Difficulties and Solutions in the Field of Teaching Parallel Programming

Sándor Szénási

Abstract

During the last decades, users and developers have become accustomed to the continuous improvement of the IT tools. Processors have become more and more faster (for the same price). But in the last years, more and more problems have appeared related to clock frequency increase, therefore processor developers have been forced to look for alternative ways for further acceleration. The processor core duplication and subsequently further reproduction is a very obvious solution, which allows the multiplication of the theoretical peak performance.

To utilize the full processing power of these new devices, we need well-written multithreaded applications. But parallel systems (however these have existed for years) have become general only in the last few years; therefore we usually did not have suitable methods, experiences and course notes about this topic. This paper focuses on the challenges and the solutions in the field of parallel programming.

Keywords: teaching programming, parallel algorithms, multi-threaded programs, program implementation, parallel debugging, I23

Introduction

In the last few years the development of processors means more and more processor cores. In fact, there are several benchmarks and test applications, which can show that the new devices will continue the speed-up. However, experience shows that it is not clearly noticeable in practical applications. The users of the two-, four- and eight-core processors do not notice the acceleration of the programs by twice, four or eight times. The clock frequency increase cause almost linearly speed-up for each programs, in contrast the increasing of the number of cores does not provide the same. The multi-core processors have advantages only in multi-threaded environments, but many applications developed in the 90s for home and small business purposes use the traditional sequential
behaviour. Therefore, these softwares need some modifications to reap the benefits of the multi-core architecture.

On the one hand this means the development of parallel variants of the well-known classical algorithms; on the other hand this also means the implementation of algorithms using some kind of parallel computer languages. In the fields of classical sequential algorithms we have decades-long experiences in teaching. But parallel systems have become general only in the last few years; therefore we usually did not have suitable methods, experiences and course notes about this topic. This is why it is advisable to reconsider many topics, for example about the establishment of the algorithmic approach, and about the programming languages.

**Teaching parallel programming**

From technical aspect, the mere appearance of parallel architectures does not cause any problems; it is easy to program these new devices. Almost all current programming languages have some possibilities of multi-threaded programming, and these are not too complicated, the students only have to read the corresponding chapter in a book (from the technical aspect, parallel programming means only the knowledge of some new classes, keywords, methods, etc.).

The biggest problem was that the change was rapid, and most of the programmers were unprepared for the new technology. Of course, there were multi-processor systems for a long time, but for most of the developers, these systems looks like some further exotic technologies (maybe, they have learned about that in the school, but usually do not have any practice in this field of this kind of programming). In the server parks with high computing power, the multi-threaded applications were usual for several years, but for the average user, the multi-processor environment was just an exciting curiosity.

Accordingly, the programmers (even if they knew the multiprogramming techniques) used common sequential algorithms, which use only one core in the multi-core architectures.

The teaching of programming was even farther behind in this area. In most cases, where the teaching of programming appears only for 2-3 semesters, this option is often completely left out. Or if it appears, then
only as a marginal topic in a two-hour period without any practical aspects. In recent years, significant changes appear in this area. In most universities, parallel programming appears (Vámossy et al, 2008).

Parallel algorithm design

Teaching programming is quite difficult, because it is about not only some lexical knowledge but a way of thinking. Therefore, various methodologies have evolved, however most of them based on the same structure (structured programming, object oriented programming, basic algorithms, sorting, data structures, etc.). Traditionally, there is no space for parallel programming in this system. Parallel programming is hard for the beginners; therefore it looks like a bad idea to teach it in the basic level.

However, this has the potential that the theory of parallel algorithms will looks like some hard and exotic thing for the students, and they will not see the casual nature of it. Today, however, it is preferable to start the presentation of the basics of these new techniques for the beginners. Of course, this does not mean that the beginners have to know everything about parallel programming (concepts, problems, solutions, synchronization techniques, etc.), but it is worth to give them a short overview about these techniques.

