Journal of Scientific & Industrial Research
Vol. 79, July 2020, pp. 631-635

Processing Real World Datasets using Big Data Hadoop Tools

N Deshai[1]*, B V D S Sekhar[1], P V G D Prasad Reddy[2] and V V S S S Chakravarthy[3]

[1]Department of Information Technology, S R K R Engineering college, Bhimavaram, AP, India
[2]Department of Computer Science & Systems Engineering, Andhra University, Visakhapatnam, AP, India
[3]Department of Electronics & Communication Engineering, Raghu Institute of Technology, Visakhapatnam, AP, India

Received 11 April 2019; revised 22 December 2019; accepted 18 April 2020

Today's digital world computations are extremely difficult and always demands essential requirements to significantly process and store an enormous size of datasets. Therefore, this is mostly structured, semi-structured and unstructured generated data with more velocity at beyond the limits and double day by day from a wide variety of applications. In the last decade, many organizations have been facing major problems in handling and processing massive chunks of data, which could not be processed efficiently due to the lack of enhancements on existing technologies. This paper, introduce advanced data processing tools to solve the extreme problems as efficiently by using the most recent and world's primary powerful Map-Reduce framework, but it has few data processing issues. Therefore, recently Apache Spark fastest tool has introduced to overcome the limitations of Map Reduce.

Keywords: Apache Spark, Big Data, Hadoop, Map Reduce

Introduction

In today's digital world many organizations generate enormous data whose size raises exponentially every year.[1] Big data become a hot topic at present decade, which seems to be of enormous size. It has unstructured data to be processed by using conventional computational tools and methods in efficient, scalable, cost-effective and reliable manner[2] with the support of apache processing tools. Big data distinctly address characteristics with six V's volume, velocity, veracity, variety, value, variability and validity. Spark assists to simplify the challenging and computationally intensive job of treating high quantities of real-time or archived data, both structured and unstructured, seamlessly combining appropriate complex abilities such as machine learning and graph algorithms. Spark significantly makes Big data processing to the volumes. Nowadays Big data analytics is one of the most effective research fields with many objectives and requirements for latest variations that influence a wide range of industries. To fulfill the computational requirements of massive data analysis, a suitable framework is necessary to create, perform and maintain the needed pipelines and algorithms. In this concern, Apache Spark has been raised as a unified engine for large-scale data analysis over a diversity of workloads. It presents a novel method for data science and engineering where wide ranges of data problems can be solved using a single processing engine with general-purpose languages.

Materials, Methods and Results

Apache Spark has been adopted as a fast and scalable framework in both academia and industry because of its advantage in advanced programming model. For an overview, comparison of features between Map Reduce and Spark is given in Table 1. It has become the most active big data open source project and one of the most active projects in the Apache Software Foundation. Separate testing operation regarding file reading computation with CERN storage system based on a dedicated testing cluster is given in Table 2. Apache Spark access and copying of special file formats is given in Table 3.

Map reduce

Google has introduced an early and latest digital world framework to significantly solve existing technology difficulties by Map-Reduce which is simple, open source, more distributed and parallely processing incredible volumes of world data on a large commodity cluster regarding reliability, high fault tolerance, great scalability and in more reliable

*Author for Correspondence
E-mail: desaij4@gmail.com

manner. In December 2004, Google issued a journal on Map Reduce. The great benefit of Map Reduce is that multiple computer nodes are easy to modify data processing. Map Reduce perform two different tasks, Map and Reduce, which takes place entirely after the completion of the mapped phase. The map task, which is actually reading and quickly process a data block at intermediate level outputs for producing key-value pairs. The reducer receive key-value frame from several map jobs, then adds with a smaller group of multiples or key-value frames (the final output) by means of intermediate datasets (intermediate key-value pair) as shown word counting in Fig. 1.

The total read time is calculated by subtracting the total CPU time from the total execution time for Cluster test of reading ROOT files from HDFS (Hadoop Distributed File System) and EOS. A total of 160 tasks were used (40 executors with 4 cores per executor) to analyze 0.5 TB of ROOT files in Apache Spark as shown in Fig. 2.

