The World Teaching of Parallel and Distributed Programming

Naylor G. Bachiega
University of São Paulo
ICMC-USP
São Carlos, Brazil
naylor@usp.br

Paulo S. L. Souza
University of São Paulo
ICMC-USP
São Carlos, Brazil
pssouza@icmc.usp.br

Simone do R. S. de Souza
University of São Paulo
ICMC-USP
São Carlos, Brazil
srocio@icmc.usp.br

Abstract

The parallel and distributed programming is widely used in cloud computing. This is due to factors such as the popularization of multicore and heterogeneous CPU, which allow significant performance gains over sequential processing. ACM and IEEE recommend to teach the parallel and distributed programming and also provide details on how this topic should be studied in computing courses. However, the teaching of parallel programming is not trivial and differs in various undergraduate courses throughout countries. The differences include teaching approaches, theoretical and practical classes, instructional materials, embedded topics, number of hours, required and elective topics, prerequisites, among others. This article presents how the main institutions of higher education in the world approach the teaching of parallel programming, with the purpose of evaluating its adherence to the syllabus proposed by ACM and IEEE. The universities considered in this study were chosen according to a specific ranking and geographically separated by continent. After that, we compared with the ACM and IEEE-Computer Society reference curricula, highlighting the main differences. Our results show that there are still significantly differences regarding teaching HPC, mainly in relation to syllabus and topics covered, and that is hard to find available documents that show clearly how such subjects are conducted.

1 Introduction

Parallel and Distributed Computing (PDC) is an essential tool for the development of diverse knowledge areas which require high computational power to run their particular applications [8].

The PDC is an area widely used in Cloud Computing [23], and it allows the use of the processing power available in parallel architectures for the solution of problems that require their own programming models with a higher degree of complexity for the execution of one or more tasks simultaneously [41].

The movement known as "parallel thinking", proposed by [3], has been emphasizing for almost a decade that programs must be developed with some level of parallelism. The aim is to reach more expressive performance gains, when compared to a sequential code that runs on CPUs with higher frequencies and/or parallelism with pipelines or multiple functional units [14]

PDC teaching, unlikely sequential teaching, addresses particular challenges. A few examples of such particularities are: the use of different primitive communication and synchronization; process abstractions or different interacting threads; parallel programming models; load balancing problems; and a more significant impact on the development and performance of applications regarding the computing platform used [12].

These questions need to be learned by students who must develop the skills and competences in order to work efficiently with these computing environments, which are more common in companies and universities nowadays [24].

Parallel Programming, a part of PDC, has become active and intrinsic for the technologies currently available, especially considering the different programming models related to these different parallel architectures.

This type of education can be splited in two parts, theoretical and practical, and it can be very challengeable. In the theoretical approach, it is necessary to verify if the teaching method is adequate for the students to acquire competences in a specific topic. In the practical approach, it is essential to define the work environment and the basic architecture to perform the practical exercises in order to develop the expected skills.

Because of the importance of PDC teaching in a global context, international institutions such as the Association for Computing Machinery (ACM) and the Institute of Electrical and Electronics Engineers (IEEE) recommend that PDC should be considered as one of the great knowledge areas in computing [1].

Regarding parallel programming teaching, three choices can be considered: (1) teaching parallel computing to a small group that will develop libraries, which will be made available to other groups; (2) adding advanced parallelism subjects or lectures to course grids; or (3) incorporating parallelism in course grids for freshmen, encouraging students to naturally think about parallelism and to use sequential computing only for exceptions [3].

For this reason, high education courses in computing need to incorporate these guidelines on PDC teaching into their course grids, so that they prepare students for an increasingly parallel and distributed world.

The goal of this paper is to present a survey of the computing pedagogical projects, focusing on the teaching of parallel programming in some of the most important universities around different continents.

The 30 universities analyzed in this paper were chosen according to the scoring systems available in two world rankings: The World University Rankings [28] and QS World University Rankings [21]. We have selected five universities from each region: North America, Europe, Asia, South America, Africa, and Oceania.

