Open access • Journal Article • DOI:10.1002/SMR.2120

Testing MapReduce programs: A systematic mapping study — Source link

Jesus Moran, Claudio de la Riva, Javier Tuya

Institutions: University of Oviedo

Published on: 01 Mar 2019 - Journal of Software: Evolution and Process (Wiley)

This paper is a pre-print paper accepted in Journal of Software: Evolution and Process.
The final version is available through the online library:https://onlinelibrary.wiley.com/doi/full/10.1002/smr.2120
Citation information: J.Morán, C. de la Riva, and J. Tuya, Testing MapReduce Programs: A Systematic Mapping Study, J Softw Evol Proc, 2018;VV:P–P e2120. https://doi.org/10.1002/smr.2120.

PERSPECTIVE ARTICLE

Testing MapReduce Programs: A Systematic Mapping Study

Jesús Morán* | **Claudio de la Riva** | **Javier Tuya**

Department of computing, Univeristy of Oviedo, Asturias, Spain

Correspondence
*Jesús Morán, Department of computing, University of Oviedo, Gijón, Spain. Email: moranjesus@uniovi.es

Present Address
Campus de Viesques, 33394, Gijón, Spain

Summary

MapReduce is a processing model used in Big Data to facilitate the analysis of large data under a distributed architecture with scale and fault tolerance mechanisms. These programs are considered critical for several enterprises causing high revenues. In order to guarantee their quality, researchers have proposed several software testing techniques and tools. This paper characterizes their state-of-the-art identifying the trends and gaps through a mapping study. The research literature of this topic is analyzed and synthetized systematically, finding that the main testing efforts are carried out by the tester in order to test the performance and, to a lesser degree, the program functionality. The principal reasons for testing the programs are performance issues, potential failures, issues related to the data, or to satisfy the agreements with efficient resources. The performance testing is carried out through simulation and evaluation, whereas the functional testing considers some program characteristics (such as specification and structure). Despite the fact that functionality is relevant to satisfy the business requirements, few studies are focused on functional testing and can indicate a potential research challenge. In addition, there is room to improve the software testing research in the MapReduce applications through more mature and standard validation methods.

KEYWORDS:
Software testing, Systematic mapping study, MapReduce, Big Data Engineering

1 | INTRODUCTION

Big Data or Data-intensive programs are those that cannot run using the traditional technology/techniques[1] and usually need novel approaches. *MapReduce* is one of the most important processing models used in *Big Data* based on the "divide and conquer" principle[2]. These programs run two functions in a distributed infrastructure, the *Map* function splits one problem into several subproblems (divide) and the *Reduce* function solves each subproblem (conquer). There are several technologies to execute and manage *MapReduce* programs such as *Spark*[3], *Flink*[4] and *Hadoop*[5], all broadly implemented in industry[6]. It is necessary to ensure the quality of these programs, especially those employed in critical sectors like health or security, such as DNA alignment with *MapReduce*[7] and image processing in ballistics with *MapReduce*[8]. These new approaches to process large data in general, and *MapReduce* in particular, have several characteristics that could have an impact on program quality, for example: (1) analysis of large quantities of data, (2) variety of the input information, (3) data without an apparent data model (schema-less), (4) program optimizations to obtain better performance, (5) implementation of the data models in each program (schema-on-read), (6) execution over heterogeneous infrastructure, and (7) automatic mechanisms to manage the resources (for example, scaling and fault tolerance).

There are several approaches to improve the quality, and software testing is one of the most used. According to the ISO/IEC/IEEE 29119-1:2013 standard[9] software testing aims to provide information of the program quality and the potential impacts/risks of poor quality. Software testing research has evolved in recent years[10], but there are several challenges to test programs in cloud and adaptive architectures[11].

The adoption and interest in these technologies/paradigms has increased over the last few years to the extent that several Fortune 1000 enterprises consider *Big Data* critical for business[12]. Despite the importance of these applications, some studies forecast that 60% of *Big Data* projects

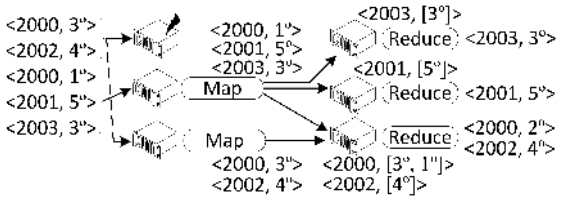

FIGURE 1 Example of the MapReduce program that calculates the average temperature per year.

fail to go beyond piloting and will be abandoned during 2017[13]. There are several challenges and concerns: poor data quality[14,15], lack of technological skills[16,17], and among others, different technological issues such as complexity[18], maturity[19], operability[20] and technical problems[14]. Some of the previously stated problems complicate the development and the *MapReduce* application could be implemented with faults. Although software testing is one of the quality assurance techniques most used to evaluate software products, there are not many studies related to *MapReduce* applications. The contribution of this paper is an evaluation and characterization about the state-of-the-art of software testing in the *MapReduce* applications through a *systematic mapping study*[21,22,23]. In this type of studies, research questions are proposed and then answered based on research studies.

A *mapping study* by Sharma et al.[24] on *Big Data* and *Hadoop*[5] indicates that the number of papers has increased significantly in recent years. This interest in *Big Data* during the previous years could have evolved the state-of-the-art about software testing in the *MapReduce* programs. Another *mapping study* was elaborated in 2013 by Camargo et al.[25] on software testing in *MapReduce* programs. Their study analyses only 14 papers and the results are focused on what types of faults the *MapReduce* programs have, how to perform the tests, the tools and the testing techniques used. In contrast to the aforementioned *mapping study*, this paper obtains more thorough results because of its deeper scope and different approach/motivation. The main differences between this *mapping study* and that of Camargo et al.[25] are: (1) broader research questions to analyze the software testing field in a more holistic way than ad-hoc or specific research questions, (2) broader and more general results obtained through the research questions, (3) relevant literature obtained through a large search involving more sources, (4) almost quadruple the number of papers analyzed in depth to improve the results, (5) deeper synthesis analysis of the papers based on several international standards in order to obtain accurate results, and (6) inclusion of the recent research lines.

The paper continues as follows. Section 2 introduces *MapReduce* and describes the main challenges from the testing point of view. The research questions are proposed in Section 3 together with the systematic steps planned to answer them. The answers and other results are detailed in Section 4. Finally, Section 5 contains the conclusions.

2 | MAPREDUCE PROCESSING MODEL

The *MapReduce* programs[2] divide one problem into several subproblems that are executed in parallel over a large number of computers. There are two principal functions: (1) *Map* that analyses part of the input data and classifies them in subproblems, and (2) *Reduce* that solves each subproblem. The data processed by these functions are in the form of <key, value> pairs in which the key is the identifier of each subproblem and the value contains the information needed to solve it. To illustrate *MapReduce*, let us imagine a program that calculates the average temperature per year. This problem could be divided into one subproblem per year, then the key (identifier of subproblem) is the year, and the value (information of the subproblem) is the temperature. Figure 1 details a distributed execution of the program analyzing the years 2000-2003. Firstly, two computers analyze the data, but one computer fails and this analysis is re-executed in a third computer. The *Map* function receives years with temperatures and creates the <key, value> pairs in order to group the temperatures per year. Then the *Reduce* function receives from all *Maps* one year with all of its temperatures, and calculates the average.

The programs are executed by a framework that automatically manages the resource allocation, the re-execution of one part of the program in case of infrastructure failures, and the scheduling of all executions between other mechanisms. The data analyzed could be stored in several distributed sources, such as for example non-relational databases and distributed file systems.

The integration of all of these technologies in the *MapReduce* program stack could be a challenge for developers and testers. Some technologies do not scale well, do not support indexing, or do not support ACID transactions, among others. Another challenge is the implementation of the data model in the program. *MapReduce* can analyze raw data without a data model (schema-less or unstructured) because the modelling of the data is codified in the program (schema-on-read). In the large data scale it is difficult to establish a model for all data and there are several issues related with poor data quality, such as for instance missing data, noise or incorrect data. Another problem is that new raw data are continuously generated and the data model could change over time, and then the program needs some changes.

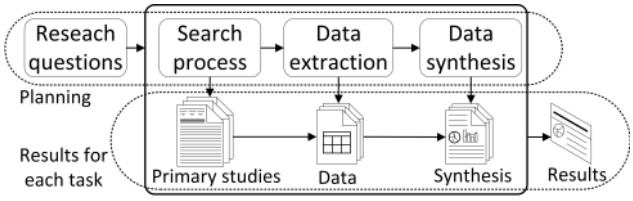

FIGURE 2 Steps of Systematic Mapping Study.

The balance and the statistical properties of the data can also change over time and they can affect the program, especially if there are performance optimizations in the code based on data property assumptions. For example, suppose that in the program that analyzes the average temperature per year, the last two years contain 80% of the data. In this case there could be at least two issues: (1) performance problems if these two years are analyzed in the same computer, and (2) memory leaks or resource issues due to the high quantity of data analyzed by one computer. A further challenge is the type of processing implemented, originally *MapReduce* analyzed the data only in batches, but nowadays there are streaming or iterative approaches, among others. For example, the temperature sensors create streams of data, then the calculation of the average temperature is more efficient using a streaming approach, but it is more difficult to implement and not all programs could be processed in this way. In some domains it is better to change the <key, value> approach to another that allows to model the program better, such as for example Pangool[26] that uses tuples, or more complex structures like graphs[27].

In the main framework of *MapReduce*, *Hadoop*, there are a lot of configuration parameters that could affect the execution in terms of resources, data replications and so on. More than 25 of these parameters are significant in terms of performance[28]. The developer does not know the resources available when the program is deployed because the cluster continuously changes (new resources adding to scale or infrastructure failures[29]), and this also makes the optimal configuration difficult. There are other advanced functionalities of *MapReduce* that could optimize the program, such as for example the *Combine* function. The problem is that if these functionalities are not well established there could be some side effects, such as incorrect output.

Also in *Big Data* there are other testing issues related to the ethical use of data. Different security procedures and policies should be considered in the *MapReduce* programs throughout the data lifecycle. For example, the analysis of some data could be forbidden in the next season due to agreements with the data provider or legal issues. In other cases, the data should be anonymized or encrypted, especially the sensitive data.

Several generic tools are used in the industry to test the *MapReduce* programs, such as JUnit[30] with mocks. In order to facilitate the testing of the *MapReduce* programs, MRUnit[31] runs the unit test cases without a cluster infrastructure. Another approach is MiniCluster[32] that simulates a cluster infrastructure in memory, or Herriot[33] that interacts with real infrastructure allowing more grained control, for example by the injection of computer failures that alter the execution of the program. There are different types of infrastructure failures that affect the test execution and several tools simplify their injection such as AnarchyApe[34], ChaosMonkey[35] or Hadoop Injection Framework[36]. The remainder of the paper analyses and summarizes the efforts of the research studies that are focused on covering the issues related to testing the *MapReduce* applications.

3 | PLANNING OF THE MAPPING STUDY

This *mapping study* aims to characterize the knowledge of software testing approaches in the *MapReduce* programs through an empirical study of the research literature. To avoid bias, the planning of the *mapping study* describes several tasks based on Kitchenham et al. guidelines[22]:

1. Formulation of the research questions (Subsection 3.1).
2. The search process to extract the significant literature (primary studies) to answer the research questions (Subsection 3.2).
3. Data extraction to obtain the relevant data from the literature (Subsection 3.3).
4. Data synthesis to summarize, mix and put the data into context to answer the questions (Subsection 3.4).

These tasks are planned and then conducted independently as described in Figure 2. The confidence of the results obtained from the planning of the *mapping study* is discussed in Subsection 3.5.

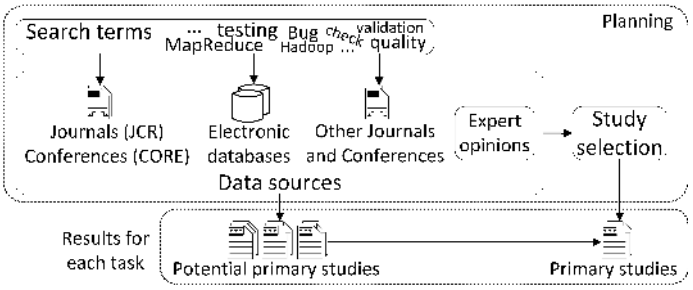

FIGURE 3 Search process to obtain the primary studies in the mapping study.

3.1 | Research Questions

The research questions are formulated to cover all the information about software testing research in the context of the *MapReduce* programs with different points of view. This work formulates the research questions based on the 5W+1H model[37,38], also known as the Kipling method[39]. This method is used in other software engineering empirical studies[40,41] and answers the questions: Why, What, How, Where, When and Who. The research questions of this *mapping study* are:

RQ1. **Why** is testing performed in the *MapReduce* programs?

RQ2. **What** testing is performed in the *MapReduce* programs?

RQ3. **How** is testing performed in the *MapReduce* programs?

RQ4. **Who**, **where** and **when** is testing performed in the *MapReduce* programs?