We illustrated comparison of performing the common HEP examination workflow on Apache Spark. The two biggest alerts from the primary studies were determined by improving the histogram mac package to fill histograms in map-reduce way technologies and to put ROOT files directly from Apache Spark, also from the CERN EOS storage system. This eliminated the requirement to transform the input data into a structured format that Apache Spark follows natively. The execution of the entire analysis workflow reading ROOT files is about 2–4 times slower on EOS than on HDFS (Fig. 3), but additional tuning and optimizations are required to achieve the gap. The scaling performance outcomes are encouraging to examine more extended and more extensive input sizes and get the purpose to make interactive analysis on very huge datasets.

Table 1 — Feature Comparison between Map Reduce and Spark

	Hadoop Map Reduce	Spark
Processing	Batch	Micro Batch, Stream
Speed	Slow	Faster than MR
Operators	NA	Time-based
Windows	No	Yes
Storage data	HDFS	In-Memory
Latency	High	Low
Fault tolerance	High	High , RDD DAG
Performance	Slow	High than MR
Remove duplicate	High	Process records exactly
Iterative data flow	Chain of states	Cyclic data flow DAG
Scalability	Incredible up to 10,000	High cluster of 8000
Visualization	Low	High, need RAM
Recovery	high fault tolerant	RDD DAG
Abstraction	NO	Spark RDD Data stream
Easy to use	Difficult	Easy
Real-time analysis	No	Good
Scheduler	Fair, capacity	Own flow scheduler
SQL	Hive	SSQL Hive, FDSL
Catching	Not	Yes
Hardware	Commodity h/w	Mid to high level h/w
Machine learning	Mahout	Mlib
Line of code	1,20,000	20,000
Deployment	Fully distribute mode	Standalonemesos/YARN

Table 2 — Size of the Files

	Apache Spark	
Data size(250 GB)	HDFS	EOS(CERN)
Text	1 Gbit/s	250 Mbit/s
PARQUET	700 Mbit/s	6-8 Gbit/s
ROOT	400 Mbit/s	2.4 Gbit/s

Table 3 — Reading Operation Completion Times

	Apache Spark	
Total Time	HDFS	EOS(CERN)
Running	4 minutes	18 minutes
Reading	2.7 hours	8.9 hours
Executing	5.6 hours	10.9 hours
CPU	2.8 hours	2.9 hours

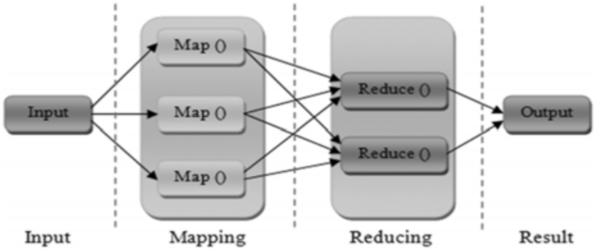

Fig. 1 — Map Reduce word counting

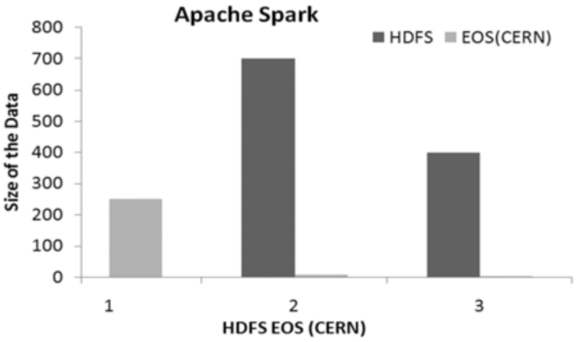

Fig. 2 — Size of the Files

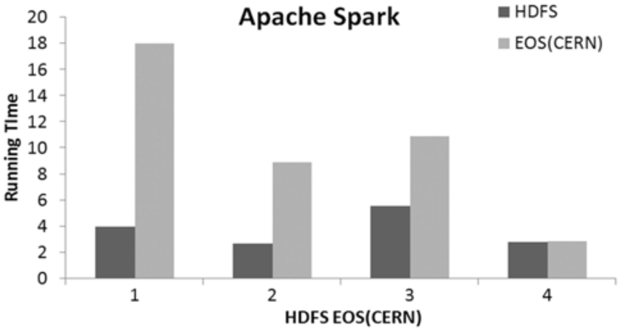

Fig. 3 — Completion Times

Features of Map Reduce

Parallel Processing

Typically, to split each job across several nodes through Map Reduce moreover, every part of the job node operates parallel and in distributed manner. Mostly the Map Reduce working depends on divide and conquers strategy, which enables us to manipulate data with multiple machines.[3] Actually, the time taken to process the data is significantly reduced by a number of machines in parallel rather than by one machine.

