Using a TV Box for Teaching Parallel Programming and High-Performance Computing

Mikael dos Santos Gois[1], Ariel Andrade Cruz[1], Kauan Gutierrez Santos Estácio[1], Talyson Rodrigues Silva Nascimento[1], Eric Santos de Jesus[1], Isabelle Saahirah Ribeiro de Lima[1], José Aprígio Carneiro Neto[1]

[1]Coordenação de Ciência da Computação – Instituto Federal de Sergipe (IFS)
Caixa Postal 49.500-543 – Itabaiana – SE – Brazil

{mikaelgois1@gmail.com, arielccomp@gmail.com, kauangt2017@gmail.com,
talyson.rodigues@gmail.com, ericjesus403@gmail.com,
isabellesaahirah28@gmail.com, jose.neto@ifs.edu.br}

Abstract. *This article demonstrated that it is possible to overcome the challenges faced by institutions in teaching parallel and distributed computing, even when using smaller equipment with physical limitations in hardware resources. The methodology used was quantitative, descriptive, and exploratory, aiming to deepen the understanding of the topic. The cluster used consisted of eleven physically identical MxQ Pro 4K TV Boxes, and for the teaching-learning process of parallel and distributed programming, parallel computational algorithms from the "Tera" family were used. The results showed that parallelism is more efficient in solving complex problems, provided that communication between the nodes is fast and efficient.*

1. Introduction

The demand for processing power has motivated studies related to parallel and distributed computing. However, educational institutions face several challenges in this context, such as the lack of adequate infrastructure for carrying out practical activities and the availability of financial resources for the acquisition of high-performance equipment. In addition, equipment of this nature has high energy consumption, requiring adequate physical space and specialized maintenance (Shoop, 2025).

To overcome these problems, low-cost, high-performance architectures emerged as alternatives to meet all this demand, known as clusters. The construction of clusters using compact and energy-efficient devices became a viable solution, used by institutions for teaching parallel and distributed computing. Among these alternatives, TV Boxes (de-characterized) stand out as computational nodes, originally developed to support multimedia applications. These devices have multicore ARM architectures, integrated network connectivity, and low energy consumption, ideal for educational and even scientific purposes (De Toledo Gomes, 2023).

The reuse of TV Boxes involves software modifications, allowing their use in the development of clusters, enabling students to have practical contact with concepts of parallelism, communication between processes, load balancing, scalability, fault tolerance, and distributed processing (David, 2019).

Furthermore, studies show that the use of compact clusters in the development of activities related to high-performance computing favors student learning, increasing their understanding of the theoretical aspects of the content covered (Shoop, 2025).

However, teaching parallel and distributed computing involves a series of steps and programming models, as well as an understanding of different hardware and software platforms, and the technical skills to work with these new programming environments.

In this context, the use of TV Boxes as didactic infrastructure for teaching parallel and distributed computing aims to explore aspects related to technical feasibility, computational performance, cost-effectiveness, and pedagogical potential, considering that the de-characterization of these devices can represent a viable and sustainable alternative for the development of educational environments focused on high-performance computing.

2. Theoretical Framework

2.1. ARM Architecture

Studies show that processors based on the ARM (Advanced RISC Machine) architecture have excelled in the development of high-performance computing clusters and in computational benchmarking environments. This growth is associated with the evolution of ARM processors in terms of computational power, scalability, and support for parallel processing applications. ARM processors combine high computational capacity with low power consumption, making them viable alternatives for modern parallel and distributed computing infrastructures (Kalapothas et al., 2023).

Furthermore, they have aroused increasing interest in academia due to their favorable cost-benefit ratio, low acquisition and maintenance costs, as well as their scalability in parallel and distributed computing environments (Göddeke et al., 2013).

The increasing adoption of ARM architectures in high-performance computing environments is driven by advances in multicore processing, heterogeneous computing, and optimized parallel execution models. As a result, these architectures are emerging as a strategic technological solution for the development of scalable, low-cost, and energy-efficient parallel computing infrastructures (Sterling et al., 2022).

2.2. Parallel Architectures

Parallel architectures are computational structures designed to support applications that require high processing capacity, scalability, and efficient execution of complex tasks. These architectures are based on the simultaneous use of multiple processing units, which cooperate in the execution of distributed operations at different computational levels. By exploiting parallelism at the instruction, data, and task levels, parallel architectures reduce execution time and improve computational performance when compared to traditional sequential systems (Hennessy & Patterson, 2011).

