Recovering Nodes Failure in Replica System by Using Data Replacement Algorithm in a HDFS Cluster

Myint Zaw, *Dr. Khine Khine Oo*
myintzaw@ucsy.edu.mm, khinekhineoo@ucsy.edu.mm

Abstract

HDFS (Hadoop Distributed File System) is designed to store big data reliably, and it has been distributed file. However, it is not replacement for failure nodes. Replication technology is a key component in every computing enterprise as it is essential to applications such as backup, file distribution, file synchronization as well as file sharing collaboration. The replacement concept is simple but its algorithmic optimization and system implementations are challenging and difficult. Data Replacement Algorithm was implemented to make the replicas to get placed on failure nodes. As this system can reduce data failure and cannot affect other Data Nodes, it is more reliable and usable.

Key words: Data Replacement Algorithm, Big Data, Hadoop, Map Reduce, HDFS (Hadoop Distributed File System), and Replica System

1. Introduction

In this electronic age, increasing number of organizations are facing the problem of explosion of data and the size of the databases used in today's enterprises that has been growing at exponential rates.

Data is generated through many sources like business processes, transactions, social networking sites, web servers, etc. and remains in structured as well as unstructured form [8]. Recent developments in the Web, social media, sensors and mobile devices have resulted in the explosion of data set sizes. For example, Facebook today has more than one billion users, with over 618 million active users generating more than 500 terabytes of new data each day [2].

Today's business applications are having enterprise features like large scale, data-intensive, web-oriented and accessed from diverse devices including mobile devices. Processing or analyzing the huge amount of data or extracting meaningful information is a challenging task.

Replica System can handle disk failure, data loss or corruption, and data replacement in failure.

2. Related Work

Replacement is a critical problem of HDFS Cluster. Dyavanur and Kavita Kori [9] surveyed a fault tolerant system. It is based on fault tolerance techniques in big data tools. This system is only specified tasks in the presence of failure. T.Cowsalya and S.R.Mugunthan [10] represented a fault tolerance system. It is using data replication and heartbeat messages. Data replication is achieved by creating copies of same data sets into more than one computing node. HDFS has automatically processed this service.

3. BIG DATA

The term "Big Data" refers to large and complex data sets made up of a variety of structured and unstructured data which are too big, too fast, or

too hard to be managed by traditional techniques. Big Data is characterized by the 6Vs [3, 11]: volume, velocity, variety, and veracity, value, variability. Volume refers to the quantity of data, variety refers to the diversity of data types, velocity refers both to how fast data are generated and how fast they must be processed, and veracity is the ability to trust the data to be accurate and reliable when making crucial decisions, value refers to valuable information within the data, variability refers to data changes during processing and lifecycle.

Big data sizes are a constantly moving target currently ranging from a few dozen terabytes to many petabytes of data in a single data set [4]. Difficulties include capture, storage, search, sharing, analytics and visualizing. Typical examples of big data found in current scenario includes web logs, RFID (Radio-frequency identification) generated data, sensor networks, satellite data, social data from social networks, Internet text and documents, Internet search indexing, call detail records, astronomy, atmospheric science, genomics, biogeochemical, biological, and other complex and/or interdisciplinary scientific research, military surveillance, medical records, photography archives, video archives, large-scale e-commerce, and so on.

3.1. What is Big Data Problem?

The world has witnessed explosive, exponential growth in recent times. Businesses have been tackling the capacity challenge for many years (much to the delight of storage hardware vendors). Therefore, the *big* in big data is not purely a statement on size [1].

One current feature of big data is the difficulty working with it by using relational databases and desktop statistics or visualization packages, required instead "massively parallel software running on tens, hundreds, or even thousands of servers"[5]. The various challenges faced in large data management include – scalability, unstructured data, accessibility, real time analytics, fault tolerance and many more.

4. Hadoop

Apache Hadoop is a top-level open source project and is governed by the Apache Software Foundation (ASF). Hadoop is not any one entity or thing. It is the best thought of as a platform or an ecosystem that describes a method of distributed data processing at scale using commodity hardware configured to run as a cluster of computing power. This architecture enables Hadoop to address and analyze vast quantities of data at significantly lower cost than traditional methods commonly found in data warehousing, especially, with relational database systems. At its core, Hadoop has two primary functions:

- Processing data (Map Reduce)
- Storing data (HDFS) [1]

4.1. Map Reduce

As a high-level programming model for processing vast amounts of data, *Map Reduce* [2] usually uses parallel and distributed computing on clusters of nodes. Map Reduce is a programming paradigm for processing large data sets in distributed environments [6]. In the Map Reduce paradigm, the *Map* function performs filtering and sorting, while the *Reduce* function carries out grouping and aggregation operations. The ‘to be or not to be’ of Map Reduce is the word counting example: it counts the appearance of each word in a set of documents.

