A Systematic Review on Teaching Parallel Programming

José Aprígio C Neto
Federal Institute of Sergipe
jose.neto@ifs.edu.br

Antônio José A Neto
Federal University of Sergipe
antonio.neto@dcomp.ufs.br

Edward David Moreno
Federal University of Sergipe
edward@dcomp.ufs.br

ABSTRACT

This work aimed to perform a systematic review of the literature on teaching parallel programming using low-cost clusters, identifying the main programming languages, hardware platforms and software tools used in teaching-learning this type of programming. The research results showed that the most used clusters in the teaching of parallel programming were assembled from multicore machines (Cluster Beowulf) and by single board computers (SBC), in addition to multicore machines with graphics acceleration cards (GPUs). Regarding the use of programming languages, software tools and parallelism libraries used in teaching parallel programming, it is observed that most of the researched works mentioned the use of C, C++, and JAVA programming languages, and as parallelism libraries the use of MPI, OpenMP, CUDA and Apache Hadoop. Furthermore, the tests on the clusters were carried out through the implementation of parallelized generic algorithms and, in some cases, using algorithms that involve matrix operations.

CCS CONCEPTS

• **Applied computing**; • **Education**; • **Interactive learning environment**;

KEYWORDS

Parallel Programming, Clusters, Teaching, Education

ACM Reference Format:

José Aprígio C Neto, Antônio José A Neto, and Edward David Moreno. 2022. A Systematic Review on Teaching Parallel Programming. In *11 Euro American Conference on Telematics and Information Systems (EATIS2022), June 01–03, 2022, Aveiro, Portugal.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3544538.3544659

1 INTRODUCTION

Parallel architectures are computational structures constituted by multiple processors that cooperate with each other to solve complex problems that involve a high hardware processing power, exploring different levels of parallelism [1].

In this context, parallel programming emerges as an important tool in solving computational problems with a high degree of complexity, aiming to explore parallelism techniques, as well as the use of libraries that work with the explicit execution of processes or threads in a way simultaneous.

Parallel programming is a technique based on data parallelism and consists of the simultaneous execution of complex tasks that involve large volumes of data [3], dividing them into smaller tasks that will be distributed and executed by several processors simultaneously [4]. Parallel programming aims to exploit the potential concurrent existent in a program's code, using parallelism libraries that work with the explicit execution of processes and/or threads simultaneously [2].

For this, the equipment used in this type of programming must have a parallel architecture [2], where these processors and/or cores can communicate with each other so that there is a synchronization in the execution of tasks [4]. Considering that parallel applications involve great computational power and demand a high response time, parallel processing seeks to reduce this time by dividing the computational load between processors and/or cores, allowing an acceleration in the execution of tasks and consequently, a reduction in the response time of the tasks [4].

However, teaching parallel programming is a complex and challenging task, considering that it involves a series of programming steps and models, as well as solving computational problems and understanding various types of hardware platforms, in addition to skills and skills to work with this new type of technology.

Therefore, this article aims to make a systematic review of the literature on studies carried out in the context of teaching parallel programming using low-cost clusters, identifying the main methodologies, architectures and software tools used in the teaching-learning process of this type of programming.

2 RELATED WORKS

Mwasaga and Joy [8] carried out a systematic literature review to investigate articles published on the teaching of high-performance computing in the period from 1988 to 2018, showing the main artifacts used as interventions in the teaching-learning process of HPC (High-performance computing), considering that these artifacts facilitate the understanding of students in the study of parallel and distributed computing.

The results of the authors' research [8] identified 211 articles related to the topic addressed. In addition, the study revealed that most publications reported having used Beowulf clusters in their practices, as well as pedagogical tools that helped in the development of these activities. In the work, the authors also report that the study was of fundamental importance for the compression of which artifacts were most used in the teaching of high-performance computing (HPC).

3 METHODOLOGY

The systematic review consists of a secondary study method that uses primary studies related to a particular research topic as a data source, with the aim of answering a specific question in an objective and impartial manner. The methods used in the preparation of systematic reviews include research question elaboration, literature search, article selection, data extraction, methodological quality

EATIS2022, June 01–03, 2022, Aveiro, Portugal

ACM ISBN 978-1-4503-9738-4/22/06…$15.00
https://doi.org/10.1145/3544538.3544659

assessment, data synthesis, evaluation of the quality of evidence, and writing and publishing the results [5].

