Performance Evaluation of Big Data Frameworks: MapReduce and Spark

Jaspreet Singh, S.N. Panda and Rajesh Kaushal

Abstract Spark and MapReduce are two prominent open-source distributed computing frameworks for big data processing and analytics. These frameworks introduce a simple programming APIs for new users and suppress the complication and fault tolerance of distributed tasks. Most of Internet companies widely deploy these frameworks to process their massive data. Furthermore, all other big communities are adopting these HPC because high-performance data analytics is required to solve big data problems. To provide an efficient framework for processing and analyzing large amount of data, today's researchers correlate both the frameworks. (1) This paper discusses the evaluation of the performance of MapReduce and Spark on page rank, sort and word count. From some existing research, we evaluate page rank and sort algorithms in these frameworks. (2) We provide in-depth analysis of task execution time on word count algorithm in both of these frameworks, through detailed experiment and quantify the performance based on different dataset sizes. Overall experimental results show that Spark is faster than MapReduce. The prime causes of speedups in Spark are the reduced DISK and CPU overheads due to RDD cashing.

Keywords Hadoop · Spark · MapReduce · HDFS · Data analytics

J. Singh (✉) · S.N. Panda · R. Kaushal
Department of Computer Science and Engineering, Chitkara University Institute of Engineering and Technology, Chitkara University, Rajpura, Punjab, India
e-mail: jaspreet.cse@chitkara.edu.in

S.N. Panda
e-mail: snpanda@chitkara.edu.in

R. Kaushal
e-mail: rajesh.kaushal@chitkara.edu.in

R. Singh et al. (eds.), *Intelligent Communication, Control and Devices*,
Advances in Intelligent Systems and Computing 624,
https://doi.org/10.1007/978-981-10-5903-2_167

1 Introduction

Big Data cannot be acknowledged as a precise term with a proper definition. Instead, it is a collection of large amount of various kinds of data, mostly unstructured. This indicates it is very difficult process for relational database managements systems to analyze and process such data, which is of large volume and unstructured nature. To business and big communities should be capable of finding out useful information from this vast amount of data, that helps to grow their existing business and to find out new requirements of their users, so as for better customer experiences in the future. Big Data analytics techniques provide more precise results and solutions [1].

Researchers and Analysts use advanced techniques of analytics such as text and predictive analytics, natural language processing (NLP), data mining and machine learning to figure out more useful insight and information which is very helpful for an enterprise to make right decision. Open-source big data analytics technologies such as Hadoop MapReduce and Spark provide adequate solutions. In this paper, we examine the comparison between Spark and MapReduce frameworks on Hadoop [2].

Section 2 covers Apache Hadoop with MapReduce framework for storage and processing of large dataset. Section 3 discusses about data processing in Apache Spark and MLlib library for iterative machine learning algorithms. Section 4 discusses Hadoop distributed file system (HDFS) to store data in a distributed manner on different nodes. Section 5 shows the comparison of MapReduce and Spark on page rank algorithm and discussion. Section 6 discusses comparison of MapReduce and Spark on sort. Section 7 shows the experimental studies and discussion generated by execution of word count program by using Spark and MapReduce programming framework. Section 8 describes the conclusion.

2 Apache Hadoop

Apache Hadoop is an open-source framework to process and manage Big Data. Big Data is large amount of which can be in any form for example structured, unstructured and semi structured [3]. Hadoop provides a parallel distributed database known as HDFS Hadoop distributed file system, and there are number of tools and frameworks related to Hadoop, which we can use to perform operations on data stored in HDFS. In Hadoop, data is replicate on multiple nodes, if one data node fails, we can recover our data from other nodes [4].

Apache Hadoop includes following modules:

(a) Hadoop Common: The common utility libraries which help Hadoop components to interact with each other.
(b) Hadoop distributed file system (HDFS): It is termed as a distributed file system for parallel processing and providing fault tolerant and high throughput which helps to store and process data on multiple nodes in a distributed manner.

(c) Hadoop YARN: Yarn supervises the cluster resource management and job scheduling during the execution of jobs [5].

3 Hadoop MapReduce

Hadoop MapReduce is a programming framework to process immense data in-parallel distributed manner on large nodes of commodity hardware [6]. It is a fault tolerant and reliable framework. MapReduce programming usually works on map and reduce paradigm [7]. Map phase splits the input dataset into multiple chunks to process in a fully parallel manner, and the output of this stage is input for the next phase which is known as reduce task, and both input and output after the execution of job are stored in HDFS [8]. The MapReduce framework typically works on a <key, value> pair, that is, it splits the input dataset into <key, value> pairs and process output of the job in similar form [9]. Figure 1 represents MapReduce workflow.

