Basic parallel and distributed computing curriculum

Claude Tadonki

Basic parallel and distributed computing curriculum

Claude Tadonki
Mines ParisTech - CRI (Centre de Recherche en Informatique) - Mathématiques et Systèmes
35, rue saint-honoré, 77305 Fontainebleau-Cedex (France)
claude.tadonki@u-psud.fr

***Abstract*—With the advent of multi-core processors and their fast expansion, it is quite clear that *parallel computing* is now a genuine requirement in Computer Science and Engineering (and related) curriculum. In addition to the pervasiveness of parallel computing devices, we should take into account the fact that there are lot of existing softwares that are implemented in the sequential mode, and thus need to be adapted for a parallel execution. Therefore, it is required to the programmer to be able to design parallel programs and also to have some skills in moving from a given sequential code to the corresponding parallel code. In this paper, we present a basic educational scenario on how to give a consistent and efficient background in parallel computing to ordinary computer scientists and engineers.**

***Keywords*-HPC; multi-core; scheduling; SIMD; accelerator; benchmark; dependence; graph; shared memory; distributed memory; thread; synchronization;**

I. Introduction

In the past, *parallel computing* courses were dedicated to HPC specialists, under appropriate prerequisites. This was due, on one hand, to the technical context, where standard processors were single-core, parallel computers being the corresponding clusters (shared or distributed memory). In addition, processors speed was increasing significantly (following the Moore's law), thus giving an argument to refrain from moving to parallel computing. Indeed, what one could achieve using a moderate cluster at a given time could be done a few years later using next generation processor. Therefore, as parallel computing could not be reasonably considered for basic issues, it was quite hard to motivate bringing it into standard courses. On the other hand, the basis to understand parallel computing and have hands on it show a significant gap from ordinary skills. Thus, one could understand a certain reluctance to such a heavy effort from both sides (students and teachers). Nowadays, the situation is no longer the same, and we have to bring parallel and distributed computing (at least at the basic level) into the standard. The corresponding courses have to be ready for a common audience.

Parallel and Distributed Computing (PDC) is a specialized topic, commonly encountered in the general context of *High Performance/Throughput Computing*. We mainly see three kind of material that could be considered when it comes to teaching PDC.

- First, the literature. There are numerous valuable books that cover general and/or specific aspects of PDC. General books that provide an overview of the topic with details on some selected aspects [1], [3], [4], [5], [20]. Some books are more educational (tutorial approach with exercises and case studies) [2], [12], [13], [14]. Other manuals focus on specific architectures [7], [8] or libraries (MPI, OpenMP, Pthreads) [9], [10]. There are books devoted to parallel algorithms design and fundamental aspects (foundation, models, scheduling, complexity) [15], [16], [17], [18], [19], [21].
- Second, conferences and assimilated events [25], [26], [27], [28], [29], [30] are good places to learn about advances and keep updated with new results. There are numerous events dedicated to PDC, some of them being tailored for student exposure and training through specialized tutorials.
- Third, summer/winter schools (or advanced schools) [22], [23], [24] are good opportunities for specialized training, through an intensive few days course. Depending on the content, purpose, or audience, such schools are intended to develop a particular skill, or give a short but consistent PDC introductory course. For instance, an ordinary student who intends to do a PhD in parallel computing could use such schools as a starting point.

Because of the prerequisites and a certain technical maturity needed to deal with parallel computing, we think that PDC courses could be reasonably considered at nearly the end of the undergraduate curriculum. At his level, the aim could be to have the students being able to design an intermediate level parallel program. The courses could be organized around that objective, taking into account the background of the student and what is really needed at that point. A typical scenario could include:

- general introduction
- parallel computation models
- distributed memory paradigm
- shared memory paradigm
- instruction level parallelism
- performance evaluation
- debugging and profiling tools
- virtualization and simulators
- specialized frameworks

In addition to how to connect the selected chapters, it is important to teach them at the right and appropriate level. Indeed, as we target an undergraduate curriculum, and following the aforementioned global objective, we just need to stay in the necessary scope. In addition, the way each chapter is introduced and handled is important. We classify the selected chapters into three main group and develop our argumentation accordingly. The following section provides a global overview of the PDC course. Next, section III describes how to provide introductory PDC elements. In section IV, we expose different ways to implement parallelism. We discuss about parallel programs execution in section V. Section VI concludes the paper.