3.2 | Search Process

The *mapping study* answers the research questions based on a series of studies that contain relevant information about these questions. These studies are called primary studies and are obtained through the tasks described in Figure 3. First, the search terms (set of several words/terms) related to software testing and *MapReduce* are searched for in different data sources (journals, conferences and electronic databases). The papers that match these searches together with other studies recommended by experts constitute the potential primary studies. Finally, these studies are filtered in the study selection in order to obtain only the studies that contain information to answer the research questions. In the following subsections each of the planning steps is described in detail.

3.2.1 | Search Terms

The search terms are obtained from the three points of view proposed by Kitchenham et al.[22]: (1) population that refers to the technologies and areas related to *MapReduce*, (2) intervention that are the issues related to software testing, and (3) outcomes that are the improvements obtained through software testing.

The search terms of this *mapping study* follow the chain "*MapReduce* technology related terms AND Quality related terms" where:

The *MapReduce* technology related terms correspond with population and are enumerated in Table 1 with synonyms. The *Big Data* paradigm and the *MapReduce* processing model are surrounded by a lot of buzzwords like other fields such as Cloud computing. For example, *Hadoop* is a distributed system that supports the execution of MapReduce programs and *non-MapReduce* programs, but there are several papers that use *Hadoop* and *MapReduce* words interchangeably in the title. Other relevant papers do not include in the title the *MapReduce* word, but contain other words related to the *MapReduce*/*Big Data* ecosystem like Hive, PIG or Spark, among others. In order to obtain the maximum relevant literature and avoid missing some primary studies due to the buzzwords and jargon, a thorough search is performed considering the *MapReduce* and *Big Data* related technologies enumerated in Table 1.

The quality related terms correspond with the Quality (sub)characteristics of ISO/IEC 25010:2008-2011[42] and ISO/IEC 9126-1:2001[43] with synonyms (outcome), together with other testing terms (intervention). Both are enumerated in Table 2.

This work performs a wide search with 9384 combinations of terms in the paper title, obtained by 92 *MapReduce* technology related terms and 102 quality related terms.

TABLE 1 MapReduce technology related terms (population).

Technology	Terms and years of creation
Field	Big Data, Massive data, Large data
Data processing	Hadoop (2006)
-Batch	MapReduce (2004)
-Iterative	Spark (2013), Tez (2013), Stratosphere (2010), Dryad (2007), Flink (2014)
-Streaming	Storm (2011), S4 (2010), Samza (2013)
-Lambda	Lambdoop (2013), Summingbird (2013)
-BSP	Giraph (2013), Hama (2011)
-Interactive	Drill (2012), Impala (2012)
-MPI	Hamster (2011)
Testing	MRUnit (2009), Junit (1998), Mock, MiniMRCluster (2006), MiniYarnMRCluster (2012), Mini cluster (2007), QuerySurge (2011)
Security	Sentry (2013), Kerberos (2007), Knox (2013), Argus (2014)
Resource Manager	Yarn (2012), Corona (2012), Mesos (2009)
MapReduce abstraction	Pig (2008), Hive (2010), Jaql (2008), Pangool (2012), Cascading (2010), Crunch (2011), Mahout (2010), Data fu (2010)
Yarn frameworks	Twill (2013), Reef (2013), Spring (2013)
Yarn integration	Slider (2014), Hoya (2013)
Data integration	Flume (2010), Sqoop (2009), Scribe (2007), Chukwa (2009), Hiho (2010)
Workflow	Oozie (2010), Hamake (2010), Azkaban (2012), Luigi (2012)
Coordinator	Zookeeper (2008), Doozerd (2011), Serf (2013), Etcd (2013)
SDK	Hue (2010), HDInsight (2012), Hdt (2012)
Serialization	Sequence File (2006), Avro (2009), Thrift (2007), Protobuf (2008)
Cluster Management	Ambari (2011), StackIQ (2011), Whte elephant (2012), Ganglia (2007), Cloudera manager (2011), Hprof (2007), MRBench (2008), HiBench (2010), GridMix (2007), PUMA (2012), SWIM (2011)
Filesystem	HDFS (2006), S3 (2006), Kafka (2011), GFS (2003), GPFS (2006), CFS (2013)
Other storage	HBase (2008), Parquet (2013), Accumulo (2008), Hcatalog (2011)
Cluster deployment	Big top (2011), Buildoop (2014), Whirr (2010)
Data Lifecycle	Falcon (2013)

3.2.2 | Data Sources

The potential primary studies may be in different data sources. This *mapping study* searches for the studies in the following data sources grouped in four categories:

a) High impact journals and conferences. The potential studies are obtained through DBLP[44] with the search terms in 31 JCR journals[45] and 53 CORE conferences[46] enumerated in Appendix A. The journals and conferences selected are related to the software testing or *Big Data*. This category contains 624 proceedings/volumes from the year of the *MapReduce* paper (2004) to June 2016.

b) Electronic databases. The search terms are queried in IEEE Xplore[47], ACM Digital Library[48], Scopus[49], Ei Compendex[50] and ISI Web of Science[51], that are employed in other *mapping studies* of software testing[52].

c) Other journals and conferences. The non-JCR journals and non-CORE conferences related to software testing or *Big Data* could be a good source of potential primary studies. This *mapping study* searches for studies through DBLP[44] with the search terms in the 33 journals and 49 conferences enumerated in Appendix B. This category contains 1687 proceedings/volumes to search.

d) Expert opinions. The three previous categories involve a wide search of software testing studies about *MapReduce* programs, but other relevant studies could not be found. The opinion of authors with experience in software testing and *MapReduce*, together with the other related *mapping study*[25] could provide potential primary studies.

This large search is difficult to carry out because the software engineering search engines do not adequately support the *mapping studies* searches[53]. To avoid this problem, we created a program that splits the 9384 combinations of search terms in 2346 searches and simulates a human performing these requests. The potential primary studies are obtained after approximately a week to avoid bans in the search engines due to a high number of requests.

TABLE 2 Quality related terms (outcome and intervention).

Quality characteristics	Terms
Functional suitability	Functionality, functional, suitability, suitable, correctness, correctable, accuracy, accurate, compliance, compliant, appropriateness, appropriate
Performance efficiency	Performance, performable, efficiency, efficient, time-behaviour, resource utilization
Compatibility	Compatibility, replaceability, replaceable, co-existence, interoperability, interoperable
Usability	Recognizability, recognizable, learnability, learnable, operability, operable, ease of use, helpfulness, helpful, attractiveness, attractive, attractivity, technical, accessibility, accessible
Reliability	Reliability, reliable, availability, available, fault tolerance, recoverability, recoverable
Security	Security, secure, safety, confidentiality, confidential, integrity, non-repudiation, accountability, accountable, authenticity, authenticable
Maintainability	Maintainability, maintainable, modularity, modular, reusability, reusable, analyzability, analyzable, changeability, changeable, modification, modifiable, stability, stable, testability, testable
Portability	Portability, portable, adaptability, adaptable, transferability, transferable, installability, installable, effective, effectiveness
Other terms	Testing, assert, assertion, check, checking, test, test case, validate, validation, verify, verification, bug, defect, fault, failure, error, quality, risk, evaluation

3.2.3 | Study Selection

Some potential primary studies obtained from the data sources could not contain information about software testing in the *MapReduce* programs. In this *mapping study* a series of filters selects only the studies that contain relevant information to answer the research questions. The potential primary studies that do not pass the filters are excluded, and the remainder make up the primary studies used to answer the research questions. The filters consist in the next criteria applied in the following order:

C1) Filter by year. A potential primary study is excluded when the publication year is before the *MapReduce* paper (2004) or before the creation of technologies/fields expressed in the search terms of Table 1.

C2) Filter by area. The potential primary studies are excluded when their research is not about Computer Science or Information systems.

C3) Filter by field. The potential primary studies are excluded when they do not contain *Big Data* information.

C4) Filter by topic. The final filter only includes the studies about software testing in the *MapReduce* programs, the remainder are excluded.

For example, the last filter excludes the papers focused on software testing of the underlying technology such as the distributed system *Hadoop*, cloud computing, net or operative system, among others. Despite the normal execution depending on all previous technologies, usually they are mature enough and the developer/tester is only focused on the *MapReduce* application. Some papers that have been excluded are intended to improve the *Hadoop* performance through infrastructure failure forecasting [54] or to inject infrastructure failures in a distributed filesystem [55], among others that do not test the *MapReduce* applications. Some other papers employ the *MapReduce* and *Big Data* capabilities to speed up testing in other *non-MapReduce* programs. For example, [56,57] are frameworks to perform unit testing and mutation testing in general programs taking advantage of the parallel capabilities of the *MapReduce* processing model.

This *mapping study* performs a wide search with more than 70000 research papers found before the filter C1. Finally, 54 primary studies are obtained after the application of all exclusion filters C1-C4 and removing all duplicates and all old versions of the research lines, as detailed in Fig. 4.

3.3 | Data Extraction

The relevant information of the primary studies is extracted through a template divided in two parts. The first part is in general based on checklists of international standards related to the research questions, and the second part is focused on other data that could be interesting to analyze. The data extracted for answering the research questions are:

RQ1 "Why is testing performed in the *MapReduce* programs?" Extraction of the arguments employed in the primary study to perform testing in the *MapReduce* programs.

RQ2 "What testing is performed in the *MapReduce* programs?" The data are extracted following two checklists that characterize the type of testing performed in each primary study: checklist of the 31 ISO/IEC 25010:2011 Quality (sub)characteristics [42], and checklist of the 17 ISO/IEC/IEEE 29119-4:2015 Quality-Related Types of Testing [58].

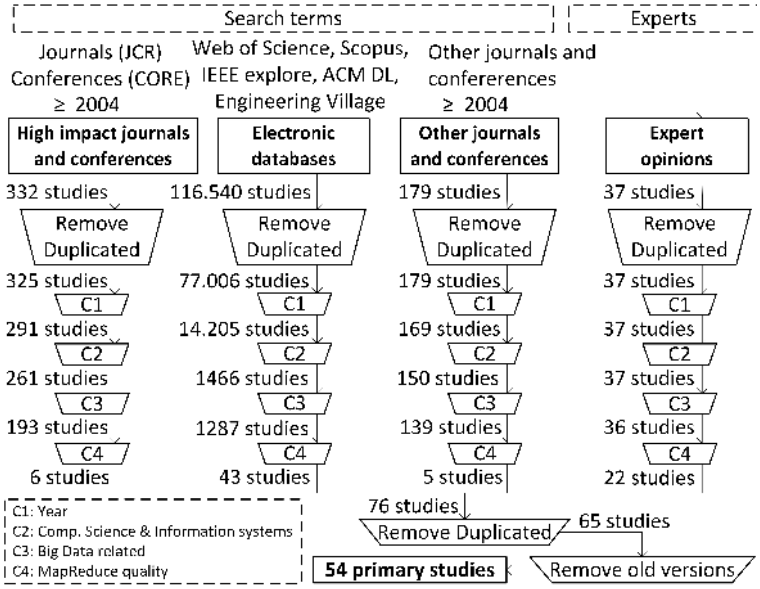

FIGURE 4 Study selection of the primary studies.

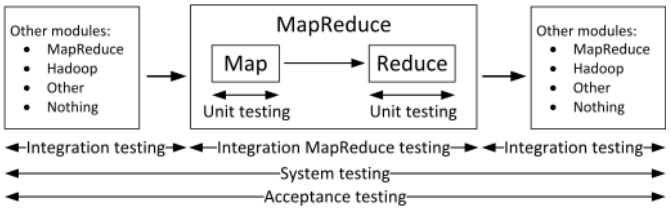

FIGURE 5 Test levels based on ITSQB and adapted to MapReduce.

RQ3 "How is testing performed in the *MapReduce* programs?" The data are extracted following a checklist of the 11 ISO/IEC/IEEE 29119-1:2013 Annex A: Test activities[9], together with a checklist of test areas: Testing specific to the *MapReduce* programs, Testing not specific to the *MapReduce* programs (other technologies/paradigms can be tested), Unclear and Not applicable. In addition, the following information about the tools used for testing is extracted: Does the study include the creation of a specific tool or use an existing tool? Is the tool based on another tool? Is the tool available? For example, if the tool is accessible via the Internet or with some type of open source license.

RQ4 "Who, where and when is testing performed in the *MapReduce* programs?" The data are extracted following three checklists focused on the roles, the lifecycle and the test level. The first checklist contains the following roles: Manager, Analyst, Architect, Tester, Test manager, Test strategist, Other stakeholders, Unclear and Not applicable. These test roles are described in the ISO/IEC/IEEE 29119-1:2013 Annex E[9]. The second checklist contains the 6 ISO/IEC 12207:2008 Software Implementation lower level Processes[59] and the 11 ISO/IEC 12207:2008 System Context Technical processes[59]. The third checklist is based on ISTQB test levels[60] and adapted to *MapReduce* with two changes represented in Figure 5: (1) Unit testing is divided in "Unit testing in *Map* function" and "Unit testing in *Reduce* function", and (2) "Integration testing" is for the integration of the *MapReduce* program with other modules, whereas "Integration *MapReduce* testing" is for the integration between *Map* and *Reduce* functions.

Other data are extracted in the *mapping study* because they can be interesting in order to characterize the results and obtain new findings. These data are extracted in a checklist with the following information about the research validation of the studies:

a) The different types of validation summarized in the Mary Shaw paper[61]: Analysis, Evaluation, Experience, Example, Persuasion and Blatant assertion.

b) Other characterizations of the research: Validation with external programs, Validation with own programs, Another type of validation, Without validation, Unclear, Other and Not applicable.

