# Methods and tools for teaching parallel and distributed computing in universities: a systematic review of the literature

*Yuriy* Sitsylitsyn*

Bogdan Khmelnitsky Melitopol State Pedagogical University, Department of Computer Science and Cybernetics, Melitopol, Ukraine

**Abstract.** As computer hardware becomes more and more parallel, there is a need for software engineers who are experienced in developing parallel programs, not only by "parallelizing" sequential designs. Teach students a parallelism in elementary courses in computer science this is a very important step towards building the competencies of future software engineers. We have conducted research on "teaching parallel and distributed computing" and "parallel programming" publications in the Scopus database, published in English between 2008 and 2019. After quality assessment, 26 articles were included in the analysis. As a result, the main tool for teaching parallel and distributed computing is a lab course with a C++ programming language and MPI library.

## 1 Introduction

Traditionally, undergraduate computer science students are taught sequential programming through a one-way programming model in higher education majors. It is typical to wait for a student to develop knowledge of serial programming before teaching parallel programming. As computer hardware becomes more and more parallel, there is a need for software engineers who are experienced in developing parallel programs, not only by "parallelizing" sequential designs. Teach students a parallelism in elementary courses in computer science this is a very important step towards building the competencies of future software engineers.

In April 2005, AMD released the AMD64 architecture Opteron 2-core processor for servers. In May 2005, Intel released the Pentium D processor x86-64 architecture, which was the first 2-core processor designed for personal computers. This was Intel's quick response to AMD's challenge. So the era of multi-core processors began. The growth of multi-core and multi-core processing has certainly added new relevance to the teaching of parallel programming. Since 2000, in the world scientific literature many articles on the topic of parallel and sequential programming and learning parallel and sequential computing.

According to Brown et al. [5], given the importance of these architectures, parallel programming becomes indispensable for undergraduate students in such specialties as computer science, computer engineering, and software engineering. Students in these specialties must be able to program in a satisfactory manner, both sequentially and in parallel. Acquaintance of students with parallelism should begin from the early periods [14], then students will consider it as a natural and general part of programming (and not as advanced and rarely used content) [4]. In addition, switching from serial to parallel programming is a difficult task [14]. There is a general opinion that the topics of parallelism should be distributed throughout the undergraduate program [3]. However, in most universities the concepts of parallelism are studied only in the last courses [18].

A tool that can be used to facilitate access from students to content is the use of distance learning, allowing the student to study the course content according to his daily routine, adapting the study to your time and allowing the teacher to monitor [24]. To conduct new research in the field of teaching parallel programming in undergraduate programs, it is important to take into account existing work and related experience, to consider the difficulties and problems that teachers face. A systematic review of the literature is a method that allows you to determine all relevant searches for a given research question. Given the above, it is necessary to organize a systematic review of the literature with the aim of analyzing scientific works related to teaching parallel programming in undergraduate programs in computer science. To do this, a study was conducted in articles indexed in the Scopus, Web of Science and Google Scholar.

### 1.1 Theoretical bases

The purpose of study is to examine the available literature on the development of "parallel thinking" among students in the specialty of computer science for the development of parallel programs.

All articles in the review mention such terms as parallel and distributed programming. Let us dwell briefly on these concepts.

Parallel programming is a method that aims to reduce the dependency between sections of the same code, which allows these sections to be executed out of order when executing processes and / or threads. However, this

* Corresponding author: yuriy@mdpu.org.ua

execution is not necessarily parallel, since the actual parallelism of execution depends on the parallel architecture of the computer, processors with support for multithreading or speculation at the thread level, and / or using parallelism libraries (i.e. parallel programming).

Parallel programming explores the potential parallelism that exists in code using parallelism libraries that allow explicit execution of processes and / or threads to simultaneously execute program code. In this case, it is important that a parallel architecture exists (for example, a multi-core processor), so that parallel execution is actually supported. Sometimes the terms "parallel computing" and "parallel programming" are used as synonyms, but it is important to note that parallel computing does not imply simultaneity, but the potential for simultaneity. Thus, parallel programming involves identifying and reducing dependencies between sections of the same code, and parallel computing involves using parallelism libraries to examine these sections in parallel architectures.