Our survey shows how adherent the course grids analyzed are concerning the PDC teaching suggested by ACM and IEEE, and highlights similarities and differences among these guidelines, topics uncovered and difficulties in finding official information on websites.

This paper is divided into six sections:

- Section 2 discusses how PDC should be studied according to the ACM and IEEE guidelines.
- Section 3 presents the methodology used to obtain the data presented in this article.
- Section 4 presents the overview of computing courses in the world and their characteristics.
- Section 5 presents the conclusions of this paper and the future work.

2 PARALLEL AND DISTRIBUTED COMPUTING IN COMPUTER SCIENCE COURSE GRIDS

To help standardize PDC teaching, ACM and IEEE-Computer Society have teamed up to establish international curriculum guidelines for undergraduate courses in computer science and related fields [1].

The guidelines proposed by the 2013 ACM present eighteen knowledge areas, which represent relevant fields of study in computing. PDC is one of these areas, due to its current importance in the development of computational solutions.

Before the review of the 2013 curriculum, PDC content, including parallel/distributed programming, was spread among other knowledge areas. In this curriculum proposal, due to PDC's current importance, it was categorized into a new area including specific topics such as: fundamentals of parallelism, parallel decomposition, parallel algorithms, analysis and programming, and parallel architectures [1], as shown in Table 1.

Table 1. Teaching PDC in 2013 curriculum guidelines for computer courses, according to ACM/IEEE

	Total Instructional Hours		
Topics	**Essential**	**Additional**	**Includes Electives**
Parallelism Fundamentals	2		No
Parallel Decomposition	1	3	No
Communication and Coordination	1	3	Yes
Parallel Algorithms, Analysis, and Programming		3	Yes
Parallel Architecture	1	1	Yes
Parallel Performance			Yes
Distributed Systems			Yes
Cloud Computing			Yes
Formal Models and Semantics			Yes

Teaching PDC in the computing curriculum proposed by ACM/IEEE is quantified in hours. This unit is defined as the time needed to present a traditional reading material. This time does not include extra work, practical classes in laboratories, lectures, among others.

The essential hours, pointed out in the Table, should be a required part of any Computer Science curriculum, i.e., the topics need to be embedded within the curriculum as a specific subject or addressed in other syllabus [1].

The additional hours are usually essential for an undergraduate course in computer science. However, some courses may allow the student to specialize in a distinct topic from the third year on, and do not need to cover the

other topics. Another problem that may influence the application of additional hours concerns administrative issues such as teacher shortage and physical and budgetary constraints.

Further, according to the ACM/IEEE guidelines, a computer science curriculum should aim to cover 90 to 100% of the additional topics, with 80% as minimum.

According to the ACM/IEEE, a curriculum must include elective material for its courses, since the essential topics are insufficient for a complete curriculum. Also, a syllabus can consist of elective material, as required in those specific courses, specializations, masters, and doctorates.

In addition to the necessary hours, the ACM/IEEE guidelines point out essential topics to be addressed in computer science curricula. Table 2 contains the subjects that should be covered in these undergraduate programs.

Considering these restructurings in PDC teaching by ACM and IEEE, and the importance of parallel/distributed programming in the training of future computer professionals, it is important to determine how parallel programming is being addressed by universities. The next sections show this scenario.

3 METHODOLOGY FOR DATA COLLECTION

In one of the steps used to carry out the survey proposed, 30 universities were chosen from two world rankings: The World University Rankings (WUR) [28] and QS World University Rankings (QS) [21].

These rankings use indicators such as: academic reputation, employer reputation, college/student ratio, college citations, international college index, global student index, among others.

In order to avoid too many results, we chose five universities in the six continents: North America, Europe, Asia, South America, Africa and Oceania.

For each score of each university, we created an index that sums the positions of both rankings. Each ranking was filtered by the best universities that offer computer science courses. For example, Stanford University (USA) is in position 1 in WUR and in position 2 in QS, so we set its score to 3.

