Advantages of Giraph over Hadoop in Graph Processing

Cristian L. Vidal-Silva
Faculty of Economics and Administration, Catholic University of the North, Antofagasta, Chile
cristian.vidal@ucn.cl

Erika Madariaga
Faculty of Engineering
Bernardo O'Higgins University
Santiago, Chile
erika.madariaga@ubo.cl

Trung Pham
Information Technology Research Center, Faculty of Economics and Business, University of Talca, Talca, Chile
tpham@utalca.cl

Jose Miguel Rubio
Academic Area of IT and Telecommunications, Technological University of Chile INACAP, Santiago, Chile
jrubiol@inacap.cl

Luis Alberto Urzua
School of Kinesiology, Faculty of Health, Santo Tomas University, Talca, Chile
lurzua@santotomas.cl

Luis Carter
Industrial Civil Engineering Department, School of Engineering, Autonomous University of Chile, Talca, Chile
luis.carter@uautonoma.cl

Franklin Johnson
Computing and Information Department, Engineering Faculty, University of Playa Ancha, Valparaiso, Chile
franklin.johnson@upla.cl

Abstract—**This article presents a comparison of the computing performance of the MapReduce tool Hadoop and Giraph on large-scale graphs. The main ideas of MapReduce and bulk synchronous parallel (BSP) are reviewed as big data computing approaches to highlight their applicability in large-scale graph processing. This paper reviews the execution performance of Hadoop and Giraph on the PageRank algorithm to classify web pages according to their relevance, and on a few other algorithms to find the minimum spanning tree in a graph with the primary goal of finding the most efficient computing approach to work on large-scale graphs. Experimental results show that the use of Giraph for processing large-size graphs reduces the execution time by 25% in comparison with the results obtained using the Hadoop for the same experiments. Giraph represents the optimal option thanks to its in-memory computing approach that avoids secondary memory direct interaction.**

Keywords-Giraph; Hadoop; graph; big data; big graph

I. INTRODUCTION

Daily, 2.5 quintillion bytes are created globally [1]. The size of the "digital universe" was 4.4 zettabytes in 2013 and a tenfold growth is forecasted by 2020 [2]. Precisely, given this growth exponential of information, the term big data is popularized. According to [3], big data is defined as: "large volumes, high speed and a great variety of information that demand innovative and profitable forms of information processing to improve understanding and decision making". Although there is a tendency to increase the space for information storage, there is no equivalent increase in the speed of access and processing. A direct solution to this problem, to reduce access time and data processing, is the use of computer clusters [4]. In this context, several distributed systems exist which permit combining data from different sources, but with high analysis and development costs, mainly for the concurrency of tasks, their synchronization, access and data transfer [4]. Hadoop, an open-source framework based on MapReduce [5, 6] appears as a solution for the mentioned issues. According to [7], big data processing closely associates with unstructured information, that is, directly unrelated data. However, in current social networks, such as Facebook and Twitter, there are links between the component nodes, so their direct representation as graphs is adequate. However, due to the structure and associations of graphs, MapReduce and Hadoop are not optimal for processing, due to the high cost of input/output of information and the possibility of requiring many chained MapReduce phases. In this context, Giraph [8, 9] emerges. Giraph is an open-source counterpart of Pregel [10], a graph processing system developed by Google for the processing of large graphs. Although the generation of data grows at a rate of 60% per year, the technologies related to big data such as Hadoop and Giraph are still under development in Chile. "We still need to know what big data can do for national companies and how they could impact different business lines" [11]. Only 17% of companies are implementing or planning to

Corresponding author: Cristian Vidal-Silva apply big data in the short term in Chile [11]. The primary objective of this work is to present a review of the application of Hadoop and Giraph for the processing of graphs, and thus highlight the practical advantages that Giraph has over Hadoop when solving problems considering relational data which usually support iterative algorithms, such as graph structures. For this, this article presents a comparison performance results of Hadoop and Giraph algorithmic solutions for a classic problem (minimum spanning tree) and a more modern problem (classification of web pages). This work extends the research of [11] mainly to confirm the previously obtained results and conclusions, and to show its applicability for the processing of big data in developing countries like Chile.

