº 611016

Outline

- P-SOCRATES at a glance
- Motivation and Vision
- Technical Approach
- Timing Analysis Methodology
- The **UpScale** SDK
- Conclusions

Quick fact sheet

- **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, Dec-2016)
- **Website:** www.p-socrates.eu
- Budget: 3.6 M€
- Partners

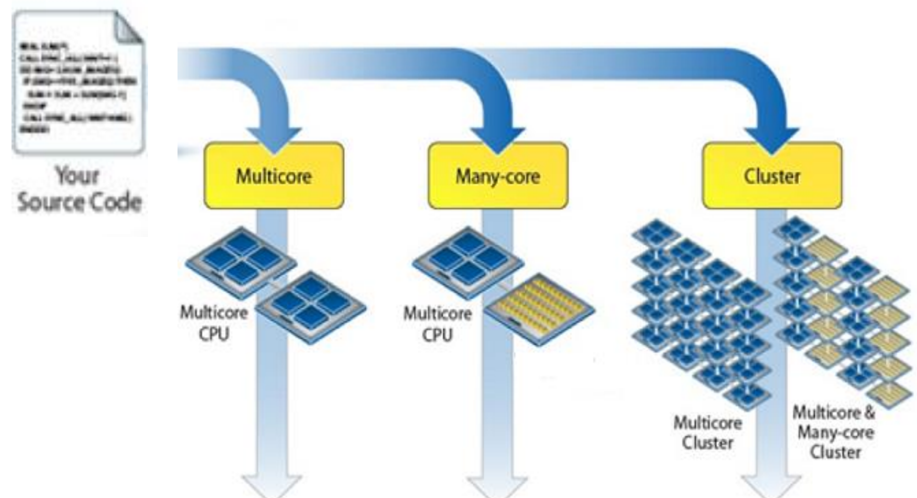
Industrial Advisory Board

- Review and prioritize requirements, ensure that the project is kept on focus, analyze and validate the results
- Members:

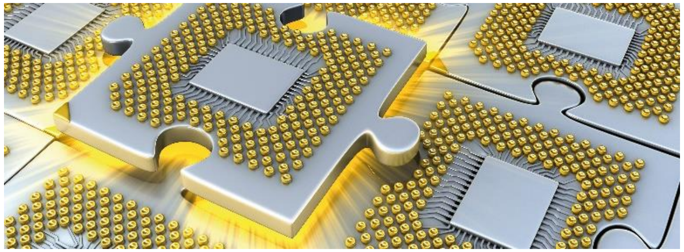
City of Bratislava

Motivation

- New applications challenge the performance capabilities of hardware platforms by crossing the boundaries of computing domains
  - Demand of increased **performance** with **guaranteed processing times**
    - Demands can only be met by **advanced parallel computing platforms**
  - **Programmability** of parallel platforms **is a major challenge**
    - Need to integrate **parallel programming models** in embedded systems



Vision

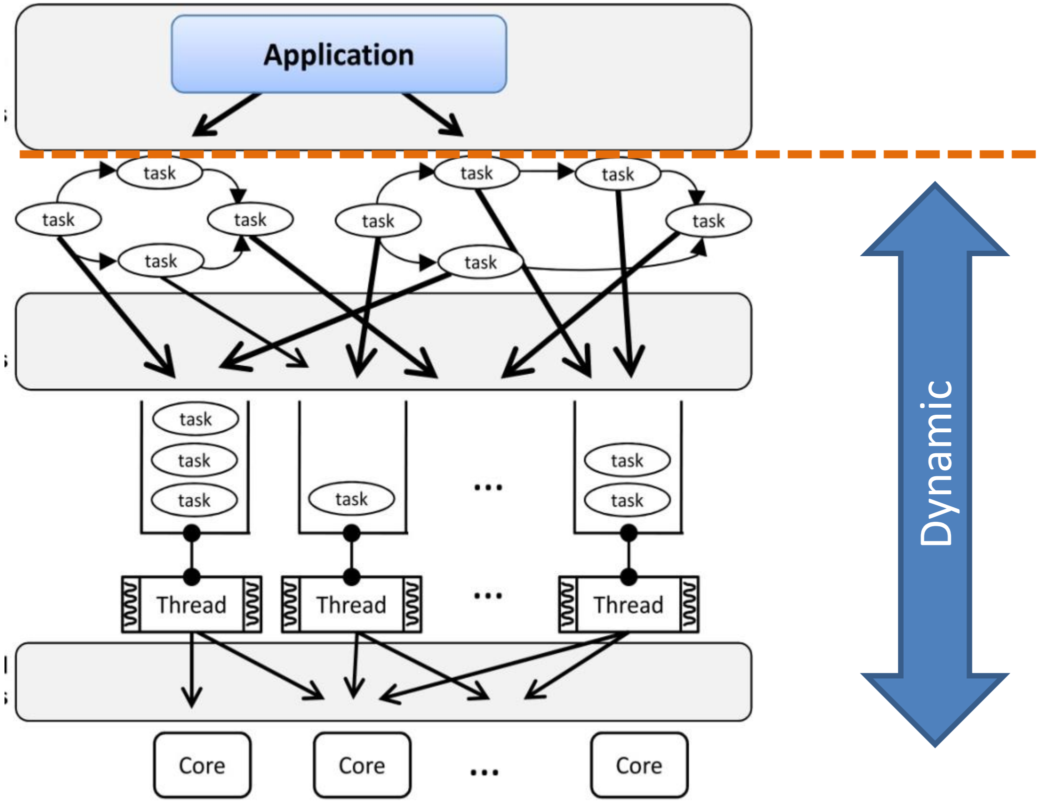


next-generation embedded
many-core accelerators

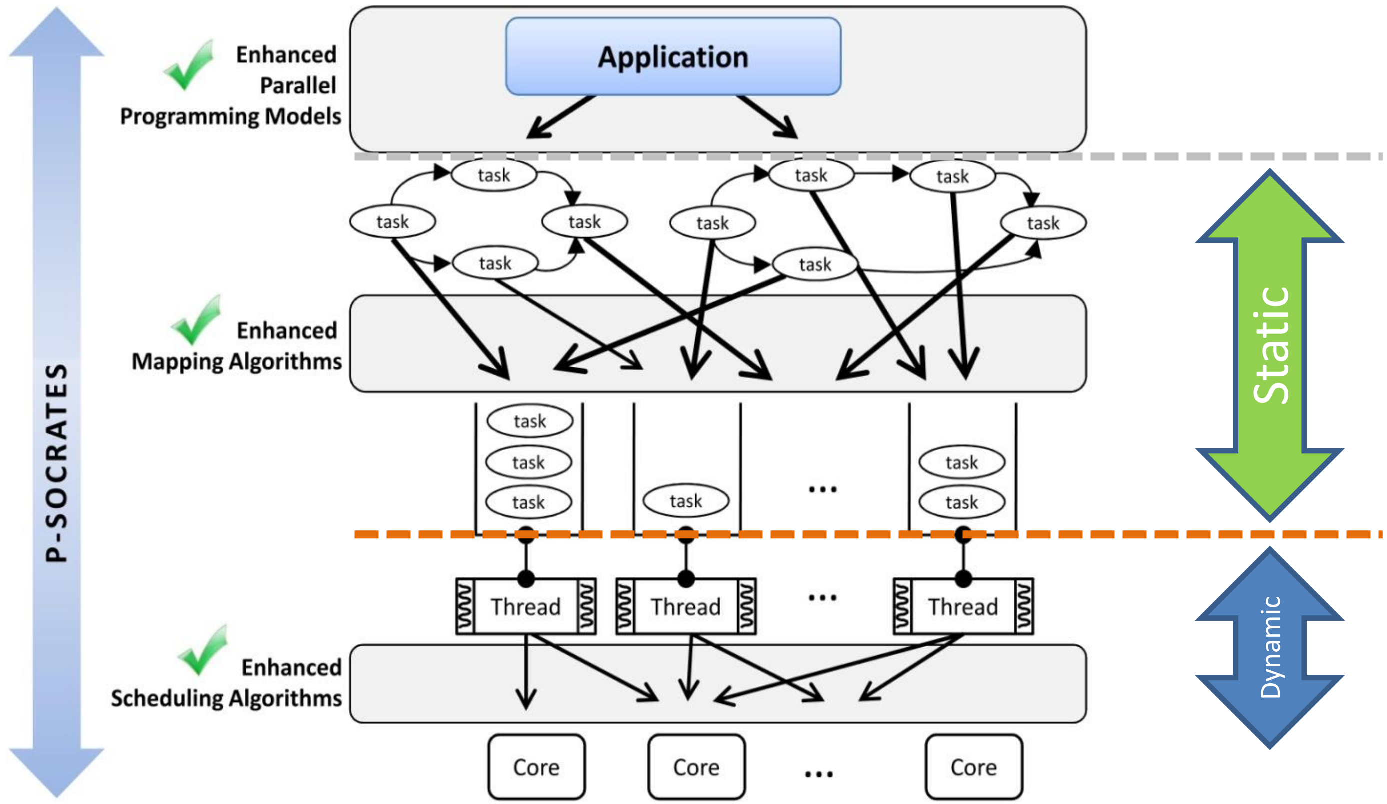
programmability of
many-core from
high-performance computing

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

Vision


Application
task
task
task
task
task
task
task
task
task
task
task
task
task
task
task
task
Thread
Thread
Thread
...
...
...
Core
Core
Core
Dynamic

