Teaching Distributed Systems Using Hadoop

48

Ronaldo C.M. Correia, Gabriel Spadon, Danilo M. Eler, Celso Olivete Jr., and Rogério E. Garcia

Abstract

Databases and Distributed Systems have a fundamental relevance in Computer Science; they are usually presented in courses where the high-level of abstraction characterizes the teaching and learning processes. Consequently, the teaching method needs to evolve to fulfill the present requirements. Therefore, grounded in these concepts, the main goal of this paper is to introduce a teaching methodology via benchmark tests. Our methodology was conducted using the Hadoop framework, and it is innovative and proved effective. Our methods allow students to be exposed to complex data, system architecture, network infrastructure, trending technologies and algorithms. During the courses, students analyzed the performance of some computational architectures through benchmark tests on local and on the cloud. Along with this scenario, they evaluate the processing time of each architecture. As a result, our methodology proved to be a support learning method, which allows students to have contact with trending tools.

Keywords

Learning methodology • Distributed systems • Hadoop • Student-centered learning

48.1 Introduction

The Big Data concept emerged from data scalability problems. Such term describes methods and paradigms of data generation, collection, and analysis in a methodological framework. Which is responsible for dealing with high scalability, reliability and fault tolerance on data [1]. Big Data is still challenged by network infrastructure and data collection. Its frameworks are supported by tools as Hadoop *MapReduce* and Hadoop *Distributed File System* (HDFS), that appears either in commercial solutions and in researches [2]. The MapReduce provides support to the analytical processing of the data while the HDFS performs a distributed file-block storage [3].

An examination of the factors that affect the distribution of the data is warranted. In this sense, Khalid, Afzal, and Aftab [4] and Souza et al. [5] formalized some of these arguments, but none of them focused on computational models as a student-centered teaching methodology. As a consequence, this paper introduces a teaching and learning methodology that focuses on assisting students in the understanding of Databases, Distributed Systems, and other related topics. Our approach has the aim of clarifying theoretical concepts. It was conducted using the Hadoop framework as a learning tool; and, though it, students are exposed to complex data, system architectures, network infrastructure, trending technologies and algorithms.

Our results are based on explaining the structure of our course as well as the description of our methods. In order to validate such methods, we show a case study, where one of our students performed a series of tests with the purpose of

R.C.M. Correia • G. Spadon (✉) • D.M. Eler • C. Olivete Jr.
R.E. Garcia
Sao Paulo State University, Sao Paulo, Brazil
e-mail: ronaldo@fct.unesp.br; spadon@fct.unesp.br; daniloeler@fct.unesp.br; olivete@fct.unesp.br; rogerio@fct.unesp.br

S. Latifi (ed.), *Information Technology – New Generations*, Advances in Intelligent Systems and Computing 558, DOI 10.1007/978-3-319-54978-1_48

building a cluster. We address this methodology to understand how much the incomprehension of basic concepts can negatively influence students that are exposed to trending technologies and its abstractions.

Accordingly, this paper is organized as follows: Sect. 48.2 presents some related works; Sect. 48.3 presents the course architecture, a brief description of relevant server layouts, and an overview of Hadoop; Sect. 48.4 presents our methodology and a case study that shows how it behaves in a real scenario; at last, conclusions come in Sect. 48.5.

48.2 Related Works

From the usage of Hadoop as a learning tool, we can derive two approaches to engaging it in the learning process. The first one describes the search for meaning to the data, in other words, data mining processes. The other one focus on the architecture that describes the hardware where Big Data tools are employed; through it, students are encouraged to search for better configurations in the framework, as well as to improve its architecture while seeking for better performance. Given these approaches, our assumption is that these two can turn Hadoop into a learning tool.