The *Map* function splits the document into words and for each word in a document it produces a (key, value) pair.

```
function map(name, document)
        for each word in document
                emit (word, 1)
```

The *Reduce* function is responsible for aggregating information received from *Map* functions. For each key and word, the *Reduce*

function works on the list of values, partialCounts. To calculate the occurrence of each word, the *Reduce* function groups by word and sums the values received in the partialCounts list.

function reduce (word, List partialCounts)

```
sum = 0
for each pc in partialCounts
        sum += pc
emit (word, sum)
```

The final output is the list of words with the count of appearance of each word. For example
(to,2)
(be,2)
(or,1)
(not,1)

Figure 1 illustrates the Map Reduce model. Each map task in Hadoop is broken into the following phases: *record reader*, *mapper*, *combiner*, and *partitioner*. The output of the map tasks, called the intermediate keys and values, are sent to the reducers. The reduce tasks are broken into the following phases: *shuffle* and *sort*, *reducer*, and *output format*. The nodes in which the map tasks run are optimally on the nodes in which the data rests. In this way, the data typically does not have to move over the network and can be computed on the local machine.

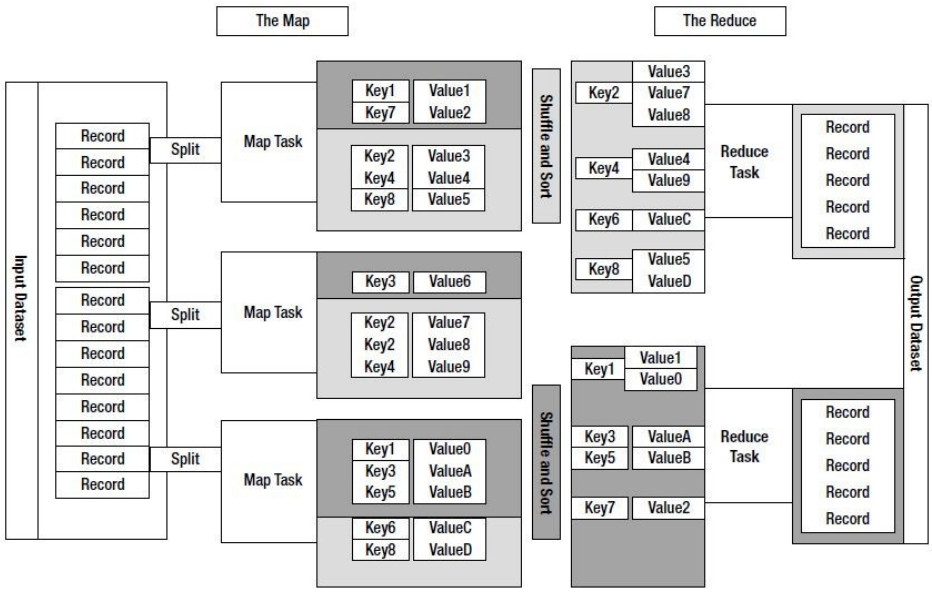

Figure 1: The Map Reduce Model [7]

The main contribution of the Map Reduce paradigm is scalability as it allows for highly parallelized and distributed execution over a large number of nodes. In the Map Reduce paradigm, the *Map* or *Reduce* task is divided into a high number of jobs which are assigned to nodes in the network. Reliability is achieved by reassigning any failed node's job to another node. A well-known open source Map Reduce implementation is Hadoop which implements Map Reduce on top of the Hadoop Distributed File System (HDFS).

4.2 HDFS (Hadoop Distributed File System)

HDFS was originally created as a part of a web search engine project called Apache Nutch, it is a

distributed file system designed to run on a cluster of cost-effective commodity hardware. HDFS provides high throughput access to application data and is suitable for applications that have large data sets. HDFS can handle large files in the gigabytes and beyond with sequential read/write operation. These large files are broken into chunks and stored across multiple data nodes as local files [1].

5. Proposed System for Text Document Application

This paper proposes Replica System by using Data Replacement Algorithm with multiple machines in HDFS Cluster, it is shown in figure 2. HDFS is a block-structure file system where each file is divided into blocks of a predetermined size. A Data Node includes multiple blocks. A block takes 64MB and stored its data is only 10 MB on the block.

