ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 3, March 2018

A Comparison on Hadoop and Spark

Namrata Pandey[1], Rajeshwari S[2], Shobha Rani BN[3], Mrs. Mounica B[4]

B.E. Student, Dept. of Information Science, New Horizon College of Engineering, Bangalore, India[1]

B.E. Student, Dept. of Information Science, New Horizon College of Engineering, Bangalore, India[2]

B.E. Student, Dept. of Information Science, New Horizon College of Engineering, Bangalore, India[3]

Sr. Asst Professor, Dept. of Information Science, New Horizon College of Engineering, Bangalore, India[4]

ABSTRACT: Data is been increasing at an exponential rate in recent times. The data collected has to be processed and analyzed carefully. Traditional algorithms and technologies are insufficient to process this data. The big data evolution is dominating in today's market. Many companies are exploring opportunities in this area of big data. To solve this problem Big Data frameworks are required. Hadoop and Spark, which are used to efficiently process the large data. With multiple big data frameworks available on the market, choosing the right one is a challenge. A detailed comparison is done between the two techniques based on their performance in terms of memory, execution time, speed and applications used. Experimental results show that Spark is more efficient than Hadoop. However, spark requires higher memory allocation, so the choice depends on performance level and memory constraints.

I. INTRODUCTION

An enormous amount of data is being generated around the world every second. This data has been continuously increasing from various platforms like social networks, e-mail, smart phones, tablets, sensors, etc. The data collected can be structured or unstructured. This huge data is kept to store and analyze every day. This large amount of data requires adequate operations to extract maximum information in minimum time.

Data warehouses are a huge database management system that is used for read-only queries for structured data. It is mainly a relational database, which is used for querying and analyzing the transaction data. Data warehouse gathers the information and delivers the processed data to their respective user. To load the data in data warehouse, it uses extraction, transformation, and loading (ETL) technique. The main disadvantage in data warehousing is its cost and flexibility. Since it is built on proprietary hardware, it is more expensive than other approaches. When it comes to flexibility, the traditional data warehouse could only operate on the data which it already has. It is not suitable for handling unstructured data.

Big data refers to the data sets whose volume and variety of data cannot match the analysis capacity of existing traditional database. Big data is used to find meaningful data and hidden pattern to understand the current market trends and help in other business application. To manage the power of big data, we require an infrastructure that can handle and process large amount of structured and unstructured data in real time while providing data privacy and security. There are a lot of big data platform like Apache Hadoop, Apache Spark, Apache Storm, etc to handle this huge amount of data.

On the other hand, Hadoop can be seen as an extension of data warehouse. Hadoop is used to manage and process a huge volume of structured and unstructured data, unlike data warehouse which only processes structured data. Data warehousing could no longer be used for handling ever increasing data storage and processing demand. Hadoop provides better scalability than data warehouse.

In this paper, we are going to discuss about Apache Hadoop and Apache Spark in terms of their performance.

II. HADOOP MAPREDUCE

The Hadoop is an open source framework written in Java that allows us to process large datasets across the clusters of a computer. When we talk about Hadoop, there are two important components of Hadoop framework i.e., HDFS (Hadoop Distributed File System) and MapReduce. Hadoop MapReduce allows parallel processing of huge amount of

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 3, March 2018

data. It breaks a large chunk into smaller ones to be processed separately on different data nodes and automatically gathers the results across the multiple nodes to return a single result. It can handle structured, semi-structured and unstructured data. It provides a cost-effective storage solution for large data volumes with no format requirements. The framework includes scheduling tasks, monitoring them and re-executing any failed tasks. Hadoop can scale up from single servers to thousands of machines, each providing storage, and computation.
Hadoop = HDFS + MapReduce

A) MapReduce

MapReduce is usually considered as a simple programming model for large data processing in a parallel manner. It has master-slave architecture. MapReduce divides the given tasks into subtasks and required operation is performed in a parallel way. It has two basic operations that are Mapper and Reducer function.

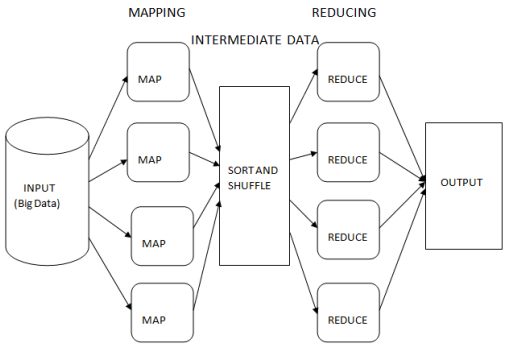

Fig 1: Architecture of MapReduce

Mapper

Maps are the tasks that are used to transform input records into intermediate records. The transformed intermediate records do not required to be of the same type as the input records. The input pair may maps to zero or many output pairs. Output pairs do not need to be of the same types as input pairs.
It is used on all input key-value pairs, which in turn generates a number of key-value pairs. Mapper then sorts the set of key-value pair by the key. The keys and values must be of the same type as the mapper output then partitioners divide the intermediate key space and assign key-value pair to reducers. The intermediate value and sorted outputs are always stored in a simple (key-len, key, value-len, value) format. The number of maps is usually driven by the total size of the inputs, that is, the total number of blocks of the input files.