Distributed programming is a method used to increase the scalability of parallel code execution using a communications network. In this context, for parallel codes, it is necessary to use message passing for collective communication, that is, the use of primitives, such as sending and receiving. Distributed programming is not necessarily used for concurrency, because processing can be distributed and sequential, but in the context of this article, the distributed potential of a cluster computer system offers the conditions for studying concurrency on a larger scale. For this reason, the concept of parallel and distributed programming is also used for systems in which several nodes with parallel architectures (for example, several cores) are connected by a high-performance communication network. In a cluster architecture of this type, parallel programming is performed by shared (operational) memory (intra-node communication) and transmission of network messages (external communication).

This research is trying to understand the methods and tools of teaching parallel programming in higher education institutions. To study existing literature, the following questions were used:

1. What methods and tools are used in teaching parallel and distribution programming?
2. What do the authors define as problems in teaching parallel and distribution programming?

### 1.2 Materials and methods

This systematic literature review utilized the PRISMA guidelines and flow chart. PRISMA guidelines include a 27-item checklist and a four-phase flow diagram outlining the items essential for transparency in conducting literature reviews [16].

In order to be included in this review, it is necessary to carry out research, to be peer-reviewed and to be published in English in a scientific journal article between 2000 and 2019. Research also needs to solve at least one of the research questions.

The following bases were used for the search: Scopus (https://www.scopus.com), Web of Science (https://apps.webofknowledge.com/) and Google Scholar (https://scholar.google.com). The last search was performed on January 29, 2020.

The search phrases used were: "teaching parallel and distributed programming", "teaching parallel and distributed computing".

The following criteria were used sequentially against article abstracts to select studies for inclusion:

Criteria 1: Study published between 2000 and 2020 in English

Criteria 2: Study published in scholarly journal

Criteria 3: Research was conducted for undergraduate computer science students

Criteria 4: Extracted data aligns with current study focus and research questions.

The results of the database search we entered in the table.

**Table 1.** The results of the database search.

| Search terms | Database | Hits |
|---|---|---|
| "teaching parallel and distributed programming" | Scopus | 41 |
| | Web of Science | 82 |
| | Google Scholar | 5 |
| "teaching parallel and distributed computing" | Scopus | 52 |
| | Web of Science | 78 |
| | Google Scholar | 51 |

Next, we remove articles from the results that do not mention teaching parallel programming. We bring all found articles into a single list. Then we delete articles that do not describe studies at the undergraduate or graduate level. Next, in a single list, we delete duplicate articles obtained from different databases. As a result, we get 26 articles.

## 2 Methods and tools for teaching parallel and distributed computing

After the screening, we received 26 research articles.

If we look at the articles in terms of practical experience in teaching parallel and distributed computing, it can be said that 85% of articles are written on the basis of their own experience and 15% of review articles in which the authors provide the results of theoretical studies on the teaching of parallel and distributed computing.

When considering methods of teaching parallel and distributed computing, the authors of the article pay attention to the following methods: a course of laboratory work that addresses selected topics or paradigms of parallel and distributed programming [1], [26]; course of lectures and laboratory work – here the authors give a complete theme of the course, which they teach: topics of lectures and laboratory work [3], [15]; some authors [2]; [8] introduce in the course of laboratory work with visualization of parallel computing, which according to the authors should promote the development of parallel thinking in students; some authors consider it appropriate to use the project method for laboratory work on real practical tasks [21] or team [31]. For a better understanding of the stages of concurrent programming, one of the authors [23] proposes to introduce a course on parallel programming in the form of a game. According to the author, this will improve the formation of students in “parallel thinking”. We have summarized the methods of teaching parallel and distributed computing into a graph (Figure 1).

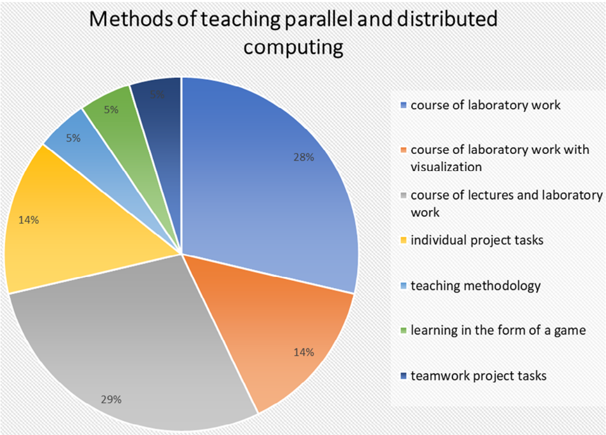


**Fig. 1.** Methods of teaching parallel and distributed computing.

The graph shows that most authors – 71%, use the method of laboratory work for practical training in parallel and distributed computing: one lab work reveals one topic or paradigm of parallel programming. Only 19% of authors consider it advisable to use the project method of teaching parallel programming and only one article about the use of the game method in teaching parallel and distributed computing.

