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. 7, Issue 2, February 2019

An Efficient Map Reduce Framework for E-Commerce Using Hadoop Technology

D.Kalaiabirami[1], B.Suganya[2], D.Nivi[3]

Assistant Professor, Department of Computer Science, Vivekanandha College of Engineering for Women, Thiuchengode, Tamilnadu, India[1]

Student, III CSE, Department of Computer Science, Vivekanandha College of Engineering for Women, Thiuchengode, Tamilnadu, India[2]

Student, III CSE, Department of Computer Science, Vivekanandha College of Engineering for Women, Thiuchengode, Tamilnadu, India[3]

ABSTRACT: The advent of the digital age has led to a rise in different types of data with every passing day. Map reduce is a programming structure for effectively composing requisitions which prepare boundless measures of information (multi-terabyte information sets) in parallel on extensive bunches (many hubs) of merchandise fittings in a dependable, shortcoming tolerant way. Map Reduce is widely been used for the efficient analysis of Big Data. Traditional DBMS techniques like Joins and Indexing and other techniques like graph search is used for classification and clustering of Big Data. These techniques are being adopted to be used in Map reduce. The computer industry is being challenged to develop methods and techniques for affordable data processing on large datasets at optimum response times. Map Reduce provides automatic parallelization and distribution of computation based on several processors. To find the useful information from massive amount of data to organizations, we need to analyze the data. The high scalability of Map Reduce is one of the reasons for adapting this model. Hadoop is an open source; distributed programming framework with enables the storage and processing of large data sets. Map Reduce framework is basically designed to compute data demanding applications to support effective decision making. Since its introduction, remarkable research efforts have been put to make it more familiar to the users subsequently utilized to support the execution of enormous data intensive applications. Map Reduce techniques have been studied at in this paper which is implemented for Big Data analysis using HDFS. In this paper we try to focus especially on Map Reduce with Hadoop for the analytical processing of big data.

I. INTRODUCTION

'Big Data' is a data, but with a huge size. 'Big Data' is a term used to describe a collection of data that is large in size and yet growing exponentially with time. This data is mainly generated in terms of photo and video uploads, message exchanges, putting comments etc….Big data' could be found in three forms: Structured ,Un-structured, Semi-structured. Any data that can be stored, accessed and processed in the form of fixed format is termed as a 'structured' data. 1021 bytes equals to 1 zettabyte or one billion terabytes forms a zettabyte.Data stored in a relational database management system is one example of a 'structured' data. A table in a database is an example of Structured Data. Any data with unknown form or the structure is classified as unstructured data. The need of big data comes from

the Big Companies like yahoo, Google, facebook etc for the purpose of analysis of big amount of data which is in unstructured form. Developers need hundreds or thousands of processing nodes and large volume of storage devices to process complex applications with large datasets, such applications process multi-terabyte to petabyte-sized datasets and using traditional data processing methods like sequential processing and centralized data processing are not

.

effective to solve these kinds of application's problems. Map Reduce system is to validate the proposed framework and to present the evaluation of the experiment based on the criteria such as speed of processing, data-storage usage, response time and cost efficiency. Traditional experience in data warehousing, reporting, and online analytic processing (OLAP) is different for advanced forms of analytics . Organizations are implementing specific forms of analytics, particularly called advanced analytics. These are an collection of related techniques and tool types, usually including predictive analytics, data mining, statistical analysis, complex SQL, data visualization, artificial intelligence, natural language processing.
Hadoop was designed especially for the analysis of large data sets to build scalable, distributed applications. To manage sizably voluminous data, Hadoop implements the paradigm called MapReduce defined by Google according to which the applications are divided into minute chunks of software, each of which can be run on a distinct node of all those who make up the system. The "MapReduce Framework" (likewise called "infrastructure" or "framework") organizes by assembling the distributed servers, running the various tasks in parallel, controlling all communications and data transfers between the pairs and Reduce merges things, so that instead of a set of key/value pair sets.
MapReduce has been facilitated by Google as a programming framework to analyse massive amounts of data. It uses for distributed data processing on large datasets across a cluster of machines. Since the input data is too large, the computation needs to be distributed across thousands of machines within a cluster in order to finish each part of computation in a reasonable amount of time.

A.Big Data Parameters(characteristics) :
As the data is too big from various sources in different form,the data's in the is characterized by the 3 Vs. i.e., Data that has extra-large Volume, comes from Variety of sources, Variety of formats and comes at us with a great Velocity is normally refer to as Big Data. The thee Vs of Big Data are: Variety, Volume and Velocity. Along with the three V's, there also exists ambiguity, viscosity, and virality.
B.Privacy:
Privacy of data is another big problem with big data. There are several strict laws for protecting the data. For example in social media we cannot get the private posts of users for sentiment analysis.

II. IMPORTANCE OF BIG DATA

Big Data can be defined as large volumes of data which is either structured or unstructured and generated at high speeds globally by various new technological devices. Big Data includes the data that is generated every second by sensors, mobiles, and consumer-driven data from social networks. Big Data is evolving from various facets within organizations legal, sales, marketing, procurement, finance, and human resources departments etc.
The significance of Big Data can be characterized as:
1) Big data is a valuable term despite the hype
2) It is gaining more popularity and interest from both business users and IT industry.
3) From an analytics perspective it still represents analytic workloads and data management solutions that could not previously be supported because of cost considerations and/or technology limitations.
4) The solutions provided enable smarter and faster decision making, and allow organizations to achieve faster time to value from their investments in analytical processing technology and products.
5) Analytics on multi-structured data enable smarter decisions. Up till now, these types of data have been difficult to process using traditional analytical processing technologies.
6) Rapid decisions are enabled because big data solutions support the rapid analysis of high volumes ofdetailed data.
7) Faster time to value is possible because organizations can now process and analyze data that is outsideof the enterprise data warehouse.

