DOI 10.12694/scpe.v19i3.1402
SCALABLE COMPUTING: PRACTICE AND EXPERIENCE ISSN 1895-1767
Volume 19, Number 3, pp. 301–307. http://www.scpe.org

EVALUATING THE SCALABILITY OF BIG DATA FRAMEWORKS

DAVID SANCHEZ[*¶], OSWALDO SOLARTE[†¶], VICTOR BUCHELI[‡¶] AND HUGO ORDONEZ[§]

Abstract. The aim of this paper is to present a method based on the isoefficiency model for assessing the scalability in big data environments. The programs word count and sort were implemented and compared in Hadoop and Spark. The results confirm that isoefficiency presented a linear growth as the size of the data sets was increased. They were checked experimentally to ensure that the evaluated frameworks are scalable and a sublinear function was obtained. This paper discusses how scalability in big data is governed by a constant of scalability (β).

Key words: Scalability, Isoefficiency, Big Data, Hadoop, Spark, MapReduce.

AMS subject classifications. 68M14

1. Introduction. The amount of data created globally is increasing sharply. Every day 2.5 Exabytes of data are generated [23]. Big data refers to data sets that cannot be managed using traditional database systems and techniques. This data may come from diverse sources, such as; emails, videos, images, logs, online transactions, search queries, health records or social networking interactions [17, 23, 6]. Moreover, the Internet of Things (IoT) technology has promoted the creation of several systems that generate enormous quantities of data such as smartphones and sensors. Big data has received increased attention in recent years from researchers and industrial community. This term has five defining characteristics: volume, variety, velocity, veracity and value [6]. Volume refers to the magnitude of data sets (multiple terabytes and petabytes) [6]. Variety refers to the structural heterogeneity in a data set (structured, semi-structured, and unstructured data). Text, images, audio, and video are examples of unstructured data, which do not have the structure required to be analyzed by traditional database systems. Velocity refers to the rate at which data is generated and the speed at which it should be analyzed. As mentioned above, the proliferation of digital devices, such as, smartphones and sensors has led to an unprecedented rate of data creation and is driving a growing need for real-time analytics. Veracity is linked to the unreliability inherent in some sources of data. For example, user feelings in social media are uncertain since they pertain to human judgment. Value is the importance of information extracted from data. The value obtained from big data sets can contribute to improving productivity and competitiveness and create enormous benefits for consumers [2].

The characteristics of big data increase the complexity of storing, processing and analyzing information and pose challenges for obtaining the real value of big data. Several technologies have been proposed for addressing this complexity, such as MapReduce, Hadoop and Spark. MapReduce is a paradigm that parallelizes and executes a program on different machines [4]. Hadoop and Spark are frameworks for non-relational storage and distributed processing. These technologies enable processing semi-structured and unstructured data. Large data sets can be stored and processed in a distributed way [3].

In this scenario, some desirable characteristics of big data environments are scalability, high performance, parallel processing, high fault tolerance and low cost. The scalability is associated with the efficient management of resources and measures the ability of a system to react and adapt to changes in the volume of data without degrading its performance [1]. For example, for opinion mining in twitter, it may be necessary to add new resources continuously, as the number of tweets increases. The models and technologies developed for big data environments have so far been shown to be scalable. Nevertheless, some questions are still only partially answered. How should scalability in big data environments be measured? Is it possible to build a mathematical model of scalability of the big data environment?

In this work, Isoefficiency is introduced as a standard measure of scalability. A model for the evaluation of the scalability of big data environments is also presented. This model was validated using Hadoop and Spark frameworks, for this purpose. Wordcount and Sort programs were developed following the MapReduce paradigm. The experimental evaluation was carried out using data sets whose sizes range from 1Mb to 30Gb (10 data sets). The results show that the Isoefficiency model for each framework is linear and for each program, the growth behavior is sublinear. Thus, the

[*](jose.d.sanchez@correounivalle.edu.co)
[†](oswaldo.solarte@correounivalle.edu.co)
[‡](victor.bucheli@correounivalle.edu.co)
[§]Universidad San Buenaventura, Cali, Colombia,(haordonez@usbcali.edu.co)
[¶]Universidad del Valle, Cali, Colombia

isoefficiency increases as the size of the data set increases, but the isoefficiency growth is low while the file size growth is accelerated. The results show that the tested frameworks are scalable and follow the sublinear behavior of the Isoefficiency function, $Y(s) = \beta X(s)$ where $\beta \approx [0.47 - 0.85] < 1$. Furthermore, scalability is governed by a constant of scalability β.

