International Journal of Modern Computer Science & Engineering, 2016, 5(1): 37-46

Modern Scientific Press

International Journal of Modern Computer Science & Engineering **ISSN: 2325-0798**

Journal homepage:www.ModernScientificPress.com/Journals/ijmcse.aspx *Florida, USA*

Article

A Compendious Research on Big Data and Hadoop

Shireesh Pyreddy

Computer Science Engineering, Brindavan Institute of Technology and Science, Kurnool, India

E-Mail: pyreddyshireesh@gmail.com

Article history: Received 24 April 2016, Revised 16 June 2016, Accepted 20 July 2016, Published 27 July 2016.

Abstract: Big data is a term that describes the large volume of both structured and unstructured data that inundates a business on a day-to-day basis. Due to the fact that the database systems like RDBMS can process the unstructured data but RDBMS finds it challenging to handle such huge data volumes. To deal with the challenges Hadoop is used. It is software framework for distributed storage and distributed processing of very large data sets. MapReduce is a back-end processing engine which is used to perform operations like map and reduce and makes the data simple and stores it into HDFS.

Keywords: Keywords: RDBMS, Hadoop, MapReduce, HDFS, Big data, Spark.

1. Introduction

In the old days, we would utilize systems to extract, transform and load data (ETL) into giant data warehouses that had business intelligence solutions built over them for reporting. Periodically, all the systems would backup and combine the data into a database where reports could be run and everyone could get insight into what was going on.

The problem was that the database technology simply couldn't handle multiple, continuous streams of data. It couldn't handle the volume of data. It couldn't modify the incoming data in real-time. And reporting tools were lacking that couldn't handle anything but a relational query on the back-end. Big Data solutions offer cloud hosting, highly indexed and optimized data structures, automatic archival and extraction capabilities, and reporting interfaces have been designed to provide more accurate analyses that enable businesses to make better decisions.

Copyright © 2016 by Modern Scientific Press Company, Florida, USA

Big data means really a big data, it is a collection of large datasets that cannot be processed using traditional computing techniques. Big data is not merely a data, rather it has become a complete subject, which involves various tools, techniques and frameworks.

Due to the advent of new technologies, devices, and communication means like social networking sites, the amount of data produces by mankind is growing rapidly every year. The amount of data produced by us from the beginning of time till 2003 was 5 billion gigabytes. If you pile up the data in the form of disks it may fill an entire football field. The same amount was created in every two days in 2011m and in every ten minutes in 2013. This rate is still growing enormously. Though all this information produced is meaningful and can be useful when processed, it is being neglected. 90%of the world's data was generated in the last few years.

1.1. Traditional Approach

In this approach, an enterprise will have a computer to store and process big data. Here data will be stored in an RDBMS like Oracle Database, MS SQL Server or DB2 and sophisticated software's can be written to interact with the database, process the required data and present it to the users for analysis purpose.

1.2. Limitation

This approach works well where we have less volume of data that can be accommodated by standard database servers, or up to the limit of the processor which is processing the data. But when it comes to dealing with huge amounts of data, it is really a tedious task to process such data through a traditional database server.

2. Benefits of Big Data

Big data is really critical to our life and its emerging as one of the most important technologies in modern world. Using the information kept in the social network like Facebook, the marketing agencies are learning about the response for their campaigns, promotions, and other advertising mediums. The information in the social media like preferences and product perception of their consumers, product companies and retail organizations are planning their production. The data regarding the previous medical history of patients, hospitals are providing better and quick service.

Big data technologies are important in providing more accurate analysis, which may lead to more concrete decision-making in greater operational efficiencies, cost reductions, and reduced risks for the business. To harness the power of big data, you would require an infrastructure that can manage and process huge volumes of structured and unstructured data in real-time and can protect data privacy and

Copyright © 2016 by Modern Scientific Press Company, Florida, USA

security. There are various technologies in the market from different vendors including Amazon, IBM, Microsoft, etc., to handle big data. While looking into the technologies that handle big data, we examine the following two classes of technology.

Table 1. Operational VS. Analytical Systems

Characteristics	**Operational**	**Analytical**
Latency	1ms - 100ms	1min – 100min
Concurrency	1000 – 100,000	1 - 10
Access pattern	Writes and reads	Reads
Data scope	operational	retrospective
End user	customer	data scientist
Technology	NoSQL	Map Reduce, MPP Data base

3. Hadoop

In this approach, an enterprise will have a computer to store and process big data. Here data will be stored in an RDBMS like Oracle Database, MS SQL Server or DB2 and sophisticated software's can be written to interact with the database, process the required data and present it to the users for analysis purpose.