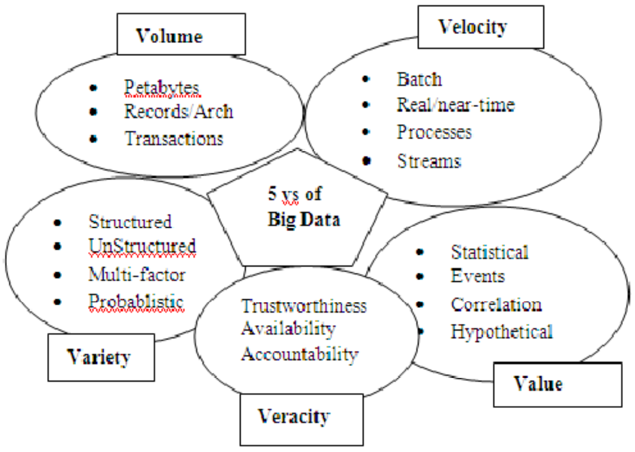
ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

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

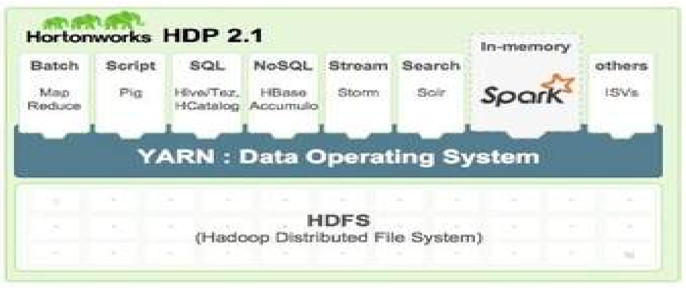
Website: www.ijircce.com

Vol. 7, Issue 2, February 2019



Hadoop:

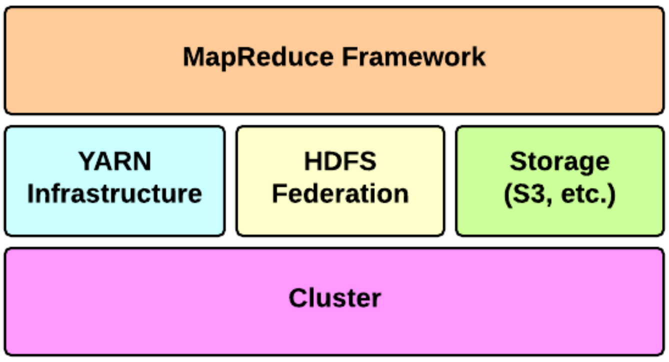
Hadoop is an open-source, java based Apache Software foundation project. It is one of the most interesting third-party implementations of MapReduce for distributed computing. Hadoop provides a framework to support large dataset distributed computing on a cluster of servers.hadoopnow being used by major companies, including Amazon, IMB, Yahoo, Facebook and a growing number of other companies. Hadoop provides its own distributed file system called Hadoop Distributed File System (HDFS), HDFS is a distributed file system designed to store multiple copies of data block on a cluster of compute nodes, to enable reliable and rapid computations. Hadoop uses HDFS to store a dataset and applies the MapReduce's power to distribute and parallelize the processing of this dataset. It means a file into HDFS is divided into 64 MB chunk and each chunk is resided on a different working machine. Hadoop is a framework that can run applications on systems with thousands of nodes and terabytes. This approach reduces the risk of catastrophic system failure. HDFS takes care of storage part of Hadoop applications. MapReduce applications consume data from HDFS.



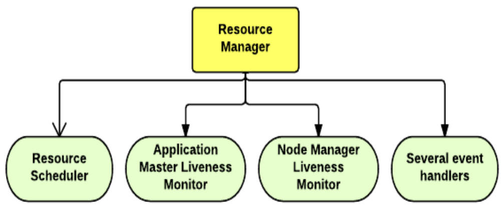
III. ARCHITETURE OF HADOOP

I. Hadoop Architecture Overview

Apache Hadoop is an open-source software framework for storage and large-scale processing of data-sets on clusters of commodity hardware. There are mainly five building blocks inside this runtime envinronment (from bottom to top):

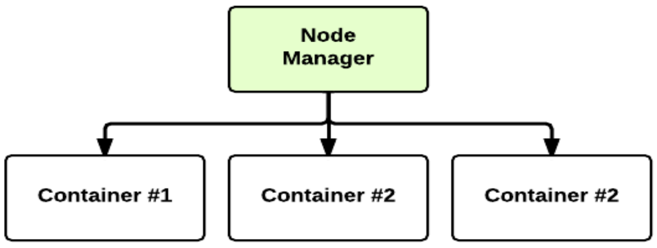


- the **cluster** is the set of host machines (**nodes**). Nodes may be partitioned in **racks**. This is the hardware part of the infrastructure.
- the **YARN Infrastructure** (Yet Another Resource Negotiator) is the framework responsible for providing the computational resources (e.g., CPUs, memory, etc.) needed for application executions. Two important elements are:the **Resource Manager** (one per cluster) is the master. It knows where the slaves are located (Rack Awareness) and how many resources they have. It runs several services,