The paper proceeds as follows. The next section describes related works. Section 3 discusses the experimental calculation of Isoefficiency, as well as the proposed model. Section 4 discusses the results of the evaluation and section 5 provides a conclusion.

2. Related work. Related works are organized into two groups: Isoefficiency based measures and comparison of big data frameworks. Regarding Isoefficiency based measures, several studies have been carried out in different distributed and parallel systems [24, 21, 27]. Moreover, other measures based on Isoefficiency have been proposed for the analysis of the scalability. Firstly, E-differentiation is a measure of scalability in parallel systems, which analyzes the change in the efficiency of a system based on two variables; workload and processors [15]. This measure makes it possible to analyze scalability based on the workload and not only on the number of processors, as is the case in Isoefficiency. Secondly, the Isoefficiency maps allow us to understand the performance of parallel systems, and the measurement is based on techniques such as temperature maps and pressure maps. Isoefficiency maps include several parameters of the parallel performance in two-dimensional planes, which allows considering additional parameters, such as, the communication between parallel processes [5].

Regarding the comparison of frameworks, the infrastructure, workload and information type are usually analyzed [4]. In some approaches, a single cluster configuration is implemented, and the size of the data set varies (this latter approach is used on the present study). This group of studies aims to measure the performance of the framework for large data sets based on the runtime [25, 4, 14].

On the other hand, some studies perform the comparison between structured and unstructured data sets [14], while other authors compare the performance between Spark and Hadoop, using different algorithms [11, 13]. In [25] a comparison of 5 frameworks was carried out. To this end, different SQL queries were executed, and some measures such as response time and fault tolerance were analyzed during the query. Conversely, [9] evaluates the scalability of Hadoop and Spark frameworks based on execution time with relatively small data sets (less than 1.5Gb).

The work shown in [19] presents an Isoefficiency study in the context of big Data. In this work the Isoefficiency is theoretically analyzed. Some variables, such as, calculation costs, synchronization and communication are studied in MapReduce applications, modeled as bulk synchronous parallel tasks. The experimental evaluation shows that the speedup follows a linear trend (100TB data set on 10,000 machines). Finally, other works are devoted to evaluating pattern recognition algorithms based on the MapReduce paradigm [16, 20, 12].

None of the studies reviewed above develop a mathematical model that describes the scalability of big data Systems, nor do they describe the behavior of the relationship between Isoefficiency and workload. Neither of the works is focused on identifying a constant of Isoefficiency that describes the behavior of the scalability in a System. Finally, the model proposed here can be applied to other experimental evaluations (frameworks, programs and data sizes) due to its three layer architecture (processing, storage, and evaluation), seen in Fig. 2.1.

3. Proposed method for measuring the scalability of big data environments. Algorithms are described based on the MapReduce paradigm. Isoefficiency is then calculated according to the size of the data set. Lastly, a model is developed, and the β constant is computed.

3.1. Modeling of the programs Word Count and Sort. The MapReduce model consists of two functions, Map and Reduce. *Map* receives a series of key/value pairs and generates a new intermediate pair key/value that is sent to the *Reduce* function. *Reduce* function combines all the values according to the key [4]. To describe an algorithm using MapReduce, it is necessary to define the Map and Reduce functions. However, there are few models, guidelines or methods that can be found to translate algorithms to the MapReduce paradigm. Therefore, some examples of the programs *Word Count* and *Sort* are presented below.

The Word Count program evaluates the number of appearances of each word in a data set. The *map* function receives a set of key/value pairs, where the key is the name of the document to be processed, and the value is the content of the document. For each word found in the content, a key/value pair is returned, where the key is the word, and the value is 1. The *Reduce* function receives this list and iterates on each element of the list of ones, adding each element. Finally, a key/value pair is returned with the word and the total number of appearances for each word.