The systematic review carried out in this article focused on articles published in the scientific databases of Scopus and Web of Science from January 2012 to November 2021.

3.1 Research Questions (RQs)

To fulfill the objective of this article, the following main question that will guide the studies (MQ) was defined: "What are the teaching strategies and hardware and software technologies used in the implementation of a cluster for teaching parallel programming?

To answer the main question (MQ) of this research, the following questions were raised in parallel:

- RQ1: What programming languages were used in the development of parallel algorithms for executing and testing clusters?
- RQ2: What hardware platforms were used in implementing the clusters, as well as in practical activities?
- RQ3: What software tools were used in the construction of clusters, as well as in the development of practical activities?
- RQ4: What types of benchmarks were used in the cluster performance analysis tests?

The query expression used in this research was "((parallel AND programming) AND (educat* OR teach* OR learn*) AND (framework OR solution* OR tool* OR code*) AND (cluster*))".

To limit the number of references, in searches carried out in bibliographic databases, the query expression was used in the following search fields: title, abstract and keywords.

3.2 Inclusion and Exclusion Criteria

The criteria used for the inclusion of articles used in this systematic review were:

- IC1: Publications focused on teaching parallel programming using a cluster.
- IC2: Publications that could respond to RQs.
- IC3: Articles published between 2012 and 2021.
- IC4: Articles written in English or Portuguese.

Regarding the exclusion criteria of the articles identified in the searches of this systematic review, the following criteria were used:

- EC1: Gray Literature.
- EC2: Studies inconsistent with the research topic.
- EC3: Duplicate articles.
- EC4: Articles not available for analysis.

3.3 Selected Publications

According to the search criteria used in this research, 479 publications were initially identified, 288 belonging to the Web of Science database and 191 to the Scopus database. Of which 69 were excluded for being duplicates and 385 for being outside the scope of the investigations proposed in this work, leaving a total of 25 publications for further analysis, 16 in the Web of Science database and 09 in the Scopus database.

It is noteworthy that all publications identified and analyzed in this research met the inclusion and exclusion criteria established in the scope of work. In addition, most publications ranked for further analysis have enough information that they can identify strategies, hardware platforms and software tools. However, some publications selected as the object of study contain only partial information related to the research questions.

4 ANALYSIS AND RESULTS

In this section, the results obtained in the searches carried out in the respective databases specified in this work will be presented, analyzed, and discussed.

4.1 Programming Languages

The survey results identified the use of various programming languages in teaching parallel programming. Among the identified languages, three of them stand out regarding their use, they are: the C, C++, and JAVA language, as shown in Table 1.

Table 1: Languages used in developing parallel algorithms

Languages	References
C	[6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [17], [19], [23], [24], [26], [29], [31]
C++	[6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [17], [19], [29], [30], [31]
JAVA	[6], [18], [19], [26], [29], [31]
Fortran	[9], [17], [31]
GCC++	[10], [13], [19]
Python	[19], [31]
GCC	[10], [13]

The use of these programming languages is more frequent because they are traditional languages, which have simple command syntaxes that are easy for students to understand. Furthermore, several parallelism libraries support its use.

In the case of the JAVA language, it is observed that its use is because it is a language that presents a compilation and execution time suitable for use in parallel programming. In addition, it allows fast and accurate error detection and debugging, as well as support the use of parallelism through the threading API.

In addition to the programming languages mentioned in Table 1, other programming languages used in teaching parallel programming were identified in the researched works, however, all with low relevance in relation to the number of times they were mentioned by the authors surveyed. Therefore, these languages were not inserted in the table, they are SCALA [6], GO [6], RUST [6], GFortran [10], C# [22], PHP [26] and HTML [26]. These languages were mentioned only once.

It is noteworthy that, when it comes to teaching programming, it is essential that undergraduate students have at least knowledge of some types of programming languages, including: a systems language, an object-oriented language, and a scripting language.

4.2 Hardware Platforms

In the analysis of the researched articles, several types of hardware were identified in the cluster solutions used in teaching parallel

Table 2: Hardware platforms used in clusters

Hardware	References
Multicore Computers	[6], [7], [8], [9], [10], [11], [15], [18], [19], [20], [22], [24], [26], [27], [28], [29], [30]
GPUs	[7], [11], [14], [18], [19], [20], [22], [29]
Raspberry Pi	[6], [10], [12], [14], [16], [31]
ODROID	[10], [11], [12],[13]
Jetson	[10], [12], [16]
Laptop's Multicore	[6], [18], [25]
Servers (multicore)	[6], [10], [11]
Servers NFS	[11], [26]
Parallella	[12], [15]

programming, from simple low-cost hardware, such as single board computers (SBCs), to supercomputers, as shown in Table 2.