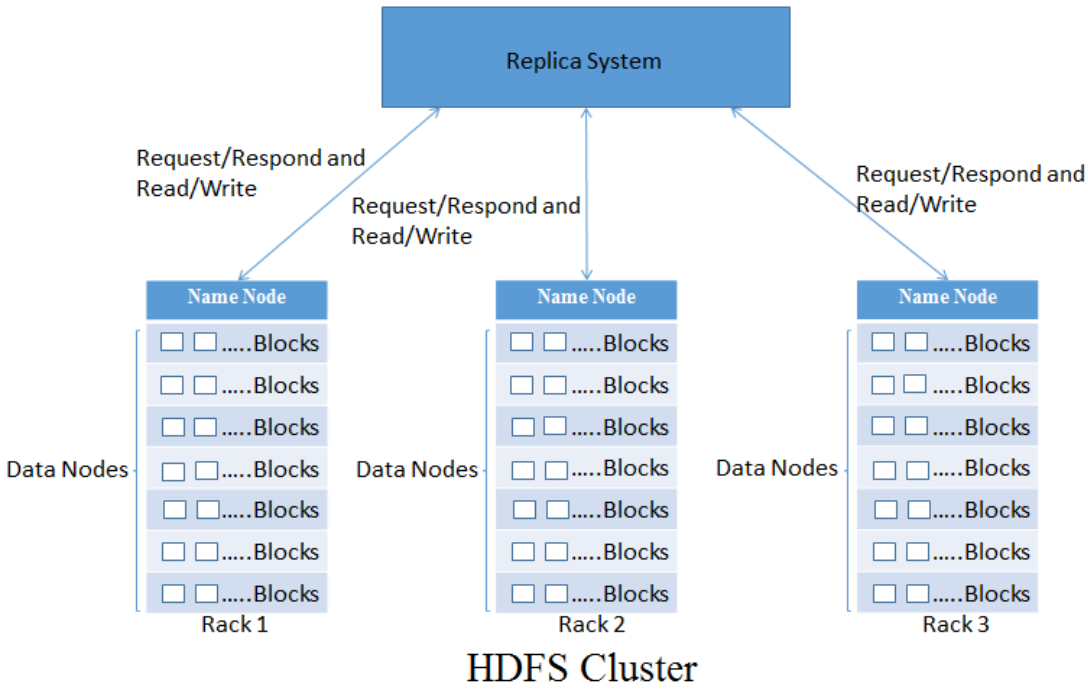

Fig 2: The proposed Architecture

In the proposed architecture, a HDFS Cluster consists of three Name Nodes and multiple Data Nodes by default. They are running on multiple machines. Name Nodes receive the data or files from the Map Reduce process, and then they maintain the data or file system, which store the directory structure to the Data Nodes. Each Data Node is responsible for storing the actual data blocks on each machine in the HDFS Cluster. They are also periodically sending the information of data message (Heartbeat message) about to the Name Nodes. Name Node is also called Master Node.

When a file input to the Name Nodes after processing Map Reduce process, it is how to break and replicate file, and how to replicate and distribute into the multiple blocks in HDFS Cluster, it is shown in figure 3.