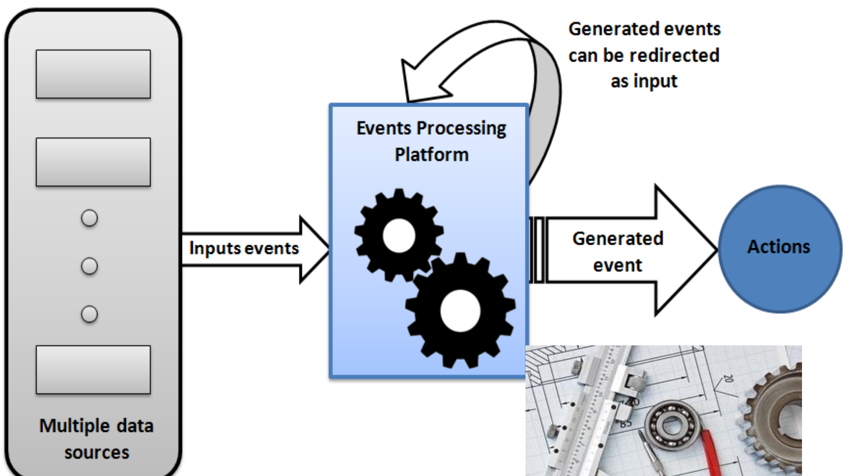
Vision

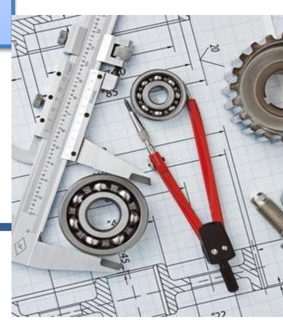

P-SOCRATES
Enhanced Parallel Programming Models
Enhanced Mapping Algorithms
Enhanced Scheduling Algorithms
Application
task
Thread
Core
...
Static
Dynamic

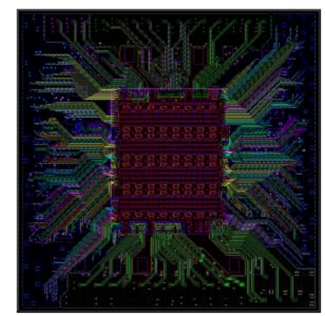
Innovation

- A **generic framework**, integrating models, tools and system software, to parallel with **high performance** and **real-time** requir

P-~~SOCRATES~~



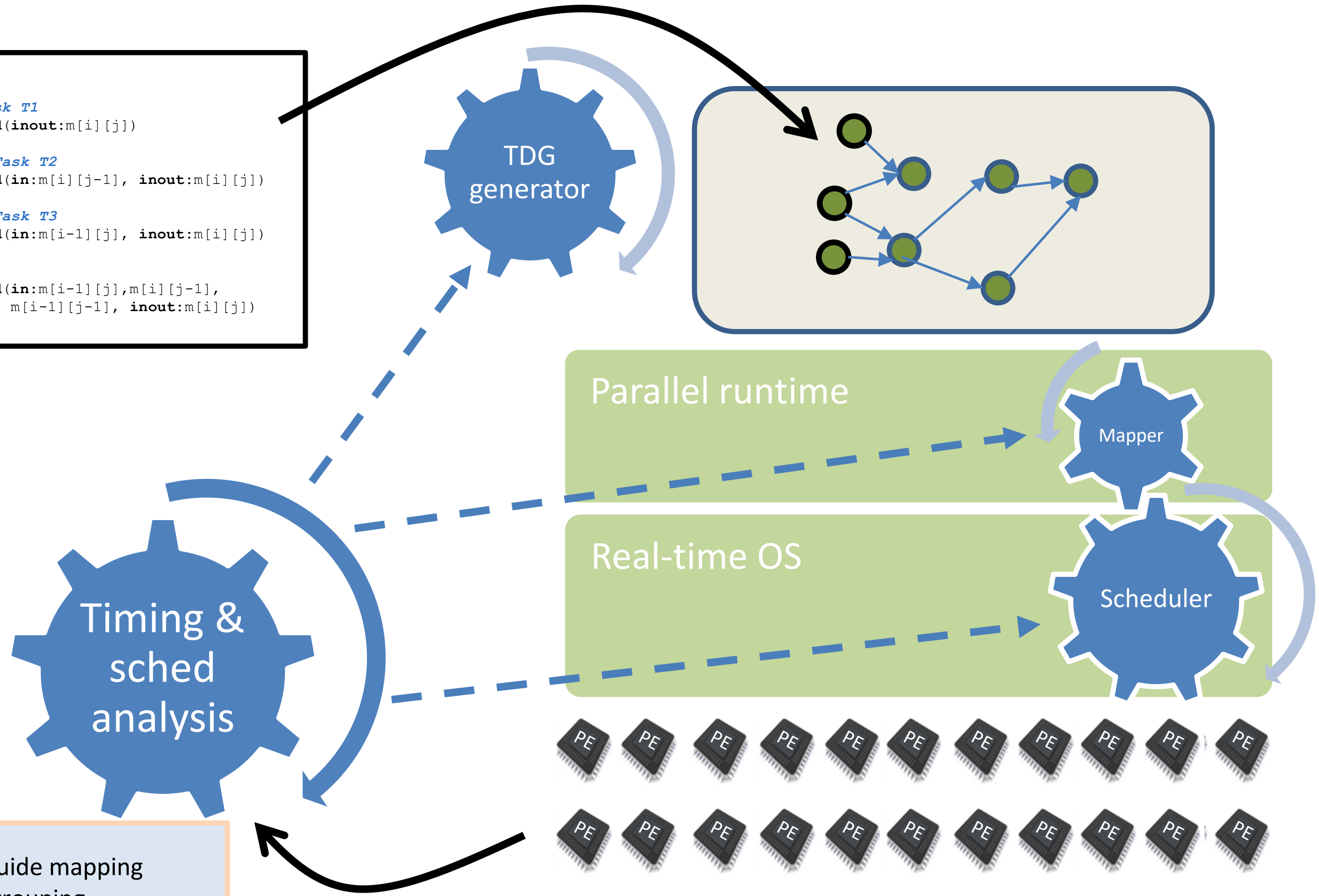




Technical Approach

Compiler phase

```
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);
```

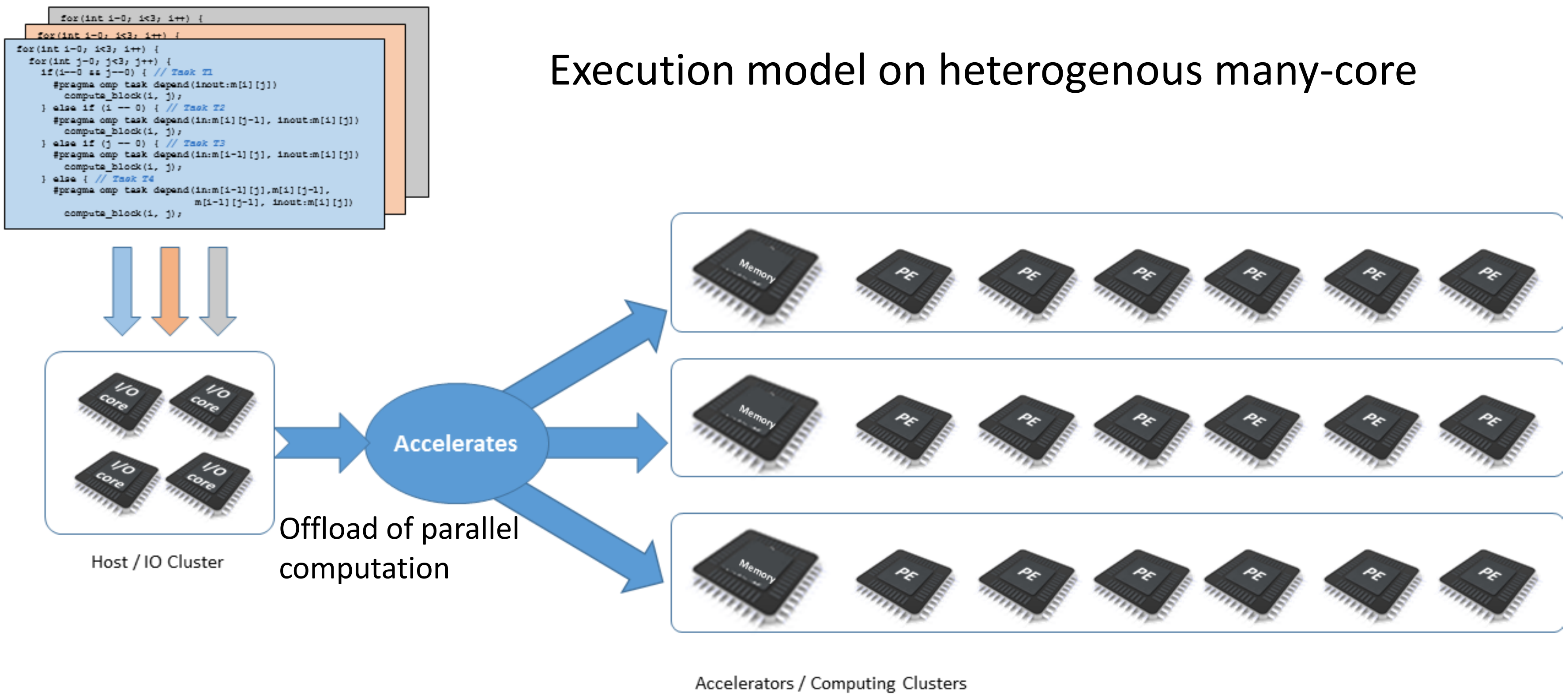


Exploring Hardware model to guide mapping
- Reducing complexity, grouping
- Scheduling communication/computation
- Explore measurement-based approaches