Among the different types of hardware identified in the survey, the highlight goes to multicore computers, used in Beowulf cluster solutions. Their greater use is because they are machines that are more easily found in the institutions' laboratories, with no need for new investments to acquire new equipment.

It is worth noting that in some studies the use of graphics accelerator cards (GPUs) and single card computers (SBCs), such as the Raspberry Pi, was also identified in cluster solutions.

Other hardware platforms appeared in the searches during the analysis of the search results, however, with little relevance in relation to the number of times they were cited by the authors. These hardware platforms were removed from Table 2, they are: LattePanda [10], VIM3Pro [10], Hikey970 [10], Servers NAS [10], Cubieboard [13] and Supercomputer [23]. These hardware platforms were mentioned only once.

4.3 Software Tools

The researchers identified several types of software tools used in the development of activities involving the teaching of parallel programming, among them the following stand out: OpenMP, MPI, CUDA, MPICH, POSIX and Apache Hadoop, as shown in Table 3.

Each of these software has its own characteristics, thus, its uses and implementations depend on the type of parallel architecture used, as well as the type of problem it is intended to solve.

The MPI (Message Passing Interface) is widely used in multiprocessing based on message-passing, where data is moved through message exchange operations.

OpenMP (Open Multi-Processing) is an API that provides parallel processing of algorithms between the cores of a processor with shared memory, which can be used in different types of architectures that have shared memory.

CUDA (Compute Unified Device Architecture) is an API that works with graphics cards (GPUs), designed to work with the programming languages C, C++, and Fortran.

Hadoop is an open-source tool developed in JAVA to work with clusters of common machines and with a high volume of data processing. Furthermore, it is a fault tolerant tool.

Table 3: Software platforms used in clusters

Software	References
OpenMP	[6], [7], [10], [11], [12], [13], [14], [15], [16], [18], [19], [21], [24], [29], [30], [31]
MPI	[6], [7], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [21], [23], [24], [26], [27], [28], [29], [30], [31]
CUDA	[6], [7], [10], [14], [18], [19], [20], [16], [29]
Apache Hadoop	[6], [18], [19], [17], [29], [31]
MPICH	[6], [9], [11], [13], [31]
POSIX	[6], [15], [17], [19], [31]
OpenCL	[6], [10], [13], [18]
Ubuntu	[10], [11], [26]

MPICH is a standard for passing messages involving distributed memory applications, used in parallel computing, and is available for the Linux operating system.

POSIX (Portable Operating System Interface) is an interface for programming APIs, which guarantees the portability of a program's source code from an operating system, also offering real-time services, interface with threads and communication between processes via network.

In addition to the software tools mentioned, other tools were also identified during the analysis of research results, however, as they were mentioned only once, they were not included in Table 3, they are: Scala [6], OpenACC, [6], Julia [6], Erlang [6], TBB [7], OpenSSH [9], Glibc [10], OmpSs [10], Cubian [13], Epiphany [15], Pilot [17], Intel Cilk Plus [18], JVM [18], MPL Express [18], JCUDA [18], SAUCE [19], NVCC [19], Task Parallel [22], Microsoft Visual Studio [22], Scholar [24], Rstudio Server [24], Thin line [24], Jupyterhub [24], OpenNebula [26], Windows 7 [27], VirtualBox [27] and NPACI Rocks Cluster [27].

4.4 Benchmarks Used in Cluster Testing

During the analysis of the results, several types of benchmark tests involving the use of parallel algorithms were identified, as shown in Table 4.

Table 4: Benchmarks and tests used in clusters

Benchmarks	References
Generic Parallel Algorithms	[6], [7], [9], [10], [11], [12], [13], [14], [15], [17], [18], [19], [16], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31]
Linpack	[9], [31]

All tests aimed to analyze the behavior and performance of the cluster solutions in question.

In addition to the generic parallel algorithms used for the tests on the clusters, the presence of a specific benchmark test was evidenced in some analyzed works, where the Linpack tool was used. This tool is used in ranking of the most powerful supercomputers in the world, the TOP500 list. Performance measured by Linpack equates to the number of floating-point operations a machine can perform per second (FLOPS).