Let us now consider the tools of parallel and distributed computing. First, consider the use of hardware. Based on the tools used by the authors of the articles, we divided them as follows: use of personal computer [6], cluster personal computers [17], personal computer with a set of video cards [26], single board computer [20], cluster of single board computers [10]. We reduce the use of learning hardware by parallel and distributed computation to the diagram (Figure 2).

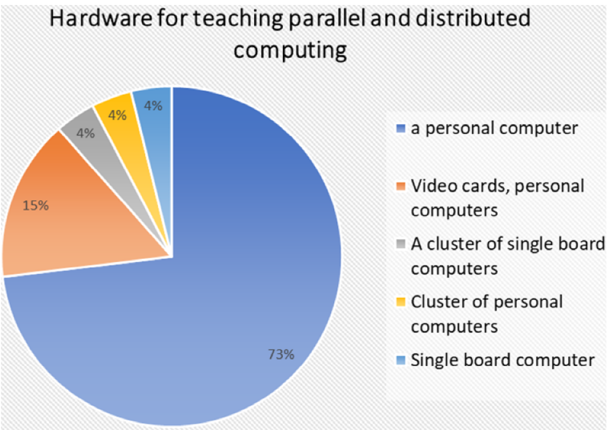


**Fig. 2.** Methods of teaching parallel and distributed computing.

The analysis of the chart shows that 88% of authors-teachers of higher education institutions use a personal computer to teach students parallel and distributed computing – that is, each student uses a personal computer (in computer class or own) to develop parallel and distributed programs. writing, debugging and running parallel and distributed programs. For 15% of authors, one or more discrete graphics cards must be present on a personal computer. Only 4% of authors use the latest technologies in the form of single-board computers for teaching parallel and distributed computing in their lab work. Laboratory work using clusters of single-board computers or personal computers is the same percentage.

Now let’s consider the software for teaching students learning parallel and distributed computing. When designing lab work to teach parallel and distributed computing, the authors use most of the programming languages or additional libraries in most. Therefore, instead of a pie chart, we used a column in which the B axis contains the number of references in programming language articles (Figure 3).

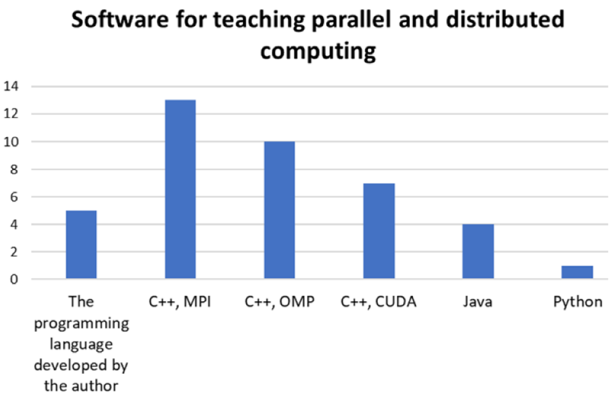


**Fig. 3.** Tools for teaching parallel and distributed computing.

Analyzing the diagram we can conclude that for laboratory work in teaching parallel and distributed computing, most authors use the C++ language together with the MPI library. The Messaging Interface (MPI) is a standardized messaging library. The messaging paradigm was developed for shared memory architecture. It offers various messaging technologies that provide shared memory. In messaging, processes are connected through a set of point-to-point primitives and collective communications. MPI is the actual standard for developing high-performance science programs [19]. MPI offers APIs for developing scalable C++ applications that are portable, efficient and flexible. The disadvantages of learning parallel and distributed computing using the MPI library are the difficulty of mastering the basic paradigms of MPI by students who previously worked only with sequential programming.

In second place by the number of mentions, the authors of the article are using the C++ programming language together with the OMP library. One of the most popular interfaces that supports multi-platform multi-core shared memory is OpenMP [10]. Using the simple semantics of this interface, a programmer can enable concurrent multicore computing in their applications. The OpenMP library uses runtime library routines, compiler directives, and environment variables in C++. The authors note that it is easier for students to learn about using this library than MPI libraries, but program development using the OMP library is limited to use only in concurrent programming whereas the MPI library can be used in concurrent, distributed, and hybrid programming.