The literature introduces former works, which focused on teaching-learning methodologies for different types computer science courses. This is the case of Souza et al. [6] which introduces a novel methodology for courses of Formal Languages and Automata Theory, while Souza et al. [7] introduced a continuous methodology that covers all the Theoretical Computing courses. Also, there is Correia et al. [8] which focused on new approaches for teaching data structures and algorithms for undergraduate students. However, when comes to methods for introducing trending technologies like Hadoop, there is a lack of a clear and precise methodology.

Consequently, the aimed result of our method is to teach how to introduce the process of evaluating the performance of a system, by guiding students in choosing the most suitable model for each particular cluster-planning case. In this sense, the first challenge is to transform Hadoop into a learning tool, which will enable the students' formation as better and up-to-date professionals.

48.3 Course Architecture

The relevance of Big Data led us to suit our course in a teaching model that can introduce trending topics to students. As it is known, there are a variety of computational models, from centralized to distributed. Many of them can be used to deploy the Hadoop tool. Consequently, it is important to review the literature and understand how each one contrasts the others. Thus, we divided our course into topics of: *Scale-up architecture*; *Scale-out architecture*; *Client–Server model*; *Client–Server with Master-Slave model*; and *Hadoop server-roles*. In the following, we present a summarized view of our course.

48.3.1 Course Topics

Coulouris et al. [9] defined a distributed system as the one where computers are independent, but they work for a common result; such computers are attached to a network where each one communicates with each other, coordinating actions by exchanging messages. Contrariwise, centralized computing is not divided at all, and every resource is located on a single server, which can be a server or a client to others. The centralized model does not face any network problems like latency and delayed packages because they do not need to be connected to a network to complete a task. Given these concepts, models, and architectures, Big Data can be generalized on *scale-up* and *scale-out*, which are stated in the following according to Henrique and Kaster [10].

48.3.1.1 Scale-Up Architecture

This architecture describes the server that centralizes hardware, which increases the processing performance, memory capability, and storage management. Specifically, scale-up servers make use of high throughput, avoiding latency and network delays. Such architecture implies in a limit where a single server cannot centralize and manage all resources. The closer to its limit higher is the monetary cost, and at this point, the distribution is the answer to improve the performance [4].

48.3.1.2 Scale-Out Architecture

The scale-out architecture focuses on model expansion, performing a distributed processing and loading balance. In this architecture, each computer manages their memory, clock, and storage. However, they work for a single purpose, and they can share their resources to solve common problems. That is because of the distribution, which makes the processing capacity higher. Most of the Big Data frameworks, such as Hadoop, are deployed through this architecture. The meaning is the fault tolerance factor is increased by the high availability of the data. Before introducing Hadoop and its roles, we introduce to students two models derived from the scale-out one. The first one is the Client-Server and the other one Client-Server with Master-Slave. These concepts enable the understanding of how works the interaction of the computers within a given architecture.

Client–Server Model

A server is a software on looping that is running on a computer. Such software is waiting to share their resources. A client, on the other hand, is an intermittent process that needs to be intentionally started on a host computer. This architecture is depicted by Fig. 48.1.

Client–Server with Master-Slave Model

As well as the previous model, the idea of servers and clients are the same. The difference between them is how the processes communicate with each other and how they are distributed. This model is unidirectional, and it usually has more slaves than masters. The master is the only process that receives requests from clients; slaves share information neither to external clients nor to each other. The slave works as a server process, but it just replies the master's requests, and the master works as a client and as a server at the same time. As a client, the master makes job requests to the slaves, and as a server, the master receives external requests and coordinate all slaves. This architecture is depicted by Fig. 48.2.

