HADOOP MAPREDUCE: THE QUINTILLION DATA ANALYZER

Adanma Cecilia Eberendu[*]

John Imhonikhe Lawal*

ABSTRACT

With the continuous advance in computer technology over the years, the quantity of data being generated is growing exponentially. Major sources of these data include social media, retailer databases, financial and medical institutions among others. These data come in different formats which include audio, video, text documents, and web pages etc. some of which are structured, semi-structured or unstructured. This poses a great challenge when these data are to be analyzed because conventional data processing techniques are not suited to handling such data. This is where Hadoop MapReduce comes in. Hadoop MapReduce is a programming model for developing applications that process large amount of data in parallel across clusters of commodity hardware in a reliable and fault-tolerant manner. This report covers the origin of Hadoop MapReduce, its features and mode of operation, describes how it is being implemented, as well as reviews how some Information Technology companies are making use of it.

Keywords: MapReduce, Hadoop, Task, JobTracker, Programming

[*] Department of Computer Science, Madonna University, Nigeria

1. INTRODUCTION

The quantity of data being generated on a daily basis is quite alarming. For example, statistics from 2014 info graphics show that every minute:

- Facebook users share nearly 2.5 million pieces of content.
- Twitter users tweet nearly 300,000 times.
- Instagram users post nearly 220,000 new photos.
- YouTube users upload 7 hours of new video content.
- Apple users download nearly 50,000 apps.
- Email users send over 200 million messages
- Amazon generates over $80,000 in online sales.

Also, with the internet of things, the amount of data generated by car sensors, railway sensors, GPS devices and other devices is also at a very high level. Various medical and research institutions also tend to gather an enormous amount of data each day. In fact, according to IBM, every day we create an average of 2.5 quintillion bytes of data (equivalent to 2.3 trillion gigabytes).Wow! These large amounts of data come in various formats ranging from images, videos, JSON strings, audios, PDFs, web pages, text documents even to 3D models. Data of this nature is known as Big Data.

Efforts to simplify data processing have always been a matter of research and development. In the year 2000, Seinst Inc. developed a C++ based distributed file sharing framework to be used for data storage and query. The system was able to store and distribute structured, semi-structured and unstructured data across multiple servers. Developers could build queries with a modified C++ language called ECL. By 2004, LexisNexis had acquired and merged Seinst Inc and ChoicePoint Inc. and their high speed parallel processing platform into HPCC Systems. That same year, Google published a paper on MapReduce - a similar architecture to the HPCC Systems. The framework turned out to be successful so others wanted to replicate the algorithm. Hence, an implementation of the MapReduce framework was adopted by an Apache open source project named Hadoop. Hadoop allows for distributed processing of large data sets across clusters of computers using simple programming models. It is very scalable as it can scale up from single servers to thousands of machines with each offering local computation and storage. The library is designed to detect and handle failures at the application layer while delivering highly-available service on top of a cluster of computers which may be prone to failures. The

Hadoop MapReduce is a software framework or programming model for developing applications that big data in parallel across clusters of commodity hardware in a reliable and fault-tolerant manner. Being a programming model, it is only made use of by developers.

Some developers still make use of conventional data storage and processing techniques (such as relational databases) to store and harness information from large data sets which proves to be inefficient these days. Hence, there is a need for them to adopt a new method of developing applications that can process information in a very efficient and fault-tolerant way.

Since its inception, the MapReduce framework has been a subject of research and discussion among various experts, professionals and institutions. Many of these researchers have praised the subject. MapReduce is arguably the most successful parallelization framework especially for processing large data sets in datacenters comprising commodity computers (Zhiqiang & Lin, 2010).MapReduce has proven to be a useful abstraction and greatly simplifies large-scale computations(Prasad, 2009).The research community uses MapReduce and Hadoop to solve data-intensive problems in bioinformatics, computational finance, chemistry, and environmental science(Serge, 2013). In praise of Hadoop MapReduce, Andrew McAfee and Erik Brynjolfsson, 2011, stated: “The evidence is clear: Data-driven decisions tend to be better decisions. Leaders will either embrace this fact or be replaced by others who do” (McAfee & Brynjolfsson, 2011).