Reducer

All intermediate values associated with a given output key are subsequently grouped by the framework, and passed to these Reducers to determine the final output. The input from the multiple mapper output partitions are gathered by the reducer.
Reducer has 2 primary phases: shuffle and sort.

Shuffle

Input to the Reducer is the sorted output of the mappers. In this stage the framework fetches the relevant partition of the output of all the mappers, via HTTP.

Sort

The framework groups Reducer inputs by keys (since different mappers may have output the same key) in this stage.
The shuffle and sort phases occur simultaneously; while map-outputs are being fetched they are merged. The output of the reduce task is typically written to the File System.

B) HDFS (Hadoop Distributed File System)

A large amount of data cannot be stored on the single node, therefore, Hadoop uses a file system called HDFS which splits the data into several smaller parts and distribute each part across multiple nodes. This HDFS is mainly designed for storage of very large datasets and to stream data as per the user application. The main objective of is to store data

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 3, March 2018

reliably even in the presence of failures. HDFS uses a master/slave architecture in which one device i.e., the master controls one or more devices i.e., the slaves.

III. SPARK

Apache Spark is an open source framework to process Big Data, which facilitates the users to run large chunks data analytics applications across the clustered system. It provides fast computation processing. Spark has several advantages over other Big Data technologies like Hadoop MapReduce and Apache Storm. Spark supports SQL queries, streaming data features, machine learning and graph algorithm.

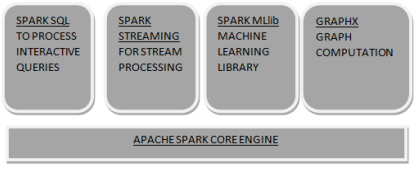

Fig 2: Components of Spark

The main occupation of spark is the resilient distributed datasets (RDDs). These are the concept at the heart of the spark framework. RDDs can deal with any type of data and is stored by spark on different partition. Apache Spark requires a cluster manager and a distributed storage system. For cluster management, Spark supports stand alone(native Spark cluster), Hadoop YARN, or Apache Mesos. For distributed storage, Spark can interface with a wide variety, including Hadoop Distributed File System (HDFS).

- Spark SQL: It is a component on top of Spark Core that introduced a data abstraction called Data Frames, which provides support for structured and semi-structured data. Spark SQL provides a domain-specific language (DSL) to manipulate Data Frames in Scala , Java, or Python. It also provides SQL language support, with command-line interfaces and ODBC/JDBC server.
- Spark Streaming: This module creates small batches of data and RDD is performed on these small batches. It allows scalable and fault tolerant processing. It can be used for real-time processing of data flow.
- Spark MLlib: It is a distributed machine learning framework. It provides multiple types of machine learning algorithms. It includes clustering, classification and regressions. It makes machine learning scalable and easy. It provides tools for construction, evaluation and tuning ML pipelines.
- GraphX: It is the new API for the treatment of graphs including parallelization.

IV. LITERATURE SURVEY

In Hibench benchmark experiment, it is used for measuring the effectiveness of any computer system. The computations have been performed many times and result values are evaluated for both Hadoop and spark.
The following are the results that have been concluded:
i) Execution Time: In most of the cases Spark outperforms Hadoop. Since Hadoop creates multiple objects for a single input. The data sharing is relatively slow in MapReduce because of duplication, serialization and the disk IO. 90% of the time is spent is spent in HDFS read/write operation.
ii) Throughput: It measures how much data can be processed in a given amount of time. In this experiment, the throughput of spark is comparatively higher than Hadoop clusters.
iii) Memory Consumption: In general, spark requires more memory resource than Hadoop. If the input size is huge, spark will be slower than Hadoop because the system does not have too much of memory since spark cannot create new RDDs, the performance will be comparatively less.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 3, March 2018

In collaborative filtering algorithm using Spark and MapReduce, E-commerce datasets are used to compare the performance of Spark and MapReduce. The amount of time required for spark is comparatively less than Hadoop. Spark performs data streaming parallelly. In case of the iterative algorithm, MapReduce involves several read/write tasks to the disk. Each task involves high latency. The next task can only begin after the completion of the previous operation. MapReduce lacks mechanism of message passing. In this process, data should be passed from one node to the other node. Hence MapReduce is efficient but fails to handle the multistage and complex algorithm. On the other sides, spark consists of a module GraphX, i.e., graph computation library, that helps to overcome the shortcomings.

