Introduction to parallel programming in economics and finance–
a tentative Syllabus

Simon Scheidegger
DBF, University of Zurich
and Hoover Institution, Stanford
simon.scheidegger@gmail.com

Open Source Macroeconomics Laboratory (OSM Lab) – Boot Camp
2017
BFI–University of Chicago

Abstract

Parallel computation has, next to theory and experiments, become in many fields a well-established and recognized third pillar of science within the last couple of years. The emerging computing power available to researchers nowadays reaches up to Petaflop/s, which is several million times faster than an average laptop. This allows for computer-aided discoveries that would otherwise be impossible. Also in modern quantitative economics and finance, parallel computing has become a key workhorse. Indeed, processing for example enormous sets of data in a timely fashion is only possible through massive parallel computing resources. Moreover, many relevant applications in quantitative finance, such as the computation of large portfolio risks or the pricing of complex financial derivative products, are heavily accelerated by means of appropriate hardware.

However, tapping parallel hardware resources efficiently is challenging. Indeed, for the greater part of computer history software has been written with a serial von Neumann computer architecture in mind, in which a computer program – in whatever programming language it is written – is ultimately translated into a stream of instructions that are executed sequentially. Parallel programming is fundamentally different. A key task of the software developer is to identify parts of an algorithm that can be run concurrently or to transform a given serial algorithm into an algorithm suitable for concurrent execution. An illustration of the interacting fields is shown in figure 1.

This course is intended to provide students in economics and finance with a self-contained introduction to the extensive and broad topic of parallel computing, with a special focus on relevant applications. Topics covered include shared memory, distributed memory, hybrid parallel programming software developement, and the good scientific conduct necessary to deal with the results from numerical applications. A key part of the course is devoted to hands-on labs based on smaller exercises as well as larger projects.

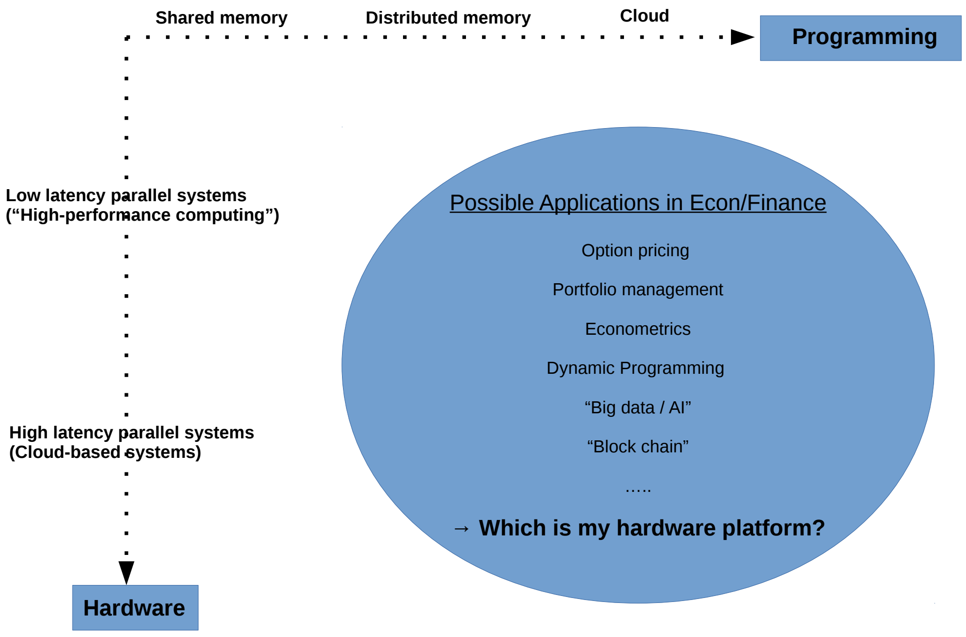

Figure 1: Interaction of hardware, software and applications in economics and finance.

1 Prerequisites:

Participants should have prior knowledge in a programming language (e.g. C++, C, Fortran, Julia or Python). Participants are encouraged to bring their own research projects to be discussed.

2 Goals of the course:

- Familiarize with parallel hardware resources available (high-latency systems; low-latency systems, cloud, etc).
- Understand which is the right platform to use in order to enable, accelerate or enhance applications.
- Learn about the most common programming paradigms that are available to harness the power of parallel computation.
- Start to think 'parallel'.
- Be aware of useful libraries (that are available to perform parallel computations).
- Work on smaller code examples plus larger projects; gain hands-on experience on systems beyond the scale of a laptop.

3 Day 1: Tuesday, July 25th, 2017

3.1 Introduction to parallel and high-performance computing (8.00-9.30)

The goal of this lecture is to provide a high-level introduction to the basic concepts of parallel programming. It will cover the emerging, available parallel hardware and how it can and should be used. Furthermore, performance concepts such as Amdahl's law be introduced. An important part of the lecture will be the high-level link between the hardware and possible applications.