The evolution of computing has intensified the need for parallel processing approaches, due to the exponential growth of data and the increasing complexity of applications. Parallel computing allows the decomposition of large computational problems into smaller tasks that can be executed simultaneously, optimizing resource

utilization and improving system performance. This capability has become essential in high-performance computing environments, where processing efficiency directly impacts scientific and industrial innovation (Kalapothas et al., 2023).

In modern computing environments, parallel architectures encompass multicore processors, Graphics Processing Units (GPUs), distributed clusters, and heterogeneous computing systems. These architectures are designed to maximize performance and improve energy efficiency, as well as system reliability (Sterling et al., 2022).

2.3. Cluster Architectures

Clusters consist of a set of independent computers, called nodes, interconnected by a communication network and configured to act as a single computing system, allowing multiple processing, memory, and storage resources to be combined to run applications in a parallel and distributed manner, providing greater performance, scalability, and availability compared to isolated systems (Pacheco, 2011).

In a cluster, each node has its own processor, memory, and operating system, characterizing a distributed memory architecture. Cooperation between nodes occurs through message exchange on the interconnection network, using parallelism libraries and communication standards. This model allows computational tasks to be divided and executed simultaneously by different nodes, reducing the total processing time and increasing the application's execution efficiency (Sterling et al., 2022).

These systems are loosely coupled, as each node operates independently, contributing cooperatively to the execution of parallel applications. Despite the nature of the architecture, clusters can be logically viewed as a single system, where middleware and resource management mechanisms are responsible for maintaining consistency, synchronization, and coordination of tasks between nodes. This flexibility allows clusters to support heterogeneous computing environments, composed of multicore processors, GPUs, ARM-based devices, and distributed storage infrastructures (Sterling et al., 2022).

2.4. Parallel Programming

Parallel programming is a technique that employs multiple interconnected computing environments to execute tasks simultaneously, aiming to increase scalability, processing performance, and resource utilization efficiency. In this approach, workloads are divided into smaller subtasks, which can be processed concurrently by different processors, cores, or distributed nodes connected through network interconnections. The effectiveness of parallel programming depends directly on efficient communication and synchronization mechanisms between processing units (McCool et al., 2012).

To enable coordination between distributed nodes, parallel programs rely on message-passing models, in which processes exchange data and synchronization information through communication primitives (send/receive) (Kalapothas et al., 2023).

Advances in parallel and distributed computing have expanded the applicability of programming to areas such as artificial intelligence, scientific simulations, cloud computing, big data processing, and edge computing. Studies indicate that the increasing availability of multicore processors, clusters, GPUs, and heterogeneous computing infrastructures has intensified the demand for efficient parallel programming

models capable of maximizing computational performance and minimizing communication overhead and energy consumption. Furthermore, modern parallel programming paradigms have incorporated optimization strategies related to load balancing, synchronization control, fault tolerance, and scalability in large-scale distributed systems (Sterling et al., 2022).

2.5. Parallel Libraries

Parallelism libraries play a fundamental role in distributed computing systems, providing communication and synchronization mechanisms between the nodes of a cluster. These libraries allow processes running on different machines to exchange data and coordinate their activities efficiently, enabling the parallel execution of applications. Among these libraries, Apache Hadoop stands out (Barney, 2010).

Apache Hadoop is an open-source framework developed in Java, designed for the distributed storage and processing of large volumes of data in clustered computing environments. Its architecture allows multiple nodes to work collaboratively, distributing both data and processing tasks across different machines. This approach provides high scalability, allowing organizations to expand their computing capacity according to the growth in data demands without the need for investments in expensive specialized equipment (Shvachko et al., 2010).

The main elements of Apache Hadoop are: MapReduce (programming model) and HDFS (distributed file system). The main advantages of this tool are: the fact that it is open source; saving implementation costs, as it does not require expenses with licenses and hiring specialized personnel; robustness, offering automatic recovery strategies in case of equipment failures; scalability, allowing the number of machines used in data processing to be increased when necessary, without the need for major changes in the code; and simplicity, as it frees the programmer from the responsibility of managing issues related to parallel computing, such as fault tolerance, scheduling, and load balancing (Goldman et al., 2012).

However, Apache Hadoop has some disadvantages: the fact that it is a constantly evolving tool, not supporting all situations to which it is subjected, due to some functionalities not being sufficiently mature; although it can use multiple machines as nodes for data processing and storage, it only has a single master node, creating a critical point susceptible to failures; difficulty in node management, fault debugging, and analysis of distributed logs; non-parallelizable problems or problems with high data dependency, which present difficulties in task division due to the impossibility of extracting the parallelizable part of the application; and the processing of small files (Goldman et al., 2012).