Unfortunately, there was no WUR for Africa. In this case, we decided to use another ranking called Center for World University Rankings [6], which is equally important. Thus, for this continent, we added the CWUR with the QS.

Table 3 shows the selected universities separated by continents and sorted by the score (S) of the rankings (R).

All the data collected were available on the official websites of each university, such as syllabus, teaching plans, lesson plans, and pedagogical projects. The data refer to the year that the subjects were offered, the hours, required or elective, bibliographic references, co-requisites, and prerequisites.

With the results obtained, in the next section, we show the overview of the parallel and distributed programming teaching in the universities selected.

4 THE SCENERY OF PARALLEL AND DISTRIBUTED PROGRAMMING

To begin our discussion of the parallel programming teaching overview in the world, in Table 4, we show the names of PDC related subjects, which refer to parallel and distributed programming teaching. It is possible to notice that there is a large number of subjects titles related to PDC.

In addition, we highlight that the subjects are offered as elective in 12 universities, as required in 7, in 5 of them it was not possible to find if they are elective or required (N/S) and 6 institutions did not offer the subject (post graduation only) or did not disclose the syllabus of their computer science course (N/A).

Figure 1 shows whether the subject is offered as elective or required. For most of the universities, it is elective and 11 of them do not provide the information.

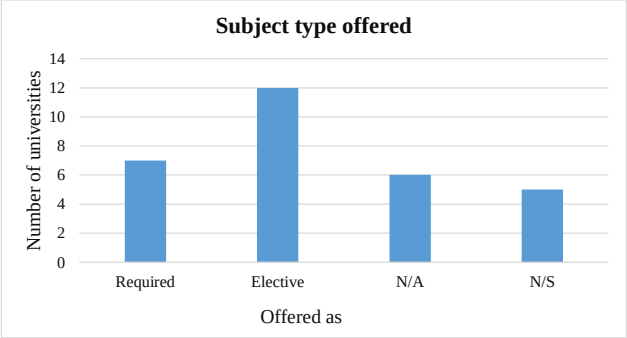

Figure 1. Subject type offer related to PDC in universities

Table 5 presents the prerequisites (subjects students need to attend before the parallel programming course) mostly used by the universities.

Operating Systems, Programming, and Architecture subjects are the most common prerequisites. Out of the 30 institutions analyzed, 11 had no prerequisites or did not make this information avaliable, and 6 of them offered no data on PDC related courses. None of the universities presented the

Table 2. Topics covered in the guidelines proposed by ACM/IEEE

Topics	Subject Addressed		
	Essential	**Additional**	**Electives**
Parallelism Fundamentals	Multiple simultaneous computations, goals of parallelism, communication, coordination, and programming errors	Not specified	Not specified
Parallel Decomposition	Need for communication and coordination. Synchronization, Independence and partitioning	Parallel decomposition concepts, task-based decomposition, data-parallel decomposition and actors and reactive processes	Not specified
Communication and Coordination	Shared memory and consistency	Message passing and atomicity	Consensus and conditional actions
Parallel Algorithms, Analysis, and Programming	Not specified	Critical paths, work and span. The relation to Amdahl's law, speed-up, scalability. Naturally (embarrassingly) parallel algorithms, and parallel algorithmic patterns	Parallel graph algorithms, matrix computations, producer-consumer and pipelined algorithms, and non-scalable parallel algorithms
Parallel Architecture	Multicore processors, shared, and distributed memory	Symmetric multiprocessing (SMP), SIMD, and vector processing	GPU, co-processing, Flynn's taxonomy, instruction level support for parallel programming, memory issues, and topologies
Parallel Performance	Not specified	Not specified	Load balancing, performance measurement, scheduling and contention, evaluating communication overhead, data management, and power usage and management
Distributed Systems	Not specified	Not specified	Faults, distributed message sending, distributed system design tradeoffs, distributed service design, and core distributed algorithms
Cloud Computing	Not specified	Not specified	Internet-Scale computing, cloud services, virtualization, and cloud-based data storage
Formal Models and Semantics	Not specified	Not specified	Formal models of processes and message passing, parallel computation, shared memory consistency, algorithmic progress, and computational dependencies. Linearizability and techniques for specifying and checking correctness