3.2 Introduction 'unix-like' environments; 'Midway' as example of a parallel computing infrastructure (10.00-10.45)

Scientific computing / number crunching is nowadays often carried out on a server or cluster. In this tutorial-like session, we will familiarize with the compute environment of clusters by using 'Midway'. We will learn how to work from a terminal (basic unix commands such as ls, cp, scp; login via ssh, submit jobs via slurm, work with an editor...).

3.3 An ultra-short introduction to CPP (10.55-11.15)

Even though there are various options on how to use parallel compute power (e.g. in Python, Matlab or Julia), the entire universe of resources still can only be addressed by low-level languages. For those who are not familiar with either Fortran or CPP, we will give a very quick introduction to CPP that later on, the provided examples can be run and modified [3]. CPP is also the starting point when one wants to efficiently use GPUs.

3.4 Introduction to larger projects (11.20-11.50)

In order to become acquainted with the parallel programming paradigms introduced in the course of the workshop, the attendants of the workshop should work on one larger project. The results will be presented in the last session of the workshop. The projects will consist of "option pricing" and "dynamic programming".

This lecture will introduce two larger projects on which the participants should work on in teams of two over the course of a couple of days in order to better familiarize with the topics of the workshop.

3.5 Exercise sheet related to the day's topic (11.50-12.00)

An exercise sheet will be provided to wrap-up the day's material.

4 Day 2: Thursday, July 27th, 2017

4.1 Basics on code optimization & OpenMP session I (8.00-9.30)

In this lecture, we address the questions how we accelerate a serial code by relatively simple means and by using profiling tools. Moreover, we will introduce the basic principles of shared memory parallelism (using OpenMP) ([1, 2]).

4.2 OpenMP session II (10.00-11.00)

In this lecture, we will cover more advanced topics of OpenMP such as work sharing constructs and reductions.

4.3 Comments on Software Engineering (11.10-11.40)

This lecture covers the topics on how to design, program, test and manage software (git repositories; unit testing) in order to guarantee replicabiliy of results.

4.4 Exercise sheet related to the day's topic (11.45-12.00)

An exercise sheet will be provided to wrap-up the day's material.

5 Day 3: Tuesday, August 1st, 2017

5.1 MPI session I (8.00-9.30)

This lecture introduces distributed memory parallelism. A basic introduction to the message passing interface (MPI; [4, 1]) will be provided. Small hands-on exercises will be provided.

5.2 MPI session II (10.00-11.00)

In this lecture, we will discuss more advanced features of MPI. Moreover, we will quickly touch the fact that Python also provides the possibility to interface with MPI.

5.3 High throughput computing (11.10-11.40)

Many parameter studies consist of running the same code dozens or hundreds of times. For such embarassingly parallel tasks (that need zero communication), high throughput computing parallelism is the way to go. In this lecture, we will learn how this could be achieved on verious hardware platforms.

5.4 Exercise sheet related to the day's topic (11.45-12.00)

An exercise sheet will be provided to wrap-up the day's material.

6 Day 4: Thursday, August 3rd, 2017

6.1 Hybrid parallelism (8.00-9.00)

I touch on the topic that compute clusters nowadays consist of heterogeneous compute hardware components. In order to harness it optimally, different parallel programming paradigms have to be combined. In the second part of this hands-on lecture, we consider on how we can couple MPI with OpenMP.

6.2 Hybrid parallelism with our projects (9.15-10.00)

We will show how a dynamic programming code for economic models with or without discrete shocks is hybrid parallelized in practice.

6.3 Advanced topics (10.15-11.00)

In this lecture, an high-level outlook to more advanced topics is given. An overview on existing libraries as well as emerging hardware (Xeon Phi, GPUs) will be delivered in this session.

6.4 Wrap-up of the projects (11.10-11.45)

We discuss with the individual teams the status of their projects.

6.5 Exercise sheet related to the day's topic (11.50-11.55)

An exercise sheet will be provided to wrap-up the day's content.

6.6 Wrap-up of the lecture series (11.55-12.00)

We will wrap up the lecture series.

References

[1] G. Hager and G. Wellein. *Introduction to High Performance Computing for Scientists and Engineers*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 2010.

[2] G. Jost, B. Chapman, and R. van der Pas. *Using OpenMP - Portable Shared Memory Parallel Programming*. MIT Press, 2007.

[3] S. Prata. *C++ Primer Plus (Fourth Edition)*. Sams, Indianapolis, IN, USA, 4th edition, 2001.

[4] A. Skjellum, W. Gropp, and E. Lusk. *Using MPI*. MIT Press, 1999.