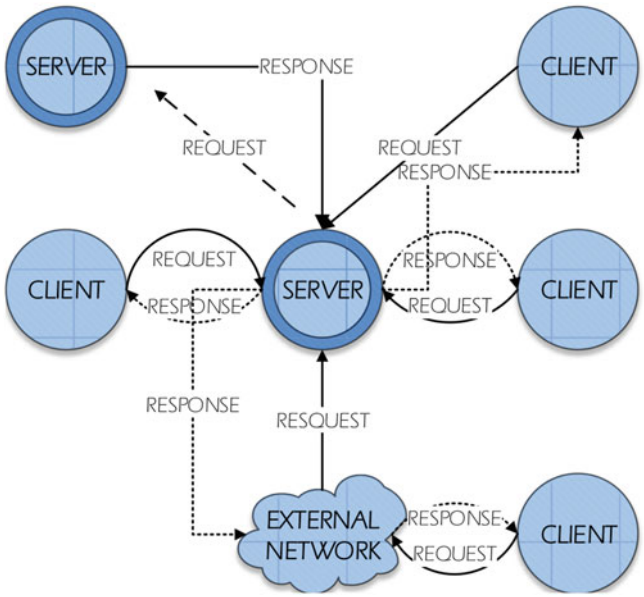


Fig. 48.1 Representation of the Client-Server model

48.3.2 Hadoop Framework

Hadoop is a framework that focuses on processing a large amount of data. It was designed to solve problems of data scalability and complex analysis, which depends on a stable model for significant processing tasks that cannot be entirely performed on database systems [10]. When Hadoop is deployed on a grid or cluster, its information has more than one copy stored on different servers, increasing the data availability. In this scenario, when a server is down, the first server that has the requested information reply to the master's request. Its architecture was planned to be independent and to solve their integrity and scalability problems. The idea behind it focuses on the problem solution and not on the environment planning.

Each computer on a cluster/grid hosts a standalone framework that is managed by the master, and each one controls a distinct group of assigned roles. That is, the master hosts the NameNode and the JobTracker/MapReduce, as well as the slaves host the DataNode/HDFS and the TaskTracker/MapReduce. Hadoop can work with more then one DataNode and TaskTracker on different slaves, as well as SecondaryNameNode, but just one NameNode and JobTracker on the master server. These roles are briefly described in the following:

(A) *JobTracker* – divide and distribute problems;
(B) *TaskTracker* – solves small problems;
(C) *DataNode* – stores the file blocks;
(D) *Secondary NameNode* – perform assistance;

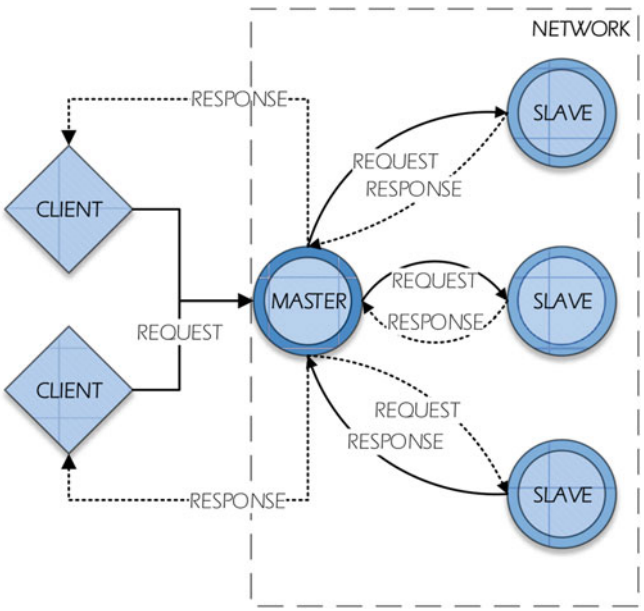


Fig. 48.2 Representation of the Client-Server with Master-Slave model

(E) *NameNode* – manage connections and requests, stores the file block list that contains the paths for each stored file.

Following, we introduce our methodology, presenting the most important points that should be considered by teachers in order to obtain good results.

48.4 Proposed Methodology

Our methodology consists of allowing the students to acquire the practical knowledge from the theoretical one. Such practical knowledge is defined as an approach to identify a better deployment model to Hadoop via benchmark. The benchmark has the aim to enhance the system performance, and on our methodology, we used it as a way to demonstrate to students the difference between computational architectures. The motivation of our work is the trending topics, which are not entirely explored by the students. Consequently, with this approach, we can present a single topic with new teaching strategies, contributing to the robustness of knowledge.