Clustered architecture to provide composability

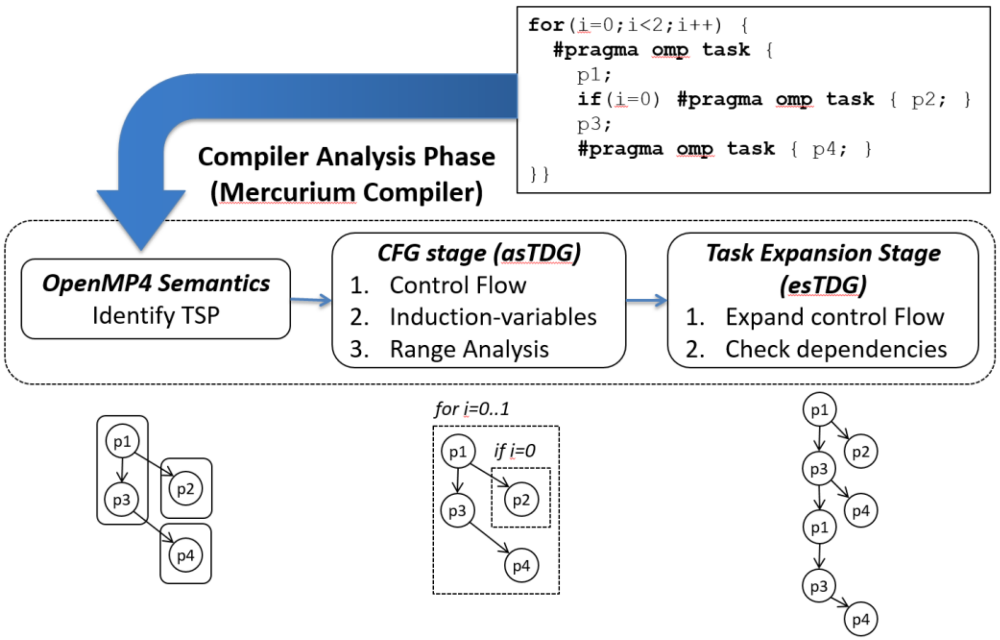
Run-time tracing

Technical Approach



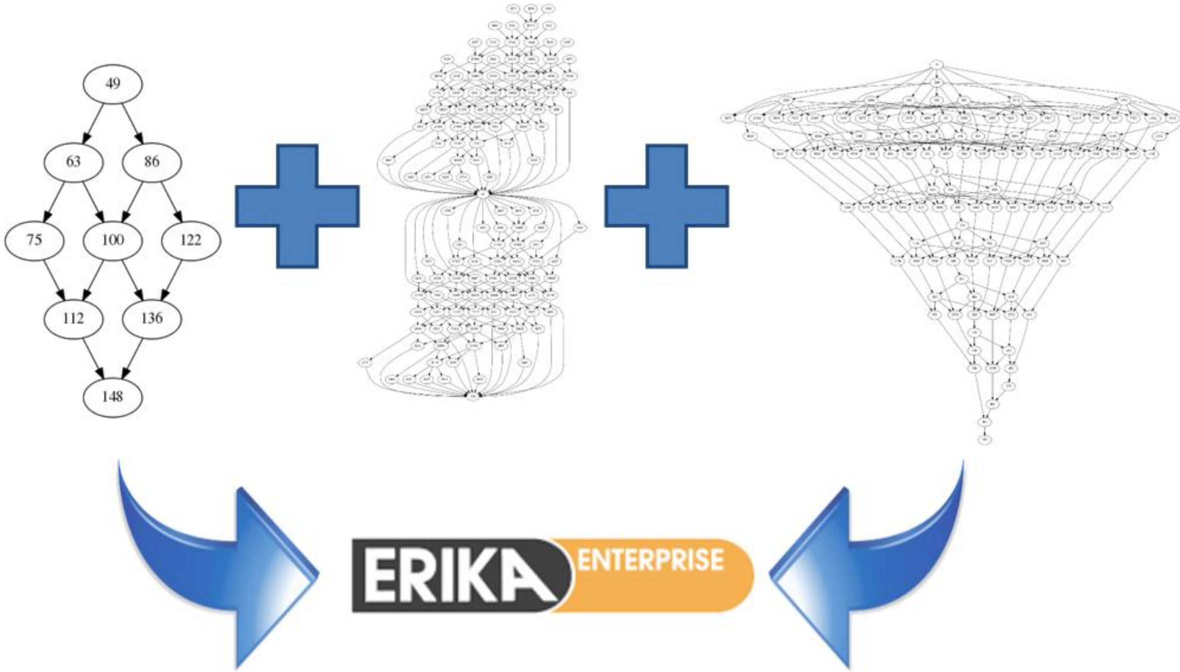
Technical Approach

- Extraction of parallelism with data-flow annotations
  - OpenMP tasking semantics generates a graph of control and data flow task execution

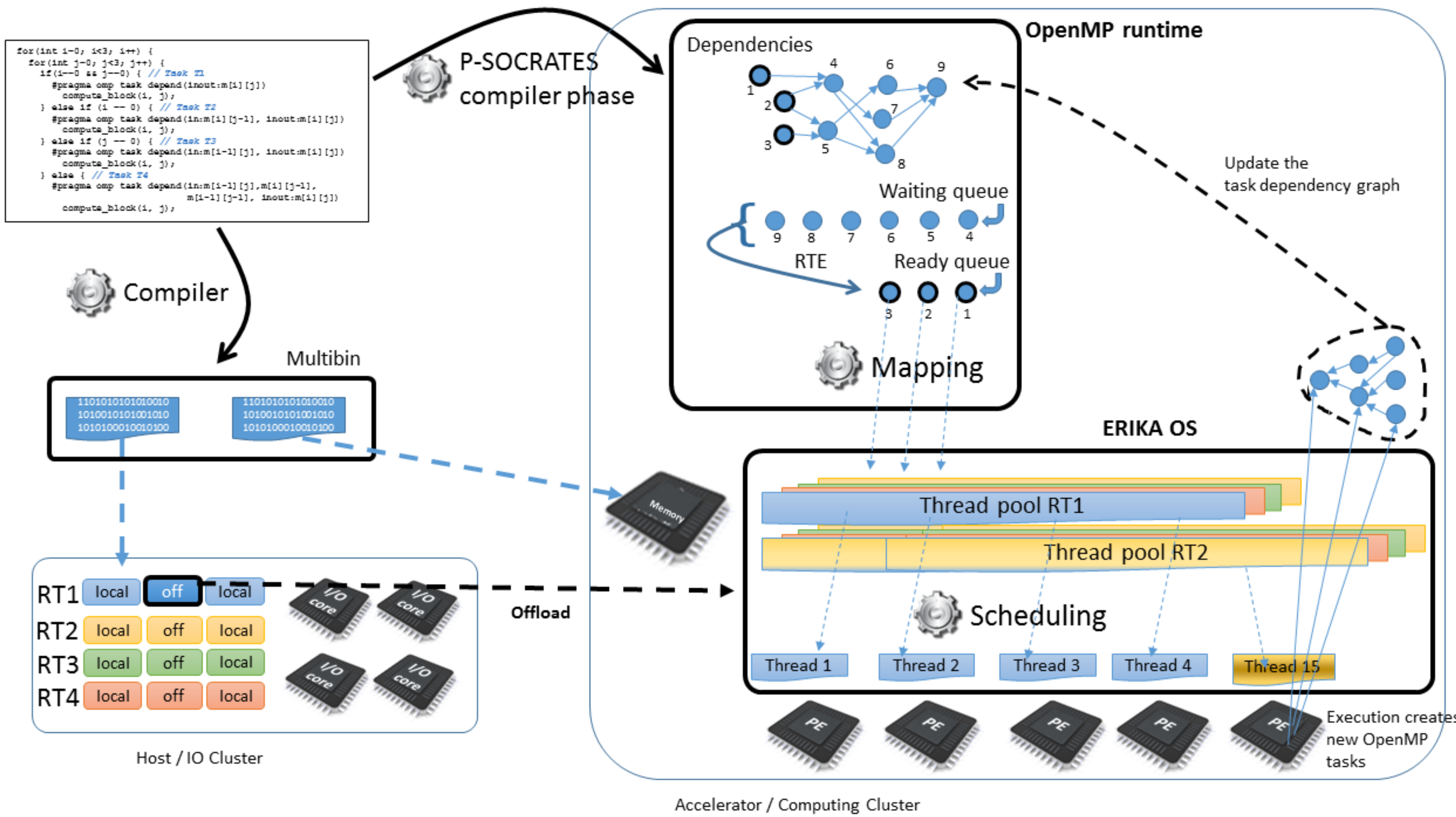


Technical Approach

- Scheduler
  - Both static mapping partitioned and dynamic mapping global scheduling approaches