II. BIG DATA

Hadoop [4, 12] is an open source framework created to achieve secure, scalable and distributed computing. Hadoop is based on Google documents for MapReduce [5, 6] and Google file system (GFS) [13] and allows the distributed processing of large data sets using computer clusters using simple programming models.

A. MapReduce

Hadoop implements a computational paradigm called MapReduce. Algorithmically, the base of MapReduce is the “divide and conquer” approach, that is, it divides the problem into small pieces for processing in parallel and thus obtains solutions in a distributed environment [12]. The MapReduce programming model composes of a Map function and a Reduce function: Map receives a key-value input pair and produces a set of intermediate key-value pairs. Then MapReduce groups all intermediate values associated with the same intermediate key as the input of the Reduce function. In this way, the Reduce function accepts an intermediate key and a set of related values for that key to mix these values and thus form a new set of final output [6]. Usually, a solution in MapReduce is performed in five stages: i) Splitting: the data are divided into multiple parts and delivered to each mapper. ii) The mapper executes the map function in charge of processing the data. iii) The combiner works directly on the output of mappers for local aggregation. iv) Shuffle: it is responsible for shuffling and ordering the key-value pairs for the function Reduce. v) Reducer: in the last step all the values with the same intermediate key are reduced to generate the final key-value pairs [6].

B. Hadoop Distributed File System (HDFS)

Hadoop includes a distributed file system (HDFS) that can handle large amounts of data. The most efficient pattern of data processing is to write once and read many times [4]. Such as in a file system of a single disk, the files in HDFS are divided into blocks for their storage and representation as independent units. An HDFS cluster presents two types of nodes which operate in a master-worker pattern: a Namenode (the master) and several Datanodes (workers) [12]. The Namenode stores and manages the metadata of the file system, and knows the Datanodes in which all the real blocks for a given file are located. When the data are retrieved, the client contacts the Namenode to obtain the list of the requested data locations and then directly contacts the Datanode to extract the real data [12]. Both, MapReduce and HDFS, manage errors automatically by the Hadoop framework [4].

III. BIG GRAPHS, PREGEL AND GIRAPH SOLUTIONS

Graphs are a finite abstract data type, which consists of a set of edges and nodes or vertex. An edge between two nodes x and y can be described mathematically by a function edge (x, y) [7]. In Big Data analysis, graph processing is considered computationally tricky due to its variable nature [7]. Besides, graph processing is not adequate with general-purpose systems such as MapReduce [14]. For the processing of large graphs (big graphs), Pregel and Giraph are used, which are based on the Bulk Synchronous Parallel (BSP) [15]. BSP is a parallel computing model, in which the calculations are divided into super passes (super-steps) separated by global barriers [16].

A. Pregel

In 2010 Google announced Pregel [10], a framework for distributed graph processing based on BSP. Its objective was to provide a certain level of abstraction so that programmers do not have to deal with distributed memory management or synchronization [8, 17]. The paradigm used by Pregel is “think as vertex”. Pregel specifies each calculation regarding what each vertex should do, and edges are the communication channels for the transmission of the results from one vertex to another. In each super-step, a vertex can execute a user-defined function and change its status from active to inactive. An argue vertex can vote to stop a super-step (inactive) and awake when it receives a message (active) [16].

B. Giraph

Apache Giraph [8, 9] is an open-source alternative of Pregel. In Pregel, to support multithreading, each worker is assigned several partitions of the graph. During each super step, a pair of available workers calculates threads with non-calculated partitions. Each worker maintains its own message store to store all incoming messages. To reduce the contention in the warehouse, and to efficiently use network resources, each computing thread has a cache buffer for all outgoing messages [18]. To implement the BSP model, the workers maintain two stores in each super-step, one for previous messages and another for current messages [18]. Giraph supports different data structures for the list of adjacent vertices [16]. It is important to note that Giraph solutions run as MapReduce tasks and they use Zookeeper to coordinate global barriers [18]. A Giraph solution is organized as a set of idealized iterative super-steps. In each super step, a vertex can send messages to the other vertices, can get its stored values or known information from its edges, and vote to stop. At the beginning of the computation (super-step 0), all vertices start with an active state. A vertex votes to stop because it decided, from its local point of view, that its work is done and the calculation can conclude. The delivery of a message changes the state of a vertex from inactive to active. Giraph finishes the calculation when all the vertices are inactive, and there are no messages to send.

