Programming Tools for Distributed and Parallel Systems (SPD)

Teacher	Massimo Coppola
Contact	massimo.coppola@isti.cnr.it , 050 621 2992
Value, period	6 credits – 4 hrs per week, 48 total – 2nd semester
Exam rules	lab project + written report + oral discussion (syllabus and project)
Pre-requisites	HPC; SPM is strongly suggested
Area	Computer Science
Course home page	hosted on https://didawiki.cli.di.unipi.it

- The course presents a selection of parallel and distributed programming languages and frameworks, covering parallelism exploitation at different scales.
- We address exploitation of parallelism via software at different architectural levels, targeting distributed systems, shared-memory/multicore CPUs and GPUs
- The course relies on knowledge about parallel skeletons, their performance models, as well as techniques to exploit them in the design and evaluation of parallel software.

Why follow the SPD course?

Develop your skills related to parallel, multithreaded, high-performance programming

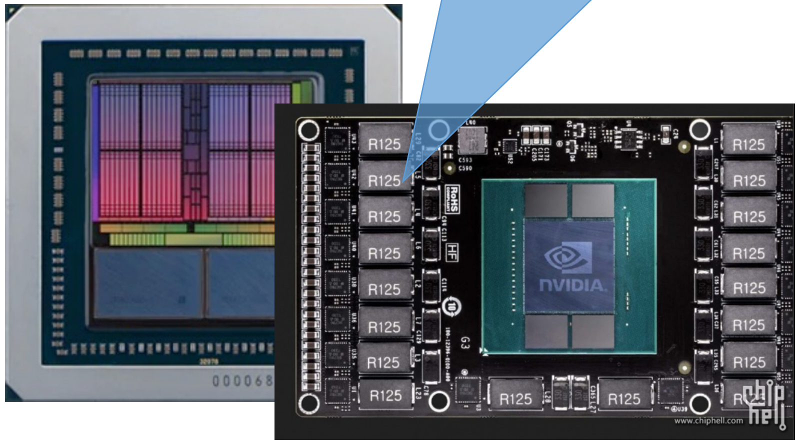
- Problem analysis and solution design for parallel and distributed applications
- Abstract modelling, experimental evaluation and critical analysis of performance, parallel scalability, efficiency

A changing landscape where parallelism is pervasive

Industry standard CPUs
Contain O(10) cores
Possibly hyperthreading
Complex, layered caches

Fujitsu Supercomputer Fugaku
New TOP500 1^{st} place 22/06/20
158K+ nodes based on
48-core A64FX ARM SoC

Nvidia Pascal – Turing GPUs
AMD Vega Pro 64
1K –5K GPU cores on-chip



RISC-V FPGA CPUs up to 1680 cores/board

FPGA as a tool to experiment in CPU/GPU design

Home made cluster of 120 Raspberry PI (ARM 32 bit core)

Syllabus

Parallel tools & platforms for HPC and large scalable systems. Lessons + lab time

- MPI – Message Passing Interface standard
  - Message passing standard, linked library with support for multiple languages
- TBB – Intel Thread Building Blocks library
  - C++ template library for shared memory multi-thread programming
  - Multi core CPUs and multiprocessor systems
- OpenCL – High-level, portable standard to exploit many-core on-chip parallelism
  - Multithread, high-memory bandwidth algorithms with streaming/regular access patterns
  - Targets graphic units (GPUs), CPU vectorization, APUs, FPGA devices …

- Other frameworks to be considered
  - Change yearly and may be related to projects, examples are Vulkan, CUDA (NVIDIA), ROC (AMD), OneAPI (INTEL); BSP-based and Map&Reduce frameworks: Spark, Graphx, Hama
- Application examples for laboratory time (change from year to year):

  Data Mining, Deep Learning, Graph / Optimization Algorithms, Stream Data Processing

Topics for Master Thesis or Research fellowships

- Clouds, Cloud-Federations and *Edge* / *Fog* computing:
  - Dynamical System Modeling, Resource Brokering, Scheduling Optimization strategies
    - Hierarchical and skeleton-based programming frameworks and performance models
    - Genetic programming, (mixed integer) linear programming, other optimization approaches to brokering and autonomic/adaptive resource management
    - Container-based and VM-based application composition, deployment and elastic scalability
    - High-performance implementation of authorization mechanisms for data security and privacy: Scalable policy evaluation and enforcing mechanisms at the hypervisor, cloud and/or federation manager levels as well as on edge devices

- Multicore CPU/GPU design and deployment on FPGA
  - HW design, digital signal synthesis/analysis, AI-accelerator design
- High-performance computing applications
  - HPC / distributed Data Mining, Stream Mining, Machine Learning, Deep Learning
  - Applications to HealthCare
  - Application of stream and Big-data Analysis for Clouds