Resource Manager
Resource Scheduler
Application Master Liveness Monitor
Node Manager Liveness Monitor
Several event handlers

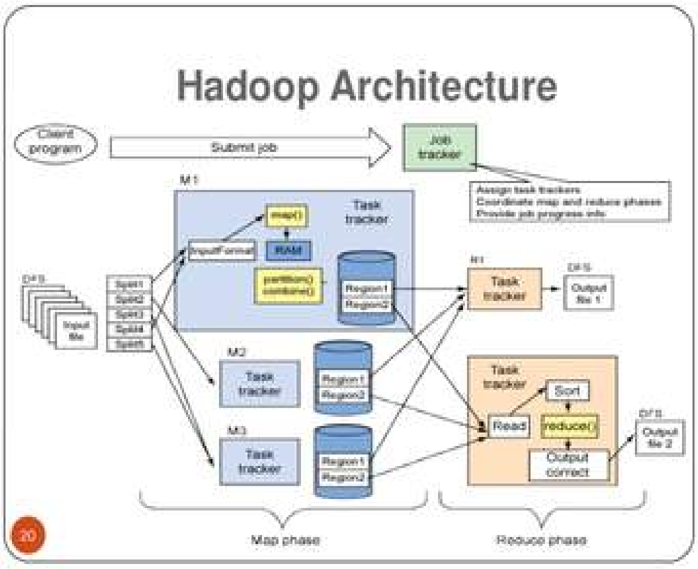
the most important is the **Resource Scheduler** which decides how to assign the resources.

- the **Node Manager** (many per cluster) is the slave of the infrastructure. When it starts, it announces himself to the Resource Manager. Periodically, it sends an heartbeat to the Resource Manager. Each Node Manager offers some resources to the cluster. Its resource capacity is the amount of memory and the number of vcores. At run-time, the Resource Scheduler will decide how to use this capacity: a **Container** is a fraction of the NM capacity and it is used by the client for running a program.



- The **HDFS Federation** is the framework responsible for providing permanent, reliable and distributed storage. This is typically used for storing inputs and output (but not intermediate ones).
- other alternative storage solutions. For instance, Amazon uses the Simple Storage Service (S3).
- The **MapReduce Framework** is the software layer implementing the **MapReduce paradigm**.

- The YARN infrastructure and the HDFS federation are completely decoupled and independent: the first one provides resources for running an application while the second one provides storage. The MapReduce framework is only one of many possible framework which runs on top of YARN (although currently is the only one implemented).



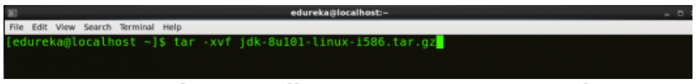
Components of Hadoop:
there are two main components used in hadoop are:
1)Storage: The Hadoop Distributed File System (HDFS): It is a distributed file system which provides fault tolerance and designed to run on commodity hardware. HDFS has master/slave architecture [5]. Files added to HDFS are split into fixed-size blocks. Block size is configurable, but defaults to 64 megabytes.

2) Processing: MapReduce : It is a programming model introduced by Google in 2004,it is the one which processes large amount of data in parallel on large clusters of hardware in fault tolerant manner.in map reduce it uses 2 functions:

A. Map Phase: Initially split the data into key value pair and fed into mapper which in turn process each key value pair and generate intermediate output.

B. Reduce Phase: Map/Reduce is a highly scalable programming paradigm capable of processing massive volumes of data by means of parallel execution on a large number of commodity computing nodes.

HADOOP INSTALLATION:
1) *Step 1:download the Java 8 Package. Save this file in your home directory.*
2) *Step 2: Extract the Java Tar File.*
***Command*:** tar -xvf jdk-8u101-linux-i586.tar.gz



```
[edureka@localhost ~]$ tar -xvf jdk-8u101-linux-i586.tar.gz
```

Fig: Hadoop Installation – Extracting Java Files

3) *Step 3: Download the Hadoop 2.7.3 Package.*
***Command*:** wget https://archive.apache.org/dist/hadoop/core/hadoop-2.7.3/hadoop-2.7.3.tar.gz

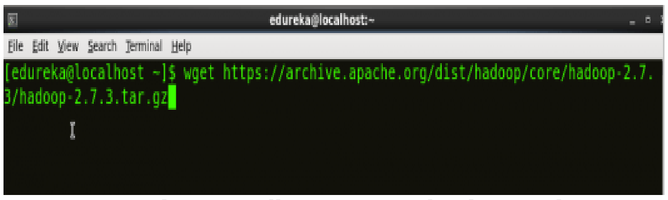
ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

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

Website: www.ijircce.com

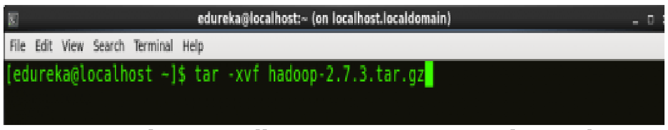
Vol. 7, Issue 2, February 2019



```
[edureka@localhost ~]$ wget https://archive.apache.org/dist/hadoop/core/hadoop-2.7.
3/hadoop-2.7.3.tar.gz
```