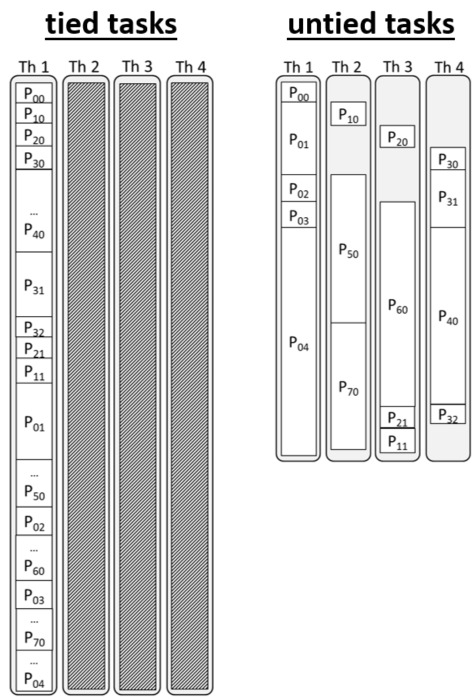
Technical Approach



Technical Approach

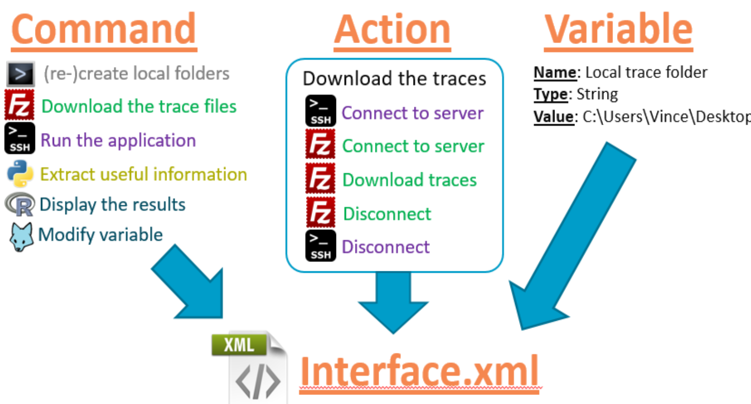
- Schedulability Analysis
  - Schedulability analysis of OpenMP tasking DAGs, considering both tied and untied tasks

```
#pragma omp parallel num_threads(N) {
#pragma omp single { // T0
   part00
   #pragma omp task { // T1
      part10
      #pragma omp task {  // T2
         part20
         #pragma omp task {  // T3
            part30
            #pragma omp task { part40 } // T4
            #pragma omp taskwait
            part31
         }
         #pragma omp taskwait
         part21
      }
      #pragma omp taskwait
      part11
   }
   part01
   #pragma omp task { part50 } // T5
   part02
   #pragma omp task { part60 } // T6
   part03
   #pragma omp task { part70 } // T7
   part04
}}
```

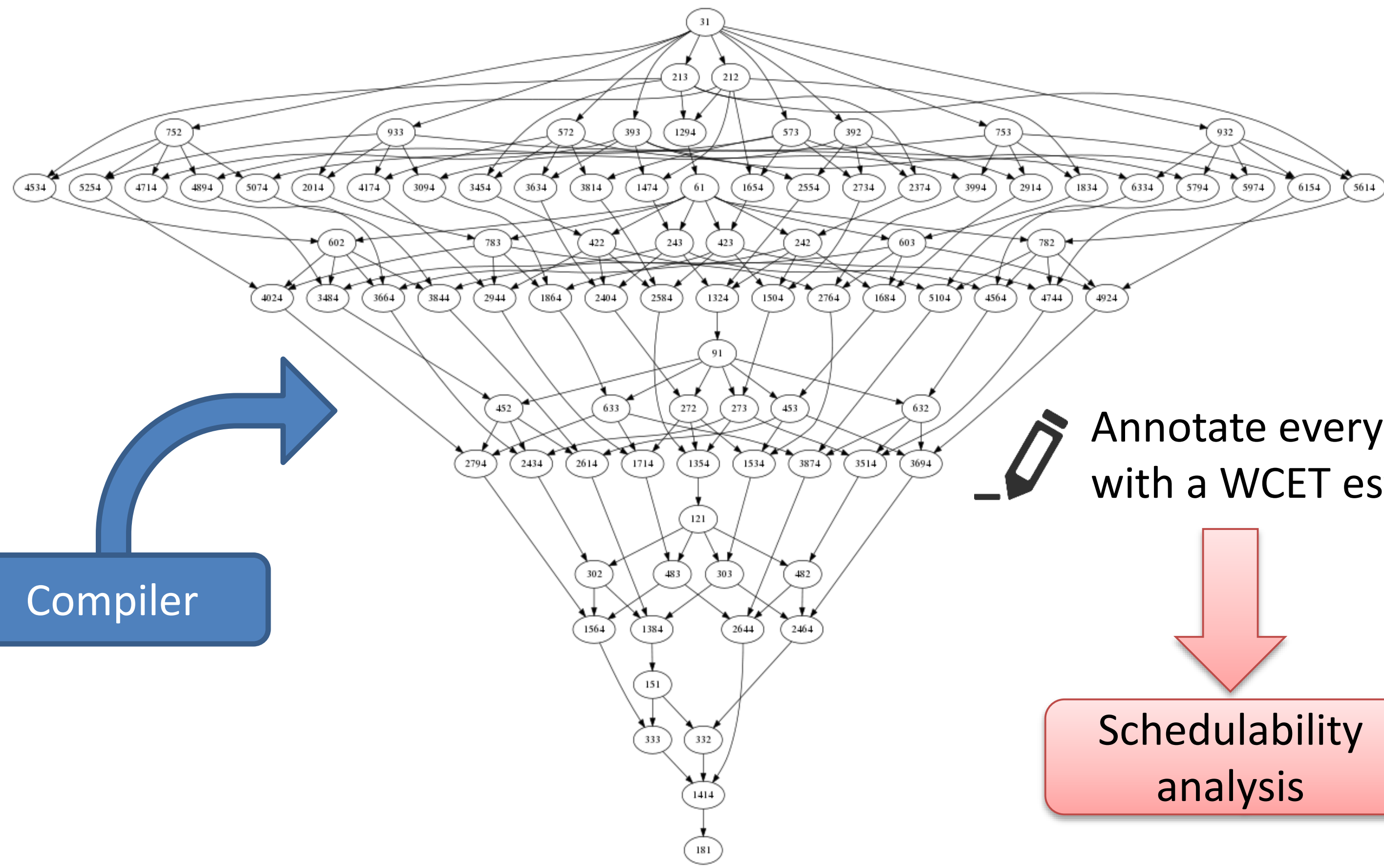


Technical Approach

- Timing Analysis
  - Exploring measurement-based approaches
    - Allowing to analyze and reason about an application timing behavior
  - Collecting execution traces is a tedious process
    - Involves many steps, in different languages
  - Developed measurement-based trace collecting and analysis tool
    - Collecting runtime execution traces is fully automatic
    - Extract and compute statistical information from the traces



P-SOCRATES TA Objectives



Compiler

Annotate every node with a WCET estimate

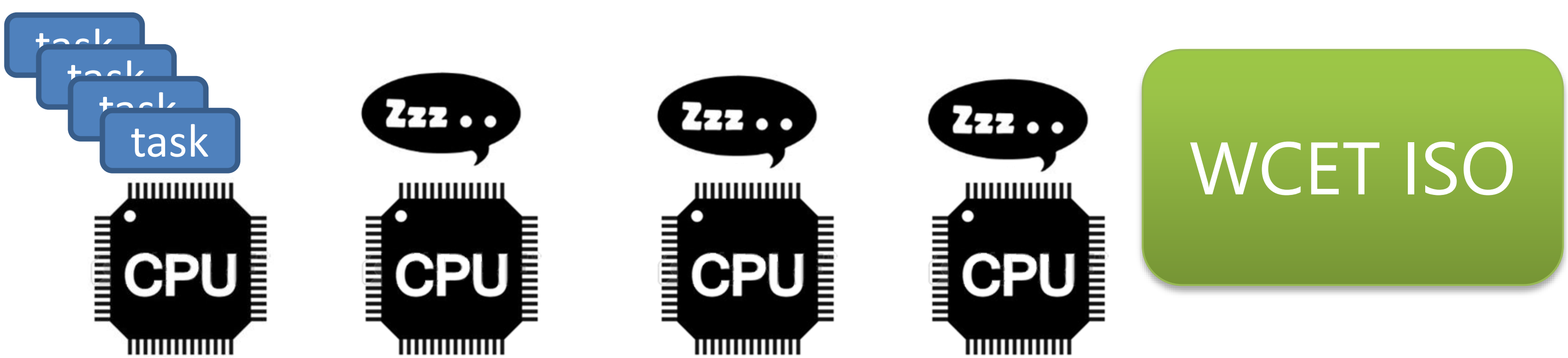
Schedulability analysis

Methodology

- A new approach to tackle the interference problem

Not one but two WCET estimates

1 One estimate is obtained by running every task in complete isolation (runs on 1 core, the rest of the system stays quiet)