Furthermore, to the mentioned tests, several other tests were identified during the analysis of the research results, however, they were removed from the table because they were mentioned only once, they are Load Tests [6], PARSEC [10], SPLASH-2 [10], Rodinia [10], SPEC OMP2012 [10], Thread Number Tests [11], Process Number Tests [11].

5 CONCLUSIONS

According to the research results, it is observed, in general, that the teachers of educational institutions have been using small clusters, formed by multicore machines or single board computers (SBCs), for practical activities involving the teaching of parallel programming, due to the fact that these equipment's have a low acquisition cost or because they are machines that are already available in the institution itself, in the laboratories, with no need to invest in the acquisition of new equipment for the development of these activities.

Regarding the libraries used in teaching parallel programming, there is a predominance of the use of libraries that involve passing through messages, as is the case with MPI. Message passing forms the basis of high-performance computing codes, making it a natural choice to begin studies of parallel programming. In addition to the MPI library, several other libraries supported practical activities in teaching parallel programming, including: OpenMP, CUDA, MPICH, POSIX and Apache Hadoop. It is worth noting that most of these libraries can be implemented with the help of programming languages C, C++, JAVA, and Python.

In the tests and performance analysis of the clusters implemented by the researched authors, it is verified the use of benchmarks executed from generic parallelized algorithms and operations involving the use of matrices, where students were able to evidence and understand in practice the scheduling of machines parallel, as well as the functioning of architectures that use shared memory.

As a future work, we intend to develop practical studies based on the construction of low-cost clusters, with the objective of encouraging the use of parallel and distributed programming in courses in the computing area of higher education institutions.