II. Course overview

A basic scenario for a consistent PDC course is displayed in figure 1.

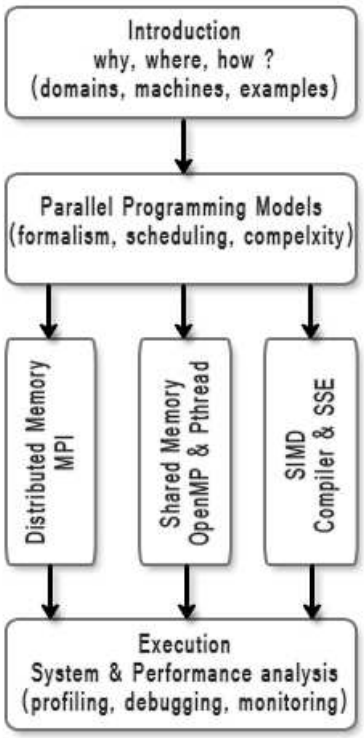

Figure 1. Integrated overview of the PDC course

The first thing to do is to buy the attention of the students by providing some illustrative examples and outline some convincing motivation items. Another purpose with the examples is to help understanding why it is important (sometimes vital) to compute faster. As the need for speed and the programming model are different from one application to another, this could be the time to talk about domain classification. Next, comes the hardware aspect, means *parallel machines* and accelerators. For undergraduate students, the topic of the *accelerators* could be postponed for postgraduate level or left as an option for particularly motivated students. This part could be ended by discussing each of the dissemination media listed in the introduction section. After this course unit, students are ready to enter into the subject.

III. Basic of parallelism

As parallel computing means simultaneous processing of several tasks, it is important here to introduce the notion of *dependence*. Dependences are the cause of *synchronization*, *data communication*, and sub-optimal performances. Students should be able to identify the main dependences between tasks and appreciate the potential of parallelism related to a given application. Deriving a (good) parallel scheduling could be the next point. Scheduling algorithms and methodologies could be explained together with their associated formalism (tasks graph, recurrence equations, ...). Some basic elements of performance prediction could be presented here, leaving the aspect of *pure complexity* for specialized students. Once the students are familial with the concept of parallelism, it could be time to consider the implementation aspect.

IV. Ways to implement parallelism

What could be done here is to present different level of parallelism and then focus on the most commonly considered solutions, namely MPI for the message passing paradigm, OpenMP and Pthread for thread level parallelism, and SSE for instruction level parallelism. It is not necessary here to to into deeper details on each programming model. It is rather important to have the students being able to derive effective implementations for some basic examples and understand that achieving a high speed program could come from a hybrid implementation. In general, there is a software gap between the hardware potential and the performance that can be attained by practical programs. Thus, it is important to handle the execution correctly and understand the performance point.

V. Running time

Compiling and running a parallel program is the last point of our PDC course scenario. After having the program running, it is important to know how to measure its performance, and thus appreciate the impact of the implemented parallelism. As for sequential programs, some mistake could have been done, either at the design stage or at the programming stage. Teaching the use of debugging techniques and tools could be considered, with the aim of being able to detect and fix programming mistakes or system issues. Hardware issues could be mentioned but not covered.

VI. Discussions and Conclusion

Teaching parallel and distributed programming at any level is a genuine requirement nowadays. In order to fulfill this crucial need, PDC course should be incorporated into standard scientific and engineer curriculum. There is certainly a pedagogical effort to bring this topic, previously reserved for specialists, into the standard. Whenever possible, the earlier it is done, the better. In this paper, we propose consistent scenario that could apply at the undergraduate level. We also thing that using well designed simulators could be very useful for this training task.