Table 3. The universities selection according to the rankings

	Index	University	R
North America	1	Stanford University [25]	W(1) Q(2)
	2	Massachusetts Institute of Technology [13]	W(2) Q(1)
	3	Carnegie Mellon University [5]	W(6) Q(3)
	4	Harvard University [9]	W(11) Q(6)
	5	Princeton University [19]	W(12) Q(8)
Europe	6	University of Oxford [36]	W(3) Q(7)
	7	University of Cambridge [31]	W(5) Q(5)
	8	ETH Zurich [7]	W(4) Q(9)
	9	Imperial College London [11]	W(9) Q(12)
	10	École Polytechnique Fédérale de Lausanne [42]	W(10) Q(18)
Asia	11	National University of Singapore [16]	W(13) Q(10)
	12	Tsinghua University [29]	W(20) Q(20)
	13	Peking University [17]	W(25) Q(17)
	14	Hong Kong University of Science and Technology [10]	W(28) Q(14)
	15	Nanyang Technological University [15]	W(31) Q(16)
South America	16	University of São Paulo [38]	W(?) Q(51)
	17	University of Chile [33]	W(?) Q(51)
	18	State University of Campinas [26]	W(?) Q(101)
	19	University of Buenos Aires [30]	W(?) Q(151)
	20	Pontifical Catholic University of Chile [18]	W(201) Q(101)
Oceania	21	University of Melbourne [34]	W(39) Q(14)
	22	University of Technology, Sydney [39]	W(83) Q(36)
	23	Australian National University [2]	W(83) Q(37)
	24	University of New South Wales [35]	W(101) Q(41)
	25	Queensland University of Technology [22]	W(98) Q(51)
Africa	26	University of Cape Town [32]	CW(223) Q(301)
	27	University of the Witwatersrand [40]	CW(230) Q(?)
	28	Stellenbosch University [27]	CW(448) Q(401)
	29	University of Pretoria [37]	CW(438) Q(401)
	30	Cairo University [4]	CW(452) Q(301)

Table 4. Subjects related to PDC

Index	Course Title	Offered as
1	Parallel Computing	Elective
2	Multicore Programming	Elective
3	Parallel Computer Architecture and Programming	Elective
4	Introduction to Distributed Computing	Elective
5	Distributed Systems	Elective
6	Concurrent Programming	Required
7	Concurrent and Distributed Systems	Required
8	Parallel Programming	Required
9	Distributed Algorithms	Elective
10	Parallelism and Concurrency	N/S
11	Parallel and Concurrent Programming	Elective
12	Distributed Computing	N/S
13	N/A	N/A
14	Parallel Programming	Elective
15	N/A	N/A
16	Concurrent Programming	Required
17	Parallel Computing and Applications	Elective
18	Introduction to Parallel Programming	Elective
19	Complex Systems in Parallel Machines	Elective
20	N/A	N/A
21	N/A	N/A
22	N/A	N/A
23	Parallel Systems	Elective
24	Distributed Systems	N/S
25	High Performance and Parallel Computing	N/S
26	N/A	N/A
27	Parallel Computing III	N/S
28	Concurrent Programming 1 and 2	Required
29	Concurrent Systems	Required
30	Parallel Processing	Required