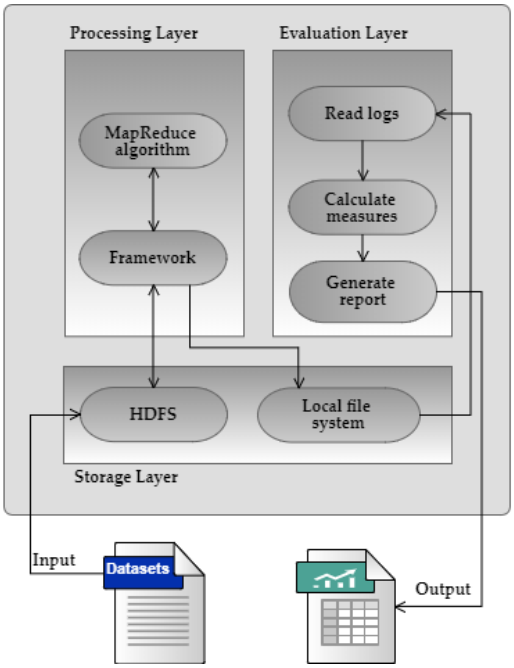

FIG. 2.1. *Architecture for the Proposed Method of the Measuring the Scalability in Big Data Environments*

The Sort program computes a sorted list in alphabetical order of the words from data sets. The *map* function receives a list of key/value pairs, where the key is the name of the document to be processed, and the value is the content of the document. For each word in the content, a key/value pair is returned, where the key is the word, and the value is 1. The *Reduce* function receives a list of key/value pairs and returns the received key, the value is not taken into account in the sort program.

3.2. Measuring the Isoefficiency. The measurement is carried out according to the size of the data set and the consumed resources. Figure 2.1 shows the architecture of the approach that consists of 3 layers; processing, storage, and evaluation. The input is a set of flat files stored in HDFS(Hadoop Distributed File System). The processing layer includes the MapReduce based algorithm for evaluation. The storage layer consists of the distributed file system of the framework to be analyzed (Hadoop and Spark), and the local file system. The distributed file system stores the input data, and the data returned by the algorithms in the processing layer. The local file system stores the logs generated by the frameworks during the processing. The logs' files contain data on the processing time, read data, returned data, and used resources among others. Finally, in the evaluation layer, the logs' files are used to calculate all measurements defined for the evaluation. A report is also created to display the results of each algorithm for the data set's different sizes.

The Evaluation was carried out using the following three steps: experimental design, configuration of the framework's infrastructure, and definition of the metrics.

3.2.1. Experimental design. Two algorithms (Word Count and Sort) are translated to the MapReduce model to run in Hadoop and Spark frameworks on a cluster with four virtual machines. The efficiency and speedup are used to calculate the Isoefficiency.

3.2.2. Configuration of the framework infrastructure. Hadoop is a framework for storing and processing large data sets. Hadoop is based on a master multislave architecture. The main component of Hadoop is the Hadoop distributed file system (HDFS). HDFS stores large data sets in distributed nodes, offering data redundancy and high-performance access. Another component is YARN that manages the resources and the MapReduce tasks. Hadoop also has a library to implement applications following the MapReduce model [18, 22, 26].

For its part, Spark is a fast and general-purpose framework for processing large amounts of data. The main feature of Spark is its memory processing. Spark is a flexible framework that can be integrated with various data storage tools,

TABLE 3.1
Hardware Description Nodes

Hardware	
Processors number	8
Processors model	Intel Xeon E5540 2.53 Ghz
Cache size	8 Mb
RAM	8 Gb
Hard drive size	200 Gb
Software	
Operating system	Ubuntu 16.04.1 LTS
Big data frameworks	Spark 2.0.2, Hadoop 2.7.3
Others	Java 8

such as, Hadoop, Cassandra, HBase, among others. Additionally, Spark can be integrated with resource managers, such as, YARN and Apache Mesos [28, 29, 30].

For the experiments, a cluster with four virtual machines connected to the same network was deployed. The cluster consists of a master node and three slave nodes. Each node has the hardware and software specifications shown in Table 3.1.