Despite its numerous benefits, a number of researchers have criticized MapReduce for different reasons. Even if high-level, declarative-style abstractions exist and have been widely adopted, Hadoop MapReduce is still far from offering interactive analysis capabilities (Vasiliki & Vladimir, 2014). The most popular of these criticisms is found in David DeWitt and Michael Stonebraker, 2008. There, MapReduce was described as “a giant step backwards”:

> *“As both educators and researchers, we are amazed at the hype that the MapReduce proponents have spread about how it represents a paradigm shift in the development of scalable, data-intensive applications. MapReduce may be a good idea for writing certain types of general-purpose computations, but to the database community, it is a giant step backward in the programming paradigm for large-scale data intensive applications”.(DeWitt & Stonebraker, 2008)*

David DeWitt and Michael Stonebraker also stated that MapReduce is not novel at all – It represents a specific implementation of well-known techniques developed nearly 25 years ago, it is missing most of the features that are routinely included in current DBMS and it is incompatible with all of the tools DBMS users have come to depend on.

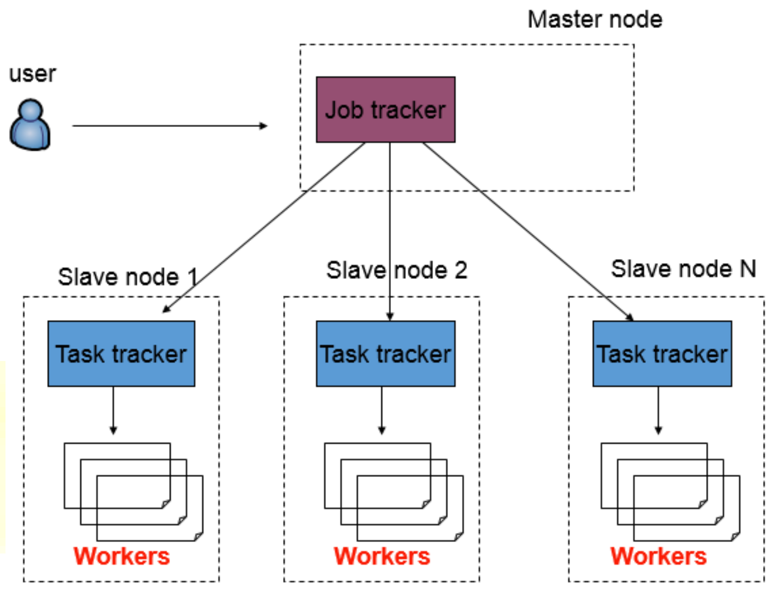
Is MapReduce truly missing most of the features that are routinely included in current database management systems? Another component of Hadoop (HBase) is responsible for having those features included in current DBMS and as such, the burden should not be placed on MapReduce. It is correct to state that MapReduce is not the best solution there can be to solving the problem of Big Data, but indeed it is good enough and has room for improvement.

The aim of this study is to discover how Hadoop MapReduce technology solves the problem of Big Data Processing. It will also aid readers to understand how Hadoop MapReduce works, the importance of Hadoop MapReduce and how Hadoop MapReduce is implemented.

In order to achieve the objective of this study as earlier stated, this report covers the history of Hadoop MapReduce, the features of Hadoop MapReduce, the mode of operation of Hadoop MapReduce, the implementation of Hadoop MapReduce, the applications of Hadoop MapReduce, the benefits of Hadoop MapReduce and the limitations of Hadoop MapReduce.Also, implementing Hadoop MapReduce is a broad subject and may not be covered in full details here. A proper introduction into the methodologies used in implementing Hadoop MapReduce is provided to aid readers who wish to go further into its implementation.

2. FEATURES OF HADOOP MAPREDUCE

The Hadoop MapReduce framework consists of a single JobTracker (known as the master) and one TaskTracker per node in the cluster (known as slaves or workers). The JobTracker is responsible for scheduling the jobs' component tasks, monitoring them and re-executing the failed tasks. The TaskTrackers execute the tasks as directed by the JobTracker. Both the input and the output of the job are stored in a file-system.



Features of Hadoop MapReduce Source: (Shi, 2008)

Applications which implement MapReduce usually specify the input and output locations and supply Map and Reduce functions via implementations of appropriate interfaces and/or abstract-classes. These and other parameters are known as the Job configuration. The Hadoop client then submits the job (either in jar/executable form) and the configuration to the JobTracker which then distributes the software and configuration to the TaskTrackers, schedules the tasks, monitors them, provide status and diagnostic information to the job client.

3. MODE OF OPERATION

A MapReduce job usually splits the input data set into independent blocks of data which are assigned to Map tasks (functions) in a completely parallel manner. The output of the map is then sorted and given as input to the Reduce tasks (functions) to produce the final result. Hadoop MapReduce works exclusively on <key, value> pairs. It views the input to the job as <key, value> pairs and produces its output from the job as a set of <key, value> pairs. It makes use of an interface known as the Writable interface to serialize the Key and Value classes. These classes have to implement the interface. Also, the key class has to implement the WritableComparable interface in order to facilitate sorting by the framework. The input and output of the MapReduce jobs can be depicted as:

(Input) <k1, v1>→ map →<k2, v2>→ combine →<k2, v2>→ reduce →<k3, v3> (output)

Applications which implement MapReduce usually implement the Mapper and Reducer Interfaces which provide several methods for different tasks.

