

# Preface

Welcome to the world of High Performance Computing! Welcome to the world of high performance Data Science!

In this textbook, we give an introduction to *high performance computing* (HPC for short) for *data science* (DS). Therefore, this textbook is organized into two parts: The first part (six chapters) covers the fundamentals of HPC while the second part (five chapters) describes the basics of DS and shows how to write distributed programs for the basic sequential algorithms in order to cope with large size datasets. Nowadays, many large-scale datasets are publicly available, and these datasets offer potentially a rich source of information that need to be purposely extracted.

We mainly distinguish two ways to design parallel algorithms: (1) Parallelizing algorithms on single multi-core shared memory machines using multi-threading, or (2) Parallelizing algorithms on a *cluster of machines* with distributed memory.

On the one hand, when parallelizing algorithms on shared memory architectures (like your smartphones, tablets, and very soon your smartwatches and other Internet of Things, IoTs, too!), computing hardware units (cores) are located on the same chip, and we can easily parallelize for example video decoding or rendering tasks using multi-threading. This kind of parallelism is said *fine-grained* but it is limited by the physical number of cores one can put on a chip (typically eight cores on a high-end smartphone in 2015). On the other hand, clusters of machines (that are distributed memory architectures) allow one to scale up resources on-the-fly according to the dataset size. There is a lot of flexibility when building a cluster of machines such as choosing heterogeneous computer nodes and deciding which topology is best for interconnecting those nodes. This kind of parallelism is called *coarse-grained* since bulks of local computations are performed independently on the machines before communications take place between them.

This undergraduate textbook focuses on designing parallel algorithms on distributed memory by using the standard interface called Message passing interface (MPI). MPI is the *de facto* standard to operate communications and global collaborative calculations between nodes of a cluster of machines. There exist several

vendor implementations of MPI that can themselves be binded with several programming languages such as C, C++, Fortran, Python, etc. We chose to use the C++ oriented-object language for implementing data science algorithms and use the C binding application programming interface (API) of OpenMPI to write parallel programs.

The two parts of this book are described concisely as follows:

**Part I** Introduction to HPC with MPI. We start by giving a quick introduction to the HPC world in Chap. 1, and describe both Amdahl's law and Gustafson's law that characterize the theoretical optimal speed-up and scaled speed-up.

Then we describe the main concepts of MPI and its programming interface: We introduce the notions of blocking/non-blocking communications, deadlocks, and the various global communication routines (like broadcast, scatter, gather, all-to-all, reduce, parallel prefix, etc.).

In Chap. 3, we emphasize on the role of the topology of the interconnection network. We shall distinguish between the *physical topology* and the *virtual topology* (or logical topology) that is considered when designing parallel algorithms. In particular, we describe communication procedures on the ring (including the optimized pipeline broadcasting) and on the hypercube. The latter one relies on a particular numbering of the nodes, called the Gray code.

In Chap. 4, we describe the main parallel algorithms for sorting on distributed memory. First, we naively parallelize the renowned Quicksort algorithm, and then introduce HyperQuicksort and the PSRS procedure (stands for parallel sorting by regular sampling) that is widely used in practice.

In Chap. 5, we study a few algorithms for multiplying matrices and vectors. We concisely explain the various techniques for computing matrix products on the topologies of the ring and of the torus.

In Chap. 6, we introduce the hot paradigm of parallel programming called "MapReduce" (that is often used with its open source counterpart called Hadoop). MapReduce allows to handle a very large number of networked computers (say, several thousands) by using programs that are built with two main user-defined functions called `map` and `reduce`. But MapReduce is also a full framework that includes a master-slave architecture that is able to take into consideration various hardware failures, or to re-route parallel computation tasks (jobs) on other machines when some machines become too slow (the so-called stragglers). We explain how to program those types of MapReduce algorithms in MPI (MPI has zero-fault tolerance) using a dedicated software library called MR-MPI.

**Part II** Introduction to Data Science. This part gives both a concise introduction to data science and furthermore explains how to parallelize those algorithms using MPI.

First, we start by describing the two most fundamental data clustering techniques that are flat partition-based clustering (in Chap. 7) and hierarchical tree-like clustering (in Chap. 8). Clustering is a very fundamental notion in exploratory data science for discovering classes in datasets, groups of homogeneous data.

Then we consider supervised classification using the  $k$ -nearest neighbor rule in Chap. 9 and make the connection with  $k$ -means.

In Chap. 10, we present yet another recent paradigm in computer science that allows one to solve optimization problems approximately on large datasets (potentially high-dimensional too!): That is, we seek core-sets that are subsets of data that guarantee an overall good approximation on the full datasets. This technique has become recently popular, and allows us to scale down Big Data to Tiny Data! Since data are usually given in high-dimension spaces, we also briefly explain a powerful technique to perform effective linear dimension reduction by stating the Johnson–Lindenstrauss’ theorem, and giving a short recipe to compute a low-distortion embedding of data into a lower dimensional space that preserves distances within a prescribed approximation factor. Interestingly, the dimension of the embedding is independent of the original extrinsic dimension but depends on logarithmically on the dataset size and the approximation factor.

In the last Chap. 11, we cover a few algorithms on graphs. Graphs are commonly encountered when performing social network analysis and in other application fields. We introduce both a sequential heuristic and a parallel heuristic to find a dense subgraph in a graph that is closed to the “densest” subgraph. Finally, we explain the simple task of testing graph isomorphism by branch-and-bound techniques on computer cluster. Testing graph isomorphism is a remarkable problem since its theoretical complexity is not yet settled (although some polynomial algorithms exist for some specific subclasses of graphs).

At the end of each chapter, we summarize the essential points to remember. The reader is invited to skim through these summaries for a first quick reading. Over 40+ exercises are given at the end of some of the chapters: These exercises are labeled with various difficulty degrees and allow the reader to test his/her understanding of the material covered. Sections that begin with an asterisk may be skipped and read later on.

The main goal of this textbook is to allow you to design parallel algorithms and then program your parallel algorithm using C++ with the C binding of MPI. The second goal is to expose you to the richness of the fields of HPC and DS, and hopefully to foster their interactions.

This undergraduate textbook has been written to be an introductory text on HPC and DS. Only basic knowledge of algorithmics and programming abilities are assumed. Therefore advanced notions in both HPC and DS fields are not covered

nor mentioned. For example, task scheduling problems and automatic parallelization of nested loops although important in HPC are not covered. Similarly, regression techniques and kernel machine learning methods for the data science field are omitted.

Happy Reading!

December 2015

Frank Nielsen

<http://www.springer.com/978-3-319-21902-8>

Introduction to HPC with MPI for Data Science

Nielsen, F.

2016, XXXIII, 282 p. 101 illus. in color., Softcover

ISBN: 978-3-319-21902-8