3.4 | Data Synthesis

The data extracted from the primary studies are synthesized in order to answer the research questions. In empirical software engineering there are several synthesis methods [62] based on different approaches according to the type of data or research questions, among others. In this *mapping study* the synthesis is performed using (1) thematic analysis [63] to answer the RQ1, and (2) meta-ethnography [64] for the remaining research questions. These synthesis methods are focused on qualitative data but synthesize the data in a different way.

The thematic analysis is selected to respond RQ1 (Why is testing performed in the *MapReduce* programs?) because it extracts a taxonomy of the reasons for testing from the primary studies. Then the RQ1 is answered by a frequency analysis of these reasons for testing. This thematic analysis is performed with a grounded approach [65] that consists of the following steps:

1. Reading of the primary studies.
2. Extraction of the segments/phrases that include the reasons for testing.
3. Create a group of labels for each previous segment/phrase based on the type of reason for testing.
4. Refine all labels several times until a few labels are obtained that compose a taxonomy of the reasons for testing.
5. Frequency analysis of the reasons for testing employed in the primary studies based on the previous taxonomy.

The meta-ethnography is selected to answer the research questions RQ2 to RQ4 because it transforms the primary studies data into a more easily analyzable shared context. This method is employed in software engineering [66] and translates all primary studies on data under several facets that contain the checklists described in the data extraction (Section 3.3). Once the data are extracted from the primary studies in these checklists, the research questions are answered by a frequency analysis. This *mapping study* follows the 7 steps proposed by Noblit et al. [64]:

1. Getting started. The topic under synthesis is software testing of the *MapReduce* programs and is well studied through *mapping study*.
2. Deciding what is relevant to the initial interest. All primary studies are important.
3. Reading the studies. The primary studies are read in order to extract the relevant data.
4. Determining how the studies are related. Primary studies could contain related concepts or very different concepts. The relationship between these concepts is established through the checklists of the data extraction of Section 3.3.
5. Translating the studies into one another. The primary studies are translated into relevant data according to the unified checklists of Section 3.3.
6. Synthesizing translations. This *mapping study* creates more general concepts by the answers of research questions. The research question RQ2 is answered by a frequency analysis of their two checklists, whereas both RQ3 and RQ4 by its three checklists described in Section 3.3.
7. Expressing the synthesis. The research questions are answered and discussed in Section 4 following the previous steps of the *mapping study*.

3.5 | Limitations of the Systematic Mapping Study

Despite the fact that planning of this *mapping study* aims to avoid bias, there could be some limitations.

- The results are limited by the academic context because the data sources are focused on the research field. Bias could be generated if the research papers do not represent the reality and motivations of the software testing in *MapReduce* programs.
- Further bias occurs if some research questions cannot be properly answered through the checklist of the data extraction. In order to minimize this bias, the majority of these checklists are based on the international standards.
- Another less important potential bias could occur during the search process if some primary studies are not found without search terms or expert opinions. In order to minimize the bias, a thorough search is performed in several databases, journals, conferences and experts.

In order to avoid bias in the results, all steps are reviewed and some countermeasures are taken in research questions, the search process, data extraction and data synthesis:

1. Research question: created by the Kipling method [39] instead of ad-hoc.

TABLE 3 Frequency of the primary studies over time.

Statistics	2010	2011	2012	2013	2014	2015	2016 until July
Frequency	1 (1.85%)	7 (12.96%)	4 (7.41%)	19 (35.19%)	12 (22.22%)	10 (18.52%)	1 (1.85%)
Absolute frequency	1 (1.85%)	8 (14.81%)	12 (22.22%)	31 (57.41%)	43 (79.63%)	53 (98.15%)	54 (100%)

2. Search process:

   Search terms: the use of a large number of terms could improve the search process by obtaining more potential primary studies. This *mapping study* searches for a combination of 92 *MapReduce* related terms and 102 testing terms obtained from ISO/IEC 25010:2011 Quality (sub)characteristics[42] with synonyms obtained through Kitchenham et al.[22] points of view.

   Data sources: this study searches 5 electronic databases and 2311 proceedings/volumes related to software testing in the *MapReduce* programs. The other data source is the expert opinions of the field in order to minimize the bias by adding primary studies that could not be found by the previous search.

   Study selection: the mapping study excludes the non-relevant studies based on 4 filters. These filters were reviewed in order to obtain the relevant studies.

3. Data extraction: the majority of the data extracted are based on checklists, in some cases obtained from international standards and in others created or adapted to the *MapReduce* processing model.

4. Data synthesis: the synthesis methods used in this work are employed in software engineering[62].

4 | RESULTS

The results are obtained through the conducting of the *systematic mapping study* that answers the research questions based on the planning of Section 3. Subsection 4.1 contains the primary studies. From them, the data are extracted, and the synthesis is developed in Subsection 4.2. Finally, the general results are discussed in Subsection 4.3.

4.1 | Primary Studies

In this work there are 54 primary studies derived from more than 70000 potential studies obtained though the search process detailed in Figure 4. These primary studies are detailed in Table C3 of Appendix C with the year of publication, type of contribution and a summary.

The *MapReduce* processing model was described in 2004, but the software testing efforts in this field according to the primary studies started in 2010 with only 1 study and after six years and six months the number of primary studies has increased to 54. Table 3 summarizes the frequencies of these primary studies over time and reveals that the research efforts of the topic may have grown because after 2013 the attention increases.

The different types of validations employed in the research are summarized in Table 4. The majority of the studies validate their research through examples (40.74%) or experience (35.19%). In 75.93% of the studies the validation is carried out applying the testing research in a program(s), but in 11.11% of the primary studies the research is not validated.

Testing in *Big Data* has open up new challenges[67], especially in the understanding of the data and its complex structures[68]. Gudipati et al.[69] establish a classification of testing in the *Big Data* field. This study includes the validation of the *MapReduce* process together with other non-functional characteristics like performance and failover. All of these characteristics are one of the main challenges in *Big Data* testing[68]. In order to overcome these challenges though software testing, it is recommended to deploy a distributed environment like production, preferably in the cloud[69,70].

Software testing can be performed in different dimensions and some authors suggest to address the three Vs of *Big Data* (Volume, Velocity and Variety). In the case of high Volume, it could be difficult to check whether the test case output is the expected, and the use of automatic tools can be helpful[69]. In the case of Variety such as semi-structured or un-structured data, it can be helpful to transform them in a structured way[69]. To test the Velocity, it is recommended to design performance tests[69]. In addition to Volume, Variety and Velocity, other authors suggest to consider the Veracity through data cleaning and normalization[70]. Those four Vs make an impact not only in the program execution, but also in the performance tests[71]. Zhenyu Liu[71] classifies the performance testing in *Big Data* in: (1) concurrent test (the impact of multiple users and

TABLE 4 Number of primary studies per type of validation.

			Number of studies			
Analysis					7 (12.96%)	54 (100%)
Evaluation					0 (0%)	
Experience					19 (35.19%)	
Example					22 (40.74)	
Persuasion					0 (0%)	
Blatant assertion					6 (11.11%)	
With validation	**Over programs**	**External programs**	30 (55.56%)	41 (75.93%)	47 (87.04%)	54 (100%)
		Own programs	12 (22.22%)			
	Other validation		8 (14.81%)			
Without validation					6 (11.11%)	

applications in concurrency), (2) load testing (realistic data loads to analyze the response of the program), (3) stress test (testing under extreme data), and (4) capacity test (the analysis of the resources that can be used).

The majority of the primary study papers are focused in the capacity and load testing. These studies are summarized in Section 4.1.1 whereas those primary studies that are more related to the functionality are described in Section 4.1.2.

4.1.1 | Performance testing and analysis

In the primary studies, the performance analysis is mainly addressed by the simulation of the program executions, or by evaluation of a performance prediction model. These prediction models characterize the performance based on different kinds of input parameters. The model of Song et al.[72] predicts the execution time given some characteristics about both the input dataset, the program functionality and the programing cluster. In addition, other models obtain the execution time considering also the filesystem[73]. The prediction models can have different goals beyond the execution time, for example the Yang et al. model[74] helps to obtain the values of the input parameters that achieve the best execution time. The tester varies the input parameters (the network or the locality of the data, among others) and then analyzes the impact in the performance.

The performance can be predicted using an stochastic approach, for example by Stochastic Petri Nets[75]. Another stochastic model[76] also considers the *MapReduce* tasks that are re-executed due its frequent failures. The performance of the *MapReduce* and *Big Data* applications can also be evaluated through large scale stochastic models by Mean Field Analysis[77].

While some models predict the performance analyzing the execution time of several samples[78] or considering the previous executions[79], other models consider some specific characteristics of the *MapReduce* execution. The Vianna et al. model[80] considers the influence over the performance of the *MapReduce* tasks that are executed in parallel. The network is another issue that can cause bottlenecks in the *MapReduce* programs and several models consider it to predict the performance[81,82], others also consider the task failures and I/O congestions[83].

Together with the network, the memory can cause performance issues, especially in iterative programs or those with high I/O operations. The performance of the shared-memory computation programs can be predicted with the Tanzil et al. model[84], whereas in those programs with Remote Direct Memory Access, the Wasi-ur-Rahman et al. model[85] can be used. The Apache Spark[3] programs process the data using distributed memory abstraction and their performance can be predicted by a model that executes a sample of data[86].

The cluster that executes the *MapReduce* programs can also influence the performance, especially when this cluster is formed by a heterogenic infrastructure. In these clusters, the Zhang et al. model[87] predicts the performance with bounds: upper and lower execution time. Another model to predict the performance in these clusters, employs the machine learning technique Support Vector Machine[88]. There are several clusters deployed in the cloud to obtain several advantages in terms of elasticity and cost. For programs executed in these clusters, the performance can be predicted modeling the systems with Layered Queueing Network[89]. In the case of an I/O intensive programs in cloud, the performance can be predicted using a CART (Classification And Regression Tree) model[90]. When the programs executed in the public cloud have an deadline requirements to satisfy, the performance can be predicted with Locally Weighted Linear Regression model considering the previous execution and the data executed in parallel[91]. For those programs that are not only executed in a public cloud, but in hybrid cloud, their performance can be predicted with the Ohnaga et al. model[92].

Several frameworks transform queries into *MapReduce* jobs, such as Hive[93] and Pig[94]. The execution time of the Hive SQL-like queries can be forecasted using multiple linear regression to predict the execution time of all the *MapReduce* jobs generated from these queries[95]. The multiple

regression analysis can be also used to predict the execution time of the join queries in Pig programs[96]. In contrast, the Zhang et al. model[97] predicts the performance of Pig programs considering the previous executions.

In addition to the prediction models, the testers can simulate the execution of the programs to analyse their performance in fine-grain way. As in the prediction models, the simulators also consider characteristics about both the input dataset, the program functionality, the programming cluster and the file system[98]. The MRPerf simulator[99] considers the inter and intra rack interactions over network using ns-2, and can be combined with other simulators, as for example DiskSim. The Chauhan et al. simulator[100] is based on MRPerf but including, among others, some random time due to operating system scheduling and network communication delays.

The execution time of the *MapReduce* programs can be also obtained using the modelling language proposed by Barbierato et al[101]. The tester can also monitor the execution of the *MapReduce* programs and test cases, obtaining charts to evaluate the performance and potential bottlenecks[102]. Villalpando et al.[103] propose a model for the *Big Data* application stablishing a relationship between the performance and reliability measures based on the international standard of quality ISO/IEC 25010[42].

Despite there are several research lines to predict the execution time, there is no comprehensible comparison between them. In general, these studies are evaluated only with few different case studies. The scientific contribution of these prediction models can be improved with empirical evaluation against other models using a standardized benchmark.

The main difference between these models is not just the technique/approach employed, but also the parameters used by the model. Different characteristics of the input dataset, program functionality, programming cluster and filesystem are considered as parameters, for example: size of data or number of <key, value> pairs (input dataset), complexity or overhead of Map (program functionality), number of CPU cores or racks (programming cluster), and number of HDFS replicas or the data transfer time for a HDFS block (filesystem).

There are a lot of different parameters, but there is no clear intuition of which parameters have more influence in performance. The contribution of the performance prediction studies can be improved evaluating which parameters really influence the performance and which not. Then the prediction models can be designed with more standardized subset of parameters that have a notorious influence in performance.

4.1.2 | Functional testing

The misconfiguration is one of the most common problems that lead to a memory/performance issues in *MapReduce*[104]. But according to the Ren et al. empirical study[105], the users rarely tune the configuration parameters that are related to performance. The users usually tune the configuration parameters only related to a failures[105]. Another empirical study analyzes 200 production failures obtaining that the majority are related to the data, and only the 1.5% are related to the performance (out of memory)[106]. In production there are several programs that does not finish their execution, Kavulya et al.[107] indicate that around the 3% of programs, and a more broader study indicates this percentage between 1.38% and 33.11%[105].