However, it should be noted that each parallelism library has specific characteristics related to the programming model, communication mechanisms, scalability, and performance.

3. Methodology

The methodology used was quantitative, descriptive, and exploratory in nature, aimed at deepening the practical understanding of the research topic, making its comprehension easier and more objective, as well as contributing to the teaching-learning process of those involved.

The implemented cluster consisted of 11 (eleven) physically identical TV Box MxQ Pro 4K, with the following characteristics: CPU - Rockchip Generic RK322x TV Box board @ 1.2GHz 4 cores, 4 threads; ARMv7 Processor rev 5 32 bits; GPU - ARM Mali-400 MP2; 1GByte RAM; 128 GByte external storage (SD memory card); 100 Mbps RJ-45 Ethernet network input; heat sinks on the internal boards; 4K Ultra HD video resolution; Dual Band Wi-Fi -– 2.4 GHz and 5 GHz; Bluetooth; HDMI port and USB ports. In the architecture, one of them functioned as the system's main node (master), and the other ten acted as slave nodes, all interconnected through cable (Cat. 6) to a TP-Link TL-SG1016D switch with 16 Gigabit Ethernet, Layer 2 (Figure 1).

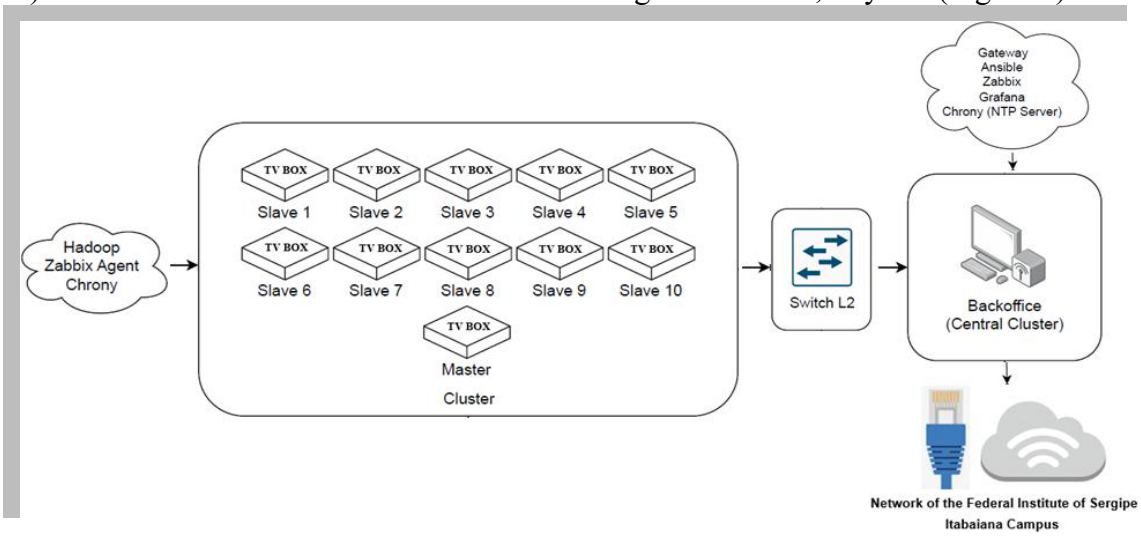


Figure 1. Cluster Architecture

For cluster configuration and execution of parallel benchmark computing algorithms, a desktop with the following physical characteristics was used: CPU - Intel(R) Core (TM)2 Duo CPU E7500 @ 2.93GHz 2 cores, 2 threads 64 bits; GPU - Intel Corporation 4 Series Chipset Integrated Graphics Controller; RAM memory - 2x SODIMM DDR3 2GByte 1333MHz 64 bits; and as an external storage source a Western Digital 500GByte SATA HDD.

The following software tools were used in the cluster configuration: Apache Hadoop 3.4.0, Java 11 (OpenJDK), Ansible 2.19.3, Chrony 4.0-8, Armbian_22.02.0-trunk_Rk322-box_bullseye_legacy_4.4.194_minimal, Debian GNU / Linux 11 (bullseye), Zabbix 6.0.42 (server, agent2, frontend), Ubuntu 24.04.3 LTS and Grafana (Enterprise v12.1.1).