(N/A) Not Available, (N/S) Not Specified

Table 5. Prerequisites for PDC subjects

Index	Prerequisite
1	Compilers; Principles of Computer Systems
2	Introduction to Algorithms
3	Intro to Computer Systems
4	Operating Systems
5	Introduction to Programming Systems; Operating Systems; Advanced Programming Techniques
6	Object Oriented Programming
7	Operating Systems
8	N/S
9	Concurrency, Maths
10	Functional programming; Algorithms; Computer Architecture
11	N/S
12	N/S
13	N/A
14	N/S
15	N/A
16	N/S
17	Design and Analysis of Algorithms
18	N/S
19	N/S
20	N/A
21	N/A
22	N/A
23	N/S
24	N/S
25	N/S
26	N/A
27	Operating Systems II; Computer Networks II; Analysis of Algorithms II
28	N/S
29	Operating Systems; Data Structures and Algorithms
30	Computer Architecture and Organization

co-requisites (subjects that can be taken concurrently with parallel computing courses).

Figure 2 shows the total, practical and theoretical course loads. Despite we have selected 30 universities, 22 of them did not detail their specific course loads and many only show the total course load but do not specify the number of hours reserved for practical and theoretical education.

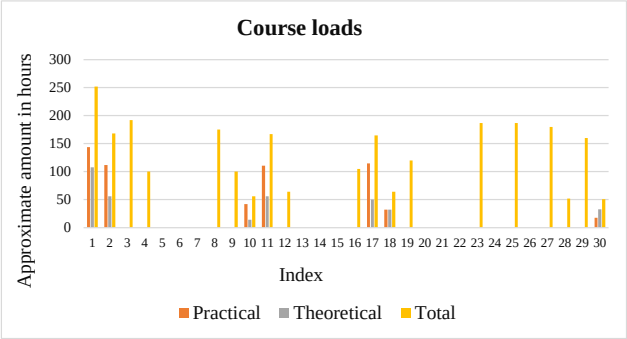

Figure 2. Course Loads of PDC subjects

Figure 3 provides the year in the course each subject was offered, and 22 institutions did not provide this data.

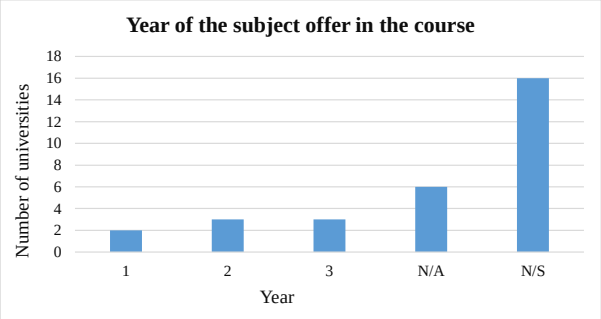

Figure 3. Year of offer of the topics related to PDC.

Most universities do not specify the year because the subject is elective and it depends on the offer period for the course announcement. Also, many students must complete the prerequisites for attending PDC-related courses.

Figure 4 brings together the years of the bibliographies

used in PDC-related courses in the universities that provided the data.

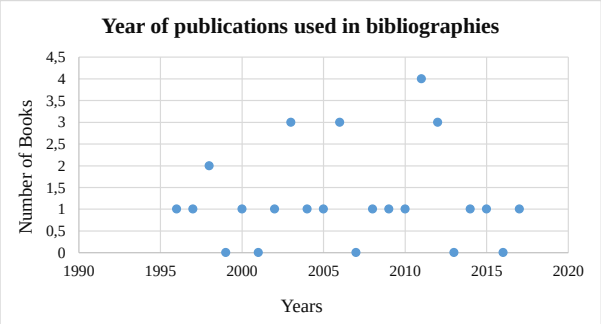

Figure 4. Year of bibliographic references used in the topics related to PDC

Most institutions have bibliographies published between 1990 and 2017. Some of the literature is more than 20 years old and, therefore, the information on the university official website might be outdated.

Table 6 presents the adherence of PDC topics to the ACM/IEEE guidelines. One can observe that most of the courses reach the required hours of Table 2, but some of them do not cover the total amount, like Parallel Architecture, which is not always approached.

Another point is that most universities condense all the topics that are essential in only one subject and perhaps do not cover specific topics deeply.