MAPPER

The Mapper is responsible for mapping input Key/Value pairs to a set of intermediate Key/Value pairs. The individual tasks that transform input records into intermediate records are known as Maps. The result of the maps does not necessarily have to be of the same data type as the input records. The output of the Maps are sorted then partitioned per Reducer. The total number of partitions depends on the number of reduce tasks for the job (this usually depends on the number of keys) and they are always equal. The MapReduce framework subsequently groups the intermediate values which are associated with a given output key and passes them to the reducer(s) to determine the final output. When developing applications with the Hadoop MapReduce framework, users can control which keys go to which Reducer by implementing an interface known as Partitioner.

REDUCER

After the Mapper has successfully mapped the input, the Reducer reduces the set of intermediate values which share a common key into smaller set of values. The user specifies the number of reduces for the job using specific functions. The reducer consists of three (3) primary phases.

1. Shuffle
2. Sort
3. Reduce

Shuffle: In this phase, the MapReduce framework fetches the relevant partition of all the mappers via Hyper Text Transfer Protocol (HTTP).

Sort: In this phase, the framework groups Reducer input by Keys (because different mappers may have produced output with the same key). The shuffle and sort framework occur simultaneously. That is, as map-outputs are being fetched, they are merged.

Reduce: In this phase, the reduce method is called for each <key, list of values > pairs in the grouped input. Then the output is written to the file system.

SAMPLE PROBLEM

To understand the whole MapReduce process, let us assume that we have an application which counts the number of times a Computer Science student of Madonna University used certain words during CSC317 examination. Let's also assume we have the following words appearing in different sentences:

Variable Array Pointer

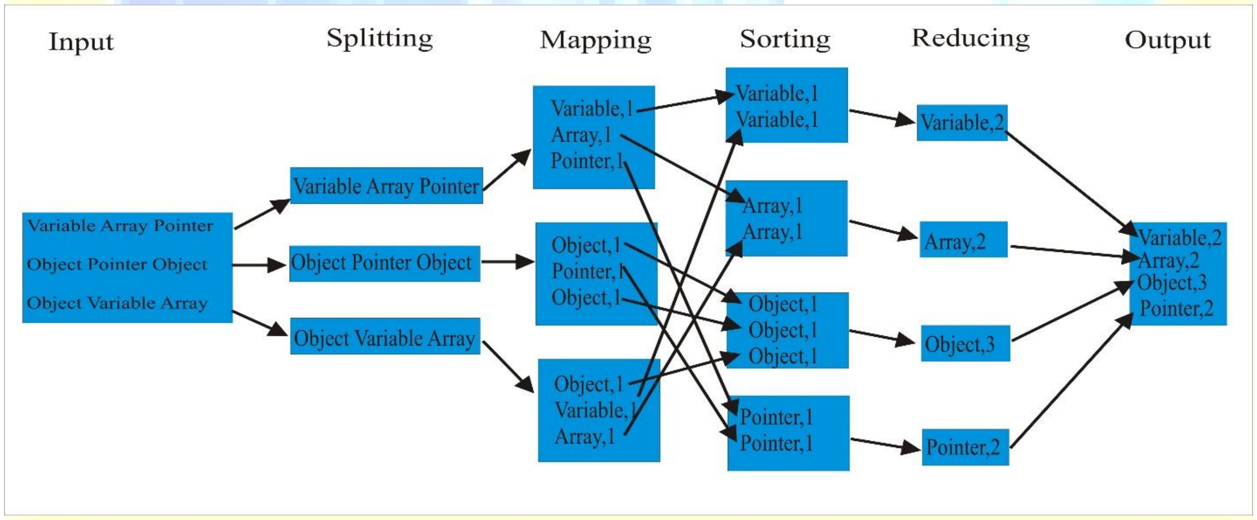
Object Pointer Object

Object Variable Array

The main steps MapReduce would take to process the data would be:

1. Get the input data.
2. Split the data into separate blocks.
3. Assign the blocks to Map tasks.
4. Sort the output of the Map tasks.
5. Reduce the sorted data using the Reduce tasks.
6. Store the output in the file system.

