Demo

High-Performance Parallelisation of Real-Time Applications with the Upscale SDK

Luis Miguel Pinho

CISTER-TR-170204

2017/03/27
High-Performance Parallelisation of Real-Time Applications with the Upscale SDK

Luis Miguel Pinho
*CISTER Research Centre
Polytechnic Institute of Porto (ISEP-IPP)
Rua Dr. António Bernardino de Almeida, 431
4200-072 Porto
Portugal
Tel.: +351.22.8340509, Fax: +351.22.8321159
E-mail: lmp@isep.ipp.pt
http://www.cister.isep.ipp.pt

Abstract

Nowadays, the prevalence of computing systems in our lives is so ubiquitous that it would not be far-fetched to state that we live in a cyber-physical world dominated by computer systems. These systems demand for more and more computational performance to process large amounts of data from multiple data sources, some of them with guaranteed processing response times. In other words, systems are required to deliver their results within pre-defined (and sometimes extremely short) time bounds. Examples can be found for instance in intelligent transportation systems for fuel consumption reduction in cities or railway, or autonomous driving of vehicles. To cope with such performance requirements, chip designers produced chips with dozens or hundreds of cores, interconnected with complex networks on chip. Unfortunately, the parallelization of the computing activities brings many challenges, among which how to provide timing guarantees, as the timing behaviour of the system running within a many-core processor depends on interactions on shared resources that are most of the time not know by the system designer. P-SOCRATES (Parallel Software Framework for Time-Critical Many-core Systems) is an FP7 European project, which developed a novel methodology to facilitate the deployment of standardized parallel architectures for real-time applications. This methodology was implemented (based on existent models and components) to provide an integrated software development kit, the UpScale SDK, to fully exploit the huge performance opportunities brought by the most advanced many-core processors, whilst ensuring a predictable performance and maintaining (or even reducing) development costs of applications. The presentation will provide an overview of the UpScale SDK, its underlying methodology, and the results of its application on relevant industrial use-cases.
High-performance parallelization of real-time applications with the UpScale SDK

Luis Miguel Pinho
Outline

• P-SOCRATES at a glance
• Motivation and Vision
• Technical Approach
• 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

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
Vision

Enhanced Parallel Programming Models

Enhanced Mapping Algorithms

Enhanced Scheduling Algorithms

Application

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Task

Tas
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

Run-time tracing

P-SOCRATES (Grant agreement nº 611016)
Technical Approach

Execution model on heterogeneous many-core

Accelerates

Offload of parallel computation

Host /I/O 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
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
    part_00
  }
  #pragma omp task { // T1
    part_10
  }
  #pragma omp task { // T2
    part_20
    #pragma omp task { // T3
      part_30
      #pragma omp task { part_40 } // T4
      #pragma omp taskwait
      part_31
    }
    #pragma omp taskwait
    part_21
  }
  #pragma omp taskwait
  part_11
}
#pragma omp task { part_50 } // T5
#pragma omp task { part_60 } // T6
#pragma omp task { part_70 } // T7
#pragma omp task { part_04 }
```
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
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
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).