An analysis of 507 programs indicates at least 5 different kinds of faults caused by the non-deterministic execution of the *MapReduce*[108]. Camargo et al.[109] classify the specific faults of the *MapReduce*, whereas Mor'an et al.[110] classify those caused by the non-determinism. Chen et al.[111] propose a formal approach to detect these faults caused by the non-determinism. In contrast, Csallner et al.[112] employs symbolic execution to check the program under test. Another technique to detect these faults caused by non-determinism checks dynamically the properties of the program under test with random data[113].One of the reasons for the non-deterministic execution is the tolerance of infrastructure failures. There are several studies that proposed to inject infrastructure failures in the test case design[114]. Failure Scenario as a Service (FSaaS)[115] injects the infrastructure failures in a cluster deployed in the cloud.

Several testing techniques are devised in order to generate test inputs aimed to detect the functional faults, such as those caused by the non-deterministic execution or other semantic errors. The MRFlow testing technique[116] generates the test coverage items that can be used to generate the test inputs based on the data-flow technique adapted to the *MapReduce* processing model. Another technique to generate the data of the test cases, employs a bacteriological algorithm aimed to kill some semantic mutants specific for *MapReduce*: varies both the number of the Reducers and the existence or not of the Combiner functionality[117]. In those *Big Data* ETL (Extract, Transform and Load) programs that integrate several technologies (*MapReduce*, Pig, Hive, among others), a subset of representative data for test can be obtained from the dataset through input space partition together with constraints[118]. In the dataflow programs like Pig, the test inputs can be generated using dynamic-symbolic execution in the control-flow graph of the program[119].

Other kind of checks can be performed in the *MapReduce* programs. Dörre et al.[120] propose an automatic checker that detects statically incompatibilities between the types of the <key, value> pairs processed by the *MapReduce* programs. Rabkin et al.[121] analyse statically the configuration parameters used by different frameworks, including Hadoop. The *MapReduce* developers and testers should analyse the configuration parameters used because the 17% of Hadoop options are not documented and the 6% is not used in the code. The main *Big Data* frameworks can be affected in the same way than Hadoop because these issues are common in the open-source programs[121]. The correctness of the *MapReduce* programs can also be verified formally through proofs modelling the specification as Coq functions[122].

TABLE 5 Number of primary studies per type of reason for testing.

Types of reasons	Number of papers	Number of formal reasons	Number of informal reasons	Number of total reasons
Performance related	30	5 (6.02%)	36 (43.37%)	41 (49.4%)
Failure related	11	3 (3.61%)	9 (10.84%)	12 (14.46%)
Improper use	2	3 (3.61%)	1 (1.2%)	4 (4.82%)
Data related	9	2 (2.41%)	8 (9.64%)	10 (12.05%)
Configuration related	3	2 (2.41%)	1 (1.2%)	3 (3.61%)
Time related	2	2 (2.41%)	0 (0%)	2 (2.41%)
Cost related	7	0 (0%)	7 (8.43%)	7 (8.43%)
Other	4	0 (0%)	4 (4.82%)	4 (4.82%)
		17 (20.48%)	66 (79.52%)	83 (100%)

4.2 | Synthesis

The primary studies contain the answers to the research questions, but this information is hidden inside. The synthesis obtains valuable information in order to answer the research questions based on the data extracted from the primary studies. The data are extracted following the template defined in Subsection 3.3 and then synthesized by the methods described in Subsection 3.4. In the following subsections the primary studies are analyzed, classified and summarized in order to obtain the answer to each research question systematically.

4.2.1 | RQ1 Why is testing performed in the MapReduce programs?

The *MapReduce* programs are tested for several reasons. A model/taxonomy of these reasons is obtained applying the synthesis method meta-ethnography to the primary studies, as described in Subsection 3.4. The reasons for testing obtained are:

- *Performance related*: issues derived from the performance goals, service level agreements, size of the data, performance under infrastructure failures and prediction/analysis/optimization of the performance.
- *Failure related*: the specific faults of the *MapReduce* programs and the number of the programs that fail in production.
- *Improper use*: not all programs fit correctly in the *MapReduce* processing model.
- *Data related*: the challenges related to schema-less data and poor data quality.
- *Configuration related*: the misconfiguration of the infrastructure or program parameters may produce a failure.
- *Time related*: the programs may fail after a long time of resource usage.
- *Cost related*: testing can be carried out in order to reduce the cost of development, resource utilization and so on.
- *Other*: the reasons that do not fall in another category of the model/taxonomy of the reasons but do not constitute a new category of reasons.

Per each one of the above categories of reasons for testing, Table 5 indicates the number of primary studies that details these reasons. Note that a primary study can contain one or several reasons for testing. In Table 5 each reason for testing is also classified based on the degree of formality of the evidence in accordance with the following types: reasons with formal evidence and with informal evidence.

Reason with formal evidence: the reason for testing is detailed in the primary studies empirically or with some rigorous evidence of this reason to test. For example, if one paper performs an extensive analysis of several programs and detects that testing is necessary because a lot of programs crash in production.

Reason with informal evidence: the reason for testing is not clearly explained or not detailed in the primary studies due to the absence of rigorous analysis of the evidence for this reason to test. For example, if a paper indicates that the testing is necessary because the developers do not know how to configure the performance parameters of *MapReduce* programs.

TABLE 6 Number of primary studies per type of reason related to performance.

Types of "performance related" reasons	Number of papers	Number of formal reasons	Number of informal reasons	Number of total reasons
Optimization/improvement of application performance	11	0 (0%)	11 (26.83%)	11 (26.83%)
Analysis of application performance	11	0 (0%)	11 (26.83%)	11 (26.83%)
Influence of infrastructure in application performance	4	3 (7.32%)	2 (4.88%)	5 (12.2%)
Influence of dataset in application performance	2	0 (0%)	2 (4.88%)	2 (4.88%)
Fulfil SLA or performance goals	10	2 (4.88%)	8 (19.51%)	10 (24.39%)
Other	2	0 (0%)	2 (4.88%)	2 (4.88%)
		5 (12.2%)	36 (87.8%)	41 (100%)

The most frequent type of reason for testing is "performance related" described in 30 primary studies and represents 49.4% of the total reasons explained in all primary studies, followed by "failure related" with 14.46%, "data related" with 12.05%, and "cost related" with 8.43% of the total number of reasons. Considering the formal evidence of the testing reasons, "performance related" is also the main reason in the primary studies with 6.02% of the total reasons (5 of formal evidence out of a total of 17 of formal evidence), followed by "failure related" and "improper use" with 3.61% of total reasons (3 of formal evidence out of a total of 17 of formal evidence).

In the model/taxonomy obtained through the synthesis of the primary studies, the 41 "performance related" reasons for testing are sub-divided in the following sub-categories of reasons:

- *Optimization/improvement of application performance*: testing is aimed to the improvement of the program performance.
- *Analysis of application performance*: understanding of performance to detect bottlenecks, among other issues.
- *Influence of the infrastructure in application performance*: whereas the *MapReduce* applications can be designed without consider the infrastructure, the program performance is influenced by the production infrastructure.
- *Influence of dataset in application performance*: in the same way that the infrastructure impacts the performance, the dataset used in production also make an influence.
- *Fulfil SLA or performance goals*: the reason for test the program is to fulfill service level agreements or other performance goals such as deadlines.
- *Other*: the reasons that do not fall in another sub-category of the model/taxonomy of the performance reasons but do not constitute a new sub-category of reasons.

Per each one of the above sub-categories for testing that are related to performance, Table 6 indicates the number of the primary studies and its reasons for testing.

From the 41 "performance related" reasons for testing, the most frequent are focused in the analysis (36.83% of "performance related" reasons) and optimization of the performance (26.83% of "performance related" reasons), followed by the fulfillment of the performance goals (24.39% of "performance related" reasons). The remainder of reasons for testing related to performance, analyze the influence of the infrastructure (12.2% of "performance related" reasons) and the dataset (4.88% of "performance related" reasons), followed by other issues (4.88% of "performance related" reasons).

Of all the reasons for testing the programs, only 20.48% are based on formal evidence, and the remaining 79.52% are based on informal evidence. Regardless of the formality of evidence, the reasons for testing the *MapReduce* programs most described in the primary studies include "performance related", especially for the analysis, optimization and fulfillment of performance goals. The least reasons for testing described are "time related", "configuration related", "improper use" and "other".

4.2.2 | RQ2 What testing is performed in the MapReduce programs?

The planning of Subsection 3.4 proposes a meta-ethnography[64] to answer this research question. The data extracted from each primary study has two facets in order to answer RQ2:

TABLE 7 Number of primary studies per ISO/IEC 25010:2011 Quality (sub)characteristic.

			Number of studies		
ISO 25010:2011 System/software product quality	**Functional suitability**	**Functional Completeness**	2 (3.7%)	14 (25.93%)	46 (85.19%)
		Functional correctness	14 (25.93%)		
		Functional appropriateness	2 (3.7%)		
	Performance efficiency	**Time-behaviour**	32 (59.26%)	35 (64.81%)	
		Resource utilisation	14 (25.93%)		
		Capacity	1 (1.85%)		
	Reliability	**Maturity**	1 (1.85%)	3 (5.56%)	
		Availability	1 (1.85%)		
		Fault tolerance	1 (1.85%)		
		Recoverability	3 (5.56%)		
Other studies	**Characterization studies**			4 (7.41%)	8 (14.81%)
	Overview of testing			4 (7.41%)	

TABLE 8 Number of primary studies per ISO/IEC/IEEE 29119-4:2015 Quality-Related Type of Testing.

		Number of studies	
ISO/IEC/IEEE 29119-4:2015 Types of testing	**Performance-Related Testing**	32 (59.26%)	44 (81.48%)
	Functional Testing	12 (22.22%)	
	Backup/Recovery Testing	2 (3.7%)	
Other studies	**Characterization studies**	5 (9.26%)	10 (18.52%)
	Overview of testing	5 (9.26%)	

a. Quality (sub)characteristics for each study according to the ISO/IEC 25010:2011[42] represented in Table 7.

b. Quality-Related Types of Testing proposed in each study based on ISO/IEC/IEEE 29119-4:2015[58] and summarized in Table 8.

The majority of efforts are focused on "performance efficiency" with 64.81% of the studies, then on "functional suitability" with 25.93% of the studies, and finally on "reliability" with 5.56% of the studies. Regarding the type of testing, 59.26% apply "performance-related testing", 22.22% employ "functional testing" and 3.7% use "backup/recovery testing".

The results obtained through the combination of both facets are more or less those expected: the "performance-related testing" is used to "performance efficiency" characteristics, the "functional testing" to "functional suitability", and "backup/recovery testing" to "reliability".

4.2.3 | RQ3 How is testing performed in the MapReduce programs?

This research question is answered through the meta-ethnography[64] proposed in Subsection 3.4. In order to answer RQ3, the primary studies are analyzed considering three facets:

a. Testing methods/techniques are summarized in Table 9 according to the test activities proposed in Annex A of ISO/IEC/IEEE 29119-1:2013[9].

b. Dependency between the primary studies and the *MapReduce* processing model is depicted in Table 10. This table describes whether the testing methods, techniques or studies are specific for the MapReduce or could be applied to other paradigms/technologies.

c. Tools created or used in the primary studies to perform software testing are characterized in Table 11.

The majority of the papers (74.07%) focus on testing only the *MapReduce* specific parts of the program. These programs have challenges related to performance issues and the correct operation of the program under parallel architecture. These issues among others are tested mainly by

TABLE 9 Number of primary studies per ISO/IEC/IEEE 29119-1:2013 Test activity of Annex A.

					Number of studies		
ISO/IEC/ IEEE 29119-1:2013 Annex A: Test activities	V&V analysis		Evaluation	26 (48.15%)		29 (53.7%)	43 (79.63%)
			Simulation	9 (16.67%)			
	Testing	Dynamic testing	Structure based	4 (7.41%)	7 (12.96%)	12 (22.22%)	
			Specification based	1 (1.85%)			
			Experienced based	1 (1.85%)			
			Other	1 (1.85%)			
		Static testing	Static analysis	3 (5.56%)	5 (9.26%)		
			Other	3 (5.56%)			
	Formal methods		Model checking	1 (1.85%)		2 (3.7%)	
			Proof of correctness	1 (1.85%)			
Other studies	Characterization studies				6 (11.11%)	11 (20.37%)	
	Overview of testing				5 (9.26%)		

TABLE 10 Number of primary studies per test area covered.

		Number of studies	
Specific of MapReduce		40 (74.07%)	50 (92.59%)
Not specific of MapReduce		10 (18.52%)	
Other studies	Characterization studies		4 (7.41%)

TABLE 11 Number of primary studies per tool created in their research.

				Number of studies	
Tool created or used	Based on other	Tool available	3 (5.56%)	11 (20.37%)	19 (35.19%)
		Tool not available	8 (14.81%)		
	Not based on other tools	Tool available	2 (3.7%)	8 (14.81%)	
		Tool not available	6 (11.11%)		
No tool created or used					35 (64.81%)

"evaluation" according to 48.15% of the studies and "simulation" in 16.67% of the studies. Other testing activities are used to a lesser degree, such as for example "structure based" in 7.41% of the studies or static analysis in 5.56% of the studies.

More than half of the studies (64.81%) do not create or use testing tools in their research. There are in total 19 tools, where 11 are based on other software testing related tools, and only 5 are freely available on the Internet with open source license.

