SPD 2015 –16 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. 2016- May. 2016)
  - **6** credits
    - Note: old SPD, 9 credits, can still be taken only *if you have it already on your study plan*
  - 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 and also large scalable systems: Clouds
- Many different parallelism levels
  - Many-core systems
  - Multiprocessor systems
  - Distributed Systems / Clusters
  - Clouds (& Grids)

Message Passing and Shared Memory

- **MPI** – Message Passing Interface
  - message passing standard
  - Cluster and Cloud computing
  - linked library
  - Support for several languages
    - C, C++, Fortran + several 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
  - Exploit Many-core on-chip parallelism targeted at graphics for general purpose programs
    - General Purpose GPU programming
    - High-level approaches tied to GPU producers and dev-kit : CUDA and Brooks+
    - Modern CPUs vector instruction support
    - Digital Signal Processors
    - APU development: soon to merge with standard programming?
    - Vulkan /Spir-v 1.0

- ASSIST and other Structured Parallel Programming approaches
  - High-Level SPP language for Clusters/Clouds, dynamic and autonomic management
  - BSP-based approaches (e.g. Apache Hama / Giraph, or MulticoreBSP)

Execution environments

- Ordinary multicore CPUs
- Large compact multicore CPUs (Intel Phi)
- General purpose computing enabled GPUs
- Clouds, Clusters, multi / many-core systems
  - *Contrail*
    - *Federation of Clouds*
  - *Specific open-source Cloud platforms:*
    - *OpenNebula*
    - *OpenStack*

Links to other courses

- **SPA** 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

QoS an SLA in {networking, virtualization, services}

- SRT Real time systems
- P2P Peer to Peer Systems
- MOR Network Optimization Methods

Important prerequisite notions

- Computer architecture
- Basic parallelism patterns/skeletons
  - Structure and meaning
  - Use in programs
  - Abstract implementation
  - Performance models
    - use and analysis of standard ones,
    - basic skills at developing/refining models and verifying them against experimental data
- Example: we will study how a farm skeleton can be best implemented on tech. X, ***starting from*** the knowledge of what a farm is and what is its standard implementation and model.
- C / C++ knowledge is required in order to use the programming frameworks

Related M. Thesis Proposals

- Master Thesis on SPD-related topics are possible
  - 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 23/02/2016
  - Some lessons will be skipped due to work constraints (e.g. *possibly* 8-9/03/2016)
    - If so, they will be moved to a different day
- 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 can as well do, and it is available in the University Library of the Science Faculty, [C.1.2 w74 INF]
- M. McCool, A. Robinson, J. Reinders Strctured 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
  - Labs of the C.S. Dept. used as a cluster
  - Intel PHI boards at the C.S. Dept.
  - Other devices on a case-by-case
- Survey: which virtual machine manager do you prefer?
  - VirtualBox, Vmware …?
  - In order to provide a standard VM for the first programming tests

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) about the project
     1. explaining problem, approach,
     2. explain design choices & work done, describe code results, analyze test results
   - 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)

Provisional Timetable

- Initial timetable
  - Tuesday 11-13 (N1)
  - Wednesday 16-18 (N1)

- Question time
  - Thu 16-18
  - at CNR Reseach Area buildings, Room C33 (entrance 19)
  - As there are only a few students, we can agreed on a different time, please ask during the lessons/ by email
  - Check the official timetable and the didawiki page for updates