Fig: Hadoop Installation – Downloading Hadoop

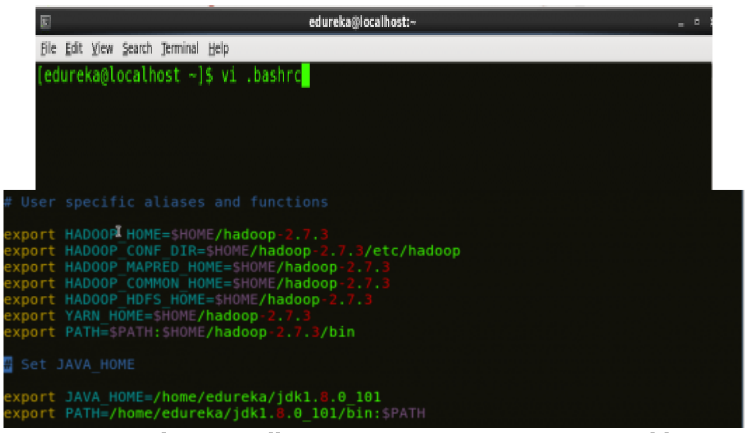
4) Step 4: Extract the Hadoop tar File.
Command: tar -xvf hadoop-2.7.3.tar.gz



```
[edureka@localhost ~]$ tar -xvf hadoop-2.7.3.tar.gz
```

Fig: Hadoop Installation – Extracting Hadoop Files

5) Step 5: Add the Hadoop and Java paths in the bash file (.bashrc).
Open**. bashrc** file. Now, add Hadoop and Java Path as shown below.
Command: vi .bashrc

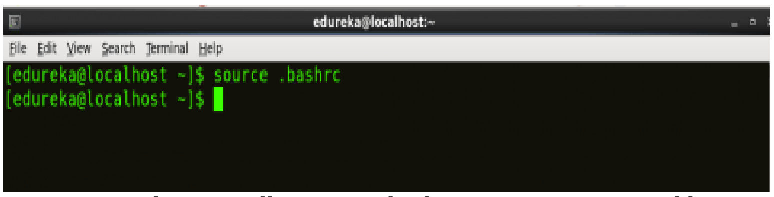


```
[edureka@localhost ~]$ vi .bashrc
```

```
# User specific aliases and functions

export HADOOP_HOME=$HOME/hadoop-2.7.3
export HADOOP_CONF_DIR=$HOME/hadoop-2.7.3/etc/hadoop
export HADOOP_MAPRED_HOME=$HOME/hadoop-2.7.3
export HADOOP_COMMON_HOME=$HOME/hadoop-2.7.3
export HADOOP_HDFS_HOME=$HOME/hadoop-2.7.3
export YARN_HOME=$HOME/hadoop-2.7.3
export PATH=$PATH:$HOME/hadoop-2.7.3/bin

# Set JAVA_HOME

export JAVA_HOME=/home/edureka/jdk1.8.0_101
export PATH=/home/edureka/jdk1.8.0_101/bin:$PATH
```

Fig: Hadoop Installation – Setting Environment Variable

Then, save the bash file and close it.
For applying all these changes to the current Terminal, execute the source command.
Command: source .bashrc



```
[edureka@localhost ~]$ source .bashrc
[edureka@localhost ~]$ 
```

Fig: Hadoop Installation – Refreshing environment variables

To make sure that Java and Hadoop have been properly installed on your system and can be accessed through the Terminal, execute the java -version and hadoop version commands.
Command: java -version

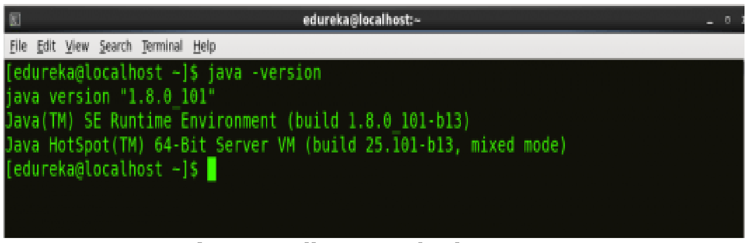
ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

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

Website: www.ijircce.com

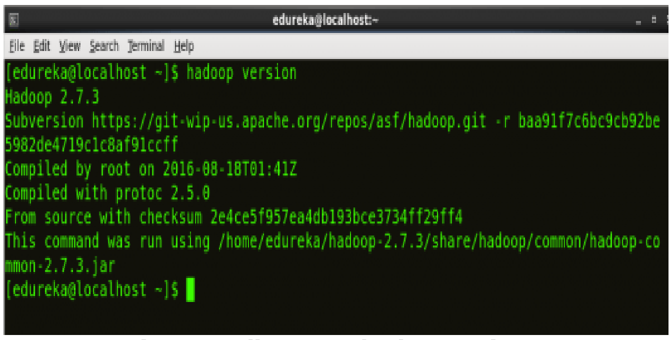
Vol. 7, Issue 2, February 2019



```
[edureka@localhost ~]$ java -version
java version "1.8.0_101"
Java(TM) SE Runtime Environment (build 1.8.0_101-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.101-b13, mixed mode)
[edureka@localhost ~]$
```