The following diagram illustrates the process:



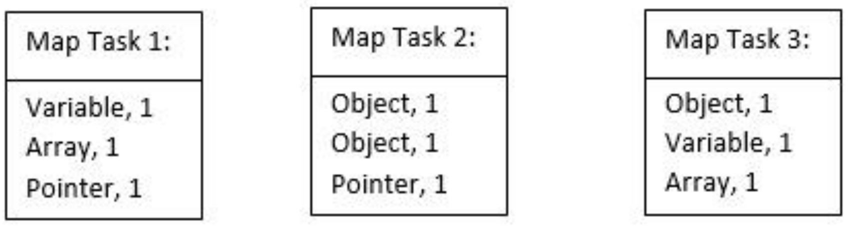
Execution of a MapReduce count program Modified from: (Zaharia, 2012)

Breaking down these steps, we can understand how each activity took place.

First the input was split into 3 blocks:

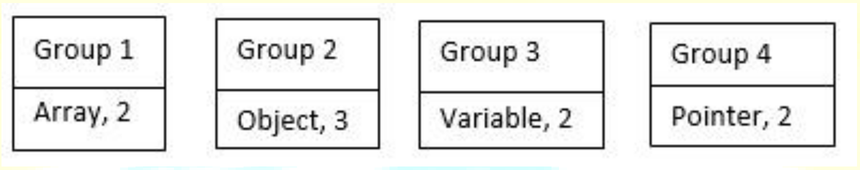
- Block 1-Variable Array Pointer
- Block 2-Object Pointer Object
- Block 3-Object Variable Array

Then the Mapper transforms the inputs to intermediate <key, value> pairs:



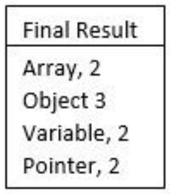
Map Task 1:	Map Task 2:	Map Task 3:
Variable, 1	Object, 1	Object, 1
Array, 1	Object, 1	Variable, 1
Pointer, 1	Pointer, 1	Array, 1

These values are passed to the Reducer. In the Reducer, the data passes through the Shuffle and Sort phases which then group the data by keys:



Group 1	Group 2	Group 3	Group 4
Array, 2	Object, 3	Variable, 2	Pointer, 2

Finally, the Reduce phase reduces the data and stores them in the file system.



Final Result
Array, 2
Object 3
Variable, 2
Pointer, 2

When you picture this behaviour on larger data, you will really appreciate Hadoop MapReduce.

4. IMPLEMENTING HADOOP MAPREDUCE

Hadoop has tools that help the successful development of a MapReduce application, e.g. HBase, Hive, Pig, Oozie, Mahout Etc. One thing is common for all of them – they must include the Hadoop library, and the programs must follow the same structure of having map and reduce functions or methods. The following pseudocode shows an example of a MapReduce program. The program counts the occurrence of each word in a large input document – similar to the sample problem we saw while understanding the mode of operation of Hadoop MapReduce.

```
Map (String fileName, String fileContents)
{
        For each word key in fileContents:
        EmitIntermediate(key, “1”);
}
Reduce(String word, Iterator Values)
{
```

```
            int occurrence = 0
            for each v in values:
            occurrence+= 1;
            Emit(AsString(occurrence));
    }
```

In the Map method, the fileName parameter is input key while the fileContents parameter is input value. With the aid of a MapReduce API's, the code loops through the fileContents and finds each word. It then passes the word and the value '1' to another MapReduce method to emit the intermediate <key, value> pairs. This method sorts the data and triggers the reducer.

In the Reduce method, the word parameter represents the input key and the Values parameter represents a list of counts. Just as we've seen in the sample problem, the reduce method is called for each <key, list of values > pairs in the grouped input. It adds up the number of occurrence of the keys and then the output is written to the file system.

5. SOME EXISTING USERS

Hadoop MapReduce is applicable in all areas of life where data is being generated. This is because the data being generated will certainly need to be stored and processed. Due to its awesome benefits, a number of "Big" companies have already started using MapReduce. They include: Facebook, Yahoo, Amazon, eBay, Google, IBM, The New York Times, and Walmart

Facebook started using Hadoop in 2007. As at then, developers at Facebook were using it to import a few data sets and writing MapReduce jobs to manipulate them. At the success of this, they started using it for major projects such as Facebook Lexicon – the tool where you can see the buzz surrounding different words and phrases on Facebook Walls. Most data stored in Hadoop's file system is published as Tables, classic data warehouse features like partitioning, sampling and indexing were added to the environment. This in-house data warehousing layer over Hadoop is called Hive – One of the software mentioned in the preceding sections implementing Hadoop MapReduce.

Amazon implements MapReduce in their Amazon Elastic MapReduce (Amazon EMR). It is a web service that makes it easy to process vast amounts of data quickly and cost-effectively.