Recently, new features in hardware for parallel and distributed computing, such as the use of video cards with a library of parallel and distributed computing CUDA. The structure of the CUDA program reflects the coexistence of the host (CPU) and one or more devices (GPUs) in the computer. Each CUDA output file may contain a mixture of both host and device code. By default, any traditional C program is a CUDA program that contains only the host code. You can add device features and data declarations to any source file C. Device features or data declarations are clearly identified by CUDA special keywords. These are usually features that exhibit a great deal of data concurrency [2]. The CUDA library is a good tool for teaching students how to develop parallel programs based on data concurrency – complex mathematical calculations for matrices. The disadvantages include the availability of expensive equipment – graphics cards and the need to combine with other technologies to develop a program for distributed computing.

Some teachers [27] use the Java programming language to simplify learning the basic principles and paradigms of parallel programming. As the author points out, Java's programming language advantages over C or Fortran include higher-level programming concepts, improved compile-time and runtime checking, resulting in faster problem detection and debugging. In addition, Java's automatic garbage collection during operation saves the programmer from many lower-level language pitfalls. Built-in thread support provides a way to insert concurrency into Java applications. The Java Development Kit (JDK) includes a large set of libraries that developers can use to quickly develop applications. Another interesting argument in favor of Java is its large pool of developers – the main reason is that Java is taught as one of the main languages in many universities around the world [27]. The disadvantages of learning parallel and distributed computing in the Java programming language can be attributed to the much slower speed of running parallel programs than when used to develop the C++ programming language.

One author explores the use of Python programming language to teach students parallel programming [19]. The author points out that Python is gaining popularity in academia as the best language for teaching beginners to consistent programming. Python syntax is clean, easy and easy to understand. At the same time, it is a high-level programming language that supports the paradigms of many programs, such as imperative, functional and object-oriented. Therefore, by default, it's almost obvious to think that Python is also the right language for teaching parallel programming paradigms. In conclusion, A. Marowka's study concludes: Python is still not mature enough to teach concurrent programming. It does not support adequate multithreaded programming, which is the main paradigm of parallel programming today. However, it has relevant modules that support the messaging paradigm and heterogeneous programming. Moreover, Python has drawbacks that make it an inappropriate language for training inexperienced programmers, such as the lack of visual debuggers and profiles [19].

It should be noted separately several experimental studies on the implementation in the parallel programming of integrated programming environments or programming languages developed by the authors of the studies themselves [12, 23, 28]. Y. Wepathana states that students should consider different concurrent architectures and programming models in their learning process. To achieve this, it is advisable to use an integrated system that has different concurrent architectures and supporting programming languages. The process of developing such environments or programming languages is still ongoing, and working with them, despite the student's learning success, has many limitations, as the authors of the research note.

Consider the authors' vision of the process of organizing learning by parallel and distributed computing. For this purpose, we will put the following questions in the articles: the number of subjects of the curriculum on which parallel and distributed calculations are taught or required to be taught; what course you need to study concurrent programming; what are the directions of parallel programming. Let us summarize these questions into a table.

**Table 2.** Font styles for a reference to a journal article.

| | Author, year of publication | Number of items | What course to study | Computer science only |
|---|---|---|---|---|
| 1 | Adams, 2013 | one | not specified | Yes |
| 2 | Anderson, 2010 | one | magistracy | Yes |
| 3 | Arroyo, 2013 | few | at all | Yes |
| 4 | Breuer, 2012 | one | first | Yes |
| 5 | Cesar, 2015 | one | magistracy | No |
| 6 | Delistavrou, 2011 | one | not specified | No |
| 7 | Eijkhout, 2018 | one | not specified | Yes |
| 8 | Franczak, | one | not specified | Yes |
| 9 | Gardner, 2017 | one | not specified | Yes |
| 10 | Gregg, 2012 | one | school | |
| 11 | Grossman,2017 | one | second | Yes |
| 12 | Kuhail, 2018 | one | not specified | Yes |
| 13 | Lin, 2013 | few | at all | Yes |
| 14 | Marowka, 2008 | one | third | Yes |
| 15 | Marowka, 2017 | one | not specified | Yes |
| 16 | Matthews, 2018 | one | not specified | Yes |
| 17 | Mosin, 2018 | one | not specified | Yes |
| 18 | Muresano, 2010 | one | not specified | Yes |
| 19 | Ontañón, 2017 | one | not specified | Yes |
| 20 | Paprzycki, 2006 | one | first | Yes |
| 21 | Prasad, 2018 | one | not specified | Yes |
| 22 | Shafi, 2014 | one | not specified | Yes |
| 23 | Wepathana, 2015 | one | not specified | Yes |
| 24 | Wilkinson, 2013 | one | not specified | Yes |
| 25 | Wilkinson, 2016 | few | first and second | Yes |
| 26 | Younis, 2019 | one | third | Yes |