There are several conditions to be considered before choosing the cluster architecture. To this end, there is the need to evaluate the performance of each possible scenario before making a choice. As a consequence, we need to cause the same load and stress in each architecture and perform metrical analysis though benchmarks.

As a first step, we instruct our students to dismantle all the available computers and to build groups of computers with similar hardware. In the face of more than one similar group, they evaluated each one singularly. Subsequently, for each group, we instructed them to perform a stress test, seeking for information about overheat and overhead of the processor. To this end, we used *TeraSort* method. Such method allows us to take advantage of all cores of the processors in the task of sorting a terabyte file of numbers.

The students were encouraged to remove from the cluster, machines with less than two terabytes of storage space or without gigabyte Ethernet driver. This step is particular to each case, but it is an important one; without it, a cluster could have an inferior performance when the computers try to balance its hardware to match an inferior computer.

When the stress test is complete, it is important to retest different distributions. The students used Open Source, Enterprise, and Professional ones. At this point, it is important to have a group discussion with the students to analyze each group of machines. Once they have selected the ones that will be in the cluster, they worked to guarantee the lowest CPU time, which are the ones that may achieve a better platform for data analysis. In this sense, they tested different computing architectures among scale-up, scale-out and geographically distributed server.

The others distributions were tested the same as the first, using a benchmark. Once we already tested processing time, core capability and the hard drives, the students checked the processing time considering the network impact. For that, they made network attacks to the cluster computers using Distributed Denial of Service – DDoS, making some of them fail in the middle of the test. With the computers failing, the Hadoop should reach the next computer, with the requested information, and that will influence in the processing time by causing the abortion of the job.

At the end of the test, there should be sufficient information to make a choice about the machines to be used, the distribution of Hadoop and its computing architecture. To validate the obtained results, the collected information should be compared to other clusters of similar hardware.

To illustrate the usage of our methodology and its steps, we demonstrate at next it appliance and the decisions made by a student that with our guidance was able to build a cluster.

48.4.1 Case Study

To perform this case study, it was used 30 tower computers. The hardware that was available were not uniform, and this was the first problem faced. To overcome it, the computers were disassembled in order to separate the common hardware and to allocate it to new machines. As a result, it was achieved different processing architectures, half AMD, and the other half Intel. By using a stress test, more specifically the *TeraSort*, it was decided to use the Intel architecture. That was because the Intel hardware does not overheat as easily as the AMD one.

Subsequently, it was proceeded with investigations about the distribution of Cloudera,[1] Hortonworks[2] and MapR[3] of the Hadoop. In this sense, it was required to differentiate each one from each other by testing them. Such test was made by using Pi Estimator. This method requires two input, the number of mappings and another one for the samples. For each hundred looping, it was increased the exponential factor until it reaches five hundred and twelve mappings and one billion of samples.