Mapper Phase<key one, value one> =list (<key two, value two>) (1)

Reducer phase<key two, list (value two)>=list (<key three, value three>) (2)

Fault Tolerance

Hadoop extremely controls the faults with the help of replication factor. Whenever user store a particular file in Hadoop storage component HDFS, which partitions the file into a number of blocks and distribute data blocks over the various machines in HDFS cluster.[4] In addition, to generate the default replica value of each block is on other cluster machines, if one machine in the cluster is failed during critical circumstances. Therefore, the user could gain data from other machines.

Scalability

Typically, Hadoop has one of the major strength as scalability. Hence, it is very easy to add new nodes with no downtime. Hadoop supports horizontal scalability. So latest nodes can join on the fly manner to the machine. In Apache Hadoop, every application can run significantly on more than thousands of nodes.

Reliability

In Hadoop, the whole data become more reliable which is stored on the cluster of machines. Regardless of machine failure, replication mechanism can support to gain the same data from a different place. Therefore, if any of the nodes fails, then also we can store data more reliably.

High Availability

Due to the more number of copies of datasets the actual data is easily available and accessible even though hardware faces failures. Therefore, even if any device goes down, our required data could be retrieved in one way or the other.

Data locality

The major limitation of Hadoop is more crossing-switching system traffic due to processing of the enormous quantity of data. Therefore to beat this problem, Data Locality came into reality. Hadoop can support to move the computation very closely tied with real data, which actually resides on the cluster node.[5] Therefore, it efficiently decreases network congestion and broadens the system throughput.

Major issues on Hadoop

Today's digital world computations are mostly based on complete real-time orientation but could not be process continuously for every aspect in an efficient way with Map-Reduce.[6] Since the Map Reduce process intermediary aspects as it responds to each job run in separation, huge data could be shuffled across the network. Whenever you require handling stream processing with Map Reduce, it is highly difficult. Hence, Map Reduce is excellent and suitable for batch process on en enormous amounts of data. For small files, processing speed is very low, high latency, less security, poor real-time stream processing, efficiently support up to batch processing only, more uncertainty, line of code is complex, no mechanism of caching, difficult ease of use, generally vulnerable, lack of delta iterations, poor interactive processing, lack of in memory and graph processing. If it's programming interface is low, Map Reduce and its open source application Hadoop face performance and latency problems during the frequently rising actual size of the data. Apache Spark is created to address the problems and drawbacks of Map Reduce.

Spark

This is world's fastest general-purpose, more distributed, much parallel and completely open-source cluster computing model. It is originally developed in

2009 and opened in 2010 as an Apache project in the UC Berkeley's AMPLab. This has fairly been since lightning fast and in-memory processing. In recent year, Hadoop could do Spark tool place and the whole world witnessed with implementation of a standard examination concerning organization of 100 terabytes of data just in 23 minutes - the earlier world proof of 71 minutes. Developers from Spark conclude that when properly processed massive data in memory approach it can work effectively, and is lightning faster means 100 times faster than Map Reduce also 10 times faster than disk. Spark simply provides more scalability, more fault tolerance, reliability, and several other features. Spark and its RDDs did reveal in 2012 while the response to weaknesses in the Map-Reduce cluster-computing model, which makes a selective straight dataflow construction on distributed applications. Map Reduce applications gather input data during the disk source, then the map takes responsibility for the data, degrade the outcome of the map task, and store reduce task outcome on disk. Spark's RDDs could perform similarly to working set to distributed applications, which contribute an intentionally reduced pattern of distributed oriented shared memory. Apache Spark has complete service of an architectural establishment with the resilient distributed dataset (RDD), which support read-only operation on a number of data items circulated across the cluster, and implement in a fault-tolerant manner.[7]