This approach works well where we have less volume of data that can be accommodated by standard database servers, or up to the limit of the processor which is processing the data. But when it comes to dealing with huge amounts of data, it is really a tedious task to process such data through a traditional database server.

Google solved this problem using an algorithm called Map Reduce. This algorithm divides the task into small parts and assigns these parts to many computers connected over the network, and collects the results to form the final result dataset.

Doug Cutting, Mike Cafarella and team took the solution provided by Google and started an open source project call hadoop in 2005 and Daug named it after his son's toy elephant. Now apache hadoop is a registered trademark of the apache software foundation.

Hadoop runs applications using the Map Reduce algorithm, where the data is processed in parallel on different CPU nodes. In short, hadoop framework is capable enough to develop applications capable of running on clusters of computers and they could perform complete statistical analysis for a huge amounts of data.

Copyright © 2016 by Modern Scientific Press Company, Florida, USA

4. Hadoop Distributed File System (HDFS)

The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware. HDFS provides high throughput access to application data and is suitable for applications that have large data sets. HDFS relaxes a few POSIX requirements to enable streaming access to file system data. HDFS was originally built as infrastructure for the Apache Nutch web search engine project. HDFS is now an Apache Hadoop subproject. HDFS is a Java-based file system that provides scalable and reliable data storage, and it was designed to span large clusters of commodity servers.

HDFS has demonstrated production scalability of up to 200 PB of storage and a single cluster of 4500 servers, supporting close to a billion files and blocks. Hadoop distributed file system was developed using distributed file system design. It is run on commodity hardware. Unlike other distributed systems, HDFS is highly fault tolerant and designed using low-cost hardware.

HDFS holds very large amount of data and provides easier access. To store such huge data, the files are stored across multiple machines. These files are stored in redundant fashion to rescue the system from possible data losses in case of failure. HDFS also makes applications available to parallel processing.

4.1. NameNode

The NameNode is commodity hardware that contains the GNU/Linux operating system and the NameNode software. It is a software that can be run on commodity hardware. The system having the Name Node act as the maser server and it does the following tasks like manages the file system namespace, regulates the client's access to the files and it also executes file system operations such as renaming, closing, and opening files and directories.

4.2. DataNode

The DataNode is a commodity hardware having the GNU/Linux operating system and DataNode software. For every node in cluster, there will be a data node. These nodes manage the data storage of their system. Data nodes perform read-write operations on the file systems, as per client request. They also perform operations such as block creation, deletion, and replication according to the instructions of the NameNode.

4.3. Block

Copyright © 2016 by Modern Scientific Press Company, Florida, USA

Generally, the user data is stored in the file of HDFS. The file in a file system will be divided into one or more segments and/or stored in individual data nodes. These files are called as blocks. In other words, the minimum amount of data that HDFS can read or write is called a block. The default block size is 64MB, but it can be increased as per the need to change in HDFS configuration.