Fig: Hadoop Installation – Checking Java Version

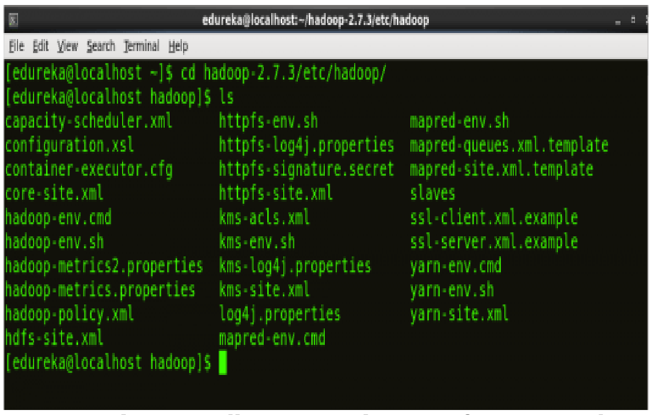
Command: hadoop version



```
[edureka@localhost ~]$ hadoop version
Hadoop 2.7.3
Subversion https://git-wip-us.apache.org/repos/asf/hadoop.git -r baa91f7c6bc9cb92be
5982de4719c1c8af91ccff
Compiled by root on 2016-08-18T01:41Z
Compiled with protoc 2.5.0
From source with checksum 2e4ce5f957ea4db193bce3734ff29ff4
This command was run using /home/edureka/hadoop-2.7.3/share/hadoop/common/hadoop-co
mmon-2.7.3.jar
[edureka@localhost ~]$
```

Fig: Hadoop Installation – Checking Hadoop Version

6) Step 6: Edit the Hadoop Configuration files.
Command: cd hadoop-2.7.3/etc/hadoop/
Command: ls
All the Hadoop configuration files are located in **hadoop-2.7.3/etc/hadoop** directory as you can see in the snapshot below:



```
[edureka@localhost ~]$ cd hadoop-2.7.3/etc/hadoop/
[edureka@localhost hadoop]$ ls
capacity-scheduler.xml      httpfs-env.sh               mapred-env.sh
configuration.xsl           httpfs-log4j.properties     mapred-queues.xml.template
container-executor.cfg      httpfs-signature.secret     mapred-site.xml.template
core-site.xml               httpfs-site.xml             slaves
hadoop-env.cmd              kms-acls.xml                ssl-client.xml.example
hadoop-env.sh               kms-env.sh                  ssl-server.xml.example
hadoop-metrics2.properties  kms-log4j.properties        yarn-env.cmd
hadoop-metrics.properties   kms-site.xml                yarn-env.sh
hadoop-policy.xml           log4j.properties            yarn-site.xml
hdfs-site.xml               mapred-env.cmd
[edureka@localhost hadoop]$
```

Fig: Hadoop Installation – Hadoop Configuration Files

7) Step 7: Open core-site.xml *and edit the property mentioned below inside configuration tag:*
core-site.xml informs Hadoop daemon where NameNode runs in the cluster. It contains configuration settings of Hadoop core such as I/O settings that are common to HDFS & MapReduce.
Command: vi core-site.xml

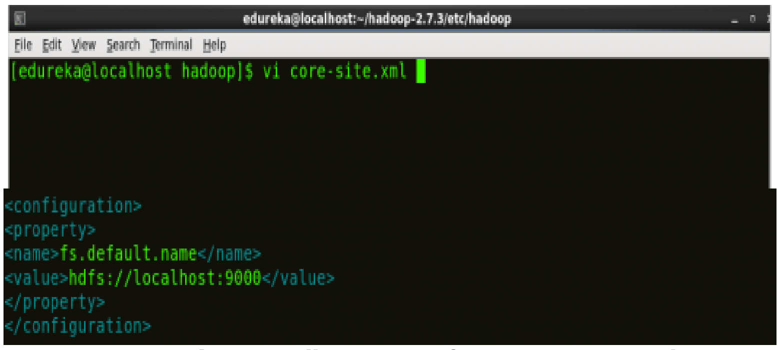
ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

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

Website: www.ijircce.com

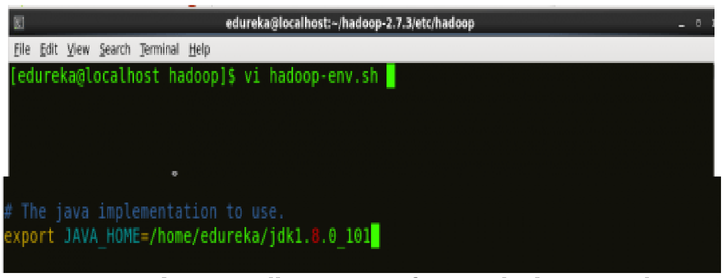
Vol. 7, Issue 2, February 2019



```
edureka@localhost:~/hadoop-2.7.3/etc/hadoop
[edureka@localhost hadoop]$ vi core-site.xml

<configuration>
<property>
<name>fs.default.name</name>
<value>hdfs://localhost:9000</value>
</property>
</configuration>
```

Fig: Hadoop Installation – Configuring core-site.xml

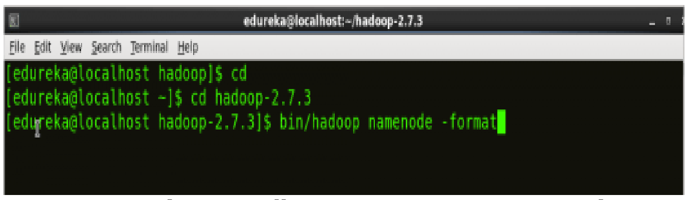
8) Step 8: Edit hadoop-env.sh *and add the Java Path as mentioned below:*
hadoop-env.sh contains the environment variables that are used in the script to run Hadoop like Java home path, etc.
Command: vi hadoop–env.sh



```
edureka@localhost:~/hadoop-2.7.3/etc/hadoop
[edureka@localhost hadoop]$ vi hadoop-env.sh

# The java implementation to use.
export JAVA_HOME=/home/edureka/jdk1.8.0_101
```

Fig: Hadoop Installation – Configuring hadoop-env.sh

9) Step 9: Go to Hadoop home directory and format the NameNode.
Command: cd
Command: cd hadoop-2.7.3
Command: bin/hadoop namenode -format



```
edureka@localhost:~/hadoop-2.7.3
[edureka@localhost hadoop]$ cd
[edureka@localhost ~]$ cd hadoop-2.7.3
[edureka@localhost hadoop-2.7.3]$ bin/hadoop namenode -format
```

Fig: Hadoop Installation – Formatting NameNode

This formats the HDFS via NameNode. This command is only executed for the first time. Formatting the file system means initializing the directory specified by the dfs.name.dir variable.

