Network and Complex Systems
ISSN 2224-610X (Paper) ISSN 2225-0603 (Online)
Vol.4, No.2, 2014
www.iiste.org

A Mathematical Appraisal: Evolution of Distributed File System and Hadoop

Atul Saurabh
Assistant Professor
Babaria Institute of Technology, BITs edu Campus Varnama, Vadodara, Gujrat
Mob: +91-9574465442 E-mail: atul.saurabh@gmail.com

Swapnil M Parikh
Assistant Professor
Babaria Institute of Technology, BITs edu Campus Varnama, Vadodara, Gujrat
Mob: +91-8238046537 E-mail: swapnil.parikh@gmail.com

The research is financed by Asian Development Bank. No. 2006-A171(Sponsoring information)

Abstract
The fast growing technology has left a great impact on the human life. Many traditional systems are either replaced or running in parallel with their electronic counterpart. As for example: - the traditional postal system is now nearly replaced by mobile phones and emails. The electronic system is providing more functionalities than their traditional counterparts. Due to social media, peoples may communicate with each other, share their thoughts and moments of life in form of texts, images or videos. On the other hand, to enhance technologies and knowledge many research activities are propelled and data from different sources are gathered in large volume for further analysis. In short today's world is surrounded with large volume of data in different form. This put a requirement for effective management of these billions of terabytes of electronic data generally called BIG DATA. The effective management must be based on proven mathematical concepts so that chance of casualties may be reduced.

This paper presents a mathematical appraisal for evolution of distribution of file data and explains some basic solution of primitive problems based on probability theory.

Keywords: BIG DATA, Distributed System, DFS, Commodity Hardware, Hadoop, Hadoop File System, HDF.

1. INTRODUCTION

We are living in the electronic world. The electronic world lives with one single motto: every task, if possible, will be achieved using some automated tool. Due this course many useful tools like facebook, stock exchange software, scientific research software etc. are developed. These tools in turns generate a large volume of data. As for example The New York Stock Exchange generates about one terabyte of new trade data per day. The generated data either may be structured (easy to process) or unstructured. Some systems also produce a large amount of binary data like facebook hosts around one petabyte of images. This large volume of data is addressed by a concept called Big Data.

1.1 Problems with Big Data

In computer world it is said that problems either move around time or space. Big Data is no exception. As the technologies evolve, the storage capacity of hard disk and transfer rate increases a lot. But the rate, at which the data is growing, creates two small problems:

- How to store this Big growing data (space problem)?
- How to process and analyze Big Data in significantly low amount of time (time problem)?

1.2 Studied Solutions

1.2.1 Adding hard disk in serial

The obvious solution to the storage is to introduce more hard disk with bigger capacity. This will solve the storage problem but introduces two more problems.

I. The more hard disk we introduce, the more we increase the rate of failure or data lost.

Mathematical Analysis

Let us suppose a system of n hard disks with probability of failure $p1, p2, p3 \cdots pn$. The event of a failure of a hard disk is independent of others. So, the chance of failure of hard disks, P_f, is as mentioned below:

$$P_f = p_1 \times p_2 \times p_3 \times \dots \times p_n \qquad (1)$$

Since,

$$0 \leq p_i \leq 1 \qquad (2)$$

So,

$$P_f \leq p_i \qquad (3)$$

Here the probability of failing hard disk is decreased. But the system (of hard disk) will fail if any one of the hard disk fails. So, the probability that the system gets fail is

$$P_{fss} = MAX(p_1, p_2, p_3, \dots, p_n) \qquad (4)$$

Where P_{fss} indicates the probability of failure of the system when disks are arranged in series. Hence the probability of a system failure is

$$P_{fss} \geq p_i, i = 1, 2, 3, \dots, n \qquad (5)$$

If we increase the number of hard disks in a system, overall probability of the system failure also increases.

II. The bigger is the capacity of the hard disk, the bigger it takes time to retrieve the data from the disk.

Mathematical Analysis

Suppose maximum seek time of i^{th} hard disk is S_i and probability that data will be available in i^{th} disk is p_{ai}.

So, total time T_S to seek data in n-system serial hard disk is

$$T_s = (p_{a1} \times S_1) + (1 - p_{a1}) \times p_{a2} \times S_2 + \dots + (1 - p_{a1}) \times (1 - p_{a2}) \times \dots \times (1 - p_{an-1}) S_{n-1} \qquad (6)$$

Thus overall seek time increases a lot. So, it seems that adding more hard disk is impractical.
Even though mathematical analysis says that adding more hard disk degrades the overall performance of the system, it indicates one ray of hope.
The total probability of hard disk failure from equation (1) is less than the individual one [see equation- (3)]. So, adding more hard disk gives good news. The total failure rate of hard disk gets decreases.

