SPD 2018 –19
COURSE INTRODUCTION

Programming Tools for Distributed and Parallel Systems

Strumenti di programmazione per sistemi paralleli e distribuiti (SPD)

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

Course structure

- Programming Tools for Parallel and Distributed Systems (SPD)

  - 2nd term (Feb. 2021- May. 2021)
  - **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

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
  - Clouds
  - Distributed Systems / Clusters
  - Multiprocessor systems
  - Many-core systems
  - Specialized multicores: GPU
  - Reconfigurable Hardware : FPGA

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 are often tied to chip producers and their dev-kit : e.g. CUDA
  - Exploit Many-core on-chip parallelism for general purpose programs
    - General Purpose GPU programming
    - Modern CPUs vector instruction support
    - Digital Signal Processors
    - Vulkan / Spir-V
  - **SYCL**
    - Single source C++ code for transparent OpenCL exploitation
    - on CPU as well as on all kind of supported accelerator devices: GPU, FPGA…

High-Level Parallel Prog. Frameworks

- **oneAPI**
  - Umbrella project or unifying methodology?
  - Encapsulates several other frameworks: DPC++, OpenMP, SYCL, TBB into a common API
    - it is expected to support a broad range of parallel computing devices, including GPUs and FPGAs

- 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)

- Low-level structured parallelism for FPGA devices

Execution environments

- Ordinary multicore CPUs
- GPUs
  - Commercial and high-end devices (OpenCL or CUDA)
- Clouds, Clusters, multi / many-core systems
- FPGA devices
  - Exploit the options of oneAPI to FPGA, or OpenCL-to-FPGA
  - There are recent advances on Open Source CPU Cores
    - RiscV, openRisc.

- Support tools
  - Using the **SLURM** Workload Manager
  - **Python** as a scripting mechanism for HPC applications

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
- verifying models against experimental data

C / C++ knowledge
- required in order to use the programming frameworks

Prerequisite notions

- Example:
  - We may study a farm skeleton implemented on a given technology (SW+HW)
  - We will assume
    - it is known what a farm skeleton is
    - what is its purpose
    - and what are its standard implementation and performance model
  - We will require from the students
    - to learn how to code the farm implementation on the technology
    - to learn how to apply/customize the performance model to the technology
    - to design experiments that can validate their model and its basic assumptions
    - to experimentally evaluate results, possibly revising the model and/or identifying issues within the implementation

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
- PAD Distributed Enabling Platforms
  - PAD focuses on Cloud platforms, distributed programming, containers, related programming and management tools

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
     - oneAPI is a special case
   - Submit **-1-** project proposal summary before and **-2-** a written report after the project work
     - explains the problem, your approach; explains design choices & work done, describes code results, analyzes test results and their modeling
   - Discuss project and report
2. Discussion on course topics
   - Either together with or after project discussion, about any topic in the course program

- Course evaluation (required by the administration)
  - Please submit by the end of the course semester

Examples of projects topics

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

- 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

Timetable

- 4 hours per week (standard)
  - Starting on 17/02/2021
  - Some lessons may be skipped due to work constraints
    - If so, they will be moved to a different day
    - See the course didawiki for rescheduling information
    - This year we already skipped the first lesson due to technical issues with the online teaching support
- Timetable changes
  - *if needed* to get non conflicting time slot for all WIN students
  - only as a last resort
  - 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.
- M. Voss, R. Asejo, J. Reinders – Pro TBB Book code samples ported to oneAPI -- Springer open access
  - Useful as reference to use TBB and oneAPI
- J. Reinders et al. - Data Parallel C++ -- Springer open access
  - May be used during the course

- Reading the slides is 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

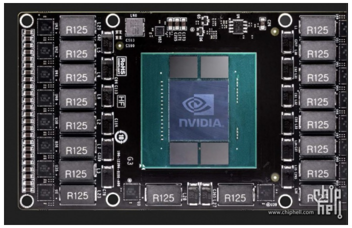
- Practice on your laptops/desktop
  - Ok for development with most of the programming tools MPI, TBB, GPGPU, etc…
- For execution, testing and actual experiments
  - Virtual Cluster / devices from the University ITC
  - Still in the arrangement phase, details to be provided soon

Provisional Timetable

- Initial timetable
  - Monday 14.15-16 WTW/2
  - Wednesday 16.15-18 WTW/2

- Question time
  - TBD
  - Via telco, possibly a channel on the course MS teams

Programming Tools for Distributed and Parallel Systems (SPD)



- Goal: learn to choose and use programming tools that exploit parallelism at different levels: data-center, multi-processor, multicore and GPU/FPGA
- Distributed and parallel processing
- Apply performance and behavioral models
  - Problem analysis and solution design
  - Abstract modelling → experimental evaluation → critical analysis
- **Exam:** project with written report + oral discussion
- **Period:** second semester, 4h/week

Programming Tools for Distributed and Parallel Systems (syllabus)

- Standard tools and frameworks
  - Distributed / parallel programming with MPI (Message Passing Interface)
  - Multithreaded programming with oneTBB (Thread Building Blocks)
  - Support Tools
- OneAPI and other unifying approaches to multiprocessing/manycores and on-chip parallelism
  - OpenCL, SyCL, TBB; ROC
  - Targets: multi-core CPU, CPU vectorization, GPUs, APUs, FPGA devices
- Application examples:
  - Data Mining, Deep Learning, Graph / Optimization Algorithms
  - Distributed and parallel compute- and data-intensive algorithms
  - Multithread, high-memory bandwidth algorithms