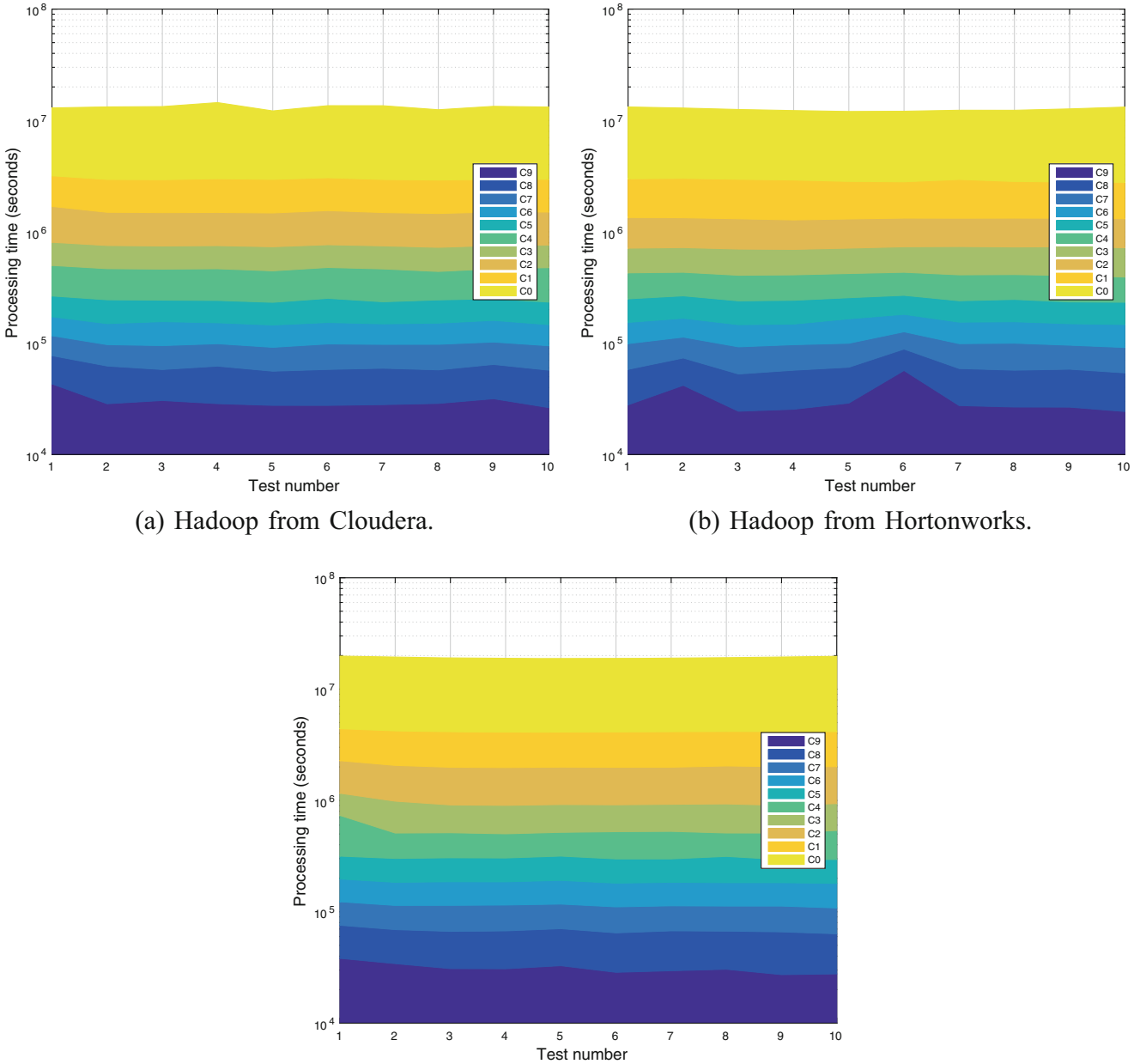
The test was estimated to be completed in 36 hours for each distribution. For each one, the output was not considered. This is because the tests focused just on the processing time. Also, time is crucial to fit performance to the cluster.

As so, the first test was on the Hortonworks; the test took 34 hours and did not present any error. The second test was on the MapR. The test took 53 hours and returned one error message, omitted almost 24 results, kept on looping on two experiments and aborted the Pi Estimator at the last step of the test. The last test was on Cloudera where the test took 36 hours and did not present any error. The benchmark results were divided by collection cycle, and they are depicted in Fig. 48.3a, b, c.

[1] www.cloudera.com

[2] www.hortonworks.com

[3] www.mapr.com



(a) Hadoop from Cloudera.

(b) Hadoop from Hortonworks.

(c) Hadoop from MapR.