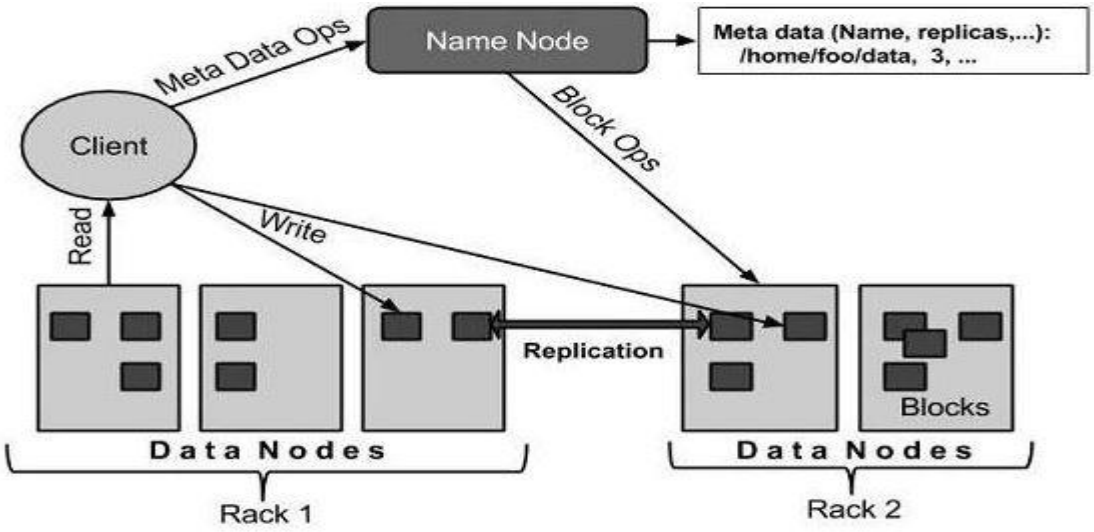

Figure 1. HDFS Architecture

5. MapReduce

MapReduce is a framework using which we can write applications to process huge amounts of data, in parallel, on large clusters of commodity hardware in a reliable manner.

MapReduce is a processing technique and a program model for distributed computing based on java. The mapreduce algorithm contains two important tasks, namely map and reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into smaller set of tuples. As the sequence of the name mapreduce implies, the reduce task is always performed after the map job.

The major advantage of mapreduce is that it is easy to scale data processing over multiple computing nodes. Under the MapReduce mode, the data processing primitives are called mappers and reducers. Decomposing a data processing applications into mappers and reducers is sometimes nontrivial. But, once we write an application in the MapReduce form, scaling the application to run over hundred, thousands, or even tens of thousands of machines in a cluster is merely a configuration change. This simple scalability is what has attracted many programmers to use the MapReduce model.

5.1. Job Tracker

Copyright © 2016 by Modern Scientific Press Company, Florida, USA

The Job Tracker is the service within Hadoop that farms out MapReduce tasks to specific nodes in the cluster, ideally the nodes that have the data, or at least are in the same rack. Client applications submit jobs to the Job tracker. The Job Tracker talks to the NameNode to determine the location of the data. The Job Tracker locates Task Tracker nodes with available slots at or near the data. The Job Tracker submits the work to the chosen Task Tracker nodes.

The Task Tracker nodes are monitored. If they do not submit heartbeat signals often enough, they are deemed to have failed and the work is scheduled on a different Task Tracker. A Task Tracker will notify the Job Tracker when a task fails. The Job Tracker decides what to do then: it may resubmit the job elsewhere, it may mark that specific record as something to avoid, and it may even blacklist the Task Tracker as unreliable. When the work is completed, the Job Tracker updates its status. Client applications can poll the Job Tracker for information.

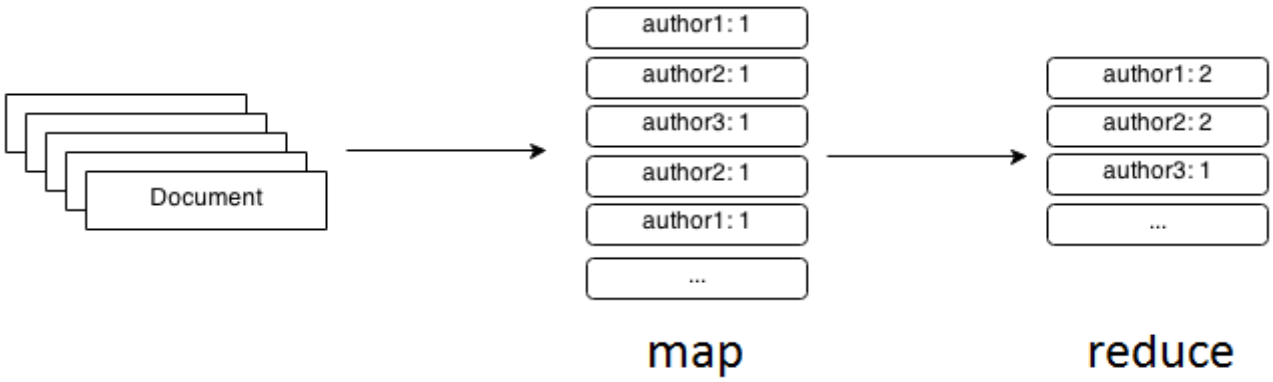

Figure 2. MapReduce

5.2. Task Tracker

A Task Tracker is a node in the cluster that accepts tasks - Map, Reduce and Shuffle operations - from a Job Tracker. Every Task Tracker is configured with a set of slots; these indicate the number of tasks that it can accept. When the Job Tracker tries to find somewhere to schedule a task within the MapReduce operations, it first looks for an empty slot on the same server that hosts the DataNode containing the data, and if not, it looks for an empty slot on a machine in the same rack. The Task Tracker spawns a separate JVM processes to do the actual work; this is to ensure that process failure does not take down the task tracker.

