Experience with Teaching PDC Topics into CS Courses of UBB University

Virginia Niculescu[1] and Darius Bufnea[2]

[1] Faculty of Mathematics and Computer Science
Babeş-Bolyai University, Cluj-Napoca
vniculescu@cs.ubbcluj.ro
[2] bufny@cs.ubbcluj.ro

Abstract. In this paper, we present an analysis of the outcomes of teaching Parallel and Distributed Computing within the Faculty of Mathematics and Computer Science from Babeş-Bolyai University of Cluj-Napoca. The analysis considers the level of interest of the students for different topics as being determinant in achieving the learning outcomes. Our experiences have been greatly influenced by the specific context defined by the fact the majority of the students are already enrolled into a software company either as an intern in an internship program or as an employee. The study emphasises that the level of interest of the students for a specific topic is determined by the context of the IT industry development in the region. The learning activity is in general influenced by this specific context, and a new, high demanding topic as Parallel and Distributed Computing is even much more influenced, when is to be taught to the undergraduate level. This analysis further leads to a more general analysis on the appropriateness of introducing PDC topics, or other relatively advanced topics, to all undergraduate students in CS, or to consider new defined educational degrees.
Keywords: Parallel and distributed programming, curricula, courses, undergradute, IT industry, workforce.

1 Introduction

The last years have brought an explosive growth in multiprocessor computing, including multi-core processors and distributed data centers. The mass marketing of multicores and general-purpose graphics processing units induces the possibility for common users to rely on its effectiveness. This enforces the software developers to efficiently use it, and also to contribute to the technology development.

As a consequence there is a clear need for students in computing related general education courses to be aware of the role that parallel and distributed computing technologies play in the computing landscape.

But now, it is considered no longer sufficient for even basic programmers to acquire only the conventional programming skills.

The ACM/IEEE Curricula 2013 Report [7] and the NSF/IEEE-TCPP Curriculum Initiative on Parallel and Distributed Computing [4], argue that the

undergraduate computer science programs should include topics in parallel and distributed computing (PDC).

This approach implies important changes and their impact should be carefully analysed.

Babeş-Bolyai University is the biggest university of Transilvania, and also one of the biggest universities of România. It is the oldest university of România, but at the same time it is a dynamic and constructive institution well integrated into society and oriented towards the future.

The Faculty of Mathematics and Computer Science [http://www.cs.ubbcluj.ro] is today a blend of tradition, experience and modernity that try to mix the abstract world of Mathematics and the perpetually effervescent universe of Computer Science. In the last decade the number of students attending Computer Science has rapidly increased. The faculty follows the Bologna system of study.

Parallel and distributed computing topics have been studied at our faculty until 2015 especially at master level programs, but still there were some modules related to concurrency, multithreading and client-server application, RPC, RMI included into some curriculum courses. Since 2015 a dedicated required course "Parallel and Distributed Programming" has been introduced for the students in the third year of study. Before this it was also an elective course "Paradigms and Techniques of Parallel Programming" that uses to introduce the main concepts and paradigms of parallel programming; after the curriculum changes this course addresses more advanced topics.

From 2014 a master program with the title "High Performance Computing and Big Data Analytics" has been included into academic program of our faculty (http://www.cs.ubbcluj.ro/education/academic-programmes/masters-programmes/high-performance-computing-and-big-data-analytics-programme-profile/). It offers the students the possibility of acquisition of theoretical, applicative and practical knowledge in high performance computing but also in the very actual domain of big data analytics. Also an important focus is on using HPC in data analysis.

This paper intend to present the evolution of the courses that includes modules from PDC topic with a focus on those that are dedicated to them. Also a broad analysis of the outcomes of these teaching subjects in correlation to the level of interest of the students for them is presented.
The next section presents the existing courses, and modules, and section 3 describes the particular context of our region that has an important influence on the level of interest of the students for different subjects. Section 4 shows the analysis results and their correlation, and the final conclusions are presented in section 5.