Concerning the question, it is necessary to consider parallel and distributed calculations of more authors (26 out of 29), which should be enclosed in one subject. It is only the authors that need to develop "parallel thinking" in all streams of all undergraduate courses [3].

More real authors do not change in explicit age for any course, it is necessary to hang up the objects that exist with parallel and different exchanges. There are no others who share the same opinion: from the beginning of the course to graduate school. One of the authors provided an example of an experiment in basic parallel high school programming [11].

In the third question, we try to execute more authors (26 out of 29) by conducting experiments in their own fields of work.

## 3 Conclusions

At present, the main method of teaching parallel and distributed computing is a laboratory course, where each work studies a separate topic or paradigm of parallel programming, but to improve teaching, teachers try to implement a design method in the laboratory.

The main tool for teaching parallel and distributed computing is a lab course with a C programming language and MPI library. But many teachers find that the use of such tools leads to the loss of interest in parallel programming by students, so many authors publish the results of experiments to improve learning by means of developing their own environments or the organization of cluster computing systems.

## References

1. J. Adams, R. Brown, E. Shoop, Patterns and Exemplars: Compelling Strategies for Teaching Parallel and Distributed Computing to CS Undergraduates, in *2013 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum* (2013). doi:10.1109/IPDPSW.2013.275
2. N. Anderson, J. Mache, W. Watson, Learning CUDA: Lab Exercises and Experiences, in *OOPSLA '11* (2010), pp. 201–202. doi:10.1145/2048147.2048206
3. M. Arroyo, Teaching Parallel and Distributed Computing to Undergraduate Computer Science Students, in *2013 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum* (2013). doi:10.1109/IPDPSW.2013.276
4. S. A. Bogaerts, One step at a time: Parallelism in an introductory programming course. Journal of Parallel and Distributed Computing **105**, 4–17 (2017). doi:10.1016/j.jpdc.2016.12.024
5. R. Brown, Strategies for preparing computer science students for the multicore world, in *ITiCSE-WGR '10*. June 2010, pp. 97–115. doi:10.1145/1971681.1971689
6. A. Breuer, M. Bader, Teaching Parallel Programming Models on a Shallow-Water Code, in *2012 11th International Symposium on Parallel and Distributed Computing*. doi:10.1109/ISPDC.2012.48
7. E. Cesar, A. Cortés, A. Espinosa, T. Margalef, J. C. Moure, A. Sikora, R. Suppi, Teaching Parallel Programming in Interdisciplinary Studies, in *Euro-Par 2015: Parallel Processing Workshops*. LNCS 9523 (2015), pp. 66–77. doi:10.1007/978-3-319-27308-26
8. C. T. Delistavrou, K. G. Margaritis, Towards an Integrated Teaching Environment for Parallel Programming, in *2011 15th Panhellenic Conference on Informatics*. doi: 10.1109/PCI.2011.16
9. V. Eijkhout, Teaching distributed memory programming from mental models, in *2011 15th Panhellenic Conference on Informatics*, p. 107. doi:10.1109/PCI.2011.16
10. T. Franczak, A. Nkansah, T. Marrinan, M. E. Papka, A Path from Serial Execution to Hybrid Parallelization for Learning HPC, in *Proceedings of the 2017 Workshop on Education for High-Performance Computing*. doi:10.1145/1734263.1734339
11. W. B. Gardner, Should We Be Teaching Parallel Programming?, in *WCCCE '17*. doi:10.1145/3085585.3085588
12. C. Gregg, L. Tychonievich, J. Cohoon, K. Hazelwood, J. EcoSim: A language and experience teaching parallel programming in elementary school, in *SIGCSE '12*, February 2012, pp. 51–56. doi:10.1145/2157136.2157155
13. M. Grossman, M. Aziz, H. Chi, A. Tibrewal, S. Imam, V. Sarkar, Pedagogy and tools for teaching parallel computing at the sophomore undergraduate level. J. Parallel Distrib. Comput. **105**, 18–30 (2017). doi:10.1016/j.jpdc.2016.12.026
14. Y. Ko, B. Burgstaller, B. Scholz, Parallel from the beginning: The case for multicore programming in the computer science undergraduate curriculum, in *SIGCSE '13*, pp. 415-420. doi:10.1145/2445196.2445320
15. M. A. Kuhail, S. Cook, J. W. Neustrom, P. Rao, Teaching Parallel Programming with Active Learning. IPDPSW. (2018). doi:10.1109/IPDPSW.2018.00069
16. A. Liberati, D. G. Altman, J. Tetzlaff, C. Mulrow, The PRISMA statement for reporting systematic reviews and metanalyses of studies that evaluate health care interventions: explanation and elaboration. Journal of Clinical Epidemiology **62**, 1–34 (2009). doi:10.1016/j.jclinepi.2009.06.006
17. H. Lin, Teaching Parallel and Distributed Computing Using a Cluster Computing Portal, in *2013 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum*. doi:10.1109/IPDPSW.2013.35
18. A. Marowka, Think Parallel: Teaching Parallel Programming Today. IEEE Distributed Systems Online **9**(8) (2008). doi:10.1109/MDSO.2008.24
19. A. Marowka, On parallel software engineering education using Python. Education and Information Technologies. (2018). doi:10.1007/s10639-017-9607-0
20. S. J. Matthews, J. C. Adams, R. A. Brown, E. Shoop, Portable Parallel Computing with the