IV. EXPERIMENTS AND RESULTS

For performance comparison of graph processing between Hadoop and Giraph, two algorithmic situations are considered:

- Classification of pages - PageRank: PageRank is an algorithm created by Google [14] to classify web pages, based on the idea that the most relevant web pages probably receive more links from other web pages [16]. The web conceptualizes as a directed graph in which all web pages are nodes, and there are arcs between pages if there are links between those pages. Typically, PageRank iterates until reaching a convergence, that is, when the PageRank values for each node no longer change. Therefore, at the end of each iteration, PageRank should check the convergence state. On the other hand, as authors in [19] highlight, PageRank can also fix a fixed number of iterations.
- Minimal Spanning Tree - Kruskal and Boruvka Algorithm: The algorithms of Kruskal and Prim [20, 21] are standard solutions to obtain a minimal spanning tree of a graph. This paper uses an implementation of the Kruskal algorithm in Hadoop, and one implementation of a Giraph solution to find a minimal distributed spanning tree, Boruvka algorithm [16].

Hadoop 2.4.0 and Giraph 1.1 in Ubuntu 15.10 were used for both experiments. For the execution tests of both solutions, due to hardware limitations, we decided to use Hadoop in “Single Node Setup” mode [4] in a machine with two CPUs and 8 GB of RAM. For comparison, PageRank, Kruskal and Boruvka algorithms testing will be carried out with implementations in Hadoop and Giraph, respectively. Besides, we used a representative data set consisting of a graph with 31 vertices and 82 edges of weight one for the case of PageRank. This data set corresponds to a set of cities of the region of Valparaiso, Chile and their connection, that is, if there is a direct road to connect these cities. Although these data are small, regarding big data, we can extrapolate them for larger computing scenarios.

A. Results

Both PageRank in Hadoop and Giraph were defined with 30 iterations / super-steps respectively. The calculation times of each of these tests are shown in Figures 1-2.

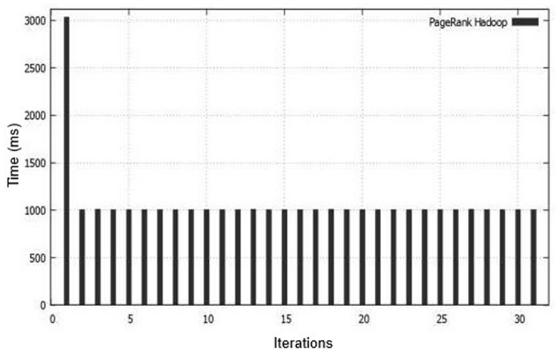

Fig. 1. Results of the PageRank algorithm in Hadoop.

Kruskal’s algorithm found the minimum spanning tree (MST) in around 3 seconds in Hadoop (Figure 3) in a single iteration, while its implementation in Giraph found the solution in around 0.7 seconds in 2 super-steps (Figure 4).

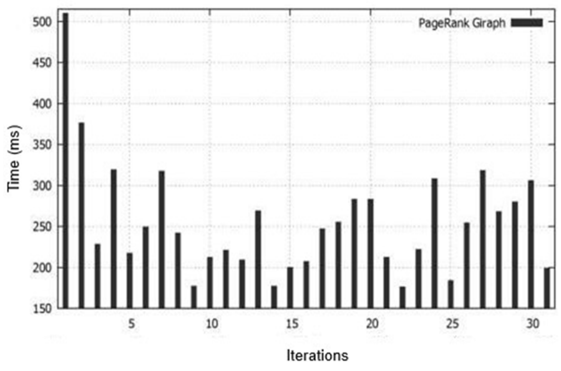

Fig. 2. Results of the PageRank algorithm in Giraph.

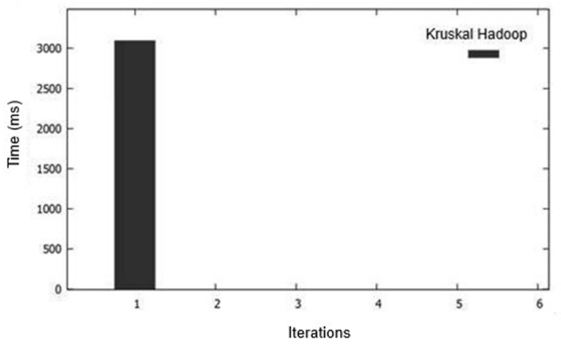

Fig. 3. Results of the Kruskal algorithm in Hadoop.