The TaskTracker monitors these spawned processes, capturing the output and exit codes. When the process finishes, successfully or not, the tracker notifies the Job Tracker. The Task Trackers also send out heartbeat messages to the JobTracker, usually every few minutes, to reassure the Job Tracker that it is still alive. These message also inform the Job Tracker of the number of available slots, so the Job Tracker can stay up to date with where in the cluster work can be delegated.

Copyright © 2016 by Modern Scientific Press Company, Florida, USA

6. Spark

6.1. Why Spark, why not MapReduce?

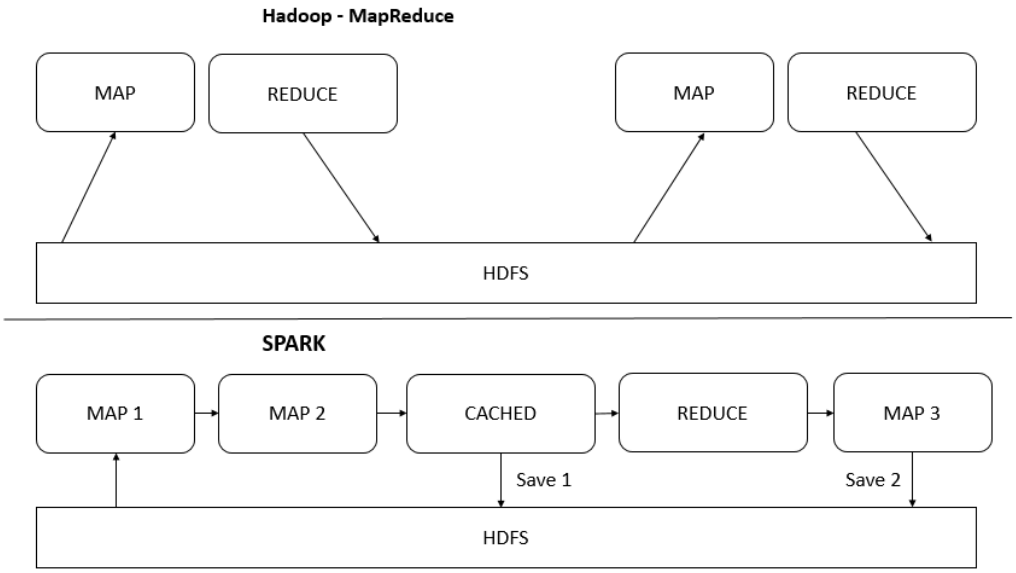

Figure 3. Why Spark, why not MapReduce?

Hadoop - MapReduce is parallel data processing framework that has traditionally been used to run map/reduce jobs. These are long running jobs that take minutes or hours to complete. Spark has designed to run on top of Hadoop and it is an alternative to the traditional batch map/reduce model that can be used for real-time stream data processing and fast interactive queries that finish within seconds. So, Hadoop supports both traditional map/reduce and Spark. We should look at Hadoop as a general purpose Framework that supports multiple models and we should look at Spark as an alternative to Hadoop MapReduce rather than a replacement to Hadoop.

Spark uses more RAM instead of network and disk I/O its relatively fast as compared to hadoop. But as it uses large RAM it needs a dedicated high end physical machine for producing effective results. It all depends and the variables on which this decision depends keep on changing dynamically with time.

Apache Spark is a lightning-fast cluster computing technology, designed for fast computation. It is based on Hadoop MapReduce and it extends the MapReduce model to efficiently use it for more types of computations, which includes interactive queries and stream processing. Spark uses Hadoop in two ways – one is storage and second is processing. Since Spark has its own cluster management computation, it uses Hadoop for storage purpose only. The main feature of Spark is its in-memory cluster computing that increases the processing speed of an application. Spark is designed to cover a wide range of workloads such as batch applications, iterative algorithms, interactive queries and streaming. Apart from supporting all these workloads in a respective system, it reduces the management burden of maintaining separate tools.

