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:** Parallel Software framework for time-critical Many-core Systems

• Three-year FP7 STREP project (Oct-2013, Dec-2016)

• **Website:** [www.p-socrates.eu](http://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:

[Logos of companies]

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
next-generation embedded many-core accelerators

real-time methodologies to provide time predictability

programmability of many-core from high-performance computing
Vision

[Diagram showing a flow of tasks, cores, threads, and application layers with notes on enhanced parallel programming models, mapping algorithms, and scheduling algorithms, with labels for static and dynamic processes.]
Innovation

• A **generic framework**, integrating models, tools and system software, to parallelize applications with **high performance** and **real-time** requirements.
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

Real-time OS

Parallel runtime

Scheduler

Mapper

Run-time tracing

P-SOCRATES (Grant agreement nº 611016)
Technical Approach

Execution model on heterogenous many-core

Offload of parallel computation

Accelerates

Host /IO Cluster

Accelerators / Computing Clusters
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

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

```c
#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

#pragma omp task { part50 } // T5
  part01
#pragma omp task { part60 } // T6
  part02
#pragma omp task { part70 } // T7
  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

Annotate every node with a WCET estimate

Compiler

Schedulability analysis
Methodology

- A new approach to tackle the interference problem

Not one but two WCET estimates

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
Annotate the graph with the WCET in CONTENTION
Annotate the graph with the WCET in CONTENTION

Compiler phase

Time

WCET

WCET

Map

Sched

Yes

No

15-03-2017

P-SOCRATES  (Grant agreement nº 611016)
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-1], 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

SUCCESS!

FAILUER!
Annotate the graph with the WCET in ISOLATION.
The big picture (static)

Annotate the graph with the WCET observed

SUCCESS!

0100010
1100101
1100110

User-guided

WCET OBS.

WCET ISO

WCET OBS.

WCET OBS.

Yes

No

Sched

Map

Time

36

WCET

ISO

Map

Yes

No

Sched

User-guided

0100010
1100101
1100110
All-in-one

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. Static' (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 directory
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 $\rightarrow$ set of 256ffts (size=512)
Results

- Infra-red image processing

<table>
<thead>
<tr>
<th>Cluster Phase</th>
<th>Execution time (ms)</th>
<th>Speed-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOTAL</td>
<td>Sequential 63140, 8872.4, 9093.2</td>
<td>7.1, 6.9, 256</td>
</tr>
<tr>
<td>Cluster Phase 1</td>
<td>2710, 162.64, 150.34</td>
<td>16.7, 18.0, 48</td>
</tr>
<tr>
<td>Cluster Phase 2</td>
<td>562, 147.56, 120.08</td>
<td>3.8, 4.7, 8</td>
</tr>
<tr>
<td>IO Phase</td>
<td>612, 612, 612</td>
<td>1.0, 1.0, 1</td>
</tr>
<tr>
<td>Cluster Phase 3</td>
<td>8554, 804.9, 879.1</td>
<td>10.6, 9.7, 16</td>
</tr>
<tr>
<td>Cluster Phase 4</td>
<td>950, 236.9, 285.6</td>
<td>4.0, 3.3, 16</td>
</tr>
</tbody>
</table>

P-SOCRATES (Grant agreement nº 611016)
Results

- Online text semantics

![Graph showing speedup of different strategies with varying numbers of tasks and searches per task.](image-url)
Results

- Online text semantics

<table>
<thead>
<tr>
<th>ISOLATION (μs)</th>
<th>Observed (μs)</th>
<th>CONTENTION (μs)</th>
<th>Core</th>
<th>sensitivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>102</td>
<td>98</td>
<td>3 445</td>
<td>7</td>
<td>0%</td>
</tr>
<tr>
<td>98</td>
<td>241</td>
<td>3 490</td>
<td>6</td>
<td>4%</td>
</tr>
<tr>
<td>77</td>
<td>242</td>
<td>2 715</td>
<td>5</td>
<td>6%</td>
</tr>
<tr>
<td>89</td>
<td>238</td>
<td>2 463</td>
<td>4</td>
<td>6%</td>
</tr>
<tr>
<td>68</td>
<td>239</td>
<td>1 471</td>
<td>3</td>
<td>12%</td>
</tr>
<tr>
<td>68</td>
<td>239</td>
<td>1 478</td>
<td>2</td>
<td>12%</td>
</tr>
<tr>
<td>78</td>
<td>239</td>
<td>3 498</td>
<td>1</td>
<td>4%</td>
</tr>
<tr>
<td>72</td>
<td>71</td>
<td>3 484</td>
<td>0</td>
<td>0%</td>
</tr>
</tbody>
</table>
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).