Figure 3.1 depicts the cluster architecture. The data set includes scientific papers from the ISI Web of Knowledge. The data set contains metadata, including title, abstract, authors, etc. However, the analysis was done on the complete paper. The data set is composed of flat files with a total size of 15 Gb. Different samples were taken with different sizes: 100Kb, 1Mb, 13Mb, 100Mb, 512Mb, 1Gb, 5Gb, 10Gb, 20Gb, 30Gb. These data sets were stored in HDFS, and also in a Google Drive repository, available at:
https://drive.google.com/drive/folders/0B0Zl7SmxErLNMm4tZVdyRG8ya1E?usp=sha

3.2.3. Definition of the Metrics. Execution time (T) measures the time required to run a program in parallel. T is calculated as $T = TF - TI$, TF is the time when the program ends, and TI the time when the program starts [7].

Speedup (S) measures the performance gain obtained by running a program in parallel, compared to the sequential execution. This metric is defined as the ratio between the time used to solve a problem in a single processor, and the time necessary to solve the same problem in a parallel system with p identical processors. Thus, TS is the runtime of the program executed sequentially, and T the runtime of the program executed in parallel [7]. Speedup is defined by $S = TS/T$.

Efficiency is defined as the ratio between Speedup (S) and the number of processors P [7], mathematically it is defined as $E = S/P$. Furthermore, efficiency is the fraction of the time that each processor is used. In an ideal scenario, the speedup (S) is equal to the number of processors and the efficiency is equal to 1, but generally, the speedup is less than the number of processors and efficiency is between 0 and 1.

Isoefficiency (W) is a measure of the scalability in parallel systems. W measures the relationship between the workload and the number of processors. Here the Isoefficiency is defined by Eq. 3.1 as the rate at which the number of

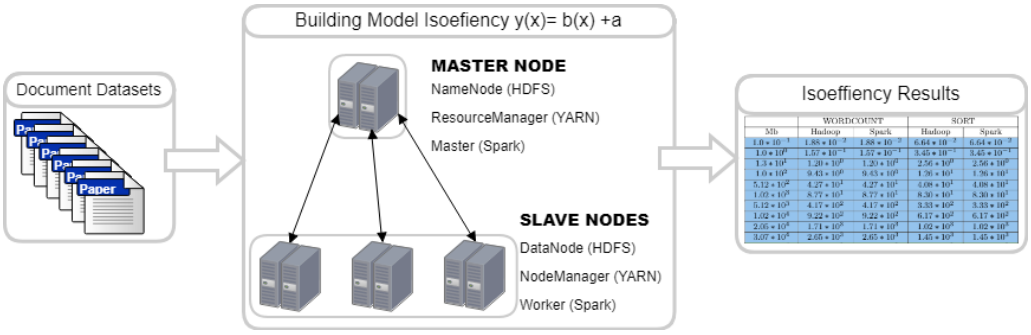

FIG. 3.1. *Hardware Architecture for the experiments*

processors must increase in order to keep the efficiency fixed as the workload increases [8, 10].

$$W = K * T_0(w, P) \tag{3.1}$$

where $K = E/1 - E$, that is the scalability constant, and $T_0 \approx T * (P - s)$ which is calculated experimentally.

3.3. The Isoefficiency model. The Isoefficiency of two programs executed in two different frameworks is calculated. This Isoefficiency is evaluated for different data set sizes(sd). Equation 3.2 represents the fit equation and is computed through the method of least squares.

$$Y(sd) = \beta X(sd) \tag{3.2}$$

where $Y(sd)$ is the Isoefficiency, $X(sd)$ is the data size evaluated in each experiment. β is the system scalability coefficient, which is estimated by linear regression. The model is represented by a linear equation, where the least squares adjust model parameters β to get a best fit. Therefore, the model constructs a model with a structure that fits the y_i for successive x values, using the parameters β; For the property RSquared, the computation of the total sum of squares is mean adjusted, other properties related to the sum of squared errors are included such as Standard error, T-statistic and P-value. The model was implemented in the Mathematica software package.

To study the relationship between the program input and the Isoefficiency, a model based on Eq. (3.2) was built. In each case, β represents the proportion in which the Isoefficiency grows according to the input. This linear function represents the scalability of the system. In most cases β grows sub-linearly depending on the data size (sd). The positive slope β less than 1 means sub-linear behavior, that is, the value on the x-axis is higher than the value on the y-axis. Thus, the sublinear growth in the Isoefficiency indicates that the system is scalable since the increase of the resources is less than the increase of the workload.