References

[1] A. Grama, G. Karypis, V. Kumar, A. Gupta, *Introduction to Parallel Computing (2nd Edition)* , 2nd ed. Addison-Wesley, 2003.

[2] P. Pacheco, *An Introduction to Parallel Programming* , 2nd ed. Morgan Kaufmann, 2011.

[3] F. T. Leighton, *Introduction to Parallel Algorithm and Architectures: Arrays, Trees, and Hypercubes*, Morgan Kaufmann,2nd ed. San Mateo CA, 1991.

[4] T. G. Lewis and H. El-Rewini, *Introduction to Parallel Computing*,2nd ed. Prentice-Hall, Englewood Cliffs, USA, 1992.

[5] G. Hager and G. Wellein, *Introduction to High Performance Computing for Scientists and Engineers* , 1st ed. CRC Press, July 2, 2010.

[6] M. Herlihy and N. Shavit, *The Art of Multiprocessor Programming* , 1st ed. Morgan Kaufmann, March 14, 2008.

[7] T. Rauber and G. Rnger, *Parallel Programming: for Multicore and Cluster Systems* , 1st ed. Springer, March 10, 2010.

[8] D. B. Kirk and W. W. Hwu , *Programming Massively Parallel Processors: A Hands-on Approach* , Morgan Kaufmann, February 5, 2010.

[9] B. Chapman, G. Jost, R. van van der Pas, *Using OpenMP: Portable Shared Memory Parallel Programming* , The MIT Press, October 12, 2007.

[10] P. Pacheco, *Parallel Programming with MPI* , 1st ed. Morgan Kaufmann, October 15, 1996.

[11] M. J. Quinn, *Parallel Programming in C with MPI and OpenMP* , McGraw-Hill Education , January 2008.

[12] L. R. Scott, T. Clark, B. Bagheri, *Scientific Parallel Computing* , Princeton University Press, March 28, 2005.

[13] G. E. Karniadakis, R. M. Kirby II, *Parallel Scientific Computing in C++ and MPI: A Seamless Approach to Parallel Algorithms and their Implementation* , Cambridge University Press, June 16, 2003.

[14] F. Gebali, *Algorithms and Parallel Computing* , Wiley, April 19, 2011.

[15] J. JaJa, *Introduction to Parallel Algorithms* , 1st ed. Addison-Wesley Professional, April 3, 1992.

[16] S. H. Roosta, *Parallel Processing and Parallel Algorithms: Theory and Computation* , 1st ed. Springe, December 10, 1999.

[17] F. T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes* , Morgan Kaufmann Pub, September 1991.

[18] J. Hromkovic, *Communication Complexity and Parallel Computing* , Springer, December 1, 2010.

[19] R. Greenlaw, H. J. Hoover, W. L. Ruzzo, *Limits to Parallel Computation: P-Completeness Theory* , Oxford University Press, USA, April 6, 1995.

[20] J. Dongarra, I. Foster, G. C. Fox, W. Gropp, K. Kennedy, L. Torczon, A. White (Eds), *The Sourcebook of Parallel Computing* , 1st ed. Morgan Kaufmann, November 25, 2002.

[21] R. Correa, I. de Castro Dutra, M. Fiallos, L. F. Gomes da Silva (Eds) , *Models for Parallel and Distributed Computation: Theory, Algorithmic Techniques and Applications* , 2nd ed. Springer, December 10, 2010.

[22] www.cineca.it/page/advanced-school-parallel-computing

[23] http://www.upcrc.illinois.edu/summer/2011

[24] http://cac.kias.re.kr/School/2011winter/

[25] http://pactconf.org/index.php/en/

[26] http://sc12.supercomputing.org/

[27] http://dynopt.org/ppopp-2012/

[28] http://ics-conference.org/

[29] http://www.cs.jhu.edu/ spaa/2012/

[30] http://www.ipdps.org/