4 Apache Spark

Apache Spark is a framework for analyzing Big Data [10] which can process and analyze massive amount of data in distributed manner. Apache Spark uses Resilient Distributed Datasets known as RDDs [11] which is distributed set of instances and an unchallengeable fault tolerant for the execution of parallel operations [12].

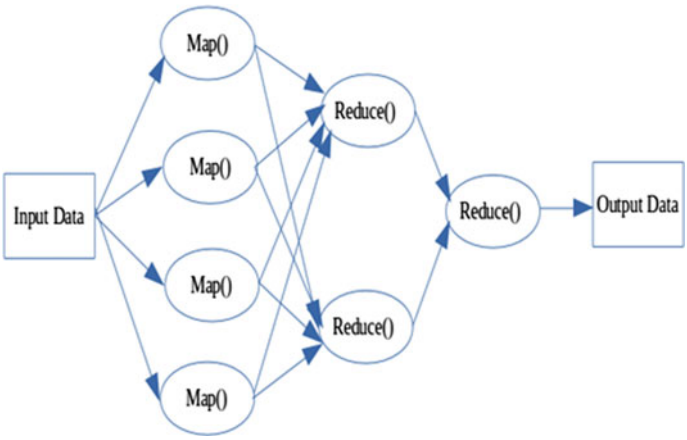

Fig. 1 MapReduce work flow

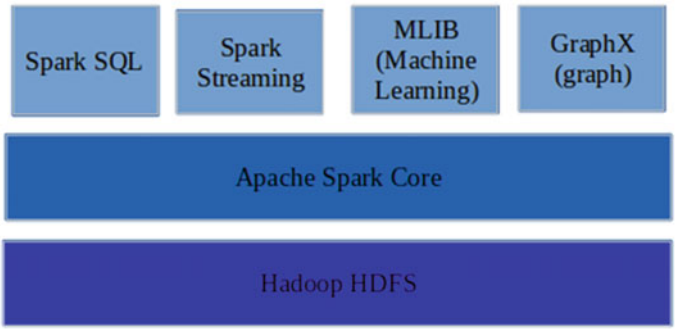

Fig. 2 Apache Spark Stack

As in MapReduce, both Map and Reduce phases use disk read/write operations number of times in the execution of a job, so these read/write operations cause more delay and increase execution time [13].

Apache Spark provides a programing interface where we can write a program, execute it in distributed manner [14], and RDD helps to keep data in RAM and process that data. In this procedure, it utilizes in-memory cache efficiently. During data processing, it puts data on local file system, which reduces read/write disk operations in Spark [15]. Figure 2 describes Apache Spark stack.

5 Page Rank

Page rank is one of the foundational algorithms of Google's indexing process: ranking every web page on the internet by the number and quality of links. It is a procedure to measuring the value and importance of a webpage. PageRank counts the number and quality of links of a webpage to estimate the weight age of webpage [16].

It is assumed that a website which accepts links from another website is very crucial. Different authors performed comparison of page rank algorithm on Hadoop and Spark and according to [2, 17, 18] Spark performs better than Hadoop but according to [19] Hadoop performs better than Spark. Table 1 describes the comparison of page rank algorithm on Hadoop and Spark.

6 Sort

Since data is not reduced through the pipeline, therefore sorting is the most perplexing operation. For moving data on all the machines, shuffle operation is performed at the sorting's core. To sort 1 TB of data, shuffling of this data needs to be

Table 1 Comparison of MapReduce and Spark on PageRank

S. No.	Apache Spark	Apache Hadoop	Results
1	1. IN memory computing (speed processing) (lower latency) 2. Average 40% bandwidth 3. Burst bandwidth is only 47% larger than average	1. Disk-based processing 2. Average 15% bandwidth 2.198% (larger) which cause traffic and speed slow	This paper shows Spark is 8x faster than Hadoop in page Rank [17]
2	1. Uses Graphx libraries (better performance and optimization) (reduce network overhead) 2. Page rank is iterative algorithm, caching the input as RDD reduces both CPU and disk i/o so it contributes 90% of speedup	1. Used Mahout libraries 2. Hadoop uses disk for all operations	This paper shows Spark is 5x faster than Hadoop in page rank [19]
3	1. There is no memory release phenomenon in Spark 2. Spark outperforms Hadoop When there is enough memory for Spark in the whole iterative process When dataset is large, there is a gradual increment between two consecutive iterations	1. When map or reduce computation ends it releases memory	Hadoop performs better if there is insufficient memory to store intermediate results [20]
4	1. OPEN-source MPI library (Message Passing Interface). HPC Load balancing, handles shuffling buffer manager 2. Use 10 iterations so overhead between the iterations is light	2. Hadoop uses file partitions and store them on disk	Spark performs better [18]

performed on the network [21]. Different authors performed comparison of sort algorithm on Hadoop and Spark, and according to [22], Spark performs better than Hadoop but according to [19], Hadoop performs better than Spark. Table 1 describes the comparison of sort algorithm on Hadoop and Spark (Table 2).