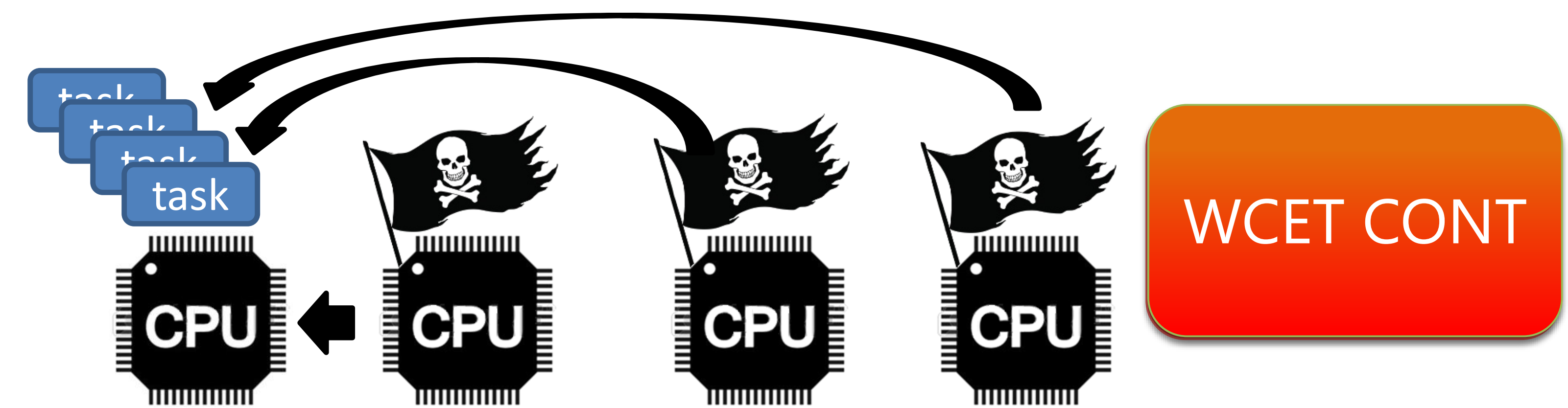


Methodology

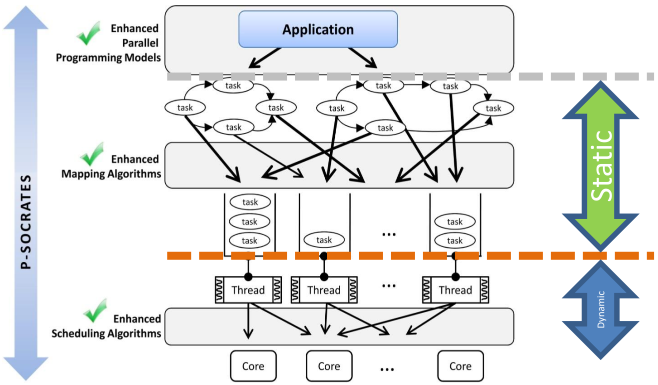
- A new approach to tackle the interference problem

Not one but two WCET estimates

The other is obtained by running every task in complete contention (runs on 1 core, the rest of the system does everything possible to interfere with its execution)



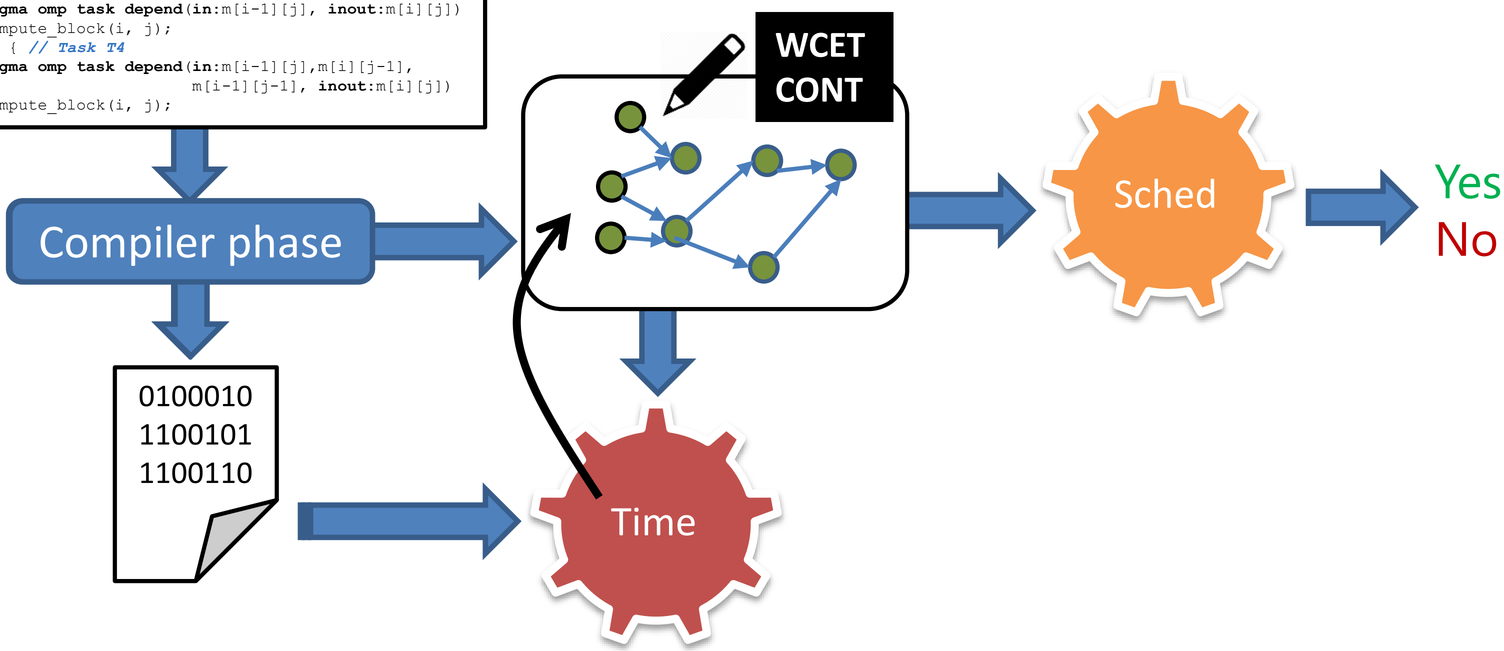
Methodology



- Processes to perform schedulability analysis
  - Based on both intrinsic and extrinsic WCET estimates
  - One process for the dynamic project approach
    - Task-to-thread mapping is with global queue
    - Thread scheduling is global with limited preemption
    - Maximize average performance
  - Another for the static process approach
    - Fixed task-to-thread mapping (heuristics to minimize makespan)
    - Partitioned per-core scheduling (with limited preemption)
    - Minimize guaranteed response time

The big picture (dynamic)

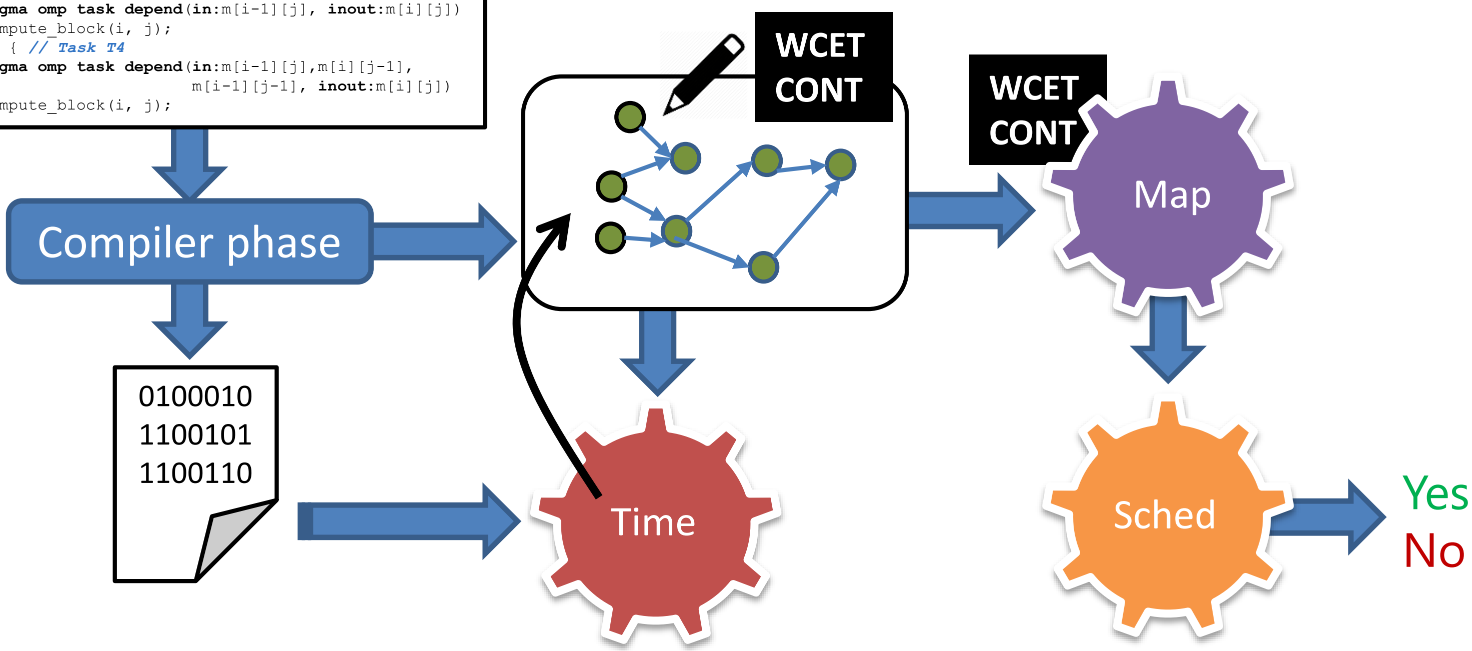
```
for(int i=0; i<3; i++) {
  for(int j=0; j<3; j++) {
    if(i==0 && j==0) { // Task T1
      #pragma omp task depend(inout:m[i][j])
        compute_block(i, j);
    } else if (i == 0) { // Task T2
      #pragma omp task depend(in:m[i][j-1], inout:m[i][j])
        compute_block(i, j);
    } else if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



Annotate the graph with the WCET in CONTENTION

The big picture (static)

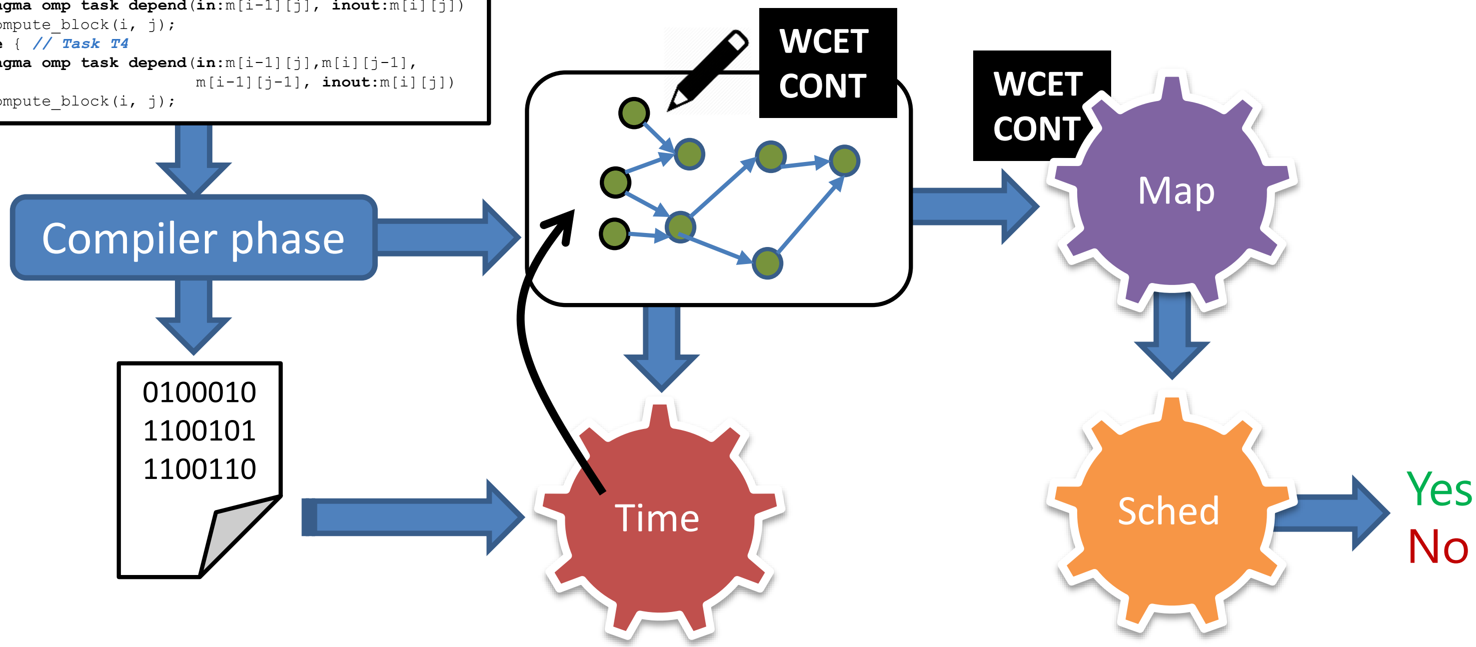
```
for(int i=0; i<3; i++) {
  for(int j=0; j<3; j++) {
    if(i==0 && j==0) { // Task T1
      #pragma omp task depend(inout:m[i][j])
        compute_block(i, j);
    } else if (i == 0) { // Task T2
      #pragma omp task depend(in:m[i][j-1], inout:m[i][j])
        compute_block(i, j);
    } else if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