It should also be noted that many courses (indexes 11, 16 and 25, i.e.) include all topics (essential, additional and elective) in only one subject, making the course load really intense and increases the number of hours exponentially.

Despite the importance of parallel programming in computing courses, some universities have not yet included this type of programming in their syllabus, such as indexes 7 and 12.

Another threat to the viability of parallel programming teaching is related to 3-year courses. In this case, after finishing their course, the students may choose to study another year and get another degree. Depending on the student's choice, there may be no contact with parallel programming; an example is index 11.

However, the university (index 11) offers an area focused entirely in Parallel Computing, which covers almost 100% of the curriculum refereed by ACM/IEEE (Figure 5).

Index 9 offers two courses that have PDC-related topics in their syllabus. In 4-year courses, the subject Scalable Distributed Systems Design is offered and, in 3-year courses, Distributed Algorithms.

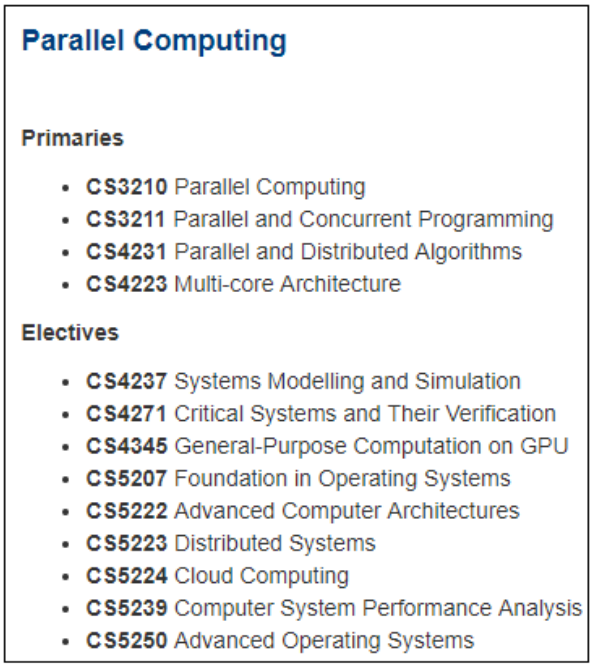

Figure 5. Parallel Computing Course [16].

We observed that in Oceania, two universities (indexes 21 and 22) do not offer PDC topics in undergraduate studies, only in postgraduation. Besides, the other universities (indexes 23, 24 and 25) in Oceania provide more than two PDC topics in undergraduate studies, encompassing more than the ACM/IEEE curriculum guidelines suggest.

- Index 23: (2 topics) High Performance Scientific Computation and Parallel Systems;
- Index 24: (3 topics) Foundations of Concurrency, Modelling Concurrent Systems and Distributed Systems; and
- Index 25: (2 topics) High Performance and Parallel Computing and Cloud Computing.

Apart from this, index 30 in Africa presented the learning relationship regarding the classification of the Bloom Taxonomy, a differential that allows students and professors to determine the knowledge level expected in the syllabus.

Another item that caught our attention was the university (index 8) that made all the course material available online and easily accessible. Materials such as classes, lectures, commented classes, exercises, resolution of exercises, program codes, among others.

Also, topics such as Parallel Performance, Cloud Computing and Formal Models and Semantics have not been explored so much by the courses at these universities.

Table 6. Relationship between the adherence of the parallel and distributed programming topics and the ACM/IEEE curriculum guidelines.