The Data frame API could be generated as a notion on top of the RDD. In Apache Spark, the starting interface that is especially an application-programming interface (API) is called the RDD. Spark strongly supports memory processing to enhance the performance of applications for big data analysis, but it could also execute traditional disk-based treatment whenever data sets are far too large for the system memory available. The RDD has specifically designed so that customers can cover up a great deal of computational complexity. It aggregates data and partitions it across a whole server cluster where it could be calculated, migrated or simply run via an analytical paradigm in another data store.[7] Spark greatly supports the deploying of iterative-based techniques, which encourage their data set various times during a loop service, and especially the data analysis with complete interactive manner. Spark achieves low latency of those applications, could be degraded by various request of dimension compared with Apache Hadoop Map Reduce execution. Between the sorts of iterative techniques are the train-based methods for machine learning operations, which made the fundamental incentive for improving Apache Spark. Apache Spark accomplishes higher performance for both batch and streaming information, just using a Direct Acyclic Graph like state-of-the-art scheduler, optimization of a query, and psychical execution tools.

Apache Spark always needs a cluster based administrator and the latest parallel and more distributed storage segment. During batch control, spark recommends standalone mode, Hadoop YARN, or Apache Mesos. During propagated accommodation, Spark makes interface among an extensive diversity, including Hadoop Distributed File System (HDFS), and conventional solutions could be executed.[8] Apache Spark further establishes a pseudo-distributed restricted method, normally utilized for improvement or examination objectives, where disseminated accommodation is not essential but the general file operation could be utilized alternately in different situations, typically, a single device including single executor through one CPU kernel operated apache spark.[9]

Spark libraries

The Spark Core engine processes mainly provide high-level API and actually support a similar set of related data-management and analysis tools. In addition to spark core, a new package of most popular code libraries to be used in data analysis and software programs actually comes with an apache spark environment. Spark SQL allows users to query stored data in different applications in the relevant SQL language. Spark streaming can simply build an application to evaluate and present information even in real-time. MLlib is a device trying to learn code library that allows customers to use advanced mathematical operations on spark information and create new applications for those analyses. GraphX, which is mostly a more graph-parallel numerical computation online tool, actually built-in library.[9] Spark typically, provides more than 80 operators, which simply make parallel applications easy to develop. Although from the Scala, Python, R and SQL shells then you really could use it interactively. Spark is enabled with a pack of libraries, along with SQL and Data Frames, MLlib, GraphX, and Spark Streaming.[10] It streams processing, dynamic in nature,

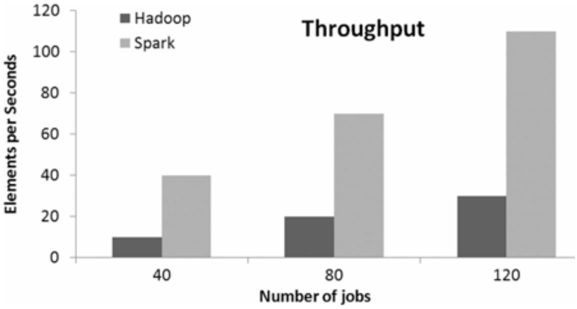

Fig. 4 — High Throughput of Spark

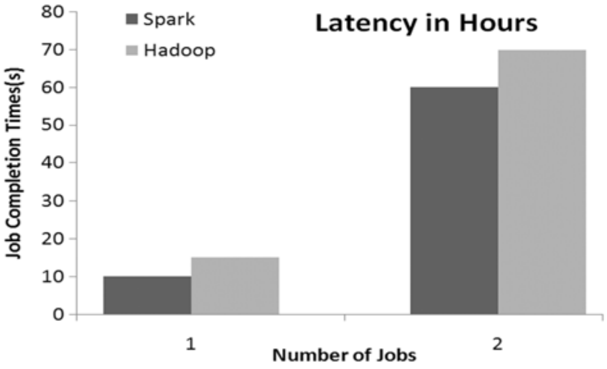

Fig. 5 — Low Latency of Spark

more in-memory computation, reusable, fault tolerant, real-time stream processing capable, lazy evaluation in apache spark, supports multiple languages, active, progressive and expanding spark community, support for sophisticated analysis, integrated with Hadoop, spark graphsx, and cost-effective.[11–14] Spark and Hadoop comparison in terms of Throughput and latency are illustrated in Figs 4 and 5.