Annotate the graph with the WCET in CONTENTION

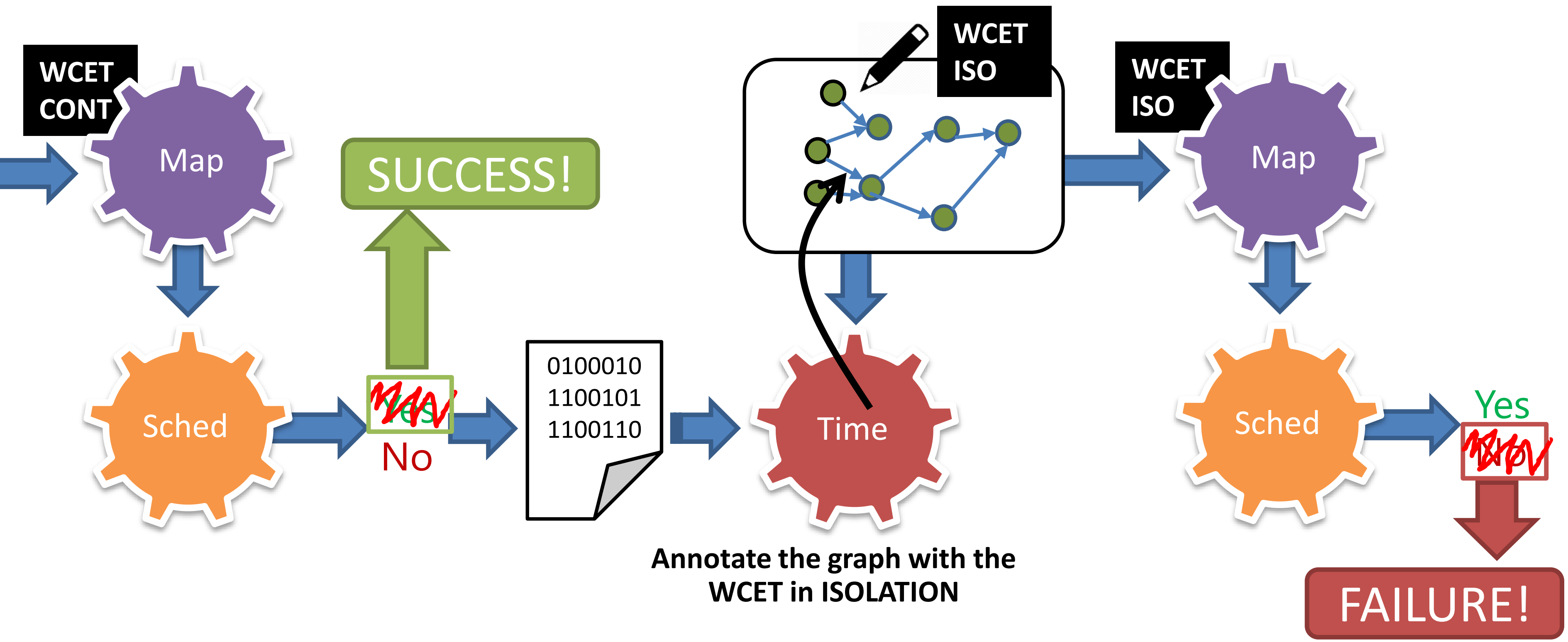
The big picture (static)

```
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);
```



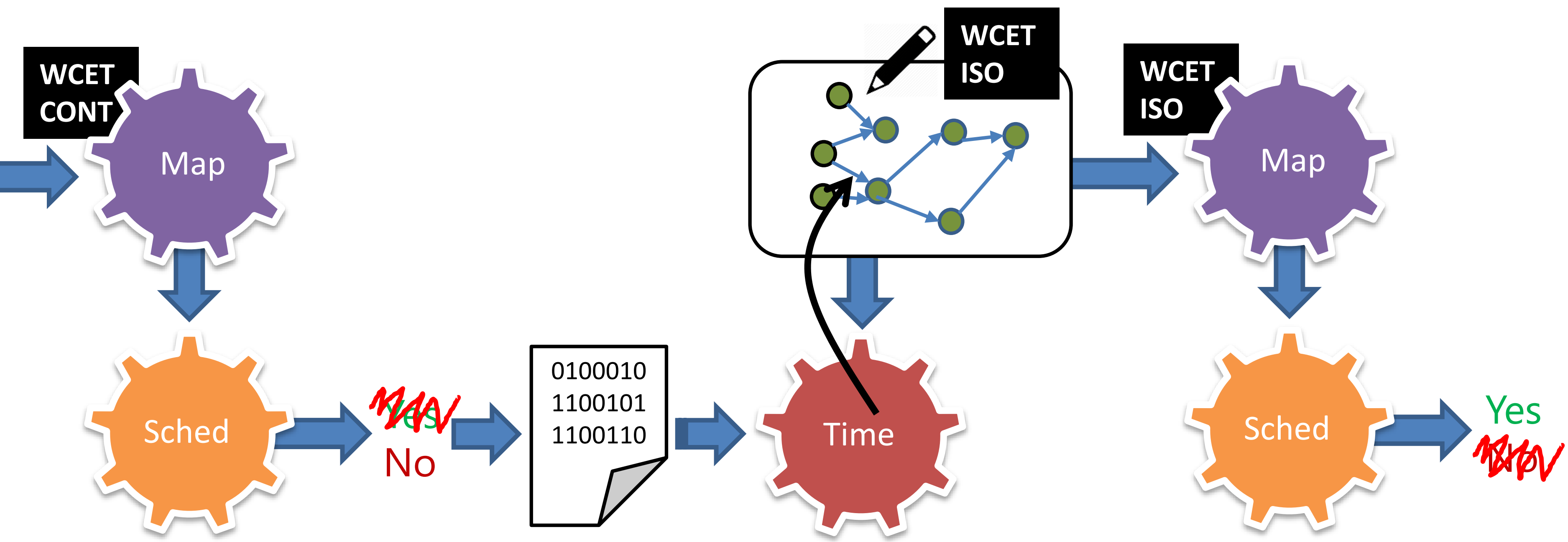
Annotate the graph with the WCET in CONTENTION

The big picture (static)