V. COMPARISON ON HADOOP AND SPARK

Data Processing
Spark- The data is stored in the memory and is sent to the disk only when it is required. The time which is required to read/write data is saved. It is good for batch processing and streaming workload.
Hadoop- It stores the data on the disk, so it can handle large datasets. Since the data works in various sequential steps, the data processing is less.
Real-time Analysis
Spark-The data in spark can be processed at the rate of millions per second. Hence spark is good for real-time processing.
Hadoop- Hadoop is not suitable for real-time data processing, as it was designed for handling a large amount of data.
Easy to use
Spark- It is user-friendly as the spark has APIs for java, python, SQL, and Scala. Since it performs batch processing, streaming in the same clusters, it easy to simplify the infrastructure for data processing.
Hadoop- On the other hand, Hadoop is written in Java, which is difficult to program. Hadoop does not have any interactive mode like Spark.
Fault Tolerance
Spark- It uses RDDs and several data storage models for fault tolerance by minimizing network I/O. In the event of data loss, RDD recovers the data through the information it already has.
Hadoop- Hadoop uses replication to achieve fault tolerance. In the event of data loss, the replicated data is used.
Security
Spark- Spark doesn't provide better security feature. It supports authentication through shared secret password which can be used by the organization.
Hadoop- Hadoop provides better security features than the spark. Hadoop MapReduce can be used in security projects, like Knox gateway and sentry.
Cost
Both Hadoop and spark are open source project, so the cost is free.
Spark- requires more amount of RAM in memory and RAM is more expensive than hard disks.
Hadoop-If the organization's requirements are about processing larger amounts of Big Data, Hadoop will be cheaper, since hard disk spaces come at a lower rate than memory space.

VI. CONCLUSION

It's your particular business needs that should determine the choice of a framework. Hadoop MapReduce has the main advantage of linear processing of a large number of datasets. It is an economical solution if no immediate results are expected. If processing of data can be done during night hours, Hadoop MapReduce can be considered, while Spark, on the other hand, delivers fast performance, iterative processing, real-time analytics, graph processing, machine learning and more. In many cases, Spark may outperform Hadoop MapReduce. The good news is the Spark is fully compatible with the Hadoop eco-system and works smoothly with Hadoop Distributed File System, Apache Hive etc.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 3, March 2018

REFERENCES

[1] Akaash Vishal Hazarika, G Jagadeesh Sai Raghu Ram, Eeti Jain, "Performance comparison of Hadoop and spark engine", IEEE International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), pp.671-774, 2017.
[2] Vladyslav Taran, Oleg Alienin, Sergii Stirenko, Yuri Gordienko, A. Rojbi, "Performance evaluation of distributed computing environments with Hadoop and Spark frameworks", IEEE International Young Scientists Forum on Applied Physics and Engineering (YSF), pp. 80-83, 2017.
[3] Yassir Samadi, Mostapha Zbakh, Claude Tadonki, "Comparative study between Hadoop and Spark based on Hibench benchmarks" IEEE 2nd International Conference on Cloud Computing Technologies and Applications (CloudTech), pp.267-275, 2016.
[4] Arya Shrihari Ramani, R. Sasikala, "Collaborative filtering algorithm using spark and MapReduce", IEEE International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), pp. 1-7, 2017.
[5] Ankush Verma, Ashik Hussain Mansuri, Neelesh Jain, "Big data management processing with Hadoop MapReduce and spark technology: A comparison", IEEE Symposium on Colossal Data Analysis and Networking (CDAN), pp. 1-4, 2016.
[6] You Dai, Jin Yan, Xiaoxin Tang, Han Zhao and Minyi Guo, "Online Credit Card Fraud Detection: A Hybrid Framework with Big Data Technologies", IEEE Trustcom/BigDataSE/ISPA, pp. 1644-1651, 2016.
[7] Y. Kltr M. U. alayan, "cardholder behavior model for detecting credit card fraud", IEEE 9th International Conference on Application of Information and Communication Technologies (AICT), pp. 148-152, 2015.
[8] Mohammad Reza HaratiNik, Mahdi Akrami; Shahram Khadivi, Mahdi Shajari, "FUZZGY: A hybrid model for credit card fraud detection", IEEE 6th International Symposium on Telecommunications (IST), pp. 1088-1093, 2012.
[9] Divya. Iyer, Arti Mohanpurkar; Sneha Janardhan; Dhanashree Rathod; Amruta Sardeshmukh, "Credit card fraud detection using Hidden Markov Model", IEEE World Congress on Information and Communication Technologies, pp. 1062-1066, 2011.
[10] Neha Bharill, Aruna Tiwari, Aayushi Malviya, "Fuzzy Based Clustering Algorithms to Handle Big Data with Implementation on Apache Spark", IEEE Second International Conference on Big Data Computing Service and Applications (BigDataService), pp. 95-104, 2016.