SPD 2014 –15 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. 2015- May. 2015)
  - **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 : www.cli.di.unipi.it/doku/magistraleinformaticanetworking/spd

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?

- 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 Cloud platforms:
  - OpenNebula
    - Open Source European Platform
    - Implementation and APIs
  - 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
- **PAD** Distributed Enabling Platforms
  - PAD focuses on Cloud/Grid platforms, related programming tools

Other courses

- 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 24/02/2015
  - Some lessons will be skipped due to work constraints (e.g. possibly 10/03/2015)
    - To be moved to a different day
- Temporary timetable changes, *if needed*
  - We need to get non conflicting time slot for all WIN students, as well as
  - slots which do not clash with fundamental courses of the other two C.S. curricula.

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

- 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 (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
- Plus oral test
  - Together with / after project discussion, about any topic in the course program
- Course evaluation
  - Please submit by the end of the course semester (you will need to, in order to take the examination)

Provisional Timetable

- Initial timetable
  - Tuesday 11-13 (N1)
  - Wednesday 14-16 (N1)

- Question time
  - Thu 16-18
  - at CNR Reseach Area buildings, Room C33 (entrance 19)