Every day, a number of users adopt Hadoop MapReduce for processing data and join in enjoying its benefits.

6. BENEFITSAND LIMITATIONSOF HADOOP MAPREDUCE

MapReduce provides a lot of benefits. They include:

1. Cost-efficiency - MapReduce requires commodity hardware to function. Also, its fault-tolerance is automatic. Hence, fewer admins are needed on the network.

2. Simplicity – Developers intending to implement MapReduce can write applications with their language of choice such as Java, C++ or Python. Also, MapReduce jobs are easy to run.

3. Scalability – MapReduce can process petabytes of data which are stored in Hadoop Distributed File System (HDFS) in one cluster.

4. Speed – With parallel processing, MapReduce can take problems that used to take days to solve and solve them in hours or minutes. For instance, in July 2008, one of Yahoo's Hadoop clusters sorted 1 terabyte of data in 209 seconds, which beat the previous record of 297 seconds in the annual general purpose (Daytona) terabyte sort benchmark.

5. Recovery – MapReduce handles failures. Due to redundancy in HDFS, if a machine with one copy of the data is unavailable, another machine has a copy of the same key/value pair, which can be used to solve the same sub-task. The JobTracker keeps track of it all.

6. Minimal Data Motion – With MapReduce, processes are moved to data and not data to processes. Processing normally occurs on the same physical node where the data resides. This is known as data locality. This reduces the network input/output patterns and contributes to the processing speed.

Despite the amazing benefits of Hadoop MapReduce, there are some limitations to it. Some are listed below:

1. The development of efficient MapReduce applications requires advanced programming skills and also a deep understanding of the architecture of the system. Analysts who are used to SQL-like or declarative languages may view MapReduce programming model as too “low-level” because MapReduce doesn’t require implementing relational operations such as joins.
2. MapReduce has a batch nature. Data needs to be uploaded to the file system always and even when the same dataset needs to be analyzed multiple times, it still needs to be read every time.
3. The master node (where the JobTracker is) has to be more sophisticated than other nodes because it can easily become a single point of failure.

4. Since Google has been granted patent, it raises the question of the long-term viability of using the mechanism in open environments.
5. The "One Way Scalability" of its design. MapReduce allows a program to scale up to process very large datasets but constrains programs ability to process smaller data items.

7. Conclusion

Hadoop MapReduce is a software framework or programming model for developing applications that process large amount of data in parallel across clusters of commodity hardware in a reliable and fault-tolerant manner.Hadoop MapReduce works exclusively on <key, value> pairs.A MapReduce job usually splits the input data set into independent blocks of data which are assigned to Map tasks (functions) in a completely parallel manner. The output of the map is then sorted and given as input to the Reduce tasks (functions) to produce the final result. Hadoop MapReduce applications can be written in several languages. One thing is common for all of them – they must include the Hadoop library, and the programs must follow the same structure of having map and reduce functions or methods. Several companies are already making use of it such as Facebook, Yahoo, and Google etc. This is due to the benefits of using MapReduce to develop applications such as the scalability, its speed, its simplicity, minimal data motion and others.

The MapReduce model is a very efficient model for processing of large data sets. Currently, the interest for Hadoop MapReduce is at its peak and there exist a lot of problems and challenges that still need to be addressed. The future ahead of Big Data is very bright, as businesses and organizations realize more the value of the information they can store and analyze. With the improvement and implementation of Hadoop MapReduce, data processing will be as fast and efficient as never before and life would be made easier. Hence, it should be adopted by developers who need to develop applications that process large amount of data.

References

DeWitt, D., & Stonebraker, M. (2008, January 17). MapReduce: A major step backwards. PM Permalink. Retrieved October 30, 2015

McAfee, A., & Brynjolfsson, E. (2011). Big Data: The Management Revolution. Retrieved October 20, 2015

Prasad. (2009). MapReduce Architecture. Retrieved October 10, 2015

Serge, B. (2013). Introduction to Hadoop,MapReduce and HDFS for Big Data Applications. Chicago: Storage Networking Industry Association.

Shi, X. (2008). Take a close look at MapReduce.

Vasiliki, K., & Vladimir, V. (2014). MapReduce: Limitations, Optimizations and Open Issues. Stockholm: KTH The Royal Institute of Technology.

Zaharia, M. (2012). Introduction to MapReduce and Hadoop.

Zhiqiang, M., & Lin, G. (2010). The Limitation of MapReduce: A Probing Case and a Lightweight Solution. *CLOUD COMPUTING 2010 : The First International Conference on Cloud Computing, GRIDs, and Virtualization*, (68-73). IARIA.