MAPREDUCE:

A MapReduce program is composed of a map procedure (or method), which performs filtering and sorting (such as sorting students by first name into queues, one queue for each name), and a *reduce* method, which performs a summary operation (such as counting the number of students in each queue, yielding name frequencies). The "MapReduce System" (also called "infrastructure" or "framework") orchestrates the processing by marshalling the distributed servers, running the various tasks in parallel, managing all communications and data transfers between the various parts of the system, and providing for redundancy and fault tolerance.

The model is a specialization of the *split-apply-combine* strategy for data analysis. It is inspired by the map and reduce functions commonly used in functional programming, although their purpose in the MapReduce framework is not the same as in their original forms. The key contributions of the MapReduce framework are not the actual map and reduce functions (which, for example, resemble the 1995 Message Passing Interface standard's*reduce* and *scatter*operations), but the scalability and fault-tolerance achieved for a variety of applications by optimizing the execution engine. Another way to look at MapReduce is as a 5-step parallel and distributed computation:

1. **Prepare the Map() input** – the "MapReduce system" designates Map processors, assigns the input key value *K1* that each processor would work on, and provides that processor with all the input data associated with that key value.
2. **Run the user-provided Map() code** – Map() is run exactly once for each *K1* key value, generating output organized by key values *K2*.
3. **"Shuffle" the Map output to the Reduce processors** – the MapReduce system designates Reduce processors, assigns the *K2* key value each processor should work on, and provides that processor with all the Map-generated data associated with that key value.
4. **Run the user-provided Reduce() code** – Reduce() is run exactly once for each *K2* key value produced by the Map step.
5. **Produce the final output** – the MapReduce system collects all the Reduce output, and sorts it by *K2* to produce the final outcome.

These five steps can be logically thought of as running in sequence – each step starts only after the previous step is completed – although in practice they can be interleaved as long as the final result is not affected.
In many situations, the input data might already be distributed ("sharded") among many different servers.
The *Map* and *Reduce* functions of *MapReduce* are both defined with respect to data structured in (key, value) pairs. *Map* takes one pair of data with a type in one data domain, and returns a list of pairs in a different domain:
Map(k1,v1) → list(k2,v2)
The *Map* function is applied in parallel to every pair (keyed by k1) in the input dataset. This produces a list of pairs (keyed by k2) for each call. After that, the MapReduce framework collects all pairs with

the same key (k2) from all lists and groups them together, creating one group for each key.

The *Reduce* function is then applied in parallel to each group, which in turn produces a collection of values in the same domain:

Reduce(k2, list (v2)) → list(v3)
Each *Reduce* call typically produces either one value v3 or an empty return, though one call is allowed to return more than one value. The returns of all calls are collected as the desired result list.

Thus the MapReduce framework transforms a list of (key, value) pairs into a list of values. This behavior is different from the typical functional programming map and reduce combination, which accepts a list of arbitrary values and returns one single value that combines *all* the values returned by map.

It is necessary but not sufficient to have implementations of the map and reduce abstractions in order to implement MapReduce. Distributed implementations of MapReduce require a means of connecting the processes performing the Map and Reduce phases. This may be a distributed file system. Other options are possible, such as direct streaming from mappers to reducers, or for the mapping processors to serve up their results to reducers that query them.

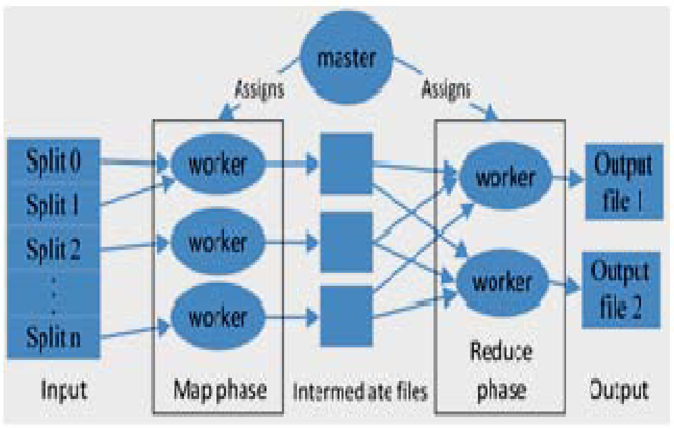
ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

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

Website: www.ijircce.com

Vol. 7, Issue 2, February 2019



Map Reduce workflow

The steps involved in working of MapReduce can be shown in as:
The applications which include indexing and search, graph analysis, text analysis, machine learning, datatransformation and many more, are not easy to implement by making the use of standard SQL which are employed by relational DBMSs. In such areas the procedural nature of MapReduce makes it easily understood by skilled programmers.