2 The Subject of Analysis

As in other reports on various curricula, we will use Bloom's classification [1] (B class) considering also a correlation to ACM level of mastery. So, we will use the following classes:

- K= Know the term ($\Leftrightarrow$ Familiarity)

– C = Comprehend so as to paraphrase/illustrate (⇔ Usage)
– A = Apply it in some way (requires operational command) (⇔ Assessment)
– N = Not in Core, but can be in an elective course

The courses from the undergraduate curriculum that address Parallel and Distributed Computing topics are the following:

1. *Operating Systems* (OS)
   first year of study, second semester
2. *Advances Programming Methods* (APM)
   second year of study, first semester
3. *Methodologies for developing software systems* (MS)
   second year of study, second semester
4. *Networking* (Net)
   second year of study, first semester
5. *Parallel and Distributed Programming*(PDP)
   third year of study, first semester
6. *Paradigms and Techniques of Parallel Programming* (elective) (PP)
   third year of study, second semester

The number of students enrolled in a course is about 180 and for an elective one varies between 30 and 70.

The next two tables emphasises (to a great extend but not completely) the PDC topics discussed at these courses. Table 1. presents some general topics with the focus on concepts, and Table 2. shows the topics for which concrete implementations are analysed.

A certain topic could be introduced to a certain course where the learning outcomes belongs to (K) or (C) Bloom's classification, and then it is discussed to a following course where the learning outcomes are moved to a more advanced Bloom's classification. So, for example, the semaphore concept is first introduced at *Operating Systems* course considering an outcome of class (K), and then is discussed again at *Parallel and Distributed Programming* course where a more deeply understanding is provided and also it is used in the context of the current implementations in Java or C#.

Examples of parallel algorithms are given especially at the *Paradigms and Techniques of Parallel Programming* course. In the curriculum there is a course of Data Structures and Algorithms, but for the moments it doesn't treat the possible parallelisation of the algorithms. Still, time-complexity and space-complexity issues are analysed for sequential algorithms, and so when the parallel programs performance metrics are introduced we may start from some already introduced concepts.

3 The Context of the Analysis

Our experience was influenced by the particular context of our region. Cluj-Napoca is now the most important educational and economic centre in Transylvania and the second largest in Romania after the Capital Bucharest and it has

Table 1. General conceptual topics.

Topic	B class	Courses
SISD/SIMD/ SPMD/ MIMD	K	PDP,PP
Computation decomposition strategies	C	PDP, PP
Data Distribution	C	PDP, PP
Functional Decomposition	C	PDP, PP
PCAM methodology	K	PP
Synchronisation Concepts	C	OS,PDP,PP
proceses, pipe, fifo	C	OS
critical section, race condition	C	OS, PDP, PP
mutex, semaphore, monitor	C	OS, PDP, PP
barriers, conditional variable	C	PDP, PP
deadlock, livelock	C	OS, PDP, PP
starvation, fairness	K	OS, PDP
Tasks and threads	C	APM, PDP, PP
Non-determinism	C	PDP, PP
Performance metrics	C	PDP,PP
Speedup, Efficiency, Cost	C	PDP,PP
IsoEfficiency	K	PP
PRAM	C	PDP,PP
Brent Theorem	K	PP
Dependencies	A	PDP,PP
Task graphs	K	PDP,PP
Divide & conquer (parallel aspects)	A	PDP,PP
Recursion (parallel aspects)	C	PDP,PP
Master-slave	A	PDP
Pipeline (parallel aspects)	A	PDP,PP
Scalability	K	PDP,PP
Granularity	K	PDP,PP

a long standing tradition in IT development. The beginnings of the computer sciences in Cluj are situated around the years 1960. The city of Cluj being one of the first academic center of SouthEaster Europe, it is also home for more than 300 IT&C companies of all sizes, accounting for approximately 70% of Romania's IT exports. During the last decade the IT industry around the city of Cluj-Napoca has increased tremendously. And because of this the requirements for workforce has increased extremely which lead to a high demand of employment of students, even starting with the third, second, or even the first year of study, too. The cooperation between students and companies starts usually with an internship program and it is also very often finalised with real employment before graduation.

So, when we discuss the impact of some changes in the academic curricula we have to consider the fact that the feedback that we obtain from students, includes also, indirectly, a feedback from the industry.

Table 2. Specific topics ⇒ implementation oriented.