1.2.1 Adding hard disk in parallel

Somehow we can take advantage of above result [eqn-3]. If the hard disk is added in parallel way then failure of one hard disk will not affect the overall system. In that case probability of system failure will not depend on individual failure of the hard disk. The system will fail if and only if all the hard disks fail and then probability of failure of the system become:

$$P_{fsp} = p_1 \times p_2 \times p_3 \times \dots \times p_n \qquad (7)$$

Where P_{fsp} is the probability of system failure when the hard disks are added in parallel. If comparing equation (4) and equation (7) we can conclude that the overall chance of system failure get decreases when the disks are added in parallel.

Adding hard disks in parallel certainly increases the performance in terms of disk failure but help us at time factor. To address this problem we need to analyze how a file is searched in one hard disk.

The file is stored as follows:
The actual content of the file is stored in data block and some basic information called metadata is stored in inode and directory. So, whenever a search is made instead of searching on data block, metadata is searched first.

We can remove metadata part from each hard disk and store in one hard disk which is dedicated for metadata storage only. In this situation we need to search only one hard disk for obtaining the data. So the seek time becomes as mentioned below

$$T_y = p_{a1} \times S_1 \qquad (8)$$

Hence effectively the seek time gets reduced [compare equation (6) to equation (8)].

2. General Architecture

From the above discussion a tree based architecture can be proposed [see figure:1]. The root of the tree keeps the metadata and other nodes keep the actual data. This architecture reduces the seek time and also have a major impact on system failure. Now the system is failed only if all the children get failed.

2.2 Pros and Cons

- The root keeps all the metadata.
- Since the size of metadata is small enough, big storage is not required at root level.
- Data searching should be faster. A faster algorithm is required to store data at root node.
- The processing power of root must be high as data can be required by many users simultaneously
- If the root node gets fail, all the data is lost.
- Since the child node only require to store data the storage capacity must be high.
- The child is not only involved in storing and transferring the data.
- The block size of child hard disk must be large enough so that storing and retrieving of data do not require higher amount of block to traverse. This makes the traversal time smaller.

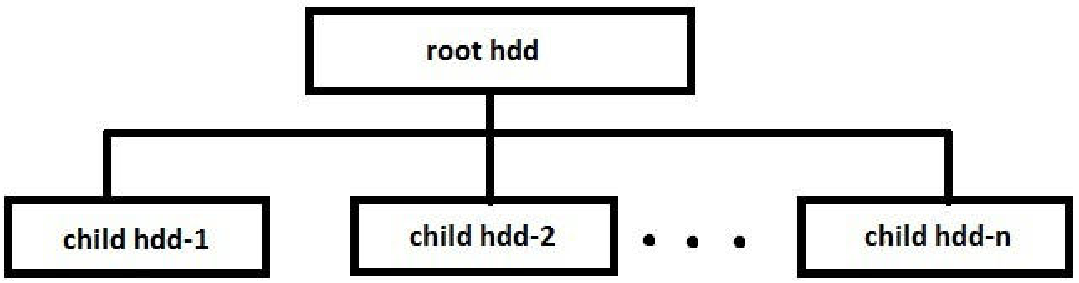

Figure 1: Hierarchical Structure of Hard disk

3. Concept of Distributed File System

The mathematical analysis suggests us to adapt a very clear architecture where data is not stored onto the single hard disk. The data is stored or spread over a range of hard disk possibly connected through the network. This architecture solves most of the problems related to the data storage and also put a requirement of a system which will manage the storage.

Definition 1: The file system which manages the storage of files across a network of machine is called Distributed File System (DFS).

The DFS performs the following tasks:

- Deciding on which node data will be stored.
- What to do when the data storage gets failed?
- How to recover data if the concerned node crashes?
- How to store meta data for fast retrieval?

4. Concept of Hadoop

This section covers architectural components of Hadoop. Hadoop makes use of master/slave architecture for both distributed storage and distributed computation. The distributed storage system is called as Hadoop Distributed File System (HDFS) and distributed computation is done with MapReduce. So, Hadoop has two major components.

- Hadoop Distributed File System (HDFS) (Storage).
- MapReduce (Processing)