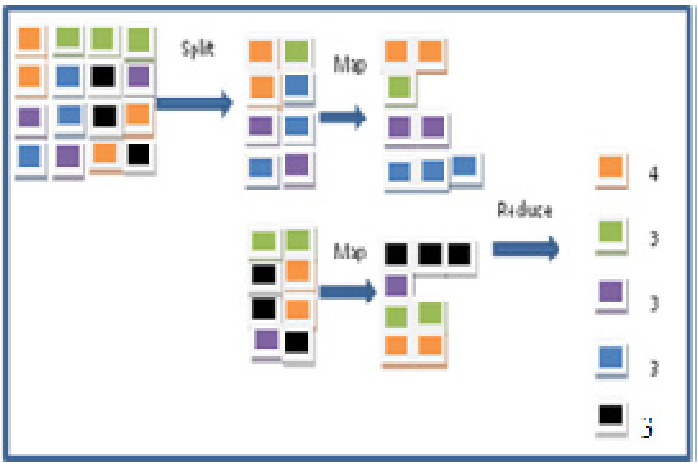


Fig. 1: Steps in MapReduce to process the database

It also has the advantage that developers do not have to be concerned with implementing parallelcomputing – this is handled transparently by the system.

Although MapReduce is designed for programmers, nonprogrammerscan exploit the value of prebuilt MapReduce applications and function libraries. The architecture of MapReduce can be depicted as:

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. 7, Issue 2, February 2019

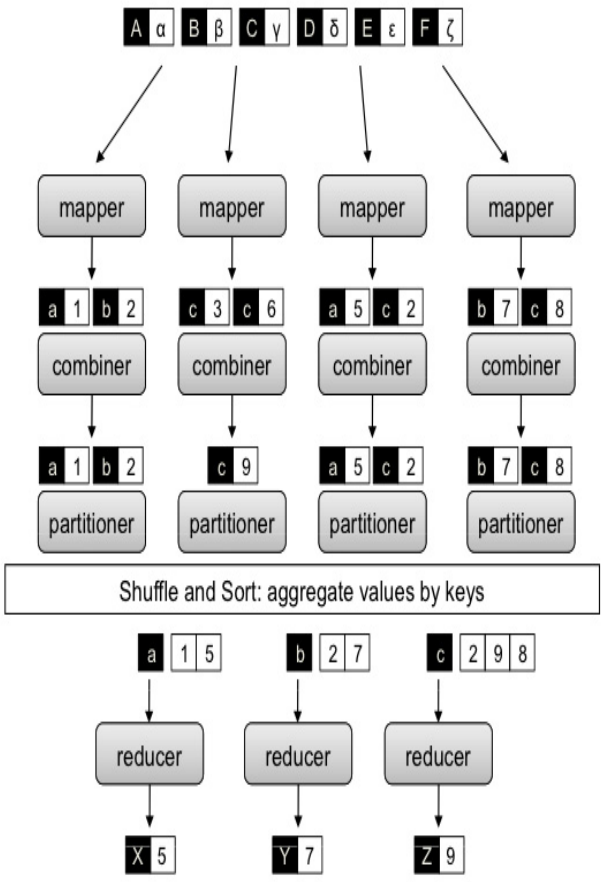


Fig. 2: MapReduce with combiners, partitioners

Map Phase: In the map phase, the master node takes the input, divides it into smaller sub-tasks, and distributes them to worker nodes. A worker node may do this again repeatedly, leading to a multi-level tree structure. The worker node processes the smaller task only, and passes the intermediated result back to its master node.

Reduce Phase: During the reduce phase, the master node collects all the intermediated outputs of all the sub-tasks generated by various worker nodes and combines them insome way to form the final output – the solution to the problem it was originally trying to solve.

a) **Input reader**: The input reader splits the input file into appropriate sizes (in practice typically 64 MB to 512 MB as per HDFS) and one split is assigned to one Map function by MapReduce framework. The input reader takes input from stable storage (typically as in our case Hadoop distributed file system) and generates the output as key/value pairs.

b) **Map function**: Each Map function takes a series of key/value pairs generated by the input reader, processes each, and in turn produces zero or more output key/value pairs.[5] The input and output types of the map can be and often are different from each other.

c) **Partition function**: Each Map function output is assigned to a particular reducer by the application' partition function for sharing purposes. The partition function is given as input the key and the number of reducers and it return the index of desired reduce.

d) **Comparison function**: The input for every Reduce is fetched from the machine where the Map run and sorted using comparison function.

e) Reduce function: The frame work calls the applications Reduce function for each unique key in the sorted order. It also iterates through the values that are associated with that key and produce zero or more outputs.

f) Output writer: It writes the output of the Reduce function to stable storage, usually a Hadoop distributed file system.

Performance:
MapReduce programs do not produce the output with high speed. The main benefit of this programming model is to make use of the optimized shuffle operation of the platform, and the only task of the programmer is to write the Map and reduce functions of the program. While execution, the author of a MapReduce program needs to shuffle the intermediate results.However, the partition function and the amount of data generated by the Map function highly influence the performance of the program. In addition to the partitioner, the Combiner function helps to reduce the amount of data written to storage (disk), and transmitted over the network.