4.2.4 | RQ4 Who, where and when is testing performed in the MapReduce programs?

The planning of the *mapping study* described in Section 3.4 proposes a meta-ethnography [64] to answer the research question through three facets:

a. The different roles that participate in the testing efforts of the *MapReduce* programs, described in Table 12.

b. Test levels summarized in Table 13 that contains a characterization of ISTQB test levels [60] adapted to the *MapReduce* processing model according to Figure 5.

TABLE 12 Number of primary studies per role.

		Number of studies	
Roles	**Tester**	45 (83.33%)	49 (90.74%)
	Developer	5 (9.26%)	
Other studies	**Characterization studies**		5 (9.26%)

TABLE 13 Number of primary studies per ISTQB Test level.

			Number of studies		
Levels of testing in ISTQB	**Unit testing**	**Unit testing Map**	16 (29.63%)	19 (35.19%)	44 (81.48%)
		Unit testing Reduce	19 (35.19%)		
	Integration MapReduce testing		35 (64.81%)		
	Integration testing		4 (7.41%)		
	System testing		2 (3.7%)		
	Acceptance testing		0 (0%)		
Other studies	**Characterization studies**			5 (9.26%)	10 (18.52%)
	Overview of testing			5 (9.26%)	

TABLE 14 Number of primary studies per ISO/IEC 12207:2008 Software Implementation lower level Process and System Context Technical process.

		Number of studies	
ISO/IEC 12207:2008 Software Implementation lower level Processes	**Software Construction Process**	3 (5.56%)	48 (88.89%)
	Software Qualification Testing Process	47 (87.04%)	
ISO/IEC 12207:2008 System Context Technical processes	**Implementation Process**	3 (5.56%)	48 (88.89%)
	System Qualification Testing Process	47 (87.04%)	
	Software Operation Process	1 (1.85%)	
Other studies	**Characterization studies**	5 (9.26%)	6 (11.11%)
	Overview of testing	1 (1.85%)	

c. Development cycle phase according to the Software Implementation lower level Processes and System Context Technical Processes of ISO/IEC 12207 [59] described in Table 14.

As expected, the main player for testing the *MapReduce* programs is the tester according to 83.33% of the studies, and then the developer according to 9.26% of the studies. Almost all primary studies, 87.04%, describe testing efforts in the "Software/System Qualification Testing Process" compared with 5.56% which focus on "Software Construction or the Implementation Process". In these processes, the studies cover in more detail the specific *MapReduce* parts of the program (*Map* and *Reduce* functions) instead of the other parts. The majority of the research efforts in 64.81% of the studies focus on the integration testing between *Map* and *Reduce* functions, and then 35.19% of the studies cover unit testing at the *Map* or *Reduce* functions. To a lesser extent the testing efforts are oriented towards the parts of the program that could not contain *MapReduce* functions: 7.41% of the studies consider the integration testing between the *MapReduce* functions with other parts of the program, and 3.7% of the studies for testing the system. All testing levels are covered by the primary studies except for acceptance testing.

From these results, it appears that the fulfillment of the contract or user requirements tested in the acceptance testing level is not greatly affected by the existence of *MapReduce* functions in the system. Despite the fact that the *Big Data* programs can contain a composite of several technologies/programs, the testing research efforts focus on testing the *MapReduce* functions in isolation from the rest of the system. Few studies

consider that a *Big Data* program can contain *MapReduce* functions together with other technologies. Regardless of the test level, the testing described in the primary studies is mainly performed in the Software/System Qualification Testing Process.

4.3 | Discussion of Results

The research questions of Subsection 3.1 are answered through the primary studies, data extraction and data synthesis. A summary is described below:

RQ1. Why is testing performed in the *MapReduce* programs? There are at least seven reasons for testing the *MapReduce* programs. The most frequent reasons are based on performance issues (analyze, optimize and fulfil performance goals), existence of several and specific failures, the type and quality of the data processed by these programs, and testing to predict and select efficiently the resources. To a lesser degree, the other reasons for testing are the improper use of the processing model or technology, misconfiguration or failures after a long period of executions.

RQ2. What testing is performed in the *MapReduce* programs? The majority of the research efforts in testing the *MapReduce* programs focus on the analysis of the performance, and to a lesser extent the functional aspects.

RQ3. How is testing performed in the *MapReduce* programs? Mainly by evaluation and simulation. In both cases testing is focused on the specific *MapReduce* functions and does not consider other parts of the program. Several tools are used to perform testing, but few are available on the Internet.

RQ4. Who, where and when is testing performed in the *MapReduce* programs? Testing is mainly performed by the tester in the Software/System Qualification Testing Process and the major efforts focus on the *MapReduce* program (unit and integration testing between *Map* and *Reduce* functions).

The analysis of several features about primary studies reveals, in addition to the answers to the research questions, other findings discussed below.

The relation between the reasons for testing the programs and the type of testing employed in each study is displayed in Figure 6. According to Table 8, 59.26% of the studies focus on performance testing (RQ2), which is very important because the *MapReduce* applications analyze large quantities of data. From RQ1 the reasons for testing the programs are obtained and 57.83% of these reasons are related to performance (48 reasons of a total of 85 according to the left side of Figure 6). The reasons for performance testing and the number of studies that test the performance are aligned. However, according to Table 8, the studies related to functionality only represent 22.22% even though 42.17% of the reasons for testing are related to functionality (35 reasons of a total of 85 according to the left side of Figure 6). There are more reasons for testing the functionality than its actual research efforts, which can indicate a challenge in the functionality testing to cover these reasons and improve the quality of the *MapReduce* applications.

The main test activities in RQ3 are evaluation in 48.15% of the studies and simulation in 16.67%. These two activities are the most frequent because the majority of studies are focused on performance testing (59.26% according to RQ2). Figure 7 characterizes the test activities (RQ3) and test levels (RQ4) regarding different types of testing (RQ2). The test levels in each type of testing are more or less similar to the answer to RQ4: the principal efforts are at integration testing level of *Map* and *Reduce* functions and to a lesser degree at unit level. However, the test activities are different depending on the type of testing: the performance testing employs evaluation and simulation to predict the time execution and resources, but functionality testing performs a variety of different test activities considering specific characteristics of the *MapReduce* processing model (static testing, structure based, formal methods, experience based and specification based).

The majority of the studies are published in conferences (75.93%) and there are few studies published in a high impact journal (12.96%). This situation could be improved with a stronger validation of the research because according to Table 4, 11.11% of studies are not validated, 40.74% are validated with examples and 22.22% employ programs created by the researcher to validate their own work. There is room for improvement in the software testing research of *MapReduce* applications.

This work analyses 54 studies in detail obtained through a wide search that ends with 1377 *Big Data* studies applying a filter (C4), in which only the studies that address software testing of *MapReduce* applications pass. Of these 1377 *Big Data* studies, 1043 are about *Big Data Engineering* and 334 about *Big Data Analytics*. Table 15 classifies the *Big Data Engineering* studies based on the research topic in order to characterize the research efforts. This classification reflects the research efforts to boost the *Big Data Engineering* field because 44.1% of the studies improve the technology, 18.31% analyse the technology through studies and surveys, 9.01% create new technologies to manage and analyse data, and 6.62% are focused on the state-of-the-art and challenges. Despite the challenges of testing in the *Big Data* area[68,70], there are few research lines which focus on testing *Big Data* programs in general and *MapReduce* programs in particular.

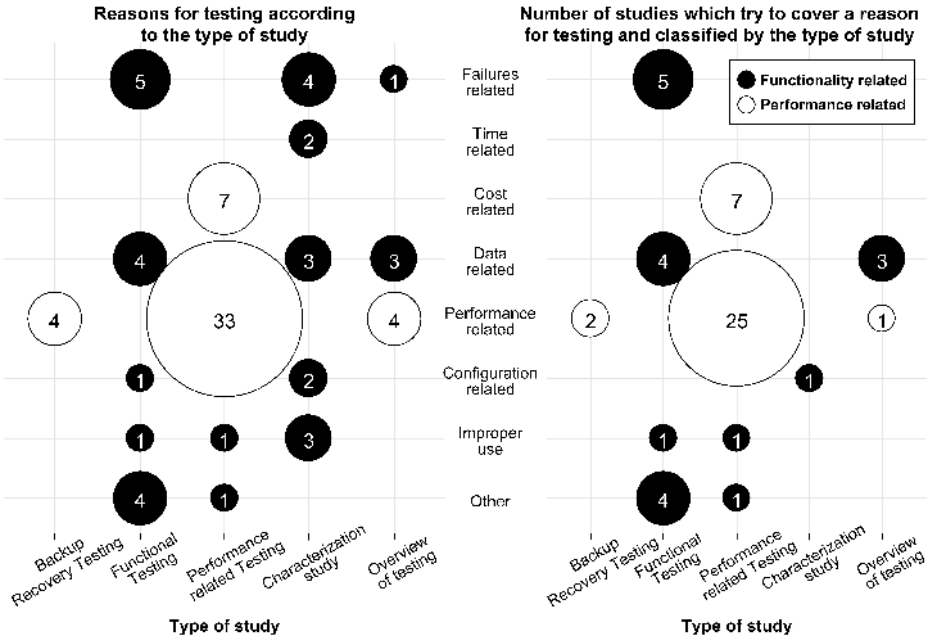

FIGURE 6 Number of reasons for testing and primary studies per type of study.

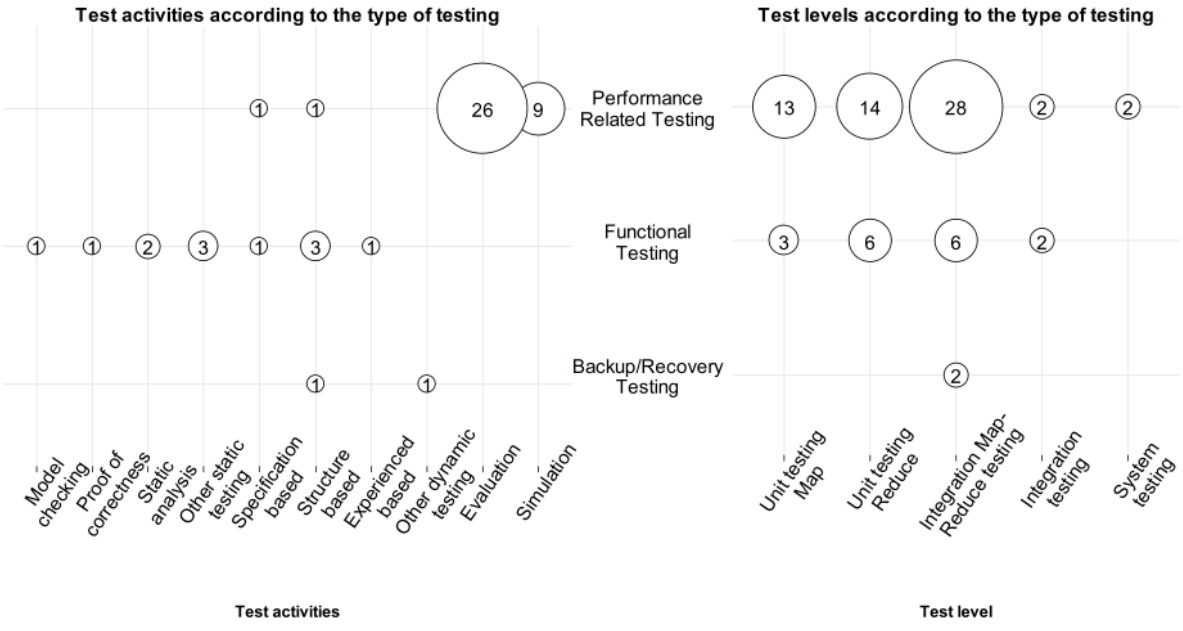

FIGURE 7 Number of primary studies per Test activity and test level according to the type of testing.

Other findings are commented in Section 4.1. Despite the number of research lines of testing in *MapReduce* is growing, the validation of these works is still simple with experience or case studies on few programs, sometimes created by the researcher. The research contribution of testing papers can be improved using controlled experiments with a standard benchmark. Specially in the performance prediction techniques that in general are not validated against other techniques. These performance prediction techniques employ a lot of different characteristics of the input dataset, program functionality, programming cluster and filesystem. In consequence, there is no clear intuition of which parameters have the more influence in performance and could be improved with rigorous validation.

TABLE 15 Number of Big Data Engineering studies in the last filter of the mapping study.