Raspberry Pi, in *SIGCSE '18*, February 2018, pp. 92–97. doi:10.1145/3159450.3159558

21. S. Mosin, N. Pleshchinskii, I. Pleshchinskii, D. Tumakov, Technique for Teaching Parallel Programming via Solving a Computational Electrodynamics Problems. RuSCDays 2018: Supercomputing (2018). doi:10.1016/j.jpdc.2018.02.023
22. R. Muresano, D. Rexachs, E. Luque, Learning parallel programming: a challenge for university students. Procedia Computer Science **1**, 875–883 (2010). doi:10.1016/j.procs.2010.04.096
23. S. Ontañón, B. Char, J. Zhu, E. Freed, Designing Visual Metaphors for an Educational Game for Parallel Programming, in *CHI EA '17*. doi:10.1145/3027063.3053253
24. K. Osadcha, O. Sysoieva, Condition, technologies and prospects of distance learning in the higher education of Ukraine. Information technologies and learning tools **70**, 271–284 (2019)
25. M. Paprzycki, Education: Integrating Parallel and Distributed Computing in Computer Science Curricula. IEEE distributed systems online **7**, 6 (2006). doi:10.1109/MDSO.2006.9
26. S. K. Prasad, A. Gupta, A. L. Rosenberg, A. Sussman, C. Weems, Topics in Parallel and Distributed Computing: Introducing Algorithms, Programming, and Performance within Undergraduate Curricula (2018). doi:10.1007/978-3-319-93109-8
27. A. Shafi, B. Carpenter, Teaching Parallel Programming Using Java, in *EduHPC '14*, November 2014, pp. 56–63. doi:10.1109/EduHPC.2014.7
28. Y.M.R.D. Wepathana, G. Anthonys, L.S.K. Udugama, Compiler for a simplified programming language aiming on Multi Core Students' Experimental Processor, in *ICIIS* (2015). doi:10.1109/ICIINFS.2015.7399025
29. B. Wilkinson, J. Villalobos, C. Ferner, Pattern Programming Approach for Teaching Parallel and Distributed Computing, in *SIGCSE '13*, March 2013, pp. 409–414. doi:10.1145/2445196.2445319
30. B. Wilkinson, C. Ferner, The Suzaku Pattern Programming Framework, in *IPDPS Workshops 2016*, pp. 978–986
31. A. A. Younis, R. Sunderraman, M. Metzler, A. G. Bourgeois, Case Study: Using Project Based Learning to Develop Parallel Programing and Soft Skills, in *IPDPSW* (2019). doi:10.1109/IPDPSW.2019.00059