ACM/IEEE	Parallelism Fundamentals	Parallel Decomposition	Communication and Coordination	Parallel Algorithms, Analysis, and Programming	Parallel Architecture	Parallel Performance	Distributed Systems	Cloud Computing	Formal Models and Semantics
Index									
1	X	X	X	X	X				
2	X	X	X	X					
3	X	X	X	X	X				
4	X	X	X	X			X		
5	X	X	X	X			X		
6	X	X	X	X					
7	X	X	X		X		X		
8	X	X	X	X					
9	X	X	X	X					
10	X	X	X	X					
11	X	X	X	X	X	X	X	X	
12	X	X	X				X		
13	-	-	-	-	-	-	-	-	-
14	X	X	X	X					
15	-	-	-	-	-	-	-	-	-
16	X	X	X	X	X	X	X		X
17	X	X	X	X	X				
18	X	X	X	X	X				
19	X	X	X	X	X				
20	-	-	-	-	-	-	-	-	-
21	-	-	-	-	-	-	-	-	-
22	-	-	-	-	-	-	-	-	-
23	X	X	X	X	X	X			
24	X	X	X	X	X	X	X		
25	X	X	X	X	X	X		X	
26	-	-	-	-	-	-	-	-	-
27	X	X	X	X					
28	X	X	X	X					
29	X	X	X	X					
30	X	X	X	X	X	X			

We found some difficulties to collect information on university websites. Besides, many institutions do not add the topics covered, teaching methodologies, syllabus, and other relevant information. On some websites, only the name of the course has been reported.

5 CONCLUSIONS

Parallel and distributed programming is still difficult in contemporary programming environments [1]. However, understanding parallel processing is vital for computer courses, and learning these programming models as soon as possible can strengthen student learning in this area.

Although many universities do not report the year of the course offer, it is possible to note that most courses chose to offer this subject in the following years because of their high degree of complexity and to meet their prerequisites. Only two universities (indexes 11 and 28) offer this subject in the first year of the course. Index 11 is targeted specifically to Parallel Computing, while at index 28, the bachelor's course is finished in one year.

Many universities use non-contemporary bibliographies, consequently, they do not address relatively new topics, such as Cloud Computing. This may happen because, unlike the academic field (which deals with with fundamental computing problems), cloud computing involves many real-world applications and tools and this makes its teaching expensive and finding qualified personnel difficult [20].

Many courses cover topics related to the essential hours within the subjects called Introduction to Distributed Computing, Distributed Systems, Concurrent Systems; but in the syllabus, it is not possible to find items related to the essential topic of Parallel Architecture.

Moreover, few institutions offer the subjects as required, and others do not even offer them, which makes the adherence to parallel programming later and, consequently, hampers the dissemination of "parallel thinking".

Another critical factor for the success of the subject, based on the knowledge acquired by the students, is the course load. In this paper, a significant divergence was observed concerning the number of hours required.

Noteworthy is the difficulty to convert different ways to count the subject hours per week, which, in North America for example, are named *unit* and may have different values for each university. In Europe, we found credits stipulated in hours and ECTS (European Credit Transfer System). Credits defined as EFTSL (Equivalent Full-Time Student Load) have also been found. This difference in terminology makes the amount of hours needed to complete a given subject a bit subjective.

About the adherence of the institutions to the ACM/IEEE curriculum guidelines, some universities go beyond the minimum proposal, covering current subjects. However,

many do not address essential topics, such as Parallel and Distributed Programming, which is an extremely important topic nowadays.

For future work, the positive and negative points aspects of each subject will be analyzed concerning their support materials, the tools used in the classroom, as well as their methodologies of practical and theoretical education.

References

[1] ACM/IEEE-CS. Computer science curricula 2013. Technical report, ACM Press and IEEE Computer Society Press, December 2013.

[2] Australian National University. Parallel Systems, Maio 2018.

[3] G. E. Blelloch. Parallel thinking. *SIGPLAN Not.*, 44(4):1–2, Feb. 2009.

[4] Cairo University. Course Specification, Maio 2018.

[5] Carnegie Mellon University. Bachelors Curriculum - Admitted 2014, 2015 & 2016, Maio 2018.

[6] Center for World University Rankings. Center for World University Rankings, Abril 2018.

[7] ETH Zurich. 252-0029-00L Parallel Programming, Maio 2018.

[8] T. Franczak, A. Nkansah, T. Marrinan, and M. Papka. A path from serial execution to hybrid parallelization for learning hpc. In *Proceedgins of the 2017 Workshop on Education for High-Performance Computing*, 2017.