Fig. 48.3 Benchmark of the Hadoop distributions. (**a**) Hadoop from Cloudera. (**b**) Hadoop from Hortonworks. (**c**) Hadoop from MapR

For a complete analysis, it was performed the same test in the *Apache Hadoop*, which results are detailed in the Table 48.1. Analyzing the results, it was concluded that among the different platforms of data management, the usage of *Apache Hadoop* was faster than others. That is because it took less than 30 hours to complete the tasks without a single error. In a superficial analysis, it is possible to list some possibilities of why this happened: the network, task starvation, deadlock, and others.

Lastly, to quantify the cloud server performance on benchmark tests, it was deployed a few servers on a VPS to compare them together with a geographically distributed one. It is important to note that most of the cloud servers are centralized on different datacenters. Consequently, computers might be faster and the results not straightly comparable.

However, for each cloud benchmark it was plotted the time spent on each Pi Estimator cycle on dispersion graphs, which are depicted on Fig. 48.4a, b, c. The cycles of each analysis showed that the distributed and the centralized model, obtained similar values, alternating between which is faster. Further, the geographically distributed model

Table 48.1 Apache benchmark results (values in seconds)

Apache Hadoop benchmark results			
Cycle	Mean (μ)	Standard deviation	Variance (σ^2)
T0	51.60	1.60	1.70
T1	51.70	1.20	0.90
T2	49.60	0.90	0.80
T3	48.60	1.30	0.30
T4	50.40	1.20	0.00
T5	56.50	1.20	0.00
T6	73.20	2.20	0.60
T7	85.40	0.50	0.20
T8	110.30	2.10	0.08
T9	1098.70	6.30	0.70
Distributed cloud server results			
Cycle	Mean (μ)	Standard deviation (σ)	Variance (σ^2)
T0	16.86	1.64	2.71
T1	17.57	1.24	1.55
T2	17.58	0.90	0.81
T3	17.94	1.31	1.72
T4	20.88	1.29	1.67
T5	26.82	1.27	1.63
T6	41.97	2.24	5.05
T7	293.73	5.48	30.00
T8	568.81	9.91	98.28
T9	6316.61	97.29	9465.80
Centralized cloud server results			
Cycle	Mean (μ)	Standard deviation (σ)	Variance (σ^2)
T0	14.49	0.47	0.22
T1	15.46	0.30	0.09
T2	16.46	0.48	0.23
T3	16.47	1.60	2.57
T4	26.50	0.35	0.12
T5	46.54	2.51	6.30
T6	79.68	1.77	3.15
T7	109.20	2.74	7.49
T8	203.97	3.10	9.58
T9	1766.34	16.59	275.25
Geographically distributed cloud server results			
Cycle	Mean (μ)	Standard deviation (σ)	Variance (σ^2)
T0	39.23	4.93	24.32
T1	291.50	78.44	6152.84
T2	125.43	47.62	2267.97
T3	986.29	434.80	189055.37
T4	601.16	243.69	59389.66
T5	2260.01	790.81	625395.16
T6	4390.56	1070.37	1145706.50

provided worst times in all cycles. The centralized results show an almost continuous processing time in eighty seconds. The distributed one shows an interesting time difference in all cycles. Taking into account this time variation, the highest processing time of it is almost half of the one from the centralized model. Finally, the geographically distributed model was able to be tested up to cycle number six (T6). On the other cycles, the exit rate prevented the execution of the test.

Considering the described process, at the end of tests, it was collected sufficient data to suppose about investment in local or cloud server. The described case points to the fact that the investment could be reverted to a highly scalable cloud server that could deploy more slaves as needed, not demanding investment in local infrastructure.

48.5 Conclusions

In this paper, we presented a methodology based on benchmark analysis to guide the learning on courses of Distributed Systems, Data Bases and Computers Network for undergraduate students, by acquiring knowledge of the most suitable architecture for Hadoop.

