UE Advanced parallel system

Niveau d'étude
Bac +5

ECTS
6 crédits

Crédits ECTS Echange
6.0

Composante
UFR IM2AG (informatique, mathématiques et mathématiques appliquées)

Période de l'année
Automne (sept. à dec./janv.)

- **Langue(s) d'enseignement:** Anglais
- **Ouvert aux étudiants en échange:** Oui
- **Crédits ECTS Echange:** 6.0
- **Code d'export Apogée:** GBX9MO59

Présentation

Description

Today, parallel computing is omnipresent across a large spectrum of computing platforms, from processor cores or GPUs up to Supercomputers and Cloud platforms. The largest Supercomputers gather millions of processing units and are heading towards Exascale (a quintillion or 10^18 flops - http://top500.org). If parallel processing was initially targeting scientific computing, it is now part of many domains like Big Data analytics and Deep Learning. But making an efficient use of these parallel resources requires a deep understanding of architectures, systems, parallel programming models, and parallel algorithms.

Overview:

The class is organized around weekly lectures, discussions and help time. The presented materials that will be available each week on the class web page. To get practical experience and good understanding of the main concepts the students are expected to develop short programs and experiments. Students will also have to prepare a presentation or a written report on research articles. Students will have access to Grid'5000 parallel machines and the SimGrid simulator for experiments.

This class is organized around 2 main blocks:

Overview of parallel systems:

1. • Introduction to parallelism from GPU to supercomputers.
   - Hardware and system considerations for parallel processing (multi-core architectures, process and thread handling, cache efficiency, remote data access, atomic instructions)
   - Parallel programming: message passing, one-sided communications, task-based programming, work stealing based runtimes (MPI, Cilk, TBB, OpenMP).
   - Modeling of parallel programs and platforms. Locality, granularity, memory space, communications.
   - Parallel algorithms, collective communications, topology aware algorithms.
   - Scheduling: list algorithms; partitioning techniques. Application to resource management (PBS, LSF, SGE, OAR).
   - Large scale scientific computing: parallelization of Lagrangian and Eulerian solvers, parallel data analytics and scientific visualization.
   - AI and HPC: how parallelism is used at different levels to accelerate machine learning and deep learning using supercomputers.

Functional parallel programming:

We propose to study a clean and modern approach to the design of parallel algorithms: functional programming. Functional languages are known to provide a different and cleaner programming experience by allowing a shift of focus from "how to do" on "what to do".
If you take for example a simple dot product between two vectors. In c language you might end up with:

unsigned int n = length(v1); double s = 0.0; for (unsigned int i = 0 ; i < n ; i++) { s += v1[i] * v2[i]; }

In python however you could write:

return sum(e1*e2 for e1, e2 in zip(v1, v2))

You can easily notice that the c language code displayed here is highly sequential with a data-flow dependence on the i variable. It intrinsically contains an ordering of operations because it tells you how to do things to obtain the final sum. On the other end the python code exhibits no dependencies at all. It does not tell you how to compute the sum but just what to compute: the sum of all products.

In this course we will study how to express parallel operations in a safe and performant way. The main point is to study parallel iterators and their uses but we will also consider classical parallel programming schemes like divide and conquer. We will both study the theoretical complexity of different parallel algorithms and practice programming and performance analysis on real machines.
All applications will be developed in the RUST programming language around the Rayon parallel programming library.
No previous knowledge of the rust language is required as we will introduce it gradually during the course. You need however to be proficient in at least one low level language (typically C or C++)

Heures d'enseignement

CM	CM	36h

Pré-requis recommandés

Students should have some base knowledge on parallel programming (some MPI or OpenMP for instance) and experience in at least one low level language (typically C or C++). No specific skills on system, processor architecture or theoretical models beyond

the base training that any computer science M1 student should have received. Students should have a taste for experimenting with advanced computer systems and ready to be exposed to a few theoretical models (mainly cost models for reasoning about parallel algorithms).

Période : Semestre 9

Compétences visées

This class will progressively enable attendees to master advanced parallel processing. This class prepare students to pursue a M2R internship in related topics in an academic or industrial research team to next pursue a PhD in computer science or to work into companies on parallel or large systems.

Infos pratiques

Contacts

Responsable pédagogique

Bruno Raffin

bruno.raffin@inria.fr

Campus

Grenoble - Domaine universitaire