REFERENCES

[1] Schepke, Claúdio. 2009. Ambientes de Programação Paralela. Trabalho Individual I, Universidade Federal do Rio Grande do Sul/PPGC.
[2] Soares, Felipe. Augusto. Lara., Nobre, Cristiane. Neri., de Freitas, H.C. 2019. Parallel programming in computing undergraduate courses: a systematic mapping of the literature. IEEE Latin America Transactions, 17(08), 1371-1381.
[3] Sato, Liria. Matsumoto., Midorikawa, Edson. Toshimi., & Senger, Hermes. 1996. Introdução a programação paralela e distribuída. Anais do XV Jornada de Atualização em Informática, Recife, PE, 1-56.
[4] Rebonatto, M. T. 2004. Introdução à programação paralela com MPI em agregados de computadores (clusters). In Congresso Brasileiro de Ciência da Computação, Itajaí (pp. 938-955).
[5] Galvão, Taís. Freire., & Pereira, Mauricio. Gomes. 2014. Revisões sistemáticas da literatura: passos para sua elaboração. Epidemiologia e Serviços de Saúde, 23, 183-184.
[6] Galvão Adams, Joel. C. 2021. Evolving PDC curriculum and tools: A study in responding to technological change. Journal of Parallel and Distributed Computing, 157, 201-219.
[7] Bednárek, David., Kruliš, Martin., & Yaghob, Jakub. 2021. Letting future programmers experience performance-related tasks. Journal of Parallel and Distributed Computing, 155, 74-86.
[8] Mwasaga, Nkundwe. Moses., & Joy, Mike. 2020. Using high-performance computing artifacts as a learning intervention: a systematic literature review. In Proceedings of the 2nd International Conference on Intelligent and Innovative Computing Applications (pp. 1-10).
[9] Datti, Ahmad. A., Umar, Hadiza. A., & Galadanci, Jamil. 2015. A beowulf cluster for teaching and learning. Procedia Computer Science, 70, 62-68.
[10] Velásquez, R. A., Isaza, S., Montoya, E., García, L. G., & Gómez, J. 2020. Embedded cluster platform for a remote parallel programming lab. In 2020 IEEE Global Engineering Education Conference (EDUCON) (pp. 763-772). IEEE.
[11] Alvarez, Lluc., Ayguade, E. & Mantovani, F. 2018. Teaching hpc systems and parallel programming with small-scale clusters. In 2018 IEEE/ACM Workshop on Education for High-Performance Computing (EduHPC) (pp. 1-10).
[12] Adams, J. C., Matthews, S. J., Shoop, E., Toth, D., & Wolfer, J. 2017. Using inexpensive microclusters and accessible materials for cost-effective parallel and distributed computing education. Journal of Computational Science Education, 8(3), 2.
[13] Toth, David. 2014. A portable cluster for each student. In 2014 IEEE Intl. Parallel & Distributed Processing Symposium Workshops (pp. 1130-1134).
[14] Adams, Joel. C., Caswell, J., Matthews, S. J., Peck, C., Shoop, E., Toth, D., & Wolfer, J. 2016. The micro-cluster showcase: 7 inexpensive beowulf clusters for teaching pdc. In Proceedings of the 47th ACM Technical Symposium on Computing Science Education (pp. 82-83).
[15] Matthews, Suzanne. J. 2016. Teaching with parallella: A first look in an undergraduate parallel computing course. Journal of Computing Sciences in Colleges, 31(3), 18-27.
[16] Wolfer, James. 2015. A heterogeneous supercomputer model for high-performance parallel computing pedagogy. In 2015 IEEE Global Engineering Education Conference (EDUCON) (pp. 799-805). IEEE.
[17] Gardner, William. B., & Carter, John. D. 2014. Using the pilot library to teach message-passing programming. In 2014 Workshop on Education for High Performance Computing (pp. 1-8). IEEE.
[18] Shafi, A., Akhtar, A., Javed, A., & Carpenter, B. 2014. Teaching parallel programming using java. In 2014 Workshop on Education for High Performance Computing (pp. 56-63).
[19] Shafi, Schlarb, Moritz., Hundt, Christian., & Schmidt, Bertil. 2015. Sauce: A web-based automated assessment tool for teaching parallel programming. In European Conference on Parallel Processing (pp. 54-65). Springer, Cham.
[20] Ludin, M., Weeden, A., Houchins, J., Thompson, S., Peck, C., Babic, I., & Sergienko, E. 2013. LittleFe: The high-performance computing education appliance. In IEEE Intl. Conf. on Cluster Computing (CLUSTER) (pp. 1-1).
[21] Cuenca, Javier., & Giménez, Domingo. 2016. A parallel programming course based on an execution time-energy consumption optimization problem. In 2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW) (pp. 996-1003).
[22] Liu, Jie. 2016. 20 years of teaching parallel processing to computer science seniors. In 2016 Workshop on Education for High-Performance Computing (EduHPC) (pp. 7-13).
[23] Moore, Shirley. V., & Dunlop, Steven. R. 2016. A flipped classroom approach to teaching concurrency and parallelism. In 2016 IEEE Intl. Parallel and Distributed Processing Symposium Workshops (IPDPSW) (pp. 987-995).
[24] Baldwin, M. E., Zhu, X., Smith, P. M., Harrell, S. L., Skeel, R., & Maji, A. 2016. Scholar: A campus hpc resource to enable computational literacy. In 2016 Workshop on Education for High-Performance Computing (EduHPC) (pp. 25-31). IEEE.
[25] Capel, Manuel. I., Tomeu, Antonio. J., & Salguero, Alberto. G. 2017. Teaching concurrent and parallel programming by patterns: An interactive ICT approach. Journal of Parallel and Distributed Computing, 105, 42-52.
[26] Maia, Régis. Matheus. Silveira. 2016. Ferramenta para criação de clusters virtuais para o ensino de programação paralela e distribuída.
[27] Beserra, D., França, M., Melo, C., Sousa, Y., Romeiro, S., Andrade, M., & Sousa, E. 2013. Ambiente virtualizado para ensino de programaçao paralela e computaçao em cluster. In XXXIII Congresso da Sociedade Brasileira de Computação/XXI Workshop sobre Educação em Computação, Maceió.
[28] Shoop, E., Brown, R., Biggers, E., Kane, M., Lin, D., & Warner, M. 2012. Virtual clusters for parallel and distributed education. In Proc. of the 43rd ACM technical symposium on Computer Science Education (pp. 517-522).
[29] Shoop Saule, Erik. 2018. Experiences on teaching parallel and distributed computing for undergraduates. In 2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW) (pp. 361-368).
[30] Mangual, O., Teixeira, M., Lopez, R., & Nevarez, F. 2014. Hybrid MPI-OpenMP versus MPI Implementations: A Case Study. POLYTECHNIC UNIV OF PUERTO RICO HATO REY.
[31] Mwasaga, N. M., & Joy, M. 2020. Implementing micro High-Performance Computing (μHPC) artifact: Affordable HPC Facilities for Academia. In 2020 IEEE Frontiers in Education Conference (FIE) (pp. 1-9).