We theorize the process of choosing a computational model, which implements Hadoop as teaching approach, where we seek for the knowledge construction to allow students to make the concept concrete. According to our proposal, for each step of benchmark analysis, the student needs to define time metrics across a reliable test method, and they need to collect and analyze statistically all results to be able to suppose about the best deployment model for each particular case.

Besides the benchmark result, the student can interact directly with the system and the hardware, and this is always important to understand a real system and how it works. The method helps to connect the student with different distributions and architecture models of an application, directly contributing to a decision about local and cloud investment.

This approach can open possibilities to make questions about what is the best platform, the best system or software architecture, encouraging new proposals, like our previous works [5, 11]. We developed our methodology while performing two similar studies in different applications, applying it in all research in development at our research group. Our evaluation consisted in confronting the results of each common application in various scaling methods at local and in the cloud. We decided to perform it as a learning methodology because it has a strong influence on past researches made by us and by others in our group.

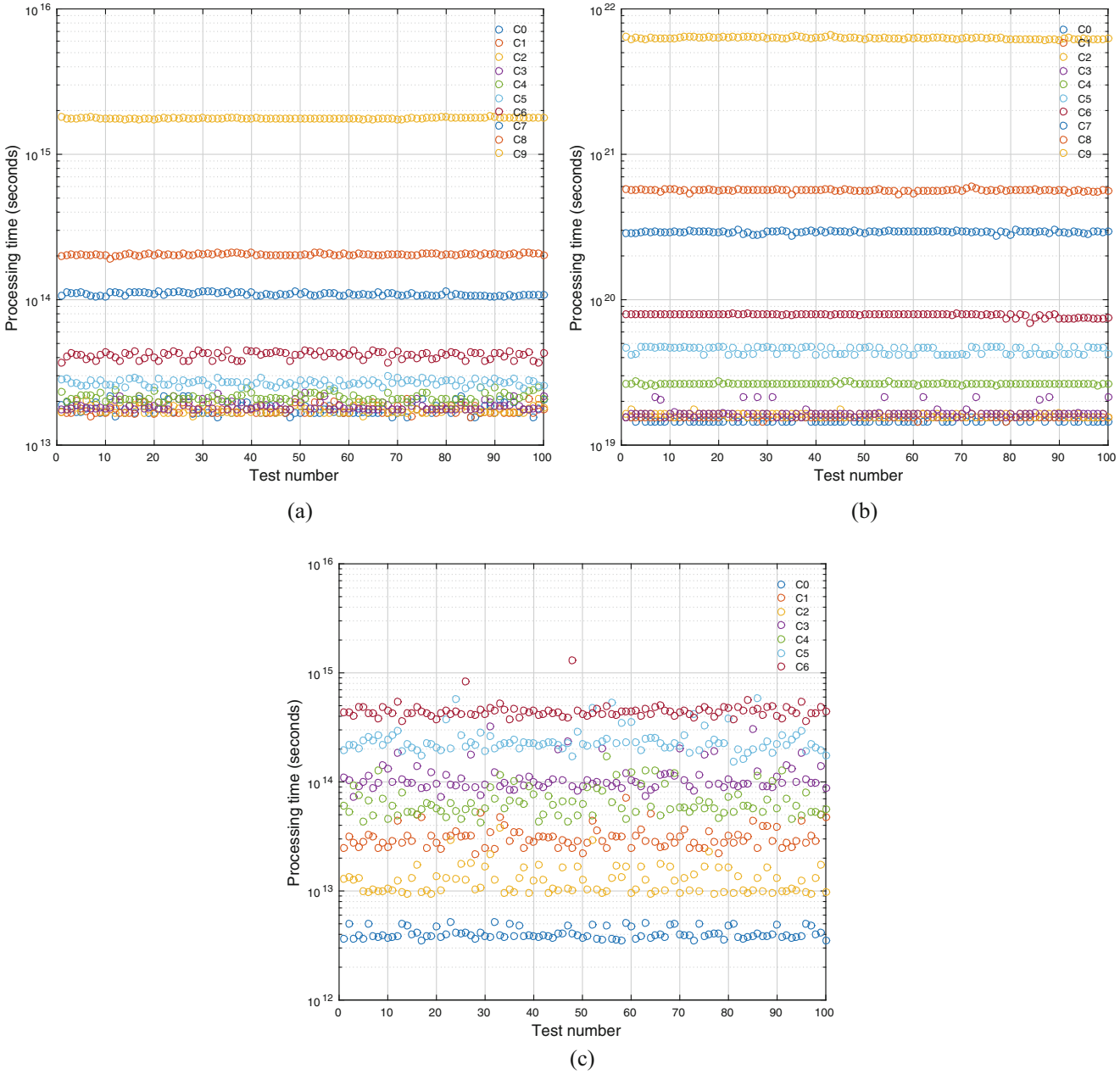


Fig. 48.4 Benchmark of the computational architectures. (**a**) Distributed Hadoop. (**b**) Centralized Hadoop. (**c**) Geographically distributed Hadoop

Acknowledgements We are grateful to the *Department of Mathematics and Computer Science* (DMC) at *Faculty of Science and Technology* (FCT) in *Sao Paulo State University* (UNESP) by the resources granted to this research.

References

1. Sagiroglu, S., & Sinanc, D. (2013). Big data: A review. In *2013 International Conference on Collaboration Technologies and Systems (CTS)* (pp. 42–47). IEEE.