[9] Harvard University. Introduction to Distributed Computing, Maio 2018.

[10] Hong Kong University of Science and Technology. BEng in Computer Science, Maio 2018.

[11] Imperial College London. CO347 Distributed Algorithms, Maio 2018.

[12] V. Kumar, A. Grama, A. Gupta, and G. Karypis. *Introduction to Parallel Computing: Design and Analysis of Algorithms*. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 1994.

[13] Massachusetts Institute of Technology. 6.816/6.836 Multicore Programming, Maio 2018.

[14] G. E. Moore. Cramming more components onto integrated circuits, reprinted from electronics, vol 38, nr 8, 1965. *IEEE Solid-State Circuits Society Newsletter*, 11(3):33–35, Sept 2006.

[15] Nanyang Technological University. Computer Science (CS) Programme, Maio 2018.

[16] National University of Singapore. Computer Science Focus Areas for BComp (CS), Maio 2018.

[17] Peking University. Undergraduate Programs, Maio 2018.

[18] Pontifical Catholic University of Chile. Bachelor: Major in Computer Science, Maio 2018.

[19] Princeton University. COS-418, Fall 2016: Distributed Systems, Maio 2018.

[20] J. Qiu, S. Kamburugamuve, H. Lee, J. Mitchell, R. Caldwell, G. Bullock, and L. Hayden. Teaching, learning and collaborating through cloud computing online classes. In *Proceedgins of the 2017 Workshop on Education for High-Performance Computing*, 2017.

[21] QS World University Rankings. Computer Science & Information Systems, Abril 2018.

[22] Queensland University of Technology. Bachelor of Information Technology (Computer Science), Maio 2018.

[23] E. Roloff, M. Diener, L. P. Gaspary, and P. O. A. Navaux. Exploiting load imbalance patterns for heterogeneous cloud computing platforms. In *Proceedings of the 8th International Conference on Cloud Computing and Services Science - Volume 1: CLOSER,*, pages 248–259. INSTICC, SciTePress, 2018.

[24] J. A. Shamsi, N. M. Durrani, and N. Kafi. Novelties in teaching high performance computing. In *IEEE Int Par and Dist Proc Sym Workshop*, pages 772–778, May 2015.

[25] Stanford University. CS149 - Parallel Computing, Maio 2018.

[26] State University of Campinas. MC970/MO644 Parallel Programming, Maio 2018.

[27] Stellenbosch University. Academic Programmes and Faculty Information, Maio 2018.

[28] The World University Ranking. Computer Science - Times Higher Education, Abril 2018.

[29] Tsinghua University. The Undergraduate and Graduate Courses Taught in English and Open to the International Visiting/Exchange Students at Tsinghua University, Maio 2018.

[30] University of Buenos Aires. Faculty of Exact and Natural Sciences, Maio 2018.

[31] University of Cambridge. Computer Science Tripos, Maio 2018.

[32] University of Cape Town. Undergraduate Courses, Maio 2018.

[33] University of Chile. Course Program, Maio 2018.

[34] University of Melbourne. Distributed Computing Project (COMP90019), Maio 2018.

[35] University of New South Wales. Distributed Systems - COMP9243, Maio 2018.

[36] University of Oxford. Concurrent Programming: 2017-2018, Maio 2018.

[37] University of Pretoria. Concurrent systems 226 (COS 226), Maio 2018.

[38] University of São Paulo. Concurrent Programming, Maio 2018.

[39] University of Technology Sydney. 42009 Parallel and Multicore Computing, Maio 2018.

[40] University of the Witwatersrand. 2018 ScI Rules Syllabuses, Maio 2018.

[41] G. Zarza, D. Lugones, D. Franco, and E. Luque. An innovative teaching strategy to understand high-performance systems through performance evaluation. *Procedia Computer Science*, 9:1733 – 1742, 2012.

[42] École Polytechnique Fédérale de Lausanne. Parallelism and concurrency, Maio 2018.