		Number of studies		
Improvements of technology	**Performance**	121 (11.6%)	460 (44.1%)	1043 (100%)
	Security	81 (7.77%)		
	Data acquisition, storage and extraction	45 (4.31%)		
	Fault tolerance and availability	42 (4.03%)		
	Energy	42 (4.03%)		
	Improvements outside of Hadoop	35 (3.36%)		
	Scheduling	34 (3.26%)		
	MapReduce model	14 (1.34%)		
	Different frameworks	10 (0.96%)		
	Other improvements	36 (3.45%)		
Studies/Surveys	**General quality in Big Data**	171 (16.4%)	191 (18.31%)	
	Other	20 (1.92%)		
Software testing	**For MapReduce programs**	64 (6.14%)	103 (9.88%)	
	For non-MapReduce programs	39 (3.74%)		
Big Data in the cloud		101 (9.68%)		
New frameworks	**New Hadoop frameworks**	85 (8.15%)	94 (9.01%)	
	Other new Frameworks	9 (0.86%)		
State-of-the-art and challenges		69 (6.62%)		
Debug		6 (0.58%)		
Other		19 (1.82%)		
Not applicable (Big Data Analytics)				334

5 | CONCLUSIONS

The number of studies on software testing in the *MapReduce* programs has increased during recent years. A characterization is carried out based on 54 research studies obtained from more than 70000 potential papers. The testing tasks in these programs are normally performed by the tester in the Software/System Qualification Testing Process due to a combination of the following 7 reasons: performance issues, potential failures, issues related to the data such as for example data quality, the reduction of the cost in resources, misconfigurations, improper use of the technology, time problems or other issues. These reasons for testing assume that both functional and performance testing are necessary, but the studies employ different approaches: functional testing considers different aspects of the program (such as specification and structure) while performance testing is more focused on simulation and evaluation. The research studies focus more on performance testing and are a challenge in functional testing due to its importance and the lack of research efforts.

The main goal of performance testing in the *MapReduce* studies is to predict the execution time and resources to execute efficiently the programs and satisfy the agreements. From the functionality point of view, the goal of the studies is to detect the faults considering the specific characteristics of the *MapReduce* processing model. Regardless of the type of testing, the majority of efforts are specific for the *MapReduce* technology at unit and integration level of the *Map* and *Reduce* functions. This situation may indicate a challenge in the integration of the MapReduce programs with other programs, especially other *Big Data* stack technologies.

The research of software testing in the *MapReduce* programs is mainly validated with programs of examples. There is room to mature with better validations and thus improve the research impact. Despite the lack of maturity, several studies create tools to support testing, but few are available on the Internet for users or other researchers. In *Big Data* there are few research studies of software testing in comparison to the research efforts to improve the technology, which indicates new opportunities in software testing of *Big Data* in general, and *MapReduce* in particular.

ACKNOWLEDGMENTS

This work was supported in part by projects TIN2016-76956-C3-1-R and TIN2013-46928-C3-1-R funded by the Spanish Ministry of Economy and Competitiveness, and GRUPIN14-007, funded by the Principality of Asturias (Spain) and ERDF funds.

How to cite this article: J.Morán, C. de la Riva, and J. Tuya, TestingMapReduce Programs: A SystematicMapping Study, J Softw Evol Proc,

APPENDIX

A HIGH IMPACT JOURNALS AND CONFERENCES FOR THE MAPPING STUDY

TABLE A1 High impact journals and conferences for the mapping study

JCR journals	
ACM Computing Surveys	International Journal of Information Processing and Management (IJIPM)
ACM SIGPLAN Notices	International Journal of Information Technology and Decision Making (IJITDM)
ACM Transactions on Database Systems (ACM TODS)	International Journal of Information Technology and Management (IJITM)
ACM Transactions on Information Systems (ACM TOIS)	International Journal of Software Engineering and Knowledge Engineering (IJSEKE)
ACM Transactions on Software Engineering and Methodology (ACM TOSEM)	Information and Software Technology (IST)
Computer Science and Information Systems (ComSIS)	Journal of Database Management (JDM)
Distributed and Parallel Databases	Journal of Information Technology (JIT)
Distributed Computing	Journal of Management Information Systems (JMIS)
Empirical Software Engineering (ESE)	Journal of Software: Evolution and Process
The International Arab Journal of Information Technology (IAJIT)	Journal of Parallel and Distributed Computing (JPDC)
IEEE Software	The Journal of Strategic Information Systems (JSIS)
IEEE Transactions on Knowledge and Data Engineering (IEEE TKDE)	Journal of Systems and Software (JSS)
IEEE Transactions on Parallel and Distributed Systems (IEEE TPDS)	Knowledge and Information Systems (KAIS)
IEEE Transactions on Software Engineering (IEEE TSE)	Software Quality Journal (SQJ)
International Journal of Data Warehousing and Mining (IJDWM)	Software Testing, Verification & Reliability (STVR)
International Journal of Information Management (IJIM)	
CORE conferences	
ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)	International Conference on Information and Knowledge Management (CIKM)
ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE)	International Conference on Intelligent Data Engineering and Automated Learning (IDEAL)
Advances in Databases and Information Systems (ADBIS)	International Conference on Management of Data (COMAD)
Australasian Data Mining Conference (AusDM)	International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)
Australasian Database Conference (ADC)	International Conference on Quality Software (QSIC)
Automated Software Engineering (ASE)	International Conference on Software and Data Technologies (ICSOFT)
Biennial Conference on Innovative Data Systems Research (CIDR)	International Conference on Software Engineering (ICSE)
Computer Aided Verification (CAV)	International Conference on Software Testing, Verification and Validation (ICST)

Databases and Programming Language (DBPL)	International Conference on Statistical and Scientific Database Management (SSDBM)
Empirical Software Engineering and Measurement (ESEM)	International Conference on Tests and Proof (TAP)
Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)	International Database Engineering and Applications Symposium (IDEAS)
European Conference on Parallel Processing (EURO-PAR)	International Symposium on Cluster Computing and the Grid (CCGRID)
European Conference on Principles of Data Mining and Knowledge Discovery (PKDD)	International Symposium on Intelligent Data Analysis (IDA)
European Software Engineering Conference (ESEC)	International Symposium on Software Testing and Analysis (ISSTA)
Evolution and Change in Data Management (ECDM)	International Workshop on Data Warehousing and OLAP (DOLAP)
IEEE International Conference on Cloud Computing (IEEE CLOUD)	International Workshop on Formal Approaches to Testing of Software (FATES)
IEEE International Conference on Cloud Computing Technology and Science (IEEE CloudCom)	Joint International Conference on Formal Techniques for Networked and Distributed Systems (FORTE)
IEEE International Conference on Data Mining (IEEE ICDM)	Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)
IEEE International Enterprise Distributed Object Computing Conference (IEEE EDOC)	Parallel Computing Technologies International Conferences Series (PaCT)
International Baltic Conference on Databases and Information Systems (DB&IS)	SIAM International Conference on Data Mining (SDM)
International Conference on Data Engineering (IEEE ICDE)	Software Engineering and Knowledge Engineering (SEKE)
International Conference on Data Warehousing and Knowledge Discovery (DaWaK)	Special Interest Group on Management of Data Conference (SIGMOD)
International Conference on Database and Expert Systems Applications (DEXA)	Symposium on Applied Computing (SAC)
International Conference on Database Systems for Advanced Applications (DASFAA)	Symposium on Large Spatial Databases (SSTD)
International Conference on Database Theory (ICDT)	Symposium on Principles of Database Systems (PODS)
International Conference on Distributed Computing Systems (ICDCS)	Very Large Data Bases Conference (VLDB)
International Conference on Extending Database Technology (EDBT)	

B OTHER JOURNALS AND CONFERENCES FOR THE MAPPING STUDY

TABLE B2 Other journals and conferences for the mapping study

Journals	
ACM DATA BASE	International Journal of Intelligent Information and Database Systems (IJIIDS
ACM SIGSOFT Software Engineering Notes (ACM SIGSOFT)	International Journal of Information Quality (IJIQ)
ACM Transactions on Management Information Systems (ACM TMIS)	International Journal of Information Systems and Change Management (IJISCM
Big Data Research	International Journal of Information Technologies and Systems Approach (IJITSA)
Computing and Information Technology (CIT)	International Journal of Parallel, Emergent and Distributed Systems (IJPEDS)
European Journal of Information Systems (EJIS)	Journal of Cases on Information Technology (JCIT)
Foundations and Trends in Databases (FTDB)	Journal of Data and Information Quality (JDIQ)
IEEE Cloud Computing	Journal of Digital Information Management (JDIM)

IEEE Computer	Journal of Enterprise Information Management (JEIM)
IEEE Distributed Systems Online (IEEE DS)	Journal of Information and Data Management (JIDM)
IEEE Transactions on Big Data	Journal of Information & Knowledge Management (JIKM)
IEEE Transactions on Cloud Computing (IEEE TCC)	Journal of Information Processing (JIP)
International Journal of Big Data Intelligence (IJBD)	The Journal of Information Processing Systems (JIPS)
International Journal of Cloud Applications and Computing (IJCAC)	Journal of Information Technology Research (JITR)
International Journal of Cloud Computing (IJCC)	Journal of Systems and Information Technology (JSIT)
International Journal of Distributed Systems and Technologies (IJDST)	Transactions on Large-Scale Data- and Knowledge-Centered Systems (Transactions LDKS)
International Journal of Enterprise Information Systems (IJEIS)	Journal of Enterprise Information Management (JEIM)
Conferences	
Advances in Model-Based Testing (A-MOST)	Industrial Conference on Data Mining (ICDM)
Alberto Mendelzon Workshop on Foundations of Data Management (AMW)	International Conference on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS)
International Conference on Big Data Analytics (BDA)	Internet and Distributed Computing Systems (IDCS)
International Conference Beyond Databases, Architectures, and Structures (BDAS)	IEEE/ACM International Symposium on Big Data Computing (BDC)
International Conference on Big Data and Smart Computing (BigComp)	IEEE International Conference on Big Data (IEEE BigData)
International Congress on Big Data (BigData Congress)	IEEE Symposium on Large-Scale Data Analysis and Visualization (IEEE LDAV)
Workshop on Scalability in Model Driven Engineering (BigMDE)	International Conference on Algorithms for Big Data (ICABD)
British National Conference on Databases (BNCOD)	International Conference on Big Data and Cloud Computing (BdCloud)
International Conference on Cloud and Autonomic Computing Conference (CAC)	International Conference on Big Data Cloud and Applications (BDCA)
International Conference on Cloud and Green Computing (CGC)	International Conference on Big Data Computing and Communications (BigCom)
International Conference on Cloud Computing and Services Science (CLOSER)	International Conference on Big Data Computing Service and Applications (BigDataService)
Cloud Computing (CloudComp)	International Multiconference on Computer Science and Information Technology (IMCSIT)
Conference on Data and Application Security and Privacy (CODASPY)	International Workshop on Machine Learning, Optimization, and Big Data (MOD)
International Computer Software and Applications Conference (COMPSAC)	Symposium on Network Cloud Computing and Applications (NCCA)
International Conference on Cloud and Service Computing (CSC)	Conference on Next Generation Information Technologies and Systems (NGITS)
European Joint Conference on Theory and Practice of Software (ETAPS)	ACM Symposium on Cloud Computing (SoCC)
International Conference on Future Data and Security Engineering (FDSE)	SPIN Workshop on Model Checking of Software (SPIN)
Federated Conference on Computer Science and Information Systems (FEDCSIS)	Symposium on Computational Intelligence in Big Data (CIBD)
International Conference on Future Internet of Things and Cloud (FICLOUD)	Symposium on Information Management and Big Data (SIMBig)
USENIX Workshop on Hot Topics in Cloud Computing (HotCloud)	Testing: Academic & Industrial Conference - Practice And Research Techniques (TAIC PART)
International Conference on Advanced Cloud and Big Data (CBD)	International Conference on Testing Communicating Systems (TestCom)
International Conference on Cloud Engineering (IC2E)	Workshop on Big Data Benchmarking (WBDB)

International Conference on Algorithms for Big Data (ICABD)	Workshop on Big Data Benchmarks, Performance Optimization, and Emerging Hardware (BPOE)
International Conference on Innovative Computing and Cloud Computing (ICCC)	Workshop on Mobile Big Data (Mobidata)
International Conference on Data Engineering and Management (ICDEM)	

C PRIMARY STUDIES

TABLE C3 Primary studies

Ref.	Year	Contribution	Summary
[106]	2013	Conference	A study and characterization of MapReduce-like failures
[72]	2013	Conference	A prediction model of individual MapReduce jobs based on important properties
[81]	2013	Conference	A performance prediction based on network properties and configuration of the cluster
[101]	2013	Conference	A performance prediction based on a representation of the architecture with some information of the MapReduce program
[118]	2015	Conference	Generator of representative data to testing Big Data programs based on input space partitioning
[107]	2010	Conference	A study and characterization of more than 170000 MapReduce executions
[74]	2011	Conference	A simple performance prediction model that considers the program and the system
[80]	2013	Journal	A model that obtains several metrics about the MapReduce programs performance and resource utilization
[69]	2013	Briefing	Classification of testing in Big Data and the underlying challenges
[109]	2013	Conference	Classification of MapReduce faults based on empirical changes in the programs
[111]	2015	Conference	Checking of the commutativity problem in the Reduce functions
[82]	2013	Conference	A performance prediction model based on information about the MapReduce program and the cluster
[99]	2012	Doctoral dissertation	A simulator of MapReduce program that obtains a prediction of the performance
[77]	2014	Journal	A performance prediction model of MapReduce program using Mean Field Analysis and information of the program, system and data
[89]	2015	Conference	A performance prediction model of MapReduce program in the cloud considering the program and the data
[115]	2012	Conference	A failure injector in the architecture using the cloud manager in order to test the MapReduce programs
[70]	2013	Conference	Challenges of software testing in Big Data
[105]	2013	Conference	A study and characterization of three Hadoop clusters
[91]	2016	Journal	Prediction of the performance and optimization of resource utilization based on deadline requirements
[102]	2011	Conference	Monitoring of the MapReduce program that generates detailed reports of the execution
[104]	2013	Journal	A study and characterization of several bugs in Big Data programs
[90]	2015	Conference	A performance prediction model of the MapReduce programs considering the deployment in virtualized cloud and the characteristics of the program
[78]	2012	Conference	A performance prediction model of the MapReduce programs considering several samplings of the input data
[114]	2015	Journal	A testing framework to run the MapReduce programs under architectural failures in order to test
[83]	2013	Conference	A performance prediction model of the MapReduce programs considering the resource contention and the task failures
[110]	2014	Conference	Classification of several MapReduce faults with a series of challenges in order to reveal the faults
[112]	2011	Conference	Functional Testing of the Reduce function based on symbolic execution
[108]	2014	Conference	Characterization of the MapReduce programs based on empirical study
[84]	2014	Conference	A performance prediction model of the MapReduce programs considering the memory shared and disk I/O
[79]	2014	Journal	Prediction of the MapReduce performance based on empirical executions and an adjustment based on micro benchmarks