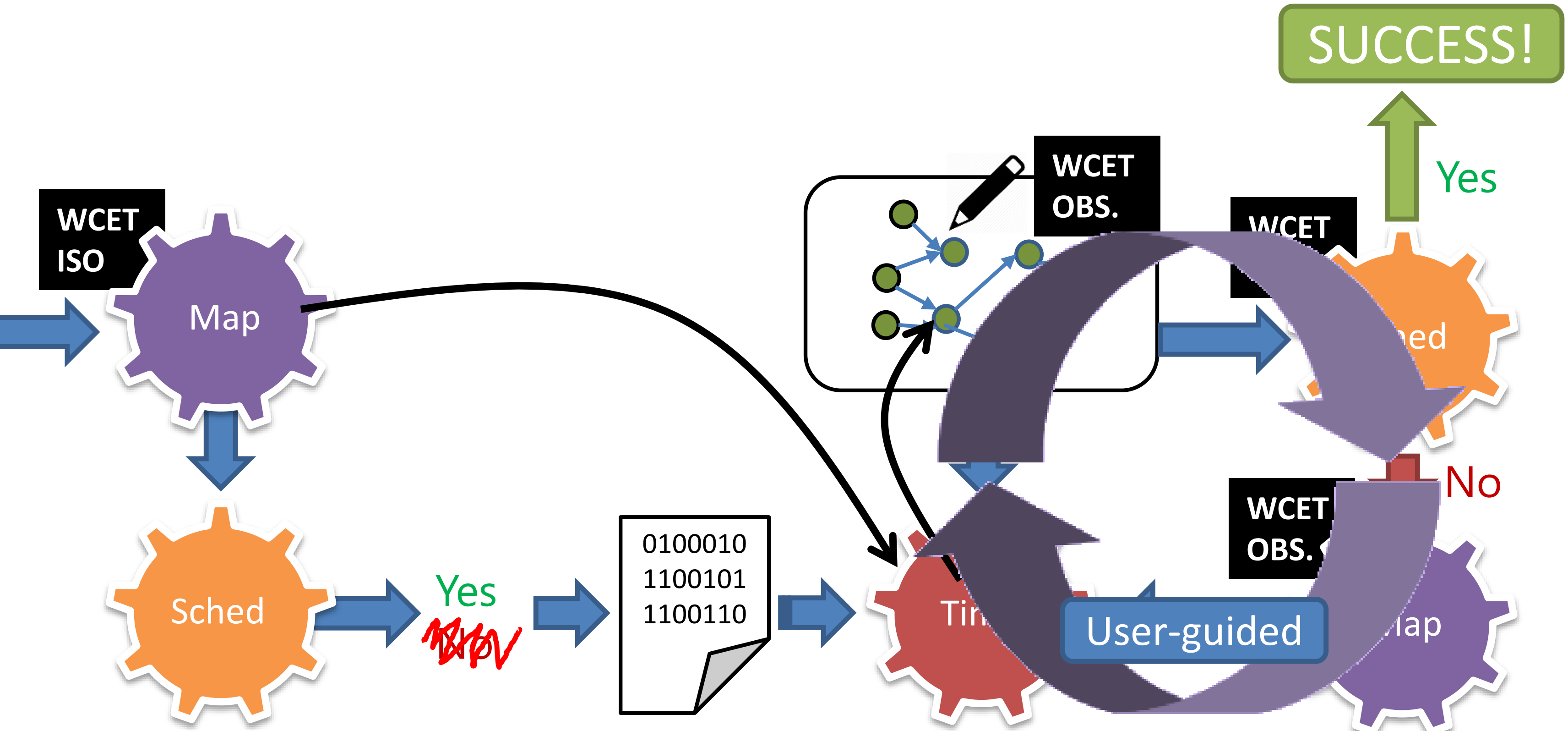
Annotate the graph with the WCET in ISOLATION

The big picture (static)



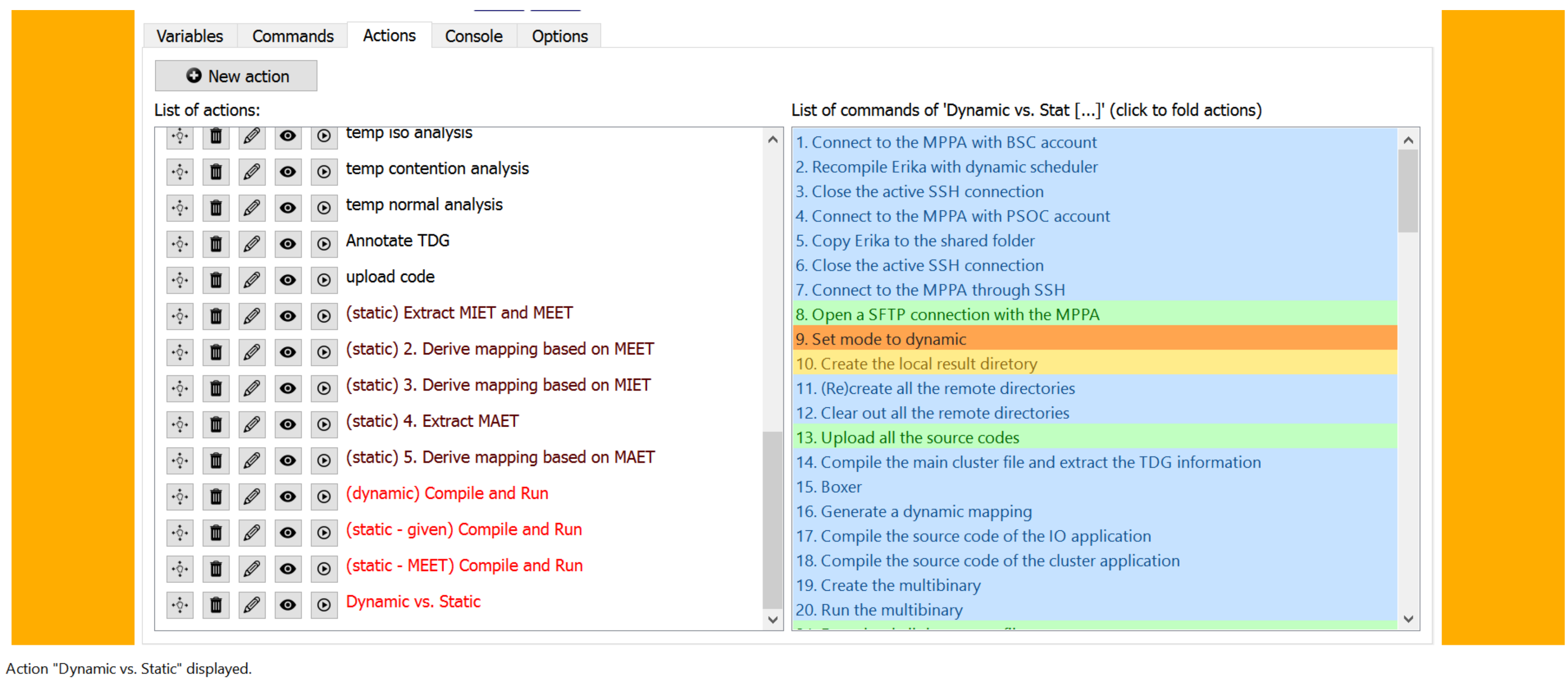
Annotate the graph with the WCET in ISOLATION

The big picture (static)



Annotate the graph with the WCET observed

All-in-one



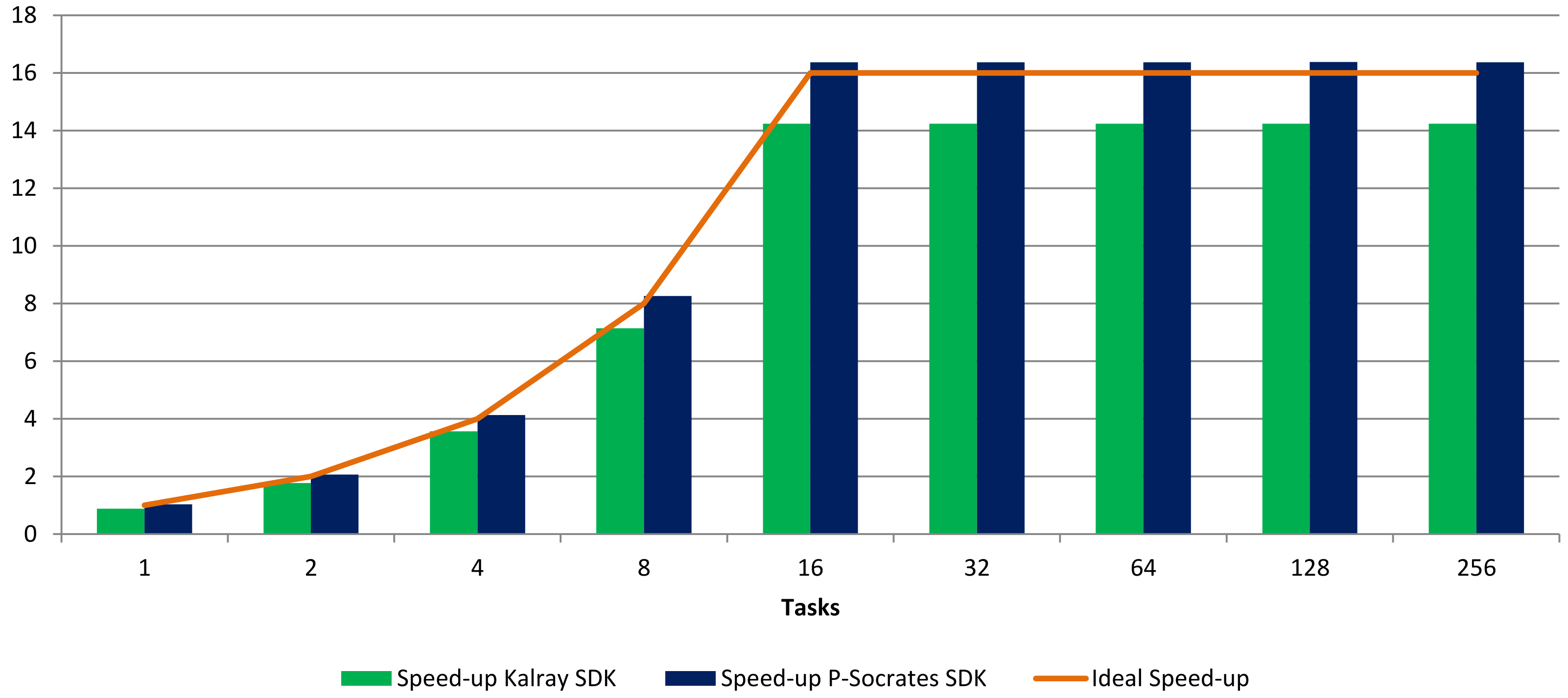
Variables
Commands
Actions
Console
Options
New action
List of actions:
temp iso analysis
temp contention analysis
temp normal analysis
Annotate TDG
upload code
(static) Extract MIET and MEET
(static) 2. Derive mapping based on MEET
(static) 3. Derive mapping based on MIET
(static) 4. Extract MAET
(static) 5. Derive mapping based on MAET
(dynamic) Compile and Run
(static - given) Compile and Run
(static - MEET) Compile and Run
Dynamic vs. Static
List of commands of 'Dynamic vs. Stat [...]' (click to fold actions)
1. Connect to the MPPA with BSC account
2. Recompile Erika with dynamic scheduler
3. Close the active SSH connection
4. Connect to the MPPA with PSOC account
5. Copy Erika to the shared folder
6. Close the active SSH connection
7. Connect to the MPPA through SSH
8. Open a SFTP connection with the MPPA
9. Set mode to dynamic
10. Create the local result diretory
11. (Re)create all the remote directories
12. Clear out all the remote directories
13. Upload all the source codes
14. Compile the main cluster file and extract the TDG information
15. Boxer
16. Generate a dynamic mapping
17. Compile the source code of the IO application
18. Compile the source code of the cluster application
19. Create the multibinary
20. Run the multibinary
Action "Dynamic vs. Static" displayed.