The cluster performance tests involved running the following parallel computational algorithms: TeraGen (data generation), TeraSort (data sorting and organization), and TeraValidate (data organization validation), with file sizes of 1 GB, 2 GB, and 5 GB. To evaluate cluster performance, the following metrics were used: real-time, CPU time, mapping time, minification time, CPU utilization, RAM usage, temperature, and data sent and received over the network. The metrics were collected using Zabbix and Grafana (dashboards).

4. Results and Discussion

This section presents a performance analysis of the implemented cluster, in relation to the benchmark tests performed. The objective was to demonstrate that the developed cluster can be used as an alternative for teaching parallel and distributed computing, since performing tests of this nature on more robust machines and supercomputers becomes a financially and academically unfeasible activity, considering the costs involved in acquiring and maintaining this equipment. Furthermore, more expensive and operationally delicate machines require qualified training, something that would not be possible for students due to the high turnover in course subjects.

4.1. Parallelism Library Performance Analysis (Apache Hadoop)

Based on the test scenarios performed, involving the parallel computing algorithms TeraGen, TeraSort, and TeraValidate, with different file sizes, the following behavior of the cluster and the Apache Hadoop parallelism library was observed (Figure 2):

Teragen

Nodes	Size (MB)	Real Time (s)	CPU Time (ms)	Map Time (ms)	Reduce Time (ms)
10	1000	4m26.143	298490	272247	-
10	2000	5m47.638	539010	443258	-
10	5000	10m3.876	1243910	953972	-

TeraSort

Nodes	Size (MB)	Real Time (s)	CPU Time (ms)	Map Time (ms)	Reduce Time (ms)
10	1000	13m37.444	1852110	1604203	530004
10	2000	25m28.175	4076800	3639943	1227092
10	5000	70m4.501	11624490	13805762	3817469

TeraValidate

Nodes	Size (MB)	Real Time (s)	CPU Time (ms)	Map Time (ms)	Reduce Time (ms)
10	1000	6m33.705	149360	172809	51957
10	2000	7m2.427	258410	267252	38132
10	5000	12m58.068	614610	608499	50794

Figure 2. Performance metrics (Apache Hadoop library)

Real-time processing increased considerably as the size of the files to be processed grew, becoming more complex due to the increased granularity of the tasks (number of data replicas generated and distributed among the nodes). This scenario was most evident in tests involving the TeraSort algorithm, due to the algorithmic complexity involved and the intensive use of available hardware resources in the system;

The same scenario was observed with regard to CPU usage time, where larger files demanded greater use of CPU resources from the nodes involved in the system, due to a high distribution in the number of blocks (tasks) that needed to be sorted and ordered;

Regarding the average time for mapping tasks in the cluster (Map Time), it is observed that the more complex and larger the task to be executed, as in the case of the TeraSort algorithm, which has a characteristic algorithmic complexity, the greater the effort required to break them down, map them, and distribute them among the nodes of the cluster, and consequently, the more hardware resources of the system will be demanded;

The TeraGen algorithm, in turn, does not perform reduce tasks during data generation, only in the data sorting/ordering activities (TeraSort) and in the validation of this sorting/ordering (TeraValidate);

The “reduce” tasks in the Apache Hadoop library represent the final stage of data processing. The goal of “reduce” is to receive the filtered results from the “map” tasks, group them by key, and execute a logical aggregation sequence to produce a final result. The “reduce” execution flow involves three steps: Shuffle and Sort (shuffling and sorting the data); Input (receives the key and a list of values associated with it); and finally, Aggregation (executes logic defined by the developer to consolidate the final result of the task). In the test scenarios performed, an increase in the execution time of these tasks is observed as the file sizes increase.

4.2. Cluster Performance Analysis Metrics (Hardware)

Regarding the cluster's hardware metrics, during the execution of the test scenarios, the following behavior and performance of the cluster and the machines involved were observed (Figures 3):

Cluster hardware metrics results - 10 Nodes - Teragen - 5G

Nodes	Size (MB)	CPU Utilization (%)	RAM Usage (%)	Temperature (°C)	Network Sent	Network Received
Master	5000	28.6	66.2	61	56.1 kb/s	58.8 kb/s
Slave 1	5000	7.48	36.6	61	326 kb/s	26.8 Mb/s
Slave 2	5000	10.4	37.2	63	303 kb/s	24.7 Mb/s
Slave 3	5000	6.56	38.1	57	79.9 kb/s	6.30 Mb/s
Slave 4	5000	9.34	34.6	59	149 kb/s	12.4 Mb/s
Slave 5	5000	7.55	34.6	56	149 kb/s	12.6 Mb/s
Slave 6	5000	41.1	49.5	65	158 kb/s	12.6 Mb/s
Slave 7	5000	12.2	36.1	60	150 kb/s	12.4 Mb/s
Slave 8	5000	72.0	46.9	67.5	53.2 Mb/s	621 kb/s
Slave 9	5000	22.6	37.2	60	364 kb/s	30.9 Mb/s
Slave 10	5000	70.4	46.9	65.8	51.5 Mb/s	603 kb/s