2. Müller, M. U., Rosenbach, M., & Schulz, T. (2013). Living by the numbers. Big data knows what your future holds. In *SPIEGEL ONLINE*, vol. 17.
3. Shvachko, K., Kuang, H., Radia, S., & Chansler, R. (2010). The Hadoop distributed file system. In *2010 I.E. 26th Symposium on Mass Storage Systems and Technologies (MSST)* (pp. 1–10). IEEE.
4. Appuswamy, R., Gkantsidis, C., Narayanan, D., Hodson, O., & Rowstron, A. (2013). Scale-up vs scale-out for Hadoop: Time to rethink? In *Proceedings of the 4th Annual Symposium on Cloud Computing* (p. 20). ACM.
5. Souza, G. S., Messias Correia, R. C., Garcia, R. E., & Olivete, C. (2015). Simulation and analysis applied on virtualization to build

Hadoop clusters. In *2015 10th Iberian Conference on Information Systems and Technologies (CISTI)* (pp. 1–7). IEEE.
6. Souza, G. S., Olivete, C., Correia, R. C. M., & Garcia, R. E. (2016). Combined methodology for theoretical computing. In *Frontiers in Education Conference (FIE)* (pp. 1–7). IEEE.
7. Souza, G. S., Olivete, C., Correia, R. C. M., & Garcia, R. E. (2015). Teaching-learning methodology for formal languages and automata theory. In *Frontiers in Education Conference (FIE)* (pp. 1–7). IEEE.
8. Messias Correia, R. C., Garcia, R. E., Olivete, C., Costacurta Brandi, A., & Cardim, G. P. (2014). A methodological approach to use technological support on teaching and learning data structures. In *2014 I.E. Frontiers in Education Conference (FIE)* (pp. 1–8). IEEE.
9. Coulouris, G. F., Dollimore, J., & Kindberg, T. (2005). *Distributed systems: Concepts and design*. Boston: Pearson Education.
10. Henrique, G. J., & Kaster, D. d. S. (2013). Consultas por similaridade em big data: Alternativas e soluções. Faculdade Estadual de Londrina, Technical Report.
11. Santana, V. J., Spadon de Souza, G., Messias Correia, R. C., Garcia, R. E., Medeiros Eler, D., & Olivete, C. (2015) Scalable information system using event oriented programming and NoSQL. In *2015 10th Iberian Conference on Information Systems and Technologies (CISTI)* (pp. 1–6). IEEE.