4. Results. Table 4.1 shows the experimental results. Blue displays the experimental data, and green the calculated Isoefficiency of Hadoop and Spark frameworks (for Word Count and Sort programs). It can be seen that for a particular size of data set, the Isoefficiency values are similar for both frameworks. The behavior is the same when the size of the data set is less than 1 Gb. On the other hand, when the size of the data sets is greater than 5 Gb, the behavior is different. The Word Count scalability is less than Sort scalability due to the higher consumption of resources.

Table 4.2 shows the statistical model obtained from the experimental evaluation of the two programs. The table presents the isoefficiency function $(m(x) + b)$, the function slope (m estimate), the standard error and the p-value, and the R2.

The results show that, the standard error values and the t-statistics are similar for both programs. A minimum square error of statistical significance 99% is also achieved. It can be inferred that the results provide reliability in the adjustment of the model. In this sense, the results are confirmed because the positive slope less than 1 indicates a sub-linear behavior. This means, that the distance between each point on the x-axis is higher than the distance between each point on the y-axis. The sub-linear growth of the Isoefficiency indicates that the system is scalable, since the Isoefficiency increases less than the workload (size of the data set) [8]. Thus, it is confirmed that the frameworks are scalable (for the word count and sort programs) since a slight increase in resources is required to keep the efficiency.

5. Conclusions and Future works. An experimental evaluation of the Isoefficiency was carried out to analyze the scalability of two big data frameworks: Hadoop and Spark. The function of Isoefficiency has been obtained according to the size of the data sets for two MapReduce programs (Wordcount and Sort). The function regulates the Isoefficiency as a linear function. It was shown that the Isoefficiency of the two frameworks was the same when executing the programs based on the MapReduce paradigm. From the Isoefficiency function, a sublinear growth of the Isoefficiency associated with an increase of the data size was evidenced. That is to say, as the data size increases the Isoefficiency of the frameworks increase in smaller proportion. The experiments show that Hadoop and Spark are an excellent choice to store and process large data sets, since the results obtained from Isoefficiency are similar. A future project might implement a MapReduce program for indexing scientific unstructured data using Hadoop, Spark, and Elasticsearch.

TABLE 4.1
Results of Scalability Experimintation and Proposed Model evaluation.

	WORDCOUNT		SORT	
Mb	Hadoop	Spark	Hadoop	Spark
$1.0 * 10^{-1}$	$1.88 * 10^{-2}$	$1.88 * 10^{-2}$	$6.64 * 10^{-2}$	$6.64 * 10^{-2}$
$1.0 * 10^{0}$	$1.57 * 10^{-1}$	$1.57 * 10^{-1}$	$3.45 * 10^{-1}$	$3.45 * 10^{-1}$
$1.3 * 10^{1}$	$1.20 * 10^{0}$	$1.20 * 10^{0}$	$2.56 * 10^{0}$	$2.56 * 10^{0}$
$1.0 * 10^{2}$	$9.43 * 10^{0}$	$9.43 * 10^{0}$	$1.26 * 10^{1}$	$1.26 * 10^{1}$
$5.12 * 10^{2}$	$4.27 * 10^{1}$	$4.27 * 10^{1}$	$4.08 * 10^{1}$	$4.08 * 10^{1}$
$1.02 * 10^{3}$	$8.77 * 10^{1}$	$8.77 * 10^{1}$	$8.30 * 10^{1}$	$8.30 * 10^{1}$
$5.12 * 10^{3}$	$4.17 * 10^{2}$	$4.17 * 10^{2}$	$3.33 * 10^{2}$	$3.33 * 10^{2}$
$1.02 * 10^{4}$	$9.22 * 10^{2}$	$9.22 * 10^{2}$	$6.17 * 10^{2}$	$6.17 * 10^{2}$
$2.05 * 10^{4}$	$1.71 * 10^{3}$	$1.71 * 10^{3}$	$1.02 * 10^{3}$	$1.02 * 10^{3}$
$3.07 * 10^{4}$	$2.65 * 10^{3}$	$2.65 * 10^{3}$	$1.45 * 10^{3}$	$1.45 * 10^{3}$
$5.12 * 10^{4}$	$4.38 * 10^{3}$	$4.38 * 10^{3}$	$2.48 * 10^{3}$	$2.48 * 10^{3}$
$6.14 * 10^{4}$	$5.25 * 10^{3}$	$5.25 * 10^{3}$	$2.97 * 10^{3}$	$2.97 * 10^{3}$
$7.17 * 10^{4}$	$6.13 * 10^{3}$	$6.13 * 10^{3}$	$3.46 * 10^{3}$	$3.46 * 10^{3}$
$8.19 * 10^{4}$	$7.0 * 10^{3}$	$7.0 * 10^{3}$	$3.95 * 10^{3}$	$3.95 * 10^{3}$
$9.22 * 10^{4}$	$7.88 * 10^{3}$	$7.88 * 10^{3}$	$4.44 * 10^{3}$	$4.44 * 10^{3}$
$1.02 * 10^{5}$	$8.75 * 10^{3}$	$8.75 * 10^{3}$	$4.93 * 10^{3}$	$4.93 * 10^{3}$
$1.05 * 10^{6}$	$8.97 * 10^{4}$	$8.97 * 10^{4}$	$5.03 * 10^{4}$	$5.03 * 10^{4}$
$1.07 * 10^{9}$	$9.18 * 10^{7}$	$9.18 * 10^{7}$	$5.14 * 10^{7}$	$5.14 * 10^{7}$
$1.10 * 10^{12}$	$9.40 * 10^{10}$	$9.40 * 10^{10}$	$5.27 * 10^{10}$	$5.27 * 10^{10}$
$1.13 * 10^{15}$	$9.63 * 10^{13}$	$9.63 * 10^{13}$	$5.39 * 10^{13}$	$5.39 * 10^{13}$
$1.15 * 10^{18}$	$9.86 * 10^{16}$	$9.86 * 10^{16}$	$5.52 * 10^{16}$	$5.52 * 10^{16}$