Cluster hardware metrics results - 10 Nodes - Terasort - 5G

Nodes	Size (MB)	CPU Utilization (%)	RAM Usage (%)	Temperature (°C)	Network Sent	Network Received
Master	5000	30.9	66.8	63	49.2 kb/s	748 kb/s
Slave 1	5000	95.8	92.3	78	16.9 Mb/s	12.8 Mb/s
Slave 2	5000	94.2	90.3	75	34.5 kb/s	31.1 kb/s
Slave 3	5000	96.8	91.3	75	15.2 Mb/s	17.4 Mb/s
Slave 4	5000	99.7	91.3	75	6.74 Mb/s	97.5 Mb/s
Slave 5	5000	96.6	91.4	72	17.9 Mb/s	17.4 Mb/s
Slave 6	5000	97.7	90.9	77	23.2 Mb/s	15.3 Mb/s
Slave 7	5000	97.7	91.0	77	24.0 Mb/s	9.75 Mb/s
Slave 8	5000	98.3	90.0	75	44.4 Mb/s	6.20 Mb/s
Slave 9	5000	96.8	92.0	76	21.4 Mb/s	14.1 Mb/s
Slave 10	5000	30.6	49.2	59	27.2 Mb/s	167 kb/s

Cluster hardware metrics results - 10 Nodes - Teravalidate - 5G

Nodes	Size (MB)	CPU Utilization (%)	RAM Usage (%)	Temperature (°C)	Network Sent	Network Received
Master	5000	28.7	64.8	60	52.8 kb/s	54.8 kb/s
Slave 1	5000	1.05	35.5	55	19.4 kb/s	15.5 kb/s
Slave 2	5000	31.2	49.6	61	29.4 kb/s	28.8 kb/s
Slave 3	5000	1.21	33.8	52	19.1 kb/s	15.2 kb/s
Slave 4	5000	7.42	34.1	55	79.5 Mb/s	154 kb/s
Slave 5	5000	1.00	35.8	50	44.3 kb/s	28.5 kb/s
Slave 6	5000	0.94	36.7	52	19.1 kb/s	15.6 kb/s
Slave 7	5000	1.11	34.2	53	32.4 kb/s	16.1 kb/s
Slave 8	5000	13.2	40.0	57	19.1 kb/s	15.6 kb/s
Slave 9	5000	1.24	34.3	53	31.8 kb/s	27.9 kb/s
Slave 10	5000	31.3	55.2	58	160 kb/s	79.0 Mb/s

Figure 3. Hardware performance metrics (“Tera” Algorithms)

The Apache Hadoop library, in the version used (3.4.0), did not demonstrate efficiency in balancing the load of hardware resources among the system's nodes. In all test scenarios, a significant imbalance in the use of these resources was identified, with some nodes being heavily demanded (overloaded) and others having practically underutilized resources, highlighting a critical problem with the library. This occurs because the library randomly chooses the nodes that will initially participate in processing the activities. Natively, it was designed to prioritize the security (fault tolerance) and stability of the cluster, and not the speed of data distribution and rearrangement;

As the cluster was scaled with respect to the number of nodes, a greater distribution of tasks was observed and, consequently, a reduction in the consumption of hardware resources among the nodes involved; however, as mentioned, this occurs in a scenario of imbalance of these resources, mainly CPU and RAM;

The most intensive use of system hardware resources was observed in tests involving the TeraSort algorithm, due to the degree of complexity of the tasks performed by it (classification and sorting of data);

As file sizes increase and become more complex, the consumption of hardware resources becomes more evident (CPU, RAM, temperature and network traffic);

During testing, the library showed difficulty in breaking files into smaller blocks, increasing task mapping time and cluster hardware resource consumption, demonstrating that the parallelism library works better with larger and more complex files;

The increase in the cluster nodes' temperature was proportional to their usage during the tests; that is, the more they were used (CPU and RAM), the higher their temperature became.