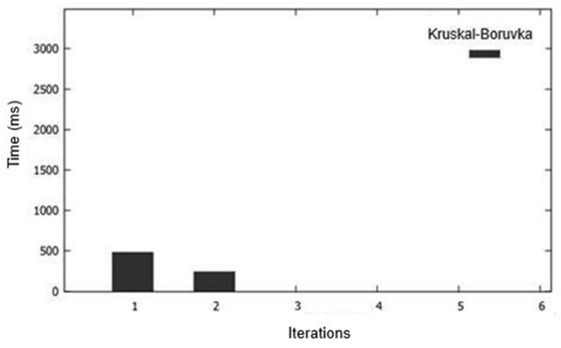

Fig. 4. Results of the Kruskal and Boruvka algorithm in Giraph.

To justify the obtained results, to process the entire graph, MapReduce performs many chained tasks to achieve its objective, and each task involves inputs and outputs to disk. Ideally, an iteration in MapReduce represents a super-step in the Giraph approach [7]. With that assumption, we can establish an average time per iteration between the total time and the number of iterations / number of super-steps, for each of the four tests. Table I presents the mentioned results. Figure 5 shows a comparison of the average times for the calculations made by Hadoop and Giraph with the Kruskal and Boruvka algorithms, respectively, to find the minimal spanning tree of a graph. When comparing the times associated with PageRank, the Giraph solution is at least four times faster than Hadoop. This difference increases to 8 times in the case of Kruskal in Hadoop and Boruvka in Giraph to find a tree of minimum expansion of a graph.

TABLE I. EXECUTION TIME IN HADOOP AND GIRAPH.

Test	Results		
	Total time (ms)	Super-steps	Avg. time (ms)
PageRank Hadoop	32.238	30	1.074,6
PageRank Giraph	7.925	30	264,2
Kruskal Hadoop	3.096	1	3.096
Boruvka Giraph	718	2	359

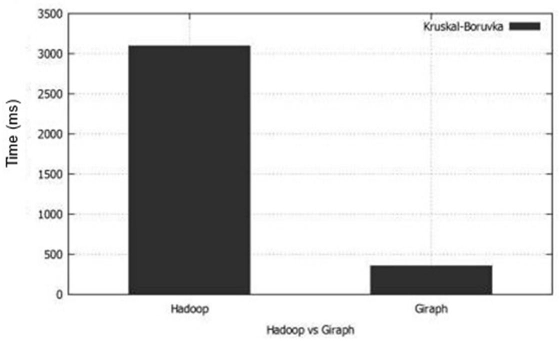

Fig. 5. A comparison between Hadoop and Giraph for the PageRank and MST algorithms.

V. CONCLUSIONS

According to the results, Giraph presents a four times higher efficiency on Hadoop in computation time of the PageRank algorithm. This advantage increases to almost eight times when it comes to finding the minimum spanning tree, because of the advantage of Giraph to maintain a global communication between each super-step without requiring inputs and outputs to disk since each node can exchange messages with others [19]. The previously described situation is not possible in Hadoop, since MapReduce does not provide any tool for global and direct communication between the participants in each iteration. Also, the nature of the graphs means that the computation of each node depends on its neighboring nodes, which implies that MapReduce solutions have to perform many chained iterations, which generates an excessive I/O traffic. It should be noted that Giraph is not a tool that comes to replace Hadoop, it should rather be a complementary solution for given problems. Hadoop has a complete ecosystem [4], which expands day by day, and it is relevant to know the advantages of both technologies, in order to assimilate them correctly. An immaturity still exists regarding big data technology and available information about its use and support use of big data tools in developing countries such as Chile [2, 11]. Because the focus of this work is a practical comparison between Hadoop and Giraph for graph problems, a future work would be to make a comparison between Giraph and current big data tools such as Apache Spark [22] and Flink [23, 24], in real and research scenarios such as model processing. Since Giraph's primary focus is graph processing, it is required to verify its potential performance advantages.