Topic	B class	Exemplification	Courses
Shared memory	A	Java, C/C++, C#	OS ,PDP
Thread/Task spawning	A	Java, C/C++, C#	OS, APM, MS, Net, PDP
Executors,Threads pools	A	Java	APM, PDP, PP
Work stealing	K	Java	PP
Synchronisation tools	A		OS,PDP,PP
mutex, semaphore	A	Java, C/C++, C#	OS, PDP, PP
barriers, conditional variable	A	Java, C	PDP, PP
Asynchrony	A		PDP, PP
Futures/promises/Async tasks	A	Java, C++	PDP, PP
Streams	A	Java	PDP, PP
Parallel loop	A	C/C++,OpenMP	PDP, PP
Hybrid	C	CUDA/C++	PDP, PP
Distributed memory	C		PDP, PP
Message passing	C	MPI, C/C++	PDP, PP
Broadcast, Scatter/Gather	C	MPI, C/C++	PDP, PP
Client Server	A	C/C++, Java, C#	MS, Net, PDP
RPC, RMI	A	Java, C#	MS
P2P	K		PDP

The IT industry of our region if very much oriented to Platform based Development (PBD) especially on Mobile-applications and Web-applications.

We may notice that there is a certain gap between academic world and the industry. The industry is very much oriented on productivity even with some expenses in the software quality. Consolidated frameworks, libraries and APIs are frequently used in the development, alongside development tools that are required in a productive environment.

This is why there are companies that can afford to hire students even from their first years of study, and encourage them for early employment with the promise that they will learn "all they have to know" at the workplace. (Of course their perspective is on the present day, without considering the future.). This comes with the drawback that students focus less on obtaining general knowledge in computer science and they start learning/using only specific fields of computer science (database, user interface development, etc.). Often enough students that are yet employed are reluctant to learn things that wouldn't help them during an internship or job interview.

The development is very often based on *"applying patterns... "* but the meaning of this idiom, in this context, is not the same to that used in the Gang of Four book: Design Patterns. Elements of Reusable Object-Oriented Software [**?**], where it is used to emphasise the situation when a design pattern (a well defined solution) is used in a new context in a creative way. Here, we have to

understand that the software is built using specific framework and technologies by composing components based on some specific recipes.

So, many times the developers build the software by using some tools and without a deep understanding of what they are really doing . The leading questions are: *"how to do"*, *"what to apply"* and not *"why"*, or *"what is hidden behind"*.

A very important remark is that the described situation has a large spreading but it is not generalised!

The university purpose is to prepare the young minds for whatever is out there in the industry without limiting the knowledge to a specific area. The graduates need to acquire enough information from all the fields in such a way that they can face the industry switches without too much effort, having the basics in place.

4 The Analysis

In order to move from "traditional" development to distributed development the students need to posses the most basic knowledge of development. It is always easier to 'build' on top of something that has solid ground. The challenges that come from the current industry context (students start focusing on employment rather than finalising their studies) trigger different approaches regarding teaching techniques:

- Before moving to a topic that requires specific background, we need to summaries and validate that students have this background; this comes with the drawback that some of the students that already have the background cannot move faster to the specific distributed programming topics, and they become distrustful.
- Some of the basic courses have been condensed or made optional in order to accommodate the students needs to have the bare minimum knowledge for employment.

In our study we went from the premise that the success of introducing new topics in the curriculum, and consequently achieving the desired learning outcomes, depend in a great measure on the level of interest of the students in that topic. In the context described in the previous section, we are aware that the level of interest of the students for one topic and another depend very much if they are working for a company or not, and when they have started to do this.

So, the first steps of our investigation was find out the level of interest of the students for the topics specified in the Table 1, and Table 2. For each topic they have been asked to choose a value between 1 and 5 (1 represents the lowest level of interest, and 5 represents the highest level of interest. The students of the second year of study have been interrogated, and also those from the third year of study. The results are reflected in Table 3 and Table 4.

The differences between the two categories are given also by the fact that the students of the second year don't know yet some included topics, but also by employment distribution that is:

Table 3. Level of interest for general conceptual topics.