Copyright © 2016 by Modern Scientific Press Company, Florida, USA

Table 2. Analyzation between Hadoop and Spark

Hadoop	**Spark**
Distributed Storage + Distributed Computation	Distributed Computation Only
MapReduce Framework	Generalized Computation
Usually data on disk (HDFS)	On disk / in memory
Not ideal for iterative work	Great at Iterative workloads (machine learning…etc)
Batch Process	- Upto 2x – 10x faster for data on disk - Upto 100x faster for data in memory

Apache Spark is a cluster computing platform designed to be fast and general-purpose. On the speed side, Spark extends the popular MapReduce model to efficiently support more types of computations, including interactive queries and stream processing. Speed is important in processing large datasets as it means the difference between exploring data interactively and waiting minutes between queries, or waiting hours to run your program versus minutes. One of the main features Spark offers for speed is the ability to run computations in memory, but the system is also faster than MapReduce for complex applications running on disk.

Spark is designed to be highly accessible, offering simple APIs in Python, Java, Scala and SQL, and rich built-in libraries. It also integrates closely with other big data tools. In particular, Spark can run in Hadoop clusters and access any Hadoop data source, including Cassandra.

6.2. Spark SQL

Spark SQL is a Spark module for structured data processing. Unlike the basic Spark RDD API, the interfaces provided by Spark SQL provide Spark with more information about the structure of both the data and the computation being performed. Internally, Spark SQL uses this extra information to perform extra optimizations. There are several ways to interact with Spark SQL including SQL, the DataFrames API and the Datasets API. When computing a result, the same execution engine is used, independent of which API/language you are using to express the computation. This unification means that developers can easily switch back and forth between the various APIs based on which provides the most natural way to express a given transformation. All of the examples on this page use sample data included in the Spark distribution and can be run in the spark-shell.

One use of Spark SQL is to execute SQL queries written using either a basic SQL syntax or HiveQL. Spark SQL can also be used to read data from an existing Hive installation. For more on how

Copyright © 2016 by Modern Scientific Press Company, Florida, USA

to configure this feature, please refer to the Hive Tables section. When running SQL from within another programming language the results will be returned as a DataFrame.

6.3. Spark Streaming

Spark Streaming is a Spark component that enables processing live streams of data. Examples of data streams include log files generated by production web servers, or queues of messages containing status updates posted by users of a web service. Spark Streaming provides an API for manipulating data streams that closely matches the Spark Core's RDD API, making it easy for programmers to learn the project and move between applications that manipulate data stored in memory, on disk, or arriving in real-time. Underneath its API, Spark Streaming was designed to provide the same degree of fault tolerance, throughput, and scalability that the Spark Core provides.

Internally, it works as follows. Spark Streaming receives live input data streams and divides the data into batches, which are then processed by the Spark engine to generate the final stream of results in batches.

Spark Streaming provides a high-level abstraction called discretized stream or DStream, which represents a continuous stream of data. DStreams can be created either from input data streams from sources such as Kafka, Flume, and Kinesis, or by applying high-level operations on other DStreams. Internally, a DStream is represented as a sequence of RDDs.

This guide shows you how to start writing Spark Streaming programs with DStreams. You can write Spark Streaming programs in Scala, Java or Python (introduced in Spark 1.2), all of which are presented in this guide. You will find tabs throughout this guide that let you choose between code snippets of different languages.

7. Conclusion

As the data increases in a humongous fashion, normal databases like RDBMS struggle to handle and process the huge data. To handle such huge data, big data came up with some tools and technologies which can do the processing of large datasets of different data formats. One such technology is Hadoop which is used to handle large datasets in distributed environment.

References

[1] N. Gershenfeld, R. Krikorian, and D. Cohen, *The internet of things*, Scientific American, 291(4)(2004): 76.

[2] V. Marx, *Biology:* The big challenges of big data, *Nature*, 498(7453)(2013): 255–260.

Copyright © 2016 by Modern Scientific Press Company, Florida, USA

[3] D. E. O'Leary, Artificial intelligence and big data, *IEEE Intelligent Systems*, 28(2)(2013): 96–99, 2013.

[4] J. Dean and S. Ghemawat, *Mapreduce: Simplified data processing on large clusters*, in OSDI\'04, 2005, pp. 137–150.

Copyright © 2016 by Modern Scientific Press Company, Florida, USA