IV. PROBLEM FORMULATION

Hadoop: Yahoo! became the primary contributor in 2006

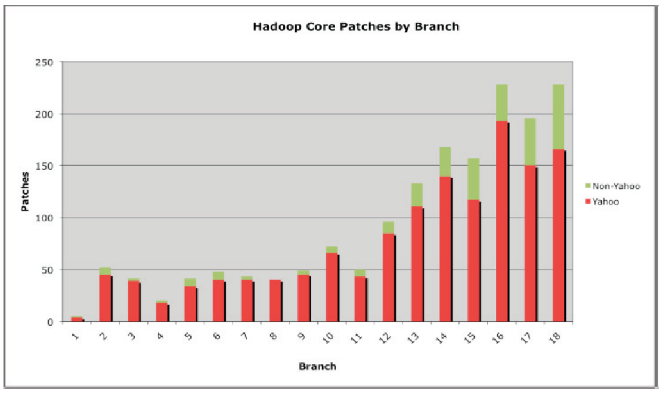


Fig. 3: Primary contribution of Hadoop

Apache Hadoop consists of several components. The ones that of are of interest from a database and analytical processing perspective are :
Hadoop Distributed File System (HDFS), MapReduce, Pig, Hive, HBase, SqoopHDFS can be a source or target file system for MapReduce programs. It is best suited to a small number of very large files. Use of data replication make possible to achieve data availability in HDFS. But it results into the rise in storage required to cope the data. The Hadoop MapReduce framework helps in distributing the map program processing so that the required HDFS data is local to the program. To process all of the output files created by the mapping process,

the Reduce program performs more movement and access to internode data. At the time of execution, both the map and reduce programs write the accomplished data to the local file system so as to reduce or even avoid the overhead of HDFS replication. HDFS supports multiple readers and one writer (MROW). The index mechanism is not available in HDFS, hence, it is best suited to read-only applications that need to scan and read the complete contents of a file. In HDFS, the actual location of the data is transparent to applications and external software.

MAPREDUCE IMPLEMENTATION:
While designing the MapReduce programs the user may not specify the mappers since it depends on the file size and the block size, where as the number of reducers can be configured by user based on number of mappers. In general the Partitioner decides to choose reducers or else Hadoop takes over the job. With the help of the combiner the network traffic will be highly reduced. If map() is not defined by the user then the output of Record reader is sent to identity mapper(without any logic) then to reduce without any reducer defined in the program then the output of the identity reducer is stored in the data node itself and is not sent to HDFS.

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. 7, Issue 2, February 2019

When multiple mappers are running there may be a situation where some mappers may be running very slow, Hadoop then identifies such slow running jobs and triggers the same job to other data node, this concept is called as Speculator execution in Hadoop.

USES:
MapReduce is useful in a wide range of applications, including distributed pattern-based searching, distributed sorting, web link-graph reversal, Singular Value Decomposition,web access log stats, inverted index construction, document clustering, machine learning,and statistical machine translation. Moreover, the MapReduce model has been adapted to several computing environments like multi-core and many-core systems,desktop grids, multi-cluster,volunteer computing environments,dynamic cloud environments,mobile environments,and high-performance computing environments.At Google, MapReduce was used to completely regenerate Google's index of the World Wide Web. It replaced the old *ad hoc* programs that updated the index and ran the various analyses.Development at Google has since moved on to technologies such as Percolator, FlumeJava and MillWheel that offer streaming operation and updates instead of batch processing, to allow integrating "live" search results without rebuilding the complete index.MapReduce's stable inputs and outputs are usually stored in a distributed file system. The transient data are usually stored on local disk and fetched remotely by the reducers.

V. CONCLUSION

Big data and the technologies associated with it can bring significant benefits to the business. But the tremendous uses of these technologies make difficult for an organization to strongly control these vast and heterogeneous collections of data to get further analysed and investigated.
With the invent of new technologies emerging at a rapid rate, one must be very careful to understand the global competition and the big data analysis that support decision making. This paper analyzes the concept of big data analysis and how it can be simplified as from existing traditional relational database technologies. This paper clearly specifies the Hadoop environment, its architecture and how it can be implemented using MapReduce along with various functions. As Big Data Analysis is still in its infancy stage we are sure that this paper helps the researchers to better understand the concepts of Big Data its processing and analysis. Big Data will definitely bring a major social change. Though programming languages like R, SPSS are evolving for Big Data analytics further research is still required to ensure integrity, security for the large data sets being processed. Big Data Analytics should be exploited for sustainable and unbiased society.

REFERENCES

[1] http://hadoop.apache.org,2010
[2] V. Patil, V.B. Nikam, "Study of Mining Algorithm in cloud computing using MapReduce Framework", Journal of Engineering, Computers & Applied Sciences (JEC&AS) Vol.2, No.7, July 2013.
[3] https://en.wikipedia.org/wiki/MapReduce
[4] D. Usha, A.P.S. AslinJenil, " A Survey of Big Data Processing in Perspective of Hadoop and Mapreduce**"**, International Journal of Current Engineering and Technology, Vol.4, No.2,April 2014.
[5] S. Ghemawat et al ."The Google File System." ACM SIGOPS Operating Systems Review, 37(5):29–43, 2003.
[6] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," in Proceedings of OSDI'04: Sixth Symposium on Operating System Design and Implementation, December 2004.
[7] T. White. Hadoop: "The Definitive Guide.",Yahoo Press,2010.
[8] Russom,P "Big Data Analytics", TDWI Best Practices Report, pp.1-40, 2011.
[11] J R Swedlow, G Zanetti, C Best. Channeling the data deluge. Nature Methods, 2011, 8: 463-465
[12] G C Fox, S H Bae, et al. Parallel Data Mining from Multicore to Cloudy Grids. High Performance Computing and Grids workshop, 2008