Use-cases

- Intelligent Traffic Application
  - Complex event processing engine for public transport
- Space Case Study
  - Pre-processing application for infrared detectors used for the Ecluid space mission
- Online Text Semantics
  - Tool performs semantic analysis, categorizes and extracts information from text
- All case studies will execute on a COTS processor
  - Kalray MPPA Bostan

Results

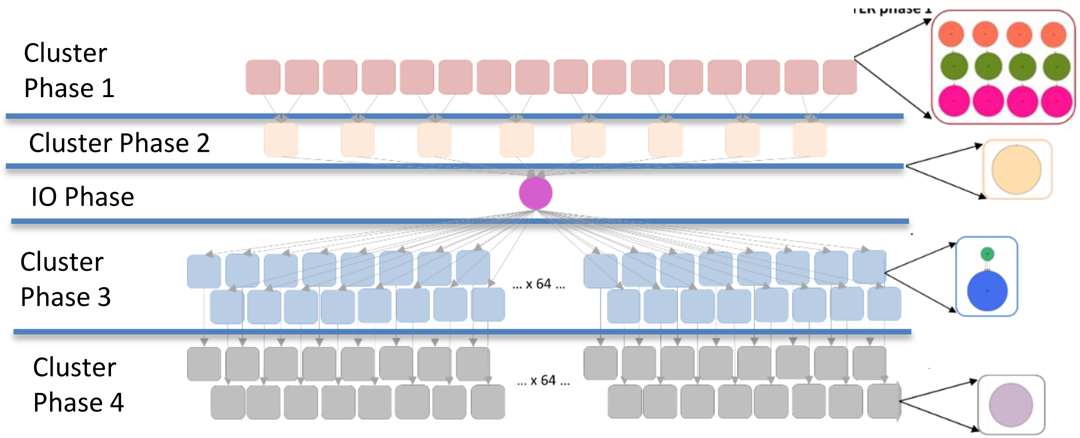
- Intelligent traffic application



Complex Function →set of 256ffts (size=512)

Results

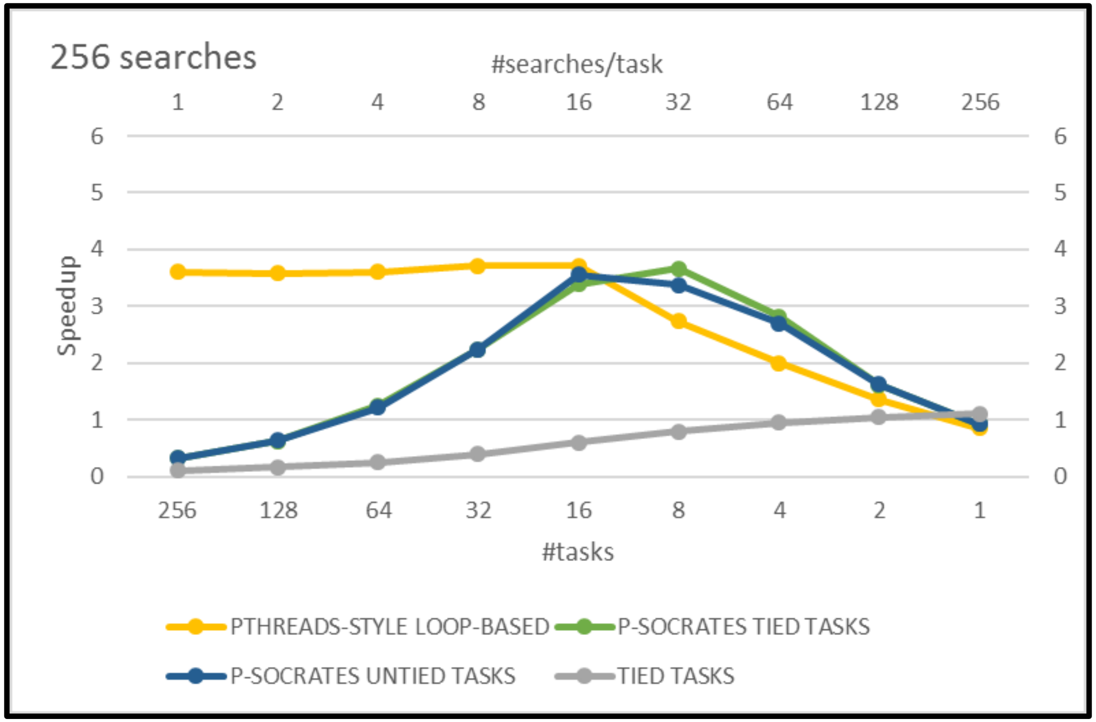
- Infra-red image processing



	Execution time (ms)			Speed-up		
	Sequential	**MPPA Native SDK**	**P-SOCRATES SDK**	**MPPA Native SDK**	**P-SOCRATES SDK**	**Maximum theorical**
TOTAL	**63140**	**8872,4**	**9093,2**	**7,1**	**6,9**	**256**
Cluster Phase 1	2710	162,64	150,34	16,7	18,0	48
Cluster Phase 2	562	147,56	120,08	3,8	4,7	8
IO Phase	612	612	612	1,0	1,0	1
Cluster Phase 3	8554	804,9	879,1	10,6	9,7	16
Cluster Phase 4	950	236,9	285,6	4,0	3,3	16

Results

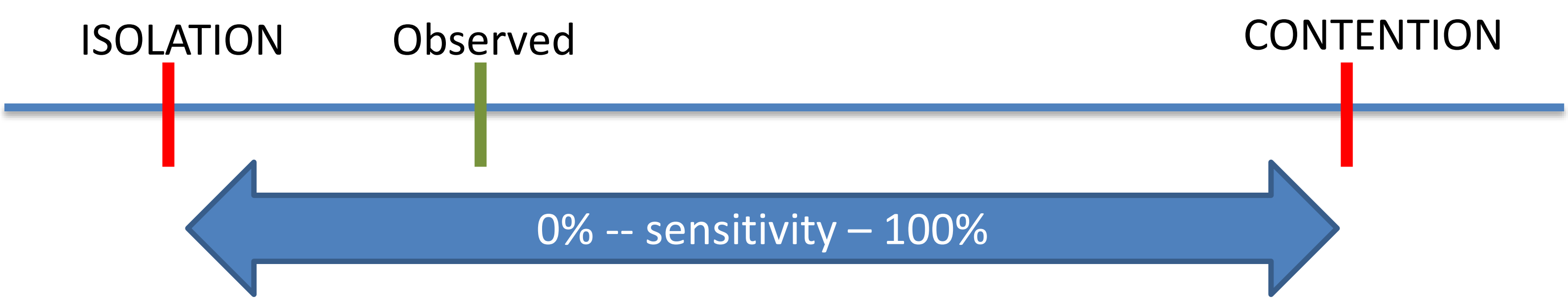
- Online text semantics



Results

- Online text semantics

ISOLATION (µs)	Observed (µs)	CONTENTION (µs)	Core	sensitivity
102	98	3 445	7	0%
98	241	3 490	6	4%
77	242	2 715	5	6%
89	238	2 463	4	6%
68	239	1 471	3	12%
68	239	1 478	2	12%
78	239	3 498	1	4%
72	71	3 484	0	0%



Conclusions

- Integrating time-predictability in high-performance embedded computing brings difficult challenges that need to be addressed
  - High-performance hardware and software stacks are not designed for predictability.
- The P-SOCRATES project tackled this important challenge by devising a methodology and the UpScale SDK
  - Allows to reason on the timing and schedulability analysis of real-time high-performance applications.
- The dynamic configuration approach achieves the same average performance than the default SDK
  - Static approach achieves higher **Guaranteed Performance**, with similar average performance (~10%)

Thank you

Post-project work partially supported by National Funds through FCT/MEC (Portuguese Foundation for Science and Technology) and co-financed by ERDF (European Regional Development Fund) under the PT2020 Partnership, within the CISTER Research Unit (CEC/04234).