Topic	Level of interest 2nd year	Level of interest 3rd year
SISD/SIMD/ SPMD/ MIMD	1	1,87
Computation decomposition strategies	1	4.5
Data Distribution	1,37	3
Functional Decomposition	1	3,12
PCAM methodology	1	3,25
Synchronisation Concepts	2,56	3,75
proceses, pipe, fifo	3,75	3
critical section, race condition	1,5	3,25
mutex, semaphore, monitor	2,25	3,18
barriers, conditional variable	1,62	3,25
deadlock, livelock	3	4,37
starvation, fairness	1	3
Tasks and threads	3,75	4.5
Non-determinism	1,31	3,87
Performance metrics	2,87	3,12
Speedup, Efficiency, Cost	3,06	4,25
IsoEfficiency	1,25	1,75
PRAM	1	2
Brent Theorem	1	1.75
Dependencies	2,37	3
Task graphs	1,62	1,87
Divide & conquer (parallel aspects)	3,68	2,75
Recursion (parallel aspects)	3,81	2,62
Master-slave	1,5	4,25
Pipeline (parallel aspects)	3	3,37
Scalability	2,06	3
Granularity	1,37	2

- 10% students of the first year of study,
- 25% students of the second year of study,
- 60% students of the third year of study,
- (75% students at the end of the third year).

(The students have a mandatory internship of 3 weeks between the 2nd and the 3rd year, and this is the moment when almost all get hired.)

Parallel programming is not easy if we have to control threads/processes executions, synchronisation, communication, etc. As the level of abstraction is increased the things could become simpler, but of course, that there is an associate performance cost [5].

Their is a large interest of the students to learn APIs and tools that implicitly use the parallelisation without the explicit control of the programmers (Java parallel streams, Scala parallel collection, OpenMP). This approach has the advantage of offering a simple and rapid development and also offers a high degree

Table 4. Level of interest for the specific topics.

Topic	Exemplification	Level of interest -2nd year	Level of interest -3rd year
Shared memory	Java, C/C++, C#	2,12	3,62
Thread/Task spawning	Java, C/C++, C#	2,93	4,12
Executors,Threads pools	Java, C#	2,31	4,5
Work stealing	Java(ForkJoin)	1,37	4
Synchronisation tools			
mutex, semaphore	Java, C/C++, C#	2,75	3,75
barriers, cond.variable	Java, C/C++	2,25	3,25
Asynchrony			
Futures/promises		1,5	4,12
Async tasks	Java, C++	3	3,12
Streams	Java	2,06	4,25
Parallel loop	C/C++,OpenMP	3,25	3,75
Hybrid	CUDA/C++	2,25	2
Distributed memory			
Message passing	MPI, C/C++	2	2,12
Broadcast, Scatter/Gather		3,25	2
Client Server	C/C++, Java, C#	4,25	4
RPC, RMI		3,68	2,75
P2P		1,75	2,75

of confidence in the correctness of the resulted code. It is known that parallel programming is sensitive to hidden errors that are very difficult to detect and then to debug. On the other hand the programmers are limited to the defined constructions, and also cannot control very well the level of performance.

The analysis included also the results obtained by the students for the tests and assignments of the curriculum required course *Parallel and Distributed Programming*. The evaluation for this course has been based on the followings tests and assignments:

1. Practical works/assignments (relative short problems that should have been implemented using discussed strategies and technologies);
2. Multithreading practical test (a problem of a medium complexity that had to be solved using threads – explicit thread creation);
3. MPI practical test (a simple problem that had to be solved using MPI);
4. Theoretical test (written exam).

The corresponding results for these evaluations are presented in Figure 1. Practical works included:

- some multithreading examples in C/C++, Java, and C#,
- a very simple CUDA example,
- a client-server application that also includes asynchronous tasks, and
- a simple MPI example.

The students had to solved them independently, at home, and then present them to the instructors.

The practical tests assume solving a given problem in a given period of time, on the students' laptops – if they chosen this way; computers from the faculty laboratories could also be used.

From these results, we may consider that MPI programming have been proved difficult for students. A deeper analysis emphasises that, in fact, the interest of the students in learning MPI was low.

