PHYS-743 **Parallel programming**

Lanti Emmanuel, Richart Nicolas

Cursus	Sem.	Type
Physics		Opt.

Language of teaching	English
Credits	3
Session	
Exam	Oral presentation
Workload	90h
Hours	**56**
Courses	20
Exercises	20
TP	16
Number of positions	**15**

Frequency

Every year

Remark

Next time: Fall (Block course)

Summary

Learn the concepts, tools and API's that are needed to debug, test, optimize and parallelize a scientific application on a cluster from an existing code or from scratch. Both OpenMP (shared memory) and MPI (distributed memory) paradigms are presented and experimented.

Content

Students are asked to bring their own computer.
This course is a two weeks long course open to all the PhD students with a basic background in programming. The lecture examples and exercises will be provided in simple C++, but could also be provided in C or Fortran if needs be. The project will be done in one of the languages supporting the concepts presented during the class, e.g. C, C++, Fortran, Python.

The first week is dedicated to the theoretical concepts and practical sessions lead by teaching assistants. Starting with the concepts of the hardware of one node of a cluster (memory hierarchy, instructions, networking, etc..), you will learn how to debug and optimize a sequential code to be run efficiently on one node of a cluster. You will go through the entire Software Development Cycle loop (analyze, implement, debug, and test) using standard GNU and proprietary tools (gdb, valgrind, perf, Intel amplifier).

Secondly, the two standard industrial parallel paradigms targeting the shared and distributed memory models are introduced: OpenMP (shared) and MPI (distributed). You will learn the basic concepts of the parallel programming. The goal is to learn how to choose the right paradigm for an application based on its needs.

The OpenMP standard will be presented in its 3.1 version (without the hardware targets nor the SIMD constructions) from the basic loop parallelization directives to the tasks concepts.

The MPI standard will be covered in details up to the standard 2 and some intersting concepts from version 3 will also be presented. You will learn how to correctly use blocking and non-blocking point-to-point and collective communications as well as advanced datatypes, one-sided communications, dynamic process management and parallel I/O.

Throughout all the first week, we'll take practice the learned concepts by using a simple toy application: a 1D Poisson solver (using a finite difference scheme). From a fully bugged sequential implementation, you'll optimize it, parallelize it using OpenMP and MPI to come up finally with an optimized hybrid implementation of the same solver.

Please note that this course will not tackle the programming of accelerators such as GPU's.

The second week is dedicated to a personal project on your own (or provided) application.

CONTENT OF THE COURSE

Week 1: Lecture + Practical session
Day 1 & 2: Optimization of a sequential code

- Compilation basics, usage of standard development tools
- Software development life cycle (debug, profile, optimize)
- Parallelization concepts

Day 3: Parallelization on a shared memory node

- NUMA nodes architecture
- Parallelization using OpenMP: Directives; Runtime Library Routines; Environment variables; Affinity: a major issue for high performance

Day 4: Parallelization on a distributed memory cluster: basic concepts

- Blocking and non-Blocking point-to-point communications
- Blocking collective communications

Day 5: Parallelization on a distributed memory cluster: advanced concepts

- Advanced MPI types. Communicators and groups. Dynamic Process management, virtual topologies
- Persistent communications and One-sided communications (RMA's)
- Parallel I/O with MPI
- Non-Blocking collective communications
- Introduction to hybrid programming

Week 2: Personal project
In order to use all the concepts learned during the first week, a one-week long project is due. You can bring your own application or we'll provide you one. Examples of projects follow:

- Mandelbrod set
- Travelling salesman problem
- N-body simulation

The work time budget for the project is 32 hours of individual work plus 8 hours in contact with an assistant. You'll have 15 minutes to present your project with 5 additional minutes for questions & answers.

Keywords

OpenMP, MPI, HPC, Parallel programming

Learning Prerequisites

Required courses

- Basic knowledge of C, C++, Fortran or Python.
- Basic knowledge of Linux and bash scripting

Recommended courses

- Scientific Programming for Engineers, MATH-611.

Learning Outcomes

By the end of the course, the student must be able to:

- Optimize sequential and parallel codes
- Implement algorithms in parallel with OpenMP and MPI
- investigate the performances of parallel code

Resources

Notes/Handbook

By the end of the course, the student must be able to:

- Optimize sequential and parallel codes
- Implement algorithms in parallel with OpenMP and MPI
- Investigate the performances of parallel code

Moodle Link

- https://go.epfl.ch/PHYS-743