TABLE 4.2
Statistical Model for Scalability

Program/ framework	Equation $Y(t) = \beta X(t)$	β estimated	Standard error	T-statistic	P-value	R^2
Wordcount Hadoop	0.0855x - 0.2874	0.0855	0.0008	110.858	$4.8987 * 10^{-14}$	0.99
Wordcount Spark	0.0855x - 0.2874	0.0855	0.0008	110.858	$4.8987 * 10^{-14}$	0.99
Sort Hadoop	0.0479x + 29.4126	0.0479	0.0015	32.2172	$9.3867 * 10^{-10}$	0.99
Sort Spark	0.0479x + 29.4126	0.0479	0.0015	32.2172	$9.3867 * 10^{-10}$	0.99

REFERENCES

[1] A. BONDI, *Characteristics of Scalability and Their Impact on Performance*, Proceedings of the 2nd international workshop on Software and performance, (2000), pp. 195–203.

[2] M. CHEN, S. MAO, AND Y. LIU, *Big data: A survey*, Mobile Networks and Applications, 19 (2014), pp. 171–209.

[3] C. COSTA AND M. Y. SANTOS, *Big Data: State-of-the-art concepts, techniques, technologies, modeling approaches and research challenges*, IAENG International Journal of Computer Science, (2017).

[4] J. DEAN AND S. GHEMAWAT, *MapReduce: Simplified Data Processing on Large Clusters*, in Proc. of the OSDI - Symp. on Operating Systems Design and Implementation, USENIX, 2004, pp. 137–149.

[5] M. DROZDOWSKI AND L. WIELEBSKI, *Isoefficiency maps for divisible computations*, IEEE Transactions on Parallel and Distributed Systems, 21 (2010), pp. 872–880.

[6] A. GANDOMI AND M. HAIDER, *Beyond the hype: Big data concepts, methods, and analytics*, International Journal of Information Management, (2015).

[7] A. GRAMA, A. GUPTA, G. KARYPIS, AND V. KUMAR, *Introduction to Parallel Computing, Second Edition*, Addison Wesley, second ed., 2003.

