IE_CSC 2. Advanced topics in scientific and parallel programming with practical application to the CAPRI HPC infrastructure

Course Area: Information Engineering

Credits: 5 (20 hours)

Instructor: Giacomo Baruzzo, Department of Information Engineering, University of Padova

e-mail: giacomo.baruzzo@unipd.it

Aim: Provide basic skills for working on remote servers, using/developing parallel software and deploying it on a containerized computer server. The course gives basic introduction to modern computer architecture and to the most important parallel programming paradigms: Multi-threading, OpenMP, MPI and CUDA with examples (mostly Python and C++). The course covers basic tools to access and to interact with remote servers, to manage remote resources, and to manage jobs. The course introduces principles of software containerization from the perspective of users, providing practical examples of Singularity/Docker. The concepts discussed are applied to simple case of studies involving writing and/or running parallel programs using the CAPRI HPC infrastructure (256 cores, 6TB shared RAM and 2 GPU Nvidia P100) recently acquired by the University of Padova for research activities.

Topics:

1. How to use a computing server (application to CAPRI)
   a. Introduction to High Performance Computing (HPC hardware and architectures, HPC software, supercomputers)
   b. Job scheduling (slurm; writing a job; running, stopping and querying status of a job)
   c. The CAPRI queuing system and policy (CAPRI hardware and architecture; access to CAPRI and projects; execution queue; how to choose queue)
2. Containerization (Singularity)
   a. Overview of containerization (definition of containers and container daemon; Singularity and Docker software; containers vs virtual machines; advantages: re-usability and reproducibility, flexibility, efficiency; disadvantages: learning curve)
   b. Using container that have already been defined (running, stopping, and resuming containers; containers options and flags)
   c. Defining new containers (new containers from scratch; extending existing containers)
   d. Sharing containers and the container repository (browsing and adding a container to the repository; guidelines for creating and documenting containers to be shared)
3. Version control (git)
   a. Basic operations (create a git repository, staging and committing changes, repository status and history, work with branches)
   b. Advanced operations and remote repository (clone a remote repository, work with a remote repository, GUI for git, git web-based hosting services)

[back to **Summary**]

4. Parallel architectures and multi-process/parallel programming
   a. Introduction to parallel programming models and architectures (basic definitions; shared vs distributed memory architectures; threading: CPU and GPU; shared memory programming; message passing programming; performance metrics)
   b. Parallel programming languages and frameworks (multi-threading; OpenMP; MPI; CUDA)
5. Hands on example (a simple parallel software for data analysis / machine learning / numerical analysis; students' proposals)

References:

- Eijkhout, V. (2013). Introduction to High Performance Scientific Computing. Lulu. com.
- Grama, A., Kumar, V., Gupta, A., & Karypis, G. (2003). Introduction to parallel computing. Pearson Education.
- Parhami, B. (2006). Introduction to parallel processing: algorithms and architectures. Springer Science & Business Media.
- Ad-hoc material by Lecturer

Schedule and room: please, see Class Schedule

Course requirements: Basics usage of tools for run/develop of scientific software (preferable unix/linux platforms)

Enrollment: add the course to the list of courses you plan to attend using the Course Enrollment Form (requires SSO authentication) and, if you are taking the course for credits, to the Study and Research Plan.

Examination and grading: Each student must produce a small parallel and containerized software (either predefined or custom built container) related to her/his research field. Each student can either a) write a simple parallel software with one of the programming paradigm presented during the course using a language of choice or b) choose a (possibly parallel) software typically used in the research activity. Containerized software must run on the CAPRI server.

[back to **Summary**]