SPD 2018 –19 Course Introduction

Strumenti di programmazione per sistemi paralleli e distribuiti (SPD)

~

Programming Tools for Distributed and Parallel Systems

M. Coppola massimo.coppola@isti.cnr.it

Course structure

- Programming Tools for Parallel and Distributed Systems (SPD)
  - 2nd term (Feb. 2019- May. 2019)
  - **6** credits
  - 48hours : ~36 lessons, ~12 laboratory
  - Final test: lab project + oral examination
    - Includes discussing the project
  - New Course pages on didawiki :

    http://didawiki.cli.di.unipi.it/doku.php/magistraleinformaticanetworking/spd/start

    (Old pages are archived per year)

Overview

Description and Analysis of parallel and distributed programming platforms and models, to tackle problems of daunting size, scale and performance requirements

Parallelism at different levels of scale

- *Theoretical foundations*
- Standards for platforms and programming systems
- State-of-the-art solutions
- Practical use
- *Applications*

Course topics

- Parallel programming tools & platforms for HPC
  - HPC as well as large scalable systems: Clouds
- Many different parallelism levels
  - Many-core systems
  - Multiprocessor systems
  - Distributed Systems / Clusters
  - Clouds

Message Passing and Shared Memory

- **MPI** – Message Passing Interface
  - message passing standard
  - distributed memory
    - Cluster and Cloud computing
  - linked library
  - multi-language standard
    - C, C++, Fortran, more from 3rd parties
- **TBB** – Intel-Thread Building Blocks library
  - C++ template library
  - shared memory
  - multiple threads
  - aims at multi-core CPUs

High-Level Parallel Prog. Frameworks

- **OpenCL**
  - High-level approach to various kind of accelerators
  - **High-level approaches tied to GPU producers and dev-kit : CUDA**
  - Exploit Many-core on-chip parallelism targeted at graphics for general purpose programs
    - General Purpose GPU programming
    - Modern CPUs vector instruction support
    - Digital Signal Processors
    - APU development: soon to merge with standard programming?
    - Vulkan /Spir-v

- Other Structured Parallel Programming approaches
  - ASSIST
  - High-Level SPP language for Clusters/Clouds, dynamic and autonomic management
  - BSP-based approaches (e.g. Apache Hama / Giraph, or MulticoreBSP)
- Lower level structured parallelism for FPGA devices

Execution environments

- Ordinary multicore CPUs
- Large compact multicore CPUs
  - Intel Phi – Knights landing from the Unipi IT Center
- GPUs
  - Commercial and high-end devices (OpenCL or CUDA)
- Clouds, Clusters, multi / many-core systems
- FPGA devices
  - Exploring the option for OpenCL-to-FPGA
  - Recent advances on Open Source CPU Cores
    - RiscV, openRisc.

Prerequisite notions

- Computer architecture
  - CPU, memory hierarchy and caching, I/O, networking
- Basic parallelism patterns/skeletons
  - Structure and meaning
  - Use in programs
  - Abstract implementation
- Parallel performance models
  - use and analysis of standard ones,
  - basic skills at developing/refining models and verifying them against experimental data
- C / C++ knowledge is required in order to use the programming frameworks
- Example:
  - we will study a farm skeleton implemented on a technology
  - **assuming it is known** what a farm is and what is its standard implementation and model
  - experimentally evaluate results, possibly adapting the model

Links to other courses

- **HPC** is a prerequisite
  - High-performance Computing Systems and Enabling Platforms
- **SPM** Distributed systems: paradigms and models
  - SPM theoretical foundations, surveys of systems
  - SPD focuses on few programming systems + lab time
  - It's assumed that you at least followed the SPM course and attempt the exams in the right order; we will not re-tell basic notions from SPM

Other related courses

- PAD Distributed Enabling Platforms
  - PAD focuses on Cloud platforms, related programming tools
- AIP Parallel & Distributed Algorithms
  - ALP provides basics of parallel algorithmic cost models
- Any course related to QoS an SLA in {networking, virtualization, services}

Final test

1. Coding an individual project
   - Agree topic with the teacher, write 2-page summary
   - Project will use at least one of the frameworks and tools presented
     - E.g. MPI, or TBB+MPI, or OpenCL + TBB …
   - Submit -1- project proposal summary **before** and -2- a written report **after** the project work
     1. explains the problem, approach,
     2. explains design choices & work done, describes code results, analyzes test results and their modeling
   - Discuss project and report
     - may be in seminar form with the class, if so agreed
2. Discussion on course topics
   - Either together with or after project discussion, about any topic in the course program
3. Course evaluation (required by the administration)
   - Please submit by the end of the course semester (you will need to, eventually, before the final examination)

Examples of projects topics

- Parallel / distributed optimization resource allocation
  - Autonomic, adaptive mechanisms
- Parallel/distributed stream-based computation
  - Summarization, mining, learning
- Parallel/distributed mining / learning

Related M. Thesis Proposals

- Some of the previous topics may be expanded to Master thesis.
  - Either as stand-alone or as a development of the course project
  - Possibly multidisciplinary
    - e.g. optimization/parallelization of algorithms
- Contact the teacher during the course or when choosing the course project topic

Timetable

- 4 hours per week (standard)
  - Starting on 19/02/2019
  - Some lessons may be skipped due to work constraints, e.g. 12/3/2019
    - If so, they will be moved to a different day
    - See the course didawiki for rescheduling information
- Temporary timetable changes
  - *if needed* to get non conflicting time slot for all WIN students
  - slots which comply with official constraints
    - e.g. do not clash with fundamental courses of the other two C.S. curricula.

Main References

- Standard MPI 3.1
  - Only those parts that we will cover during the lessons
  - They will be specified in the slides/web site.
  - Available online :
    - http://www.mpi-forum.org/docs/mpi-2.2/mpi22-report.pdf
    - http://www.mpi-forum.org/docs/mpi-3.1/mpi31-report.pdf
- B. Wilkinson, M. Allen Parallel Programming, 2nd edition. 2005, Prentice-Hall.
  - This book will be also used; the 1st edition is ok as well and it is available in the University Library of the Science Faculty, [C.1.2 w74 INF]
- M. McCool, A. Robinson, J. Reinders Structured Parallel Programming – Patterns for Efficient Computation 2012, Morgan Kaufmann
  - Useful as a comprehensive guide for TBB. However it is redundant with SPM; Cilk is not a topic of the SPD course.

- Reading the slides are not enough to pass the course
  - Should be obvious: take notes, check the references on the web site and look for them on your own when working out the exercises

Laboratory

- First exercises with laptops
- Ok for development with most of the programming tools (MPI, TBB, GPGPU, etc…)
- For testing, options are
  - Clusters/servers from the C.S. Dept.
  - Intel PHI boards at the C.S. Dept.
  - Other devices on a case-by-case

Provisional Timetable

- *Initial* timetable
  - Monday 14-16 (Room C1)
  - Friday 16-18 (Room N1)

- Question time
  - Thu 15-17
  - at CNR Research Area buildings, Room C33 (entrance 19)
  - As the number of students is usually small, we can agreed on a different time. Please ask the teacher during the lessons or by email
  - Check the *official timetable* and the one on the course didawiki page for updates