Table 2 Comparison of MapReduce and Spark on sort

S. No.	Apache Spark	Apache Hadoop	Results
1	Network overhead is caused by and is proportional to the number of files opened simultaneously Low network speed leads to increase cache buffer sparks performance degrades	MapReduce can overlap the shuffle stage with the map stage, which effectively hides the network overhead	MapReduce performs better than Spark on 100 and 500 GB data 1 Gbps ethernet speed MapReduce is 2x faster than Spark [19]
2	Cache buffer decrease	Use disk for read/write operation	100 Tbs and Pbs of data 3x faster 10x fewer machines 22] 200 machines with 10 Gbps link each while 2000 machine

7 Experimental Studies

We have evaluated the efficiency of MapReduce and Spark by implementing word count algorithm.

For these experiments, we have used the WC program which we have included in both MapReduce and Spark, and these experiments were performed on five datasets of different sizes on both MapReduce and Spark to calculate the execution time [23]. To perform this experiment, two nodes Hadoop clusters were used. Figure 1 describes the comparison of MapReduce and Spark when word count algorithm was implemented on both the frameworks [15, 24]. Table 3 describes the comparison of MapReduce and Spark on word count on different datasets (Fig. 3).

Table 3 Comparison of MapReduce and Spark on word count

Dataset	MapReduce (s)	Spark (s)
100 MB	31	11
500 MB	145	27
1 GB	317	44
5 GB	2249	220
10 GB	4972	459

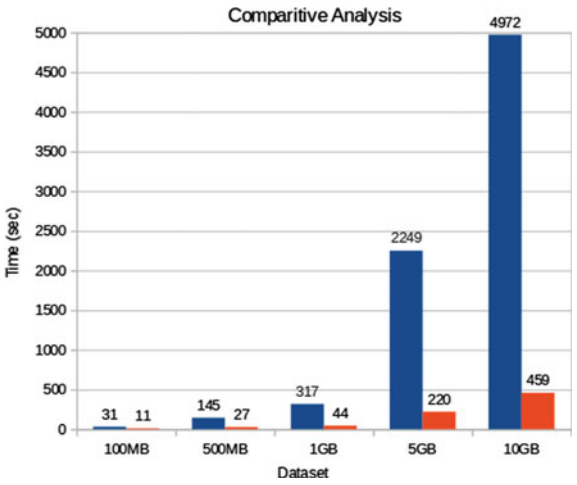

Fig. 3 Comparative analysis of Hadoop

8 Conclusion

Both the Spark and MapReduce frameworks are popular distributed computing paradigms for providing an effective solution for handling this large amount of data called Big Data. Today there are many misconceptions about the comparison of Spark and MapReduce framework. Many researchers and bloggers are claiming that Spark is 10–100 times faster than MapReduce framework. In this paper, we are performing the comparative analysis of both the programming frameworks in terms of execution time. We have run word count program on both the programming frameworks with different sized datasets and found that Spark is 10 times faster than Hadoop.

References

1. Landset, S., Khoshgoftaar, T. M., Richter, A. N., & Hasanin, T. (2015). A survey of open source tools for machine learning with big data in the Hadoop ecosystem. Journal of Big Data, 2(1). doi:10.1186/s40537-015-0032-1.
2. Elser, B., & Montresor, A. (2013). An evaluation study of Big Data frameworks for graph processing. 2013 IEEE International Conference on Big Data. doi:10.1109/bigdata.2013.6691555.
3. Vavilapalli, V. K., Seth, S., Saha, B., Curino, C., O'malley, O., Radia, S., … Shah, H. (2013). Apache Hadoop YARN. Proceedings of the 4th annual Symposium on Cloud Computing - SOCC '13. doi:10.1145/2523616.2523633.
4. HDFS Architecture. (n.d.). Retrieved March 2, 2017, from http://hadoop.apache.org/docs/current/hadoop-project dist/hadoop-hdfs/HdfsDesign.html.
5. Cugola G, Margara A (2012) Processing flows of information: from data stream to complex event processing. ACM Comput Surv 44(3):15:1–15:62.