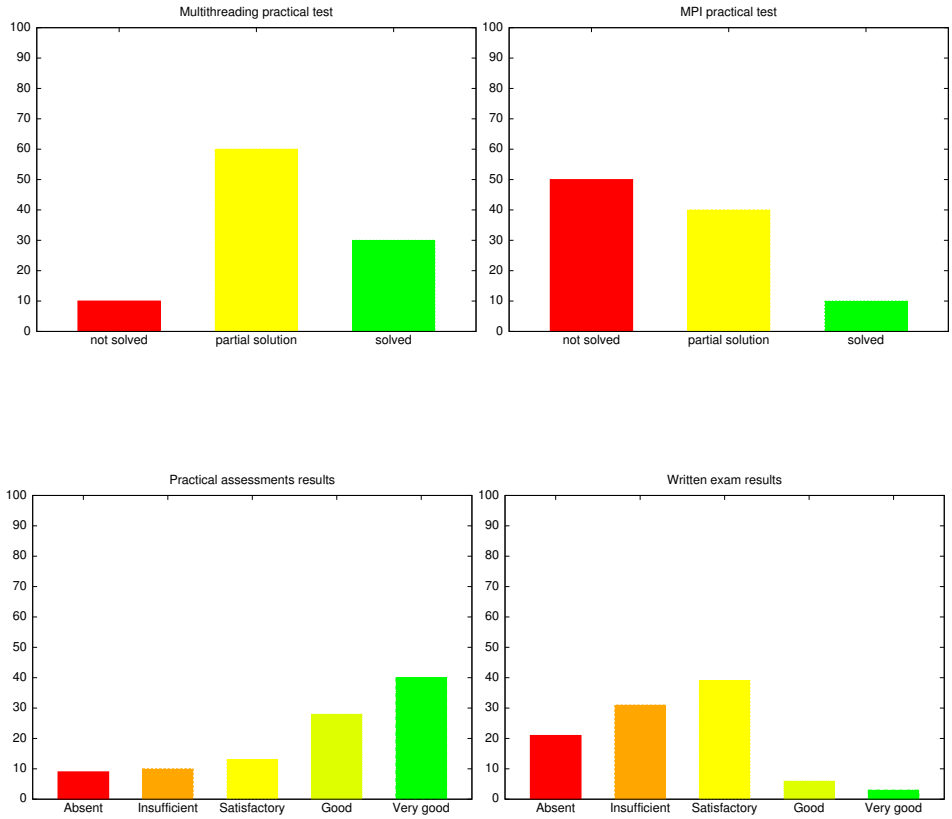

Fig. 1. Evaluation results.

The students are much more confronted to using threads, for different types of applications, and this leads to a much better knowledge acquisition.

The theoretical evaluation shows the fact that even the students declare that they have certain interest in studying concepts, still either because they don't have enough time (being involved in others activities as working for companies) or because they looses their ability for theoretical approaches, the results are not very good.

Since the students are soon enrolled in productive activities they are much oriented on practical skills, and the results obtained for practical works in correspondence to the results of the theoretical test confirm this situation.

For the elective course *Paradigms and techniques of parallel programming* (PP) the students have been allowed to choose a paradigm and a technology for solving in parallel a problem. The problem could be also chosen from a list of proposed problems but they also had the possibility to propose new problems. From the 35 students that attended this course, only one has chosen MPI as a programming model. All the others chosen to go on the multithreading paradigm and they also chosen to use for implementation different languages (Java – 23, C# – 5, C++ – 4, Scala –2). The project also required to do a written documentation that includes design pattern oriented analysis of the design decisions, theoretical performance evaluation, and results of a set of the empirical testing.

The students' choices are influenced by the mainstream technologies and their abilities in working in a specific programming language. These abilities are on their turn influenced by their personal experience, which is, in a vast majority of cases, driven by their employers, not by the academic environment.

5 Conclusions

The main conclusion of our experience with teaching PDC topics is that even if they are very necessary and important to be studied due to the last development of systems architectures and of the associated programming, it is also very important to take into the consideration the latest approaches and paradigms applied in the IT industry. The need for high productivity induces some modifications in the ways the programming activity and software systems construction are developed. All these lead to a new category of software developers which are not supposed to understand all the components that they usually assemble. An adapted and simplified curriculum should be in this case specified. In such a curriculum some PDC topics should be included, but in a pragmatic, usage-oriented way – how to use parallel programming frameworks/libraries, etc.

