Final Remarks and Perspectives

9

Chapter Summary
After reading the book, the reader will be able to start parallel programming on any of the three main parallel platforms using the corresponding libraries OpenMP, MPI, or OpenCL. Until a new production technology for significantly faster computing devices is invented, such parallel programming will be—besides the parallel algorithm design—the only way to increase parallelism in almost any area of parallel processing.

Now that we have come to the end of the book; the reader should be well aware and informed that **parallelism is ubiquitous** in computing; it is present in hardware devices, in computational problems, in algorithms, and in software on all levels.

Consequently, **many opportunities** for improving the efficiency of parallel programs are ever present. For example, theoreticians and scientists can search for and design new, improved parallel algorithms; programmers can develop better tools for compiling and debugging parallel programs; and cooperation with engineers can lead to faster and more efficient parallel programs and hardware devices.

Being so, it is our hope that our book will serve as the first step of a reader who wishes to join this ever evolving journey. We will be delighted if the book will also encourage the reader to delve further in the study and practice of parallel computing.

As the reader now knows, the book provides many basic insights into parallel computing. It focuses on three main parallel platforms, the multi-core computers, the distributed computers, and the massively parallel processors. In addition, it explicates and demonstrates the use of the three main corresponding software libraries and tools, the OpenMP, the MPI, and the OpenCL library. Furthermore, the book offers hands-on practice and miniprojects so that the reader can gain experience.

After reading the book, the reader may have become aware of the following three *general facts about the libraries* and their use on parallel computers:

R. Trobec et al., *Introduction to Parallel Computing*,
Undergraduate Topics in Computer Science,
https://doi.org/10.1007/978-3-319-98833-7_9

- OpenMP is relatively easy to use yet limited with the number of cooperating computers.
- MPI is harder to program and debug but—due to the excellent support and long tradition—manageable and not limited with number of cooperating computers.
- Accelerators, programmed with OpenCL, are even more complex and usually tailored to specific problems. Nevertheless, users may benefit from excellent speedups of naturally parallel applications, and from low power consumption which results from massive parallelization with moderate system frequency.

What about near future? How will develop high-performance computers and the corresponding programming tools in the near future? Currently, the only possibility to increase computing power is to *increase parallelism* in algorithms and programs, and to *increase the number of cooperating processors*, which are often supported by massively parallel accelerators. Why is that so? The reason is that state-of-the-art production technology is already faced with **physical limits** dictated by space (e.g., dimension of transistors) and time (e.g., system frequency) [8].

Current high-performance computers, containing millions of cores, can execute more than 10^{17} floating point operations per second (100 petaFLOPS). According to the Moore's law, the next challenge is to reach the *exascale barrier in the next decade*. However, due to abovementioned physical and technological limitations, the validity of Moore's law is questionable. So it seems that the most effective approach to future parallel computing is an interplay of controlflow and dataflow paradigms, that is, in the **heterogeneous computing**. But programming of heterogeneous computers is still a challenging interdisciplinary task.

In this book, we did not describe programming of such extremely high-performance computers; rather, we described and trained the reader for programming of parallel computers *at hand*, e.g., our personal computers, computers in cloud, or in computing clusters. Fortunately, the approaches and methodology of parallel programming are fairly independent of the complexity of computers.

In summary, it looks like that we cannot expect any significant shift in computing performance until a **new production technology** for computing devices is invented. Until then, the maximal exploitation of parallelism will be our delightful challenge.