6. Dittrich, J., & Quiané-Ruiz, J. (2012). Efficient big data processing in Hadoop MapReduce. Proceedings of the VLDB Endowment, 5(12), 2014–2015. doi:10.14778/2367502.2367562.
7. S, G. P., R, N. H., & Prabhu, S. (2017). High Performance Computation of Big Data: Performance Optimization Approach towards a Parallel Frequent Item Set Mining Algorithm for Transaction Data based on Hadoop MapReduce Framework. International Journal of Intelligent Systems and Applications, 9(1), 75–84. doi:10.5815/ijisa.2017.01.08.
8. Kabáč, M., Consel, C., & Volanschi, N. (2017). Designing parallel data processing for enabling large-scale sensor applications. Personal and Ubiquitous Computing. doi:10.1007/s00779-017-1009-1.
9. Mavridis, I., & Karatza, H. (2017). Performance evaluation of cloud-based log file analysis with Apache Hadoop and Apache Spark. Journal of Systems and Software, 125, 133–151. doi:10.1016/j.jss.2016.11.037.
10. Svyatkovskiy, A., Imai, K., Kroeger, M., & Shiraito, Y. (2016). Large-scale text processing pipeline with Apache Spark. 2016 IEEE International Conference on Big Data (Big Data). doi:10.1109/bigdata.2016.7841068.
11. Gopalani, S., & Arora, R. (2015). Comparing Apache Spark and Map Reduce with Performance Analysis using K-Means. International Journal of Computer Applications, 113 (1), 8–11. doi:10.5120/19788-0531.
12. Huang, W., Meng, L., Zhang, D., & Zhang, W. (2017). In-Memory Parallel Processing of Massive Remotely Sensed Data Using an Apache Spark on Hadoop YARN Model. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 10(1), 3–19. doi:10.1109/jstars.2016.2547020.
13. M. Zahariaetal. Resilient Distributed Datasets: A Fault Tolerant Abstraction for In Memory Cluster Computing. NSDI 2012.
14. Liang, F., & Lu, X. (2015). Accelerating Iterative Big Data Computing Through MPI. Journal of Computer Science and Technology, 30(2), 283–294. doi:10.1007/s11390-015-1522-5.
15. Wang, K., & Khan, M. M. (2015). Performance Prediction for Apache Spark Platform. 2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems. doi:10.1109/hpcc-css-icess.2015.246.
16. Barrachina, A. D., & O'Driscoll, A. (2014). A big data methodology for categorising technical support requests using Hadoop and Mahout. Journal Of Big Data, 1(1), 1. doi:10.1186/2196-1115-1-1.
17. Jiang, T., Zhang, Q., Hou, R., Chai, L., Mckee, S. A., Jia, Z., & Sun, N. (2014). Understanding the behavior of in-memory computing workloads. 2014 IEEE International Symposium on Workload Characterization (IISWC). doi:10.1109/iiswc.2014.698.
18. Liang, F., & Lu, X. (2015). Accelerating Iterative Big Data Computing Through MPI. Journal of Computer Science and Technology, 30(2), 283–294. doi:10.1007/s11390-015-1522-5.
19. Shi J., Qiu Y., Minhas U. F., Jiao L., Wang C., Reinwald B., & Ozcan F., "Clash of the titans: MapReduce vs. Spark for large scale data analytics", In Proceedings of the VLDB Endowment, 8(13), pp. 2110–2121, 2015.
20. Apache Spark the fastest open source engine for sorting a petabyte. (2016, October 27). Retrieved March 4, 2017, from https://databricks.com/blog/2014/10/10/spark-petabyte-sort.html.
21. Armbrust M., Das T., Davidson A., Ghodsi A., Or A., Rosen J., & Zaharia M., "Scaling spark in the real world: performance and usability", In Proceedings of the VLDB Endowment, 8 (12), pp. 1840–1843, 2015.
22. Awan, A. J., Brorsson, M., Vlassov, V., & Ayguade, E. (2015). Performance Characterization of In-Memory Data Analytics on a Modern Cloud Server. 2015 IEEE Fifth International Conference on Big Data and Cloud Computing. doi:10.1109/bdcloud.2015.37.

23. Liang, F., Feng, C., Lu, X., & Xu, Z. (2014). Performance Benefits of Data MPI: A Case Study with Big Data Bench. Big Data Benchmarks, Performance Optimization, and Emerging Hardware Lecture Notes in Computer Science, 111–123. doi:10.1007/978-3-319-13021-7_9.
24. Gu L., & Li H., "Memory or time: Performance evaluation for iterative operation on hadoop and spark", In High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC), 2013 IEEE 10th International Conference, pp. 721–727, IEEE, November, 2013.