[8] A. Y. Grama, A. Gupta, and V. Kumar, *Isoefficiency: Measuring the Scalability of Parallel Algorithms and Architectures*, IEEE Parallel and Distributed Technology, 1 (1993), pp. 12–21.
[9] L. Gu and H. Li, *Memory or time: Performance evaluation for iterative operation on hadoop and spark*, Proceedings - 2013 IEEE International Conference on High Performance Computing and Communications, HPCC 2013 and 2013 IEEE International Conference on Embedded and Ubiquitous Computing, EUC 2013, (2014), pp. 721–727.
[10] V. Kumar, V. N. Rao, and K. Ramesh, *Parallel Depth First Search on the Ring Architecture*, (1988).
[11] B. Kupisz and O. Unold, *Collaborative filtering recommendation algorithm based on Hadoop and Spark*, 2015 IEEE International Conference on Industrial Technology (ICIT), (2015), pp. 1510–1514.
[12] J. Li, D. Li, and Y. Zhang, *Efficient Distributed Data Clustering on Spark*, 2015 IEEE International Conference on Cluster Computing, (2015), pp. 504–505.
[13] X. Lin, P. Wang, and B. Wu, *Log analysis in cloud computing environment with Hadoop and Spark*, 2013 5th IEEE International Conference on Broadband Network & Multimedia Technology, (2013), pp. 273–276.
[14] O. O. Malley and A. C. Murthy, *Winning a 60 Second Dash with a Yellow Elephant Hadoop implementation*, Proceedings of sort benchmark, 1810 (2009), pp. 1–9.
[15] Min Wang, Weiqun Ding, and Hong Lin, *E-differentiation for analyzing scalability of parallel algorithms on parallel architectures*, Proceedings of ICICS, 1997 International Conference on Information, Communications and Signal Processing. Theme: Trends in Information Systems Engineering and Wireless Multimedia Communications (Cat. No.97TH8237), 3 (1997), pp. 1457–1461.
[16] S. Muppidi, *Document Clustering with Map Reduce using Hadoop Framework*, International Journal on Recent and Innovation Trends in Computing and Communication, (2015).
[17] S. Sagiroglu and D. Sinanc, *Big data: A review*, International Conference on Collaboration Technologies and Systems (CTS), (2013), pp. 42–47.
[18] R. D. Schneider, *Hadoop for Dummies Special Edition*, John Wiley & Sons Canada, Ltd., special ed., 2012.
[19] H. Senger, V. Gil-Costa, L. Arantes, C. A. Marcondes, M. Marín, L. M. Sato, and F. A. Da Silva, *BSP cost and scalability analysis for MapReduce operations*, in Concurrency Computation, 2016.
[20] N. Shah and S. Mahajan, *Distributed Document Clustering Using K-Means*, International Journal of Advanced Research in Computer Science and Software Engineering, 4 (2014), pp. 2277–128.
[21] S. Shudler, A. Calotoiu, T. Hoefler, and F. Wolf, *Isoefficiency in Practice: Configuring and Understanding the Performance of Task-based Applications*, Principles and Practice of Parallel Programming (PPoPP), (2017).
[22] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, *The Hadoop distributed file system*, 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies, MSST2010, (2010).
[23] S. Singh and N. Singh, *Big Data analytics*, 2012 International Conference on Communication, Information & Computing Technology (ICCICT), (2012), pp. 1–4.
[24] V. Singh, V. Kumar, G. Agha, and C. Tomlinson, *Efficient algorithms for parallel sorting on mesh multicomputers*, International Journal of Parallel Programming, 20 (1991), pp. 95–131.
[25] P. Wendell, *Big Data Benchmark*, 2014.
[26] T. White, *Hadoop: The definitive guide*, vol. 54, O'Reilly Media, Inc., fourth edi ed., 2015.
[27] T.-R. Yang and H.-X. Lin, *Isoefficiency Analysis of CGLS Algorithm for Parallel Least Squares Problems*, (1997), pp. 452–461.
[28] M. Zaharia, M. Chowdhury, T. Das, and A. Dave, *Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing*, Nsdi, (2012), pp. 2–2.
[29] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, *Spark : Cluster Computing with Working Sets*, HotCloud'10 Proceedings of the 2nd USENIX conference on Hot topics in cloud computing, (2010), p. 10.
[30] M. Zaharia, M. J. Franklin, A. Ghodsi, J. Gonzalez, S. Shenker, I. Stoica, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, and S. Venkataraman, *Apache Spark: a unified engine for big data processing*, Communications of the ACM, 59 (2016), pp. 56–65.

Edited by: Dana Petcu
Received: Jun 2, 2018
Accepted: Sep 1, 2018