[103]	2014	Journal	Performance analysis model for MapReduce applications based on ISO 25010 that establishes a relationship between the performance and reliability measures
[75]	2013	Conference	Obtains the performance of the MapReduce programs based on Stochastic Petri Nets
[95]	2015	Conference	Performance prediction of HIVE-QL queries through the underlying MapReduce applications based on multiple lineal regression
[97]	2013	Journal	Performance prediction of PIG queries through the underlying MapReduce applications
[73]	2014	Conference	Performance prediction for a MapReduce program and optimization based on the type of application and potential bottlenecks
[85]	2014	Conference	Mathematical model for performance prediction of the RDMA-Enhanced MapReduce programs
[87]	2013	Conference	A performance prediction model of the MapReduce programs considering information of the program and the performance for several parts of the program
[92]	2015	Conference	Model that predicts the performance of MapReduce applications in hybrid clouds
[86]	2015	Conference	Simulation of the Spark applications in order to obtain performance information
[88]	2014	Conference	A performance prediction model of the MapReduce programs considering the heterogeneity of the cluster
[76]	2011	Conference	A performance prediction model of the MapReduce programs based on the mean time between failures
[71]	2014	Conference	Overview and challenges of performance testing in Big Data
[119]	2013	Conference	Data generation for dataflow programs based on symbolic execution
[98]	2014	Conference	Simulating the MapReduce program under configurable hardware in order to obtain a performance prediction
[100]	2014	Conference	Simulating the scheduler of the MapReduce program in order to test the best configuration
[67]	2015	Conference	Test factory model for Big Data development
[121]	2011	Conference	Static analysis of the MapReduce configuration in order to detect misconfigurations and avoid failures
[120]	2011	Conference	Automatic checking of the java types inside MapReduce programs in order to detect incompatible types
[117]	2012	Dissertation	Data generator for MapReduce programs based on bacteriological algorithm in order to test the program
[116]	2015	Conference	Testing technique for MapReduce programs based on data flow and the MapReduce specifics
[113]	2013	Conference	Checking the correctness of the dataflow programs based on the operators properties
[96]	2013	Journal	Performance prediction of the join queries in Pig
[68]	2013	Journal	Overview and challenges of testing in Big Data
[122]	2011	Conference	Formal verification of the MapReduce program based on a model of the program/specification and invariants

References

1. *ISO/IEC JTC 1 - Big Data, preliminary report.* 2014.
2. Dean Jeffrey, Ghemawat Sanjay. MapReduce: Simplified Data Processing on Large Clusters. *Proc. of the OSDI - Symp. on Operating Systems Design and Implementation.* 2004;:137–149.
3. Apache Spark: a fast and general engine for large-scale data processing. https://spark.apache.org. Accessed: December 2017.
4. Apache Flink: Scalable batch and stream data processing. https://flink.apache.org. Accessed: December 2017.
5. Apache Hadoop: open-source software for reliable, scalable, distributed computing. https://hadoop.apache.org/. Accessed: December 2017.
6. Institutions that are using Apache Hadoop for educational or production uses. https://wiki.apache.org/hadoop/PoweredBy. Accessed: December 2017.
7. Schatz M. C.. CloudBurst: highly sensitive read mapping with MapReduce. *Bioinformatics.* 2009;25(11):1363–1369.
8. Kocakulak Hakan, Temizel Tugba Taskaya. A Hadoop solution for ballistic image analysis and recognition. In: :836–842IEEE; 2011.
9. ISO/IEC/IEEE 29119-1:2013 - ISO/IEC/IEEE International Standard for Software and systems engineering - Software testing - Part 1: Concepts and definitions. 2013:1–64.
10. Bertolino Antonia. Software Testing Research: Achievements , Challenges , Dreams. In: :85–103; 2007.

11. Bertolino Antonia, Inverardi Paola, Muccini Henry. *Software architecture-based analysis and testing: A look into achievements and future challenges*. 2013.

12. NewVantage Partners LLC . *Big Data Executive Survey 2016 An Update on the Adoption of Big Data in the Fortune 1000*. 2016.

13. Gartner. *How to Take a First Step to Advanced Analytics*. 2015.

14. Xerox. *Big Data in Western Europe Today*. 2015.

15. Cai Li, Zhu Yangyong. The Challenges of Data Quality and Data Quality Assessment in the Big Data Era. *Data Science Journal*. 2015;14:2.

16. Marr Bernard. *Where Big Data Projects Fail. http://www.forbes.com/sites/bernardmarr/2015/03/17/where-big-data-projects-fail/*. Accessed: December 2017.

17. Pure Storage. *BIG DATA'S BIG FAILURE: The struggles businesses face in accessing the information they need*. 2015.

18. Capgemini Consulting. *Big Data survey*. 2014.

19. Marx Vivien. Biology: The big challenges of big data. *Nature*. 2013;498(7453):255–260.

20. Bachlechner Daniel, Leimbach Timo. Big data challenges: Impact, potential responses and research needs. In IEEE International Conference on Emerging Technologies and Innovative Business Practices for the Transformation of Societies (EmergiTech): :257–264; 2016.

21. Kitchenham Barbara. Procedures for performing systematic reviews. *Keele, UK, Keele University*. 2004;33(TR/SE-0401):28.

22. Kitchenham Barbara, Charters S. Guidelines for performing Systematic Literature Reviews in Software Engineering. *Engineering*. 2007;2:1051.

23. Petersen Kai, Feldt Robert, Mujtaba Shahid, Mattsson Michael. Systematic mapping studies in software engineering. *EASE'08 Proceedings of the 12th international conference on Evaluation and Assessment in Software Engineering*. 2008;68–77.

24. Sharma Megha, Hasteer Nitasha, Tuli Anupriya, Bansal Abhay. Investigating the inclinations of research and practices in Hadoop: A systematic review. In Proceedings of the 5th International Conference on Confluence 2014: The Next Generation Information Technology Summit: 227–231; 2014.

25. Camargo Luiz Carlos, Vergilio Silvia Regina. Testing MapReduce Programs: A Mapping Study. In 32nd International Conference of the Chilean Computer Science Society (SCCC): 85–89; 2013.

26. Ferrera Pedro, De Prado Ivan, Palacios Eric, Fernandez-Marquez Jose Luis, Di MarzoÂăSerugendo Giovanna. Tuple MapReduce and Pangool: an associated implementation. *Knowledge and Information Systems*. 2014;41(2):531–557.

27. Lin Jimmy. Mapreduce is Good Enough? If All You Have is a Hammer, Throw Away Everything That's Not a Nail!. *Big Data*. 2013;1(1):28–37.

28. Babu Shivnath. Towards automatic optimization of MapReduce programs. *SoCC '10: Proceedings of the 1st ACM symposium on Cloud computing*. 2010; 137–142.

29. Vishwanath Kashi Venkatesh, Nagappan Nachiappan. Characterizing Cloud Computing Hardware Reliability. *Proceedings of the 1st ACM symposium on Cloud computing - SoCC '10*. 2010; 193.

30. JUnit: a simple framework to write repeatable tests. http://junit.org. Accessed: December 2017.

31. Apache MRUnit: Java library that helps developers unit test Apache Hadoop map reduce job. http://mrunit.apache.org. Accessed: December 2017.

32. Minicluster: Apache hadoop cluster in memory for testing. https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/CLIMiniCluster.html. Accessed: December 2017.

33. Herriot: Large-scale automated test framework. https://wiki.apache.org/hadoop/HowToUseSystemTestFramework. Accessed: December 2017.

34. Anarchy Ape: Fault injection tool for Hadoop cluster from Yahoo anarchyape. https://github.com/david78k/anarchyape. Accessed: December 2017.

35. Chaos Monkey: Fault injector. https://github.com/Netflix/SimianArmy/wiki/Chaos-Monkey. Accessed: December 2017.

36. Hadoop Injection Framework. https://hadoop.apache.org/. Accessed: December 2017.

37. Pan Zhongdang, Kosicki Gerald. Framing analysis: An approach to news discourse. *Political Communication*. 1993;10(1):55–75.

38. Hart Geoff. The Five W's: An Old Tool for the New Task of Audience Analysis. *Technical Communication*. 1996;43(2):139–145.

39. Kipling Rudyard. *Just so stories*. Macmillan and Co; 1902.

40. Jia Changjiang, Yu Yuen Tak. Using the 5W+1H Model in Reporting Systematic Literature Review: A Case Study on Software Testing for Cloud Computing. In 13th International Conference on Quality Software (QSIC):222–229; 2013.

41. Jia Changjiang, Cai Yan, Yu Yuen Tak, Tse T.H.. 5W+1H pattern: A perspective of systematic mapping studies and a case study on cloud software testing. *Journal of Systems and Software*. 2016;116:206–219.

42. ISI/IEC 25010:2011. *Software Process: Improvement and Practice*. 2011;2(Resolution 937):1–25.

43. *ISO/IEC 9126-1: Software Process: Improvement and Practice*. 2001.

44. DBLP - digital bibliography & library project. http://dblp.uni-trier.de. Accessed: December 2017.

45. Thomson reuters. http://thomsonreuters.com. Accessed: December 2017.

46. CORE - Computing research and education. http://www.core.edu.au. Accessed: December 2017.

47. IEEE Xplore Digital Library. http://ieeexplore.ieee.org. Accessed: December 2017.

48. ACM Digital Library. http://dl.acm.org. Accessed: December 2017.

49. Scopus. http://www.scopus.com. Accessed: December 2017.

50. EI compendex. https://www.engineeringvillage.com/. Accessed: December 2017.

51. ISI web of science. https://www.accesowok.fecyt.es. Accessed: December 2017.

52. Palacios Marcos, García-Fanjul José, Tuya Javier. Testing in Service Oriented Architectures with dynamic binding: A mapping study. *Information and Software Technology*. 2011;53(3):171–189.

53. Brereton Pearl, Kitchenham Barbara A., Budgen David, Turner Mark, Khalil Mohamed. Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software*. 2007;80(4):571–583.

54. Pan Xinghao, Tan Jiaqi, Kavulya Soila, Gandhi Rajeev, Narasimhan Priya. Ganesha: BlackBox Diagnosis of MapReduce Systems. *SIGMETRICS Perform. Eval. Rev.*. 2010;37(3):8–13.

55. Joshi Pallavi, Gunawi Haryadi S., Sen Koushik. PREFAIL: A Programmable Tool for Multiple-Failure Injection. *ACM SIGPLAN Notices*. 2011;46(10):171.

56. Tilley Scott, Parveen Tauhida. HadoopUnit: Test Execution in the Cloud. In Software Testing in the Cloud: 2012 (pp. 37–53).

57. Saleh Iman, Nagi Khaled. HadoopMutator: A Cloud-Based Mutation Testing Framework. *14th International Conference on Software Reuse, ICSR 2015*. 2014;172–187.

58. ISO/IEC/IEEE 29119-4:2015 - ISO/IEC/IEEE International Standard for Software and systems engineering - Software testing - Part 4: Test techniques. 2015;1–149.

59. *ISO/IEC 12207:2008 Systems and software engineering - Software life cycle processes*. 2008.

60. *Foundation Level Syllabus Reference, International Software Testing Qualifications Board (ISTQB) Std*. 2011.

61. Shaw Mary. Writing Good Software Engineering Research Papers. In Proceedings of the 25th International Conference on Software Engineering: 726–736; 2003.

62. Cruzes Daniela S., Dyb Tore. Research synthesis in software engineering: A tertiary study. In Information and Software Technology: 440–455; 2011.

63. Cruzes Daniela S., Dyba Tore. Recommended Steps for Thematic Synthesis in Software Engineering. *International Symposium on Empirical Software Engineering and Measurement*. 2011;(7491):275–284.

64. Noblit George W., Hare R. Dwight RD. *Meta-ethnography: Synthesizing qualitative studies*. Qualitative research methods series Newbury Park. 1988.

65. Strauss A, Corbin J. Basics of qualitative research: grounded theory procedure and techniques. *Qualitative Sociology*. 1990;13(1):3–21.

