PHYS-743

Parallel programming

Keller Vincent, Richart Nicolas

Cursus	Sem.	Type
Physics		Opt.

Language of teaching	English
Credits	3
Session	
Exam	Multiple
Workload	90h
Hours	**48**
Courses	20
Exercises	20
TP	8
Number of positions	**8**

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

This course is a two weeks long course. It is open to all the PhD students with a solid background in programming.
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 – test) using standard GNU and proprietary tools (gdb, valgrind, gprof, 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 up to the standard 3. 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 or Intel Xeon Phi's.
The second week is dedicated to a personal work on your own (or provided) application.
CONTENT OF THE COURSE
Week 1: Lecture + Practical session
Day 1: Optimization of a sequential code
- Compilation basics, usage of standard development tools
- Software development life cycle (debug – profile – optimize)
- Parallelization concepts

Day 2: 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 3: Parallelization on a distributed memory cluster: basic concepts

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

Day 4: 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

Day 5: Hybrid programming (OpenMP + MPI)

- Introduction to hybrid programming
- How to write a (successful) research project proposal in a large data center

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

Strong knowledge of C, C++ or Fortran 90
Basic knowledge of Linux and bash scripting

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

- http://moodle.epfl.ch/course/view.php?id=13817