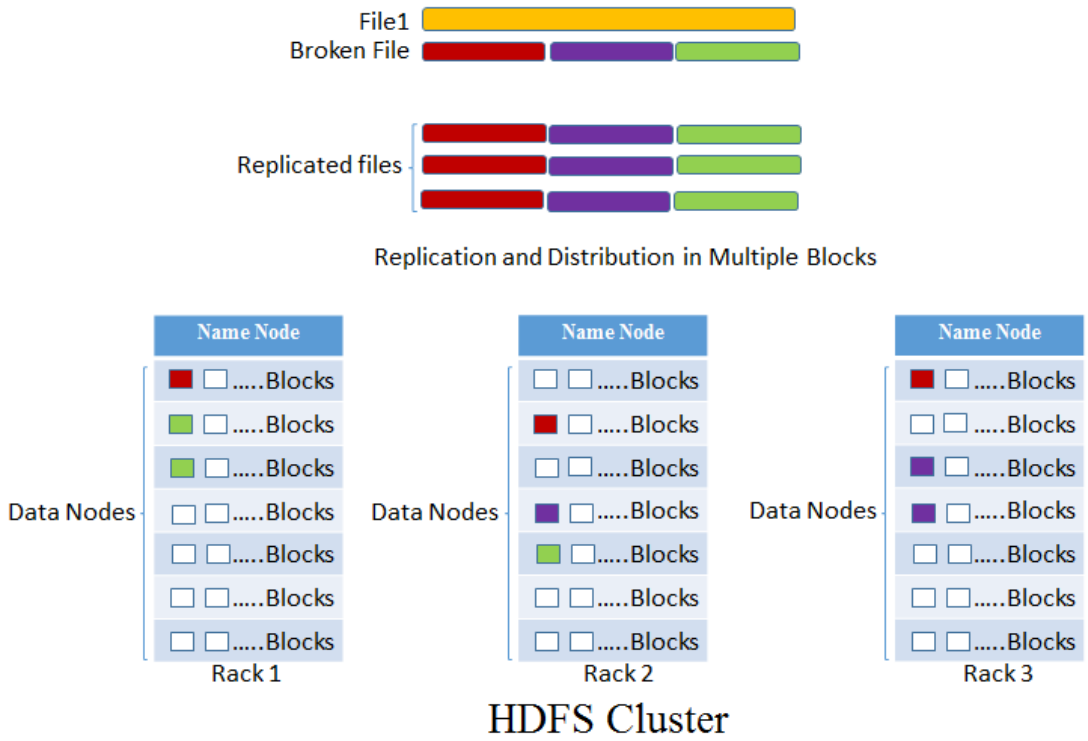

Figure 3: Shown how to brake and replicate file, and how to replicate and distribute into the multiple blocks

In the figure, files are stored in HDFS. These files are broken into smaller chunks also known as blocks. These blocks have been replicated files. In this process, they are replicated by the factors of three replications, which are the multiple replication factors of HDFS. These blocks are distributed on the cluster that takes place in multiple Data Nodes.

Data or files are very important for every organization. Hence, they need all of the data or files without losing. Hadoop is a distributed file system. It is a powerful system. When a Node is failed by hardware damage or a block is failed during dress, Name Node can detect faults and failures. In this case, HDFS is three replications for data or file by default, the data is remaining two blocks in HDFS Cluster. Therefore, data is available and Name Node is requested the data to Replica System. It searches the available data node and gets, and then gives back to the Name Node. If multi nodes are failed, like to one node failure and Replica System backup all.

5.1 Data Replacement Algorithm

HDFS store data by multiple blocks. Data Node heartbeat message send to Name Node when a block failure (j). Then Name Node request to replica system. This system make replica (i) and search available blocks (k), and then make copies and placed to Name Node (y) by pipelined.

Algorithm

1. Data Node request for j to Name Node
2. Name Node request message send to replica system
3. Then do
4. For every j create new i
5. For (i == k)
6. Place k on y
7. y place k to directory
8. for (k_n)
9. k_1 pipelined with k_2 …., k_n
10. end

Static data type is mainly apply in this algorithm, where i represents set of entire replica, j represents set of failure block, k represents set of available block, n is total number, and y represents requested Name Node.

6. Conclusion

This system proposed a Replica System by using Data Replacement Algorithm and provides an overview of Big Data, Hadoop, Map Reduce, and HDFS. Replica System is reduced the failure of Data Nodes, and all Blocks of data in Data Node. It is exploited to enhance the load balance across multiple machines. This system proposes replication failure nodes for better performance and availability.

References

[1] Adam Jorgensen, James Rowland-Jones, John welch, Dan Clark, Christopher Price, Brian Mitchell, **Microsoft® Big Data Solutions**, Published by John Wiley & Sons, Inc., www.wiley.com, 2014.

[2] J. Dean & S. Ghemawat, "MapReduce: simplified data processing on large clusters," CACM 51(1): 107–113, Jan. 2008.

[3] F. Ohlhorst, Big Data Analytics: Turning Big Data into Big Money, Hoboken, N.J, USA: Wiley, 2013.

[4] McKinsey Global Institute, 2011, Big Data: The next frontier for innovation, competition, and productivity,

[5] Thomas Herzog, Associate Commissioner, New York State, Thomas Kooy, IJIS Institute Big Data and the Cloud, IJIS Institute Emerging Technologies,
Available:http://www.correctionstech.org/meeting/2012/Presentations/Red_01.pdf, Aug, 2012.

[6] Donald Miner and Adam Shook, **MapReduce Design Patterns,** Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472., 2013.

[7] Jason Venner, **Pro Hadoop**, Copyright © 2009 by Jason Venner.

[8] Impetus white paper, March, 2011, "Planning Hadoop/NoSQL Projects for 2011" by Technologies,
Available:
http://www.techrepublic.com/whitepapers/planninghadoopnosql-projects-for-2011/2923717,
March, 2011.

[9] Manjula Dyavanur, Kavita Kori, **"Fault Tolerance Techniques in Big Data Tools: A Survey",** International Conference On Advances in Computer & Communication Engineering (ACCE), vol.2, Special Issue 2, May 2014.

[10] T. Cowsalya and S.R. Mugunthan, **"HADOOP ARCHITECTURE AND FAULT TOLERANCE BASED HADOOP CLUSTERS IN GEOGRAPHICALLY DISTRIBUTED DATA CENTER",** ARPN Journal of Engineering and Applied Sciences, VOL. 10, NO. 7, APRIL 2015.

[11] Lidong Wang and Cheryl Ann Alexander, **Big Data in Medical Applications and Health Care,** American Medical Journal 2015, 6 (1): 1.8