66. Da Silva Fabio Q B, Cruz Shirley S J O, Gouveia Tatiana B., Capretz Luiz Fernando. Using meta-ethnography to synthesize research: A worked example of the relations between personality and software team processes. In International Symposium on Empirical Software Engineering and Measurement: 153–162; 2013.

67. Thangaraj Muthuraman, Anuradha Subramanian. State of art in testing for big data. In IEEE International Conference on Computational Intelligence and Computing Research; 2016.

68. Mittal Akhil. Trustworthiness of Big Data. *International Journal of Computer Applications*. 2013;80(9):35–40.

69. Gudipati Mahesh, Rao Shanthi, Mohan Naju D, Kumar Gajja Naveen. *Big Data: Testing Approach to Overcome Quality Challenges*. Big Data: Challenges and Opportunities. (11):65–72. 2013.

70. Nachiyappan S., Justus S. Getting ready for BigData testing: A practitioner's perception. In 4th International Conference on Computing, Communications and Networking Technologies; 2013.

71. Liu Z. Research of performance test technology for big data applications. *IEEE International Conference on Information and Automation, ICIA 2014*. 2014;53–58.

72. Song Ge, Meng Zide, Huet Fabrice, Magoules Frederic, Yu Lei, Lin Xuelian. A Hadoop MapReduce performance prediction method. In IEEE International Conference on High Performance Computing and Communications, and IEEE International Conference on Embedded and Ubiquitous Computing:820–825; 2014.

73. Yin Jinsong, Qiao Yuanyuan. Performance modeling and optimization of MapReduce programs. In IEEE 3rd International Conference on Cloud Computing and Intelligence Systems:180–186; 2014.

74. Yang Xiao, Sun Jianling. An analytical performance model of MapReduce. *IEEE International Conference on Cloud Computing and Intelligence Systems*. 2011; 306–310.

75. Cheng Sheng-Tzong, Wang Hsi-Chuan, Chen Yin-Jun, Chen Chen-Fei. Performance Analysis Using Petri Net Based MapReduce Model in Heterogeneous Clusters. In Advances in Web-Based Learning: 170–179 Springer, Berlin, Heidelberg; 2015.

76. Jin Hui, Qiao Kan, Sun Xian He, Li Ying. Performance under failures of mapReduce applications. In IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing:608–609; 2011.

77. Castiglione Aniello, Gribaudo Marco, Iacono Mauro, Palmieri Francesco. Exploiting mean field analysis to model performances of big data architectures. *Future Generation Computer Systems*. 2014;37:203–211.

78. Xu Lijie. MapReduce Framework Optimization via Performance Modeling. In 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum: 2506–2509; 2012.

79. Zhang Zhuoyao, Cherkasova Ludmila, Loo Boon Thau. Parameterizable benchmarking framework for designing a mapreduce performance model. *Concurrency Computation Practice and Experience*. 2014;26(12):2005–2026.

80. Vianna Emanuel, Comarela Giovanni, Pontes Tatiana, et al. Analytical performance models for mapreduce workloads. *International Journal of Parallel Programming*. 2013;41(4):495–525.

81. Han Jungkyu, Ishii Masakuni, Makino Hiroyuki. A Hadoop performance model for multi-rack clusters. In 5th International Conference on Computer Science and Information Technology: 265–274; 2013.

82. Ishii Masakuni, Han Jungkyu, Makino Hiroyuki. Design and performance evaluation for Hadoop clusters on virtualized environment. In International Conference on Information Networking: 244–249; 2013.

83. Cui Xiaolong, Lin Xuelian, Hu Chunming, Zhang Richong, Wang Chengzhang. Modeling the Performance of MapReduce under Resource Contentions and Task Failures. *IEEE 5th International Conference on Cloud Computing Technology and Science*. 2013;(1):158–163.

84. Ahmed Sarker Tanzir, Loguinov Dmitri. On the performance of MapReduce: A stochastic approach. In IEEE International Conference on Big Data: 49–54; 2014.

85. Rahman Md., Lu Xiaoyi, Islam Nusrat Sharmin, Panda Dhabaleswar K.. Performance Modeling for RDMA-Enhanced Hadoop MapReduce. *43rd International Conference on Parallel Processing;* 2014:50–59.

86. Wang Kewen, Khan Mohammad Maifi Hasan. Performance prediction for apache spark platform. In IEEE 17th International Conference on High Performance Computing and Communications, IEEE 7th International Symposium on Cyberspace Safety and Security and IEEE 12th International Conference on Embedded Software and Systems: 166–173; 2015.

87. Zhang Zhuoyao, Cherkasova L, Loo B T. Performance Modeling of MapReduce Jobs in Heterogeneous Cloud Environments. *Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on*. 2013;:839–846.

88. Fan Yuanquan, Wu Weiguo, Xu Yunlong, et al. Performance prediction model in heterogeneous MapReduce environments. In IEEE International Conference on Computer and Information Technology: 240–245; 2014.

89. Wu Xing, Liu Yan, Gorton Ian. Exploring Performance Models of Hadoop Applications on Cloud Architecture. In 11th International ACM SIGSOFT Conference on Quality of Software Architectures: 93–101; 2015.

90. Mytilinis Ioannis, Tsoumakos Dimitrios, Kantere Verena, Nanos Anastassios, Koziris Nectarios. I/O Performance Modeling for Big Data Applications over Cloud Infrastructures. In IEEE International Conference on Cloud Engineering: 201–206; 2015.

91. Khan Mukhtaj, Jin Yong, Li Maozhen, Xiang Yang, Jiang Changjun. Hadoop Performance Modeling for Job Estimation and Resource Provisioning. *IEEE Transactions on Parallel and Distributed Systems*. 2016;27(2):441–454.

92. Ohnaga Hayata, Aida Kento, Abdul-Rahman Omar. Performance of Hadoop Application on Hybrid Cloud. In International Conference on Cloud Computing Research and Innovation (ICCCRI): 130–138; 2015.

93. Apache Hive: data warehouse software facilitates reading, writing, and managing large datasets residing in distributed storage using SQL. https://hive.apache.org/. Accessed: December 2017.

94. Apache Pig: platform for analyzing large data sets that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs. https://pig.apache.org/. Accessed: December 2017.

95. Sangroya Amit, Singhal Rekha. Performance Assurance Model for HiveQL on Large Data Volume. In 22nd IEEE International Conference on High Performance Computing Workshops: 26–33; 2016.

96. Mogrovejo Renato Javier M., Monteiro José Maria, Machado Javam C., Viana Carlos Juliano M., Lifschitz Sérgio. Towards a Statistical Evaluation of PigLatin Joins. *Journal of Information and Data Management*. 2013;4(3):483.

97. Zhang Zhuoyao, Cherkasova Ludmila, Verma Abhishek, Loo Boon Thau. Performance Modeling and Optimization of Deadline-Driven Pig Programs. *ACM Transactions on Autonomous and Adaptive Systems*. 2013;8(3):1–28.

98. Bian Zhaojuan, Wang Kebing, Wang Zhihong, et al. Simulating Big Data Clusters for System Planning, Evaluation, and Optimization. *2014 43rd International Conference on Parallel Processing*. 2014;391–400.

99. Wang G. Evaluating Mapreduce system performance: A Simulation approach. PhD thesis Virginia Polytechnic Institute and State University. 2012.

100. Chauhan Jagmohan, Makaroff Dwight, Grassmann Winfried. Simulation and performance evaluation of the hadoop capacity scheduler. In 24th Annual International Conference on Computer Science and Software Engineering: 163–177; 2014.

101. Barbierato E, Gribaudo M, Iacono M. A performance modeling language for big data architectures. *27th European Conference on Modelling and Simulation*. 511-517. 2013.

102. Dai Jinquan, Huang Jie, Huang Shengsheng, Huang Bo, Liu Yan. HiTune: dataflow-based performance analysis for big data cloud. *Proceeding USENIXATC'11 Proceedings of the 2011 USENIX conference on USENIX annual technical conference.* 2011;7.

103. Bautista Villalpando Luis, April Alain, Abran Alain. Performance analysis model for big data applications in cloud computing. *Journal of Cloud Computing: Advances, Systems and Applications.* 2014;3(1):19–38.

104. Rabkin Ariel, Katz Randy Howard. How hadoop clusters break. *IEEE Software.* 2013;30(4):88–94.

105. Ren Kai, Kwon YongChul, Balazinska Magdalena, Howe Bill. Hadoop's adolescence. *Proceedings of the VLDB Endowment.* 2013;6(10):853–864.

106. Li Sihan, Zhou Hucheng, Lin Haoxiang, et al. A characteristic study on failures of production distributed data-parallel programs. In 35th International Conference on Software Engineering (ICSE): 963–972; 2013.

107. Kavulya Soila, Tan Jiaqi, Gandhi Rajeev, Narasimhan Priya. An Analysis of Traces from a Production MapReduce Cluster. In 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing: 94–103; 2010.

108. Xiao Tian, Zhang Jiaxing, Zhou Hucheng, et al. Nondeterminism in MapReduce considered harmful? an empirical study on non-commutative aggregators in MapReduce programs. In Companion Proceedings of the 36th International Conference on Software Engineering: 44–53; 2014; New York, New York, USA.

109. Camargo Luiz C, Vergilio Silvia R. Classificacao de Defeitos para Programas MapReduce: Resultados de um Estudo Empirico. 2013.

110. Moran Jesus, Riva Claudio, Tuya Javier. MRTree: Functional Testing Based on MapReduce's Execution Behaviour. In International Conference on Future Internet of Things and Cloud: 379–384; 2014.

111. Chen Yu-Fang, Hong Chih-Duo, Sinha Nishant, Wang Bow-Yaw. Commutativity of Reducers. In Springer Berlin Heidelberg 2015 (pp. 131–146).

112. Csallner Christoph, Fegaras Leonidas, Li Chengkai. New Ideas Track: Testing Mapreduce-style Programs. *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering.* 2011;504–507.

113. Xu Zhihong, Hirzel Martin, Rothermel Gregg, Wu Kun Lung. Testing properties of dataflow program operators. In 28th IEEE/ACM International Conference on Automated Software Engineering: 103–113; 2013.

114. Marynowski João Eugenio, Santin Altair Olivo, Pimentel Andrey Ricardo. Method for testing the fault tolerance of MapReduce frameworks. *Computer Networks.* 2015;86:1–13.

115. Faghri Faraz, Bazarbayev Sobir, Overholt Mark, Farivar Reza, Campbell Roy H., Sanders William H.. Failure scenario as a service (FSaaS) for Hadoop clusters. *Proceedings of the Workshop on Secure and Dependable Middleware for Cloud Monitoring and Management - SDMCMM '12.* 2012;:1–6.

116. Morán Jesús, Riva Claudio, Tuya Javier. Testing data transformations in MapReduce programs. In 6th International Workshop on Automating Test Case Design, Selection and Evaluation: 20–25ACM Press; 2015; New York, New York, USA.

117. Mattos Antonio Junior. Test data generation for testing mapreduce systems. PhD thesis Federal University of Paraná. 2011.

118. Li Nan, Escalona Anthony, Guo Yun, Offutt Jeff. A Scalable Big Data Test Framework. In IEEE 8th International Conference on Software Testing, Verification and Validation (ICST): 1–2; 2015.

119. Li Kaituo, Reichenbach Christoph, Smaragdakis Yannis, Diao Yanlei, Csallner Christoph. SEDGE: Symbolic example data generation for dataflow programs. In 28th IEEE/ACM International Conference on Automated Software Engineering: 235–245; 2013.

120. Dörre Jens, Apel Sven, Lengauer Christian. Static type checking of Hadoop MapReduce programs. *Proceedings of the second international workshop on MapReduce and its applications - MapReduce '11.* 2011;:17.

121. Rabkin Ariel, Katz Randy. Static extraction of program configuration options. *2011 33rd International Conference on Software Engineering (ICSE).* 2011;131–140.

122. Ono Kosuke, Hirai Yoichi, Tanabe Yoshinori, Noda Natsuko, Hagiya Masami. Using Coq in specification and program extraction of Hadoop MapReduce applications. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): 350–365; 2011.

AUTHOR BIOGRAPHY

Jesús Morán received his B.Sc. degree in Computer Engineering in 2012 and an M.Sc. in Computer Engineering in 2014 from the University of Oviedo, Spain. He is currently a PhD candidate at the University of Oviedo. His research interests include software testing, Big Data technologies and distributed programming.

Claudio de la Riva is an Assistant Professor at the University of Oviedo. He is a member of the Software Engineering Research Group (GIIS, giis.uniovi.es). He obtained his PhD in Computing from the University of Oviedo. His research interests include software verification and validation and software testing, mainly focused on testing database applications and services. He is a member of ACM.

Javier Tuya is a Professor at the University of Oviedo, Spain, where he is the research leader of the Software Engineering Research Group. He received his PhD in Engineering from the University of Oviedo in 1995. He is the Director of the Indra-Uniovi Chair, member of the ISO/IEC JTC1/SC7/WG26 working group for the recent ISO/IEC/IEEE 29119 Software Testing standard and convener of the corresponding UNE National Body working group. His research interests in software engineering include verification & validation and software testing for database applications and services. He is a member of the IEEE, IEEE Computer Society, ACM and the Association for Software Testing (AST).