Also, there are some topics – as MPI – that have a great importance for the well understanding of some basic concepts of Parallel Programming, but they are not yet very much used in the industry. This leads to a low level of interest for this topic from the undergraduate students.

Also, the acquisition of the theoretical concepts and general principles is not very good, since our students are now, very much oriented on achieving practical abilities.

Master students that choose a specialisation that includes High Performance Computing, have of course, a much higher degree of interest and opening to fields as Scientific Computation, Models of Computation, or Correctness and Formal Methods.

The premise of our study was that the success of introducing new topics in the undergraduate curriculum, and most importantly achieving the desired

learning outcomes, depend in a great measure on the level of interest of the students in that topic. This premise proved to be correct.
On the other hand, the level of interest on different topics of Parallel and Distributed Computing depends very much on the students' levels. The distinction between undergraduate and master students is very clear, but between undergraduates we may emphasise at least two classes of interest.

A solution could be based on moving more topics on the elective courses. Another, more complex solution, would involved a larger aria of Computer Science fields, that could be included into a new defined educational degree. A proposal that comes from Cluj Innovation City Project [8] is to develop *Vocational Studies*. The proposal claims that this way an important part of the IT industry employees could come directly from an IT related vocational curricula, and this would reduce part of the pressure on the employment market, but most importantly would engage young people into the industry in their early 20ties. (The drawbacks of this proposal have not been studied, yet.)

There is an important evolution of the software development by using Parallel and Distributed Computing, and by using in a more efficient way the present hardware facilities. There is also a wide acceptance of the syntagma "Parallelism is the future of programming". Still, we may paraphrase the title of the paper of Domenico Talia: "Parallel computation still not ready for the mainstream" [6] and say: "Mainstream still not ready for [all kind of] Parallel Computation".

References

1. Bloom, B. S., Engelhart, M. D.; Furst, E. J., Hill, W. H.; Krathwohl, D. R. *Taxonomy of educational objectives: The classification of educational goals*. Handbook I: Cognitive domain. New York: David McKay Company.(1956).
2. David J. John, Stan J. Thomas. *Parallel and Distributed Computing across the Computer Science Curriculum*. Parallel & Distributed Processing Symposium Workshops (IPDPSW), 2014 IEEE International. DOI: 10.1109/IPDPSW.2014.121.
3. Guoming Lu, Jie Xu,Jieyan Liu, Bo Dai, Shenglin Gui, Siyu Zhan. *Integrating Parallel and Distributed Computing Topics into an Undergraduate CS Curriculum at UESTC.* Parallel and Distributed Processing Symposium Workshop (IPDPSW), 2015 IEEE International DOI: 10.1109/IPDPSW.2015.66
4. Sushil K. Prasad, Almadena Chtchelkanova, Frank Dehne, Mohamed Gouda, Anshul Gupta, Joseph Jaja, Krishna Kant, Anita La Salle, Richard LeBlanc, Andrew Lumsdaine, David Padua, Manish Parashar, Viktor Prasanna, Yves Robert, Arnold Rosenberg, Sartaj Sahni, Behrooz Shirazi, Alan Sussman, Chip Weems, and Jie Wu. 2012. *NSF/IEEE-TCPP Curriculum on Parallel and Distributed Computing - Core Topics for Undergraduates* - Version I, http://www.cs.gsu.edu/ tcpp/curriculum/index.php, 55 pages.
5. David B. Skillicorn, Domenico Talia. *Models and languages for parallel computation*. Journal ACM Computing Surveys, Volume 30 Issue 2, June 1998, pp. 123-169.
6. Domenico Talia. em Parallel computation still not ready for the mainstream. Communications of the ACM. Volume 40 Issue 7, July 1997, pp. 98-99.

7. *** *ACM Curricula 2013 Report*. https://www.acm.org/education/CS2013-final-report.pdf. pp 144- 154.
8. *** *Cluj Innovation City.* http://www.clujinnovationcity.com.