REFERENCES

[1] I. Yaqoob, I. A. T. Hashem, A. Gani, S. Mokhtar, E. Ahmed, N. B. Anuar, A. V. Vasilakos, "Big data", International Journal of Information Management, Vol. 36, No. 6, pp. 1231–1247, 2016

[2] A. K. Wahi, V. Ahuja, "The internet of things-new value streams for customers", International Journal of Information Technology and Management, Vol. 16, No. 4, pp. 360–375, 2017

[3] P. Zikopoulos, C. Eaton, Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data, McGrawHill Osborne Media, 2011

[4] T. White, Hadoop: The Definitive Guide, O'Reilly Media, Inc., 2015

[5] J. Dean, S. Ghemawat, "Mapreduce: Simplified data processing on large clusters", 6th Conference on Symposium on Operating Systems Design & Implementation, San Francisco, December 6-8, 2004

[6] J. Dean, S. Ghemawat, "Mapreduce: Simplified data processing on large clusters", Communications of the ACM, Vol. 51, No. 1, pp. 107–113, 2008

[7] M. Aurelio, B. Fagnani, G. Lotz, Dynamic Graph Computations using Parallel Distributed Computing Solutions, Science without Borders, 3-Months Project Report, Queen Mary, University of London, 2013

[8] R. Shaposhnik, C. Martella, D. Logothetis, Practical Graph Analytics with Apache Giraph, Apress, 2015

[9] S. Sakr, F. M. Orakzai, I. Abdelaziz, Z. Khayyat, Large-Scale Graph Processing Using Apache Giraph, Springer, 2017

[10] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, G. Czajkowski, "Pregel: A system for large-scale graph processing", 2010 ACM SIGMOD International Conference on Management of Data, Indianapolis, USA, June 6-11, 2010

[11] S. Valenzuela, C. Vidal, "Evaluacion de hadoop y giraph para el procesamiento de grafos", Jornadas Chilenas de la Computacin, XXVII Encuentro Chileno de Computacin. Santiago, Chile, 2015 (in Spanish)

[12] S. Karanth, Mastering Hadoop. Packt Publishing, 2015

[13] S. Ghemawat, H. Gobioff, S. T. Leung, "The Google file system", Nineteenth ACM Symposium on Operating Systems Principles, Bolton Landing, USA, October 19-22, 2003

[14] S. Brin, L. Page, "The anatomy of a large-scale hypertextual web search engine", Computer Networks and ISDN Systems, Vol. 30, No. 1-7, pp. 107–117, 1998.

[15] L. G. Valiant, "A bridging model for parallel computation", Communications of the ACM, Vol. 33, No. 8, pp. 103–111, 1990

[16] M. Han, K. Daudjee, K. Ammar, M. T. Ozsu, X. Wang, T. Jin, "An experimental comparison of pregel-like graph processing systems", Proceedings of the VLDB Endowment, Vol. 7, No. 12, pp. 1047–1058, 2014

[17] S. Salihoglu, J. Shin, V. Khanna, B. Q. Truong, J. Widom, "Graft: A debugging tool for apache giraph", 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Australia, May 31-June 4, 2015

[18] M. Han, K. Daudjee, "Giraph unchained: Barrierless asynchronous parallel execution in pregel-like graph processing systems", Proceedings of the VLDB Endowment, Vol. 8, No. 9, pp. 950–961, 2015

[19] J. Lin, C. Dyer, Data-Intensive Text Processing with MapReduce, Morgan and Claypool, 2010

[20] M. Held, R. M. Karp, "The traveling-salesman problem and minimum spanning trees: Part ii", Mathematical Programming, Vol. 1, No. 1, pp. 6–25, 1971

[21] R. C. Prim, "Shortest connection networks and some generalizations", Bell System Technical Journal, Vol. 36, No. 6, pp. 1389–1401, 1957

[22] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, A. Ghodsi, J. Gonzalez, S. Shenker, I. Stoica, "Apache spark: A unified engine for big data processing", Communications of the ACM, Vol. 59, No. 11, pp. 56–65, 2016

[23] E. Friedman, K. Tzoumas, Introduction to Apache Flink: Stream Processing for Real Time and Beyond, O'Reilly Media, 2016

[24] S. Papp, The Definitive Guide to Apache Flink: Next Generation Data Processing, Apress, 2016