5. Conclusion

Over the years, several changes have occurred in the computer science education process. These changes have happened at both the hardware and software levels. Regarding hardware, there has been the emergence of new computer architectures, with machines composed of multiple processors or cores, as well as the development of small, powerful, inexpensive, and energy-efficient computers, consisting of a single integrated board, which have allowed the execution of different types of academic projects in the area of high-performance computing without the need for large investments by educational institutions for their acquisition and maintenance. As for software, high-performance computing has come to depend on various parallelism libraries for the construction of these new architectural environments.

Thus, this research demonstrated that the best way to learn to program, in any programming language, is through constant practice, and that the implementation of the cluster contributed significantly to this teaching-learning process of parallel and distributed programming.

However, learning parallel and distributed programming is not an easy task, mainly because many educational institutions do not have dedicated clusters where each student can practice developing their parallel algorithms. Therefore, this work

demonstrates that it is possible to overcome these challenges, even if it requires the use of smaller equipment with physical limitations regarding its hardware resources.

Regarding the results of the tests involving the parallel algorithms used, it was possible to demonstrate that parallelism is more efficient in solving complex problems and that communication between the nodes of the cluster, regardless of the processor speed and the amount of RAM they possess, depends on the use of a fast and efficient network technology, aiming to avoid potential bottlenecks in data transmission and reception.

It is worth noting that the cluster implemented in this project is available to the academic community of the Federal Institute of Sergipe for conducting new studies and research involving high-performance computing

For future work, it is recommended to install, configure, and test new parallelism libraries, as well as to run different parallel computing algorithms and datasets involving Big Data.

Acknowledgments

We would like to thank the Foundation for Support of Research and Technological Innovation of the State of Sergipe – FAPITEC/SE, the State Secretariat for Economic Development and Science and Technology – SEDETEC/SE, and the Federal Institute of Sergipe – IFS, for their financial support in the development of this research project.

References

Barney, B. (2010). Introduction to parallel computing. *Lawrence Livermore National Laboratory*, *6*(13), 10.

David, N., Almeida, Y., Neto, J. S., & de Almeida, D. (2019). Cluster Beowulf de baixo custo para o estudo de Sistemas Distribuídos. In *Escola Regional de Informática de Mato Grosso (ERI-MT)* (pp. 136-138). SBC. DOI: 10.5753/eri-mt.2019.8612

De Toledo Gomes, J. P., Silva, P. H. B., de Mello Innocentini, M. D., & Formigoni, C. E. (2023). Análise de Desempenho de um Cluster Beowulf com Utilização de TV Box. *17º Jornada Científica e Tecnológica e 14º Simpósio de Pós-Graduação do IFSULDEMINAS*, *15*(3).

Göddeke, D., Komatitsch, D., Geveler, M., Ribbrock, D., Rajovic, N., Puzovic, N., & Ramirez, A. (2013). Energy efficiency vs. performance of the numerical solution of PDEs: An application study on a low-power ARM-based cluster. *Journal of Computational Physics*, *237*, 132-150. DOI: 10.1016/j.jcp.2012.11.031

Goldman, A., Kon, F., Junior, F. P., Polato, I., & de Fátima Pereira, R. (2012). Apache Hadoop: conceitos teóricos e práticos, evolução e novas possibilidades. *XXXI Jornadas de atualizaçoes em informatica*, 88-136.

Hennessy, J. L., & Patterson, D. A. (2011). *Computer architecture: a quantitative approach*. Elsevier.

Kalapothas, S., Galetakis, M., Flamis, G., Plessas, F., & Kitsos, P. (2023). A survey on risc-v-based machine learning ecosystem. *Information*, *14*(2), 64. DOI: 10.3390/info14020064

McCool, M., Reinders, J., & Robison, A. (2012). *Structured parallel programming: patterns for efficient computation*. Elsevier.

Pacheco, P. (2011). *An introduction to parallel programming*. Elsevier.

Shoop, E., Matthews, S. J., Brown, R., & Adams, J. C. (2025). Hands-on parallel & distributed computing with Raspberry Pi devices and clusters. *Journal of parallel and distributed computing*, *196*, 104996. DOI: 10.1016/j.jpdc.2024.104996

Shvachko, Konstantin; Kuang, Hairong; Radia, Sanjay; Chansler, Robert. The Hadoop Distributed File System. In: *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*. Lake Tahoe: IEEE, 2010. p. 1–10.

Sterling, T., Brodowicz, M., & Anderson, M. (2017). *High performance computing: modern systems and practices*. Morgan Kaufmann.