- It may be worth to present the currently (again) spreading alternative programming paradigms. Although, it looks like a good idea to use the common methodology for the beginners (structures programming, OOP, sequential programs), it is worth to mention the basics of the alternative paradigms (logical programming, functional programming, dataflow programming). Without going into details, we can show that the execution semantics of these alternative paradigms can be much more advantageous in multi-processor environments.
- In the field of teaching traditional algorithms, it may be also worth noting that what possibilities provide the parallel version. For example, in case of a simple search, we can show that (without the deep details) with using multiple cores, we can easily speed-up the algorithms.
- In addition to the benefits, it is preferable to present the disadvantages too. In case of several algorithms, we use global
variables for simplicity. For didactical reasons, this can be good in
the future too, but it is worth noting that in case of multi-threaded
environment, these variables can cause several problems.

Hopefully, these examples will help the students in the future, when they
have to learn about the advantages and disadvantages of parallel
algorithms.

Implementation of parallel algorithms

Parallel programming causes several changes in the implementation phase
too. Fortunately, the new functions for this are easy to understand and do
not require long learning time. Although, at first, we might think that the
implementation of parallel programs causes several problems, but in
practice, it seems it does not.

- Possibilities of programming languages to start multiple threads
  simultaneously. In most cases, there are several ways to start
  multiple threads, which is especially useful from the educational
  point of view, because we can choice, how deep to dive on this
  topic.
- Understanding various operations associated with the parallel
  programming. Typically, this might include various
  synchronization techniques. Programming languages usually have
  several features.

Our experiences clearly show that the implementation does not cause
particular problems. Developing correct parallel algorithms is much
harder than implementing the source code itself.

Parallel development process

In practice, it has turned out that during the development for multi-
processor environments, the implementation of the code causes fewer
problems than the subsequent testing and debugging.

During testing, the biggest problem is that the parallel algorithms are
often nondeterministic. Traditionally, we assume that for a given input the
answer of the program will always the same. Testing means that we
compare the output of the program to the correct output. But in case of nondeterministic applications, the program may give different outputs for the same input, which is an entirely new situation. It is essential to adopt the testing methodology to these new requirements, as well as we have to prepare the students to be not surprised, if similar events occurred in their work.

Debugging also raises a number of problems. Classically, the trace function is well known to localize errors. In case of multi-threaded applications, this feature is not as usable as in the case of sequential algorithms.

- Due to the large number of threads, it is really hard to trace the execution of an application, because we usually can see only one thread. In the case of GPGPU applications this is even more problematic, because we have to debug thousands of threads.
- The separate threads work independently, thus stopping one of them does not necessary mean the pausing of the whole system. Threads in the background may influence the state of the currently examined thread. It may be also confusing, when more than one threads run into the same breakpoint at the same time, and this means new traces.
- In case of multi-threaded programming, the debugging itself can cause several side effects. There are a lot of problems (like deadlocking, race conditions), which do not appear during debugging, so finding the exact location of these bugs may be really hard.

Unfortunately, we cannot give a satisfactory solution for these problems; perhaps, the next generations of programming environments will solve these. From the educational aspect, it is appropriate to take into account these difficulties and prepare the students for the problems which may arise.

First of all, in the case of exams, it must be taken into account these specialities. It is not expected that students can write perfect codes immediately, but it is an appropriate requirement that they must be able to find and correct the errors. In multi-threaded programs, we have definitely taking into consideration that this is an order of magnitude more difficult tasks for a moderately complex program. It is easy to make any errors, whereas the failure point localization might require a bit of luck and a lot of time.
Conclusions and recommendations

Multi-threaded programming has become an inevitable factor in software development. Therefore, it is important to present this new area for the students.

Since this topic is rather complex, it cannot be expected to start multi-thread programming in the first semester, but it is worth to mention some related definitions and techniques in basic level.

The implementation of a well-designed parallel algorithm does not present particular challenges, however the debugging and testing may raise several problems. In these cases the usual tools are often ineffective; therefore students need some creativity and deeper knowledge about parallel algorithms, for example Control Flow Graph (CFG), Program Dependence Graph (PDG) (Miller et al, 1998). Finding bugs in multi-threading environment is often quite difficult (especially for inexperienced programmers), so we have to take this account in the evaluation phase.

References


Author(s)

Sándor Szénási
PhD
assistant lecturer
Óbuda University, John von Neumann Faculty of Informatics
szenasi.sandor@nik.uni-obuda.hu