Conclusion

However, Map Reduce has many limitations, which are significantly overcoming by Apache latest lightning-fast stream processing framework is a spark. The latest spark framework was finding to be simple, more accurate, precise, specific, reproducible, low latency, more throughput, fault-tolerant, fastest batch and stream processing. Apache spark is a flexible procedure and reduces the drawbacks of Map Reduce.

References

1 Boyi S, Peng Y & Liangcun J, Big spatial data processing with Apache Spark, *6th International IEEE Conference on Agro-Geoinformatics,* 2017, Available from: 10.1109/ Agro Geoinformatics.2017.8047039.

2 Elif Y, Mehmet A, Oya K, Alper, K & Umut T, Data mining library for big data processing platforms: a case study-sparkling water platform, *3rd International Conference on Computer Science and Engineering,* (UBMK), 2018.

3 Deshai N, Sekhar B V D, VenkataRamana S, S, Srinivas K & Varma G P S, Big data hadoop map reduce job scheduling: a short survey, in *Information Systems Design and Intelligent Applications* by S Satapathy, V Bhateja, R Somanah, XS Yang, R Senkerik. Advances in Intelligent Systems and Computing, vol 862. Springer, Singapore, 2019, 349-365, Available from: 10.1007/ 978-981-13-3329-3_3.3

4 Deshai N, Sekhar B V D S, Venkataramana S, Chakravarthy V V S S S & Chowdary P S R, Study with comparing big-data handling techniques using apache hadoop map reduce Vs apache spark, *Int J Eng Technol*, **7(4)** (2018) 4839–4843, Available from:10.14419/ijet.v7i4.1.15997

5 Syue F-H, Kshirsagar V A & Lo S-C, Improving MapReduce load balancing in hadoop,14th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD),(2018) Available from: 10.1109/FSKD.2018.8687158

6 Deshai N, Venkataramana S & PardhaSaradhiVarma G, Performance and cost evolution of dynamic increase hadoop workloads of various datacenters, in *Smart Intelligent Computing and Applications* by S Satapathy, V Bhateja, S Das, Springer, Singapore, Smart Innovation, Systems and Technologies, **105** (2019) 505–516. Available from: 10.1007/978-981-13-1927-3_54

7 Sumitra Srinivas K & Gangadhara Rao Kancharla, Feature selection in big data using filter based techniques, *4th MEC International Conference on Big Data and Smart City (ICBDSC)*, 2019, 1–7, Available from:10.1109/ ICBDSC.2019.8645573 ·

8 Davor S & Ervin V, Apache spark as distributed middleware for power system analysis, *25th Telecomm Forum (TELFOR)* (Belgrade, Serbia), 2017, Available from:10.1109/TEL FOR.2017.8249455

9 Deshai N, SaradhiVarma G, P & Venkataramana S, A study on analytical framework to breakdown conditions among data quality measurement, *International Conference on Innovative Research in Science and Technology*, 7, 2018 Available from: 10.14419/ijet.v7i1.1.9276

10 Deshai N, Venkataramana S, Hemalatha. I & Varma G P S, A Study on big data hadoop map reduce job scheduling, *International Conference on Innovative Research in Science and Technology*, 7, 2017 Available from: 10.14419/ijet.v7i3.31.18202

11 Deshai N & SaradhiVarma G P, Big data challenges and analytics processing over health prescriptions, J Adv Res Dyn Control Syst, 15 (2017).

12 Georgios G, Big data software analytics with apache spark, in *IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion)*, 2018.

13 Siva Kumar Seelam & Pattabiraman V, Acceleration of hadoop MapReduce using in-memory computing, *International Conference on Recent Trends in Advance Computing (ICRTAC)*, 2018, Available from: 10.1109/ICRTAC.2018.8679199

14 Sekhar B V D S, Prasard Reddy P V G D & Varma G P S , Performance of secured and robust watermarking using evolutionary computing technique, *J Glob Inf Manag* (JGIM) **25(4)** (2017) 61–79, Available from: RePEc:igg:jgim 00:v:25:y:2017:i:4:p:61-79