SPD 2025 –26
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. 2026- May. 2026)
  - 6 credits
  - 48hours : ~36 lessons, ~12 laboratory
  - Final test:
  - Option 1 lab project + oral examination
    - Includes writing a report on the project and discussing it
  - Option 2 prepare a presentation on an agreed topic
    - Includes writing a short report
  - 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, SYCL**
  - 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 and more
  - An abstraction on top of OpenCL that developed into a programming approach

High-Level Parallel Prog. Frameworks

- **oneAPI**
  - Umbrella framework
  - Encapsulates several other frameworks: DPC++, OpenMP, SYCL, TBB into a common developer framework and set of APIs
    - supports a broad range of parallel computing devices (including GPUs, 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
- ROCm Open SW stack

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
  - Different perspective
- Data Center Design and Operation
  - platform design for scalability

Final test – Option 1

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, or SyCL
     - oneAPI is a special case
   - Submit the **-1-** project proposal summary (before) and **-2-** a written report (after) about 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

Final test – Option 2

1. Prepare and present a topic based presentation
   - Agree topic with the teacher, examples are
     - a paper or small set of papers on a research topic related to the course
     - a specific technology akin but not discussed in the course
   - Submit a short, written report and prepare a presentation in seminar form
     - about 30 minutes, 20-30 slides
2. Presentation, and discussion on course topics

- 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 16/02/2026
  - A few lessons may be skipped due to conflicting work constraints
    - If so, they will be moved to a different day
    - See the course didawiki for rescheduling information, or contact teacher
- Timetable changes
  - if needed to get non conflicting time slot for all WIN students
  - report conflicts via email
  - we won't change twice, so please inform the teacher about your other constraints
  - 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
  - Only those parts that we will cover during the lessons, they will be specified in the slides/web site.
  - Available online in different revisions and formats
    - http://www.mpi-forum.org/docs/
- M. Voss, J. Reinders - Today's TBB (C++ Parallel Programming with Threading Building Blocks).
  - Springer Open access book available at Today's TBB
- B. Wilkinson, M. Allen Parallel Programming, 2nd edition. 2005, Prentice-Hall.
  - This book will be also used; the 1st edition available in the University Library of the Science Faculty
- 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
- Academic papers on specific topics will be distributed/referenced via the wiki when needed
- 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
  - As well as on CNR/ISTI premises
  - Still in the arrangement phase, details to be provided soon

Provisional Timetable

- Initial timetable
  - Thursday 09.00-11.00 Fib X3
  - Friday 14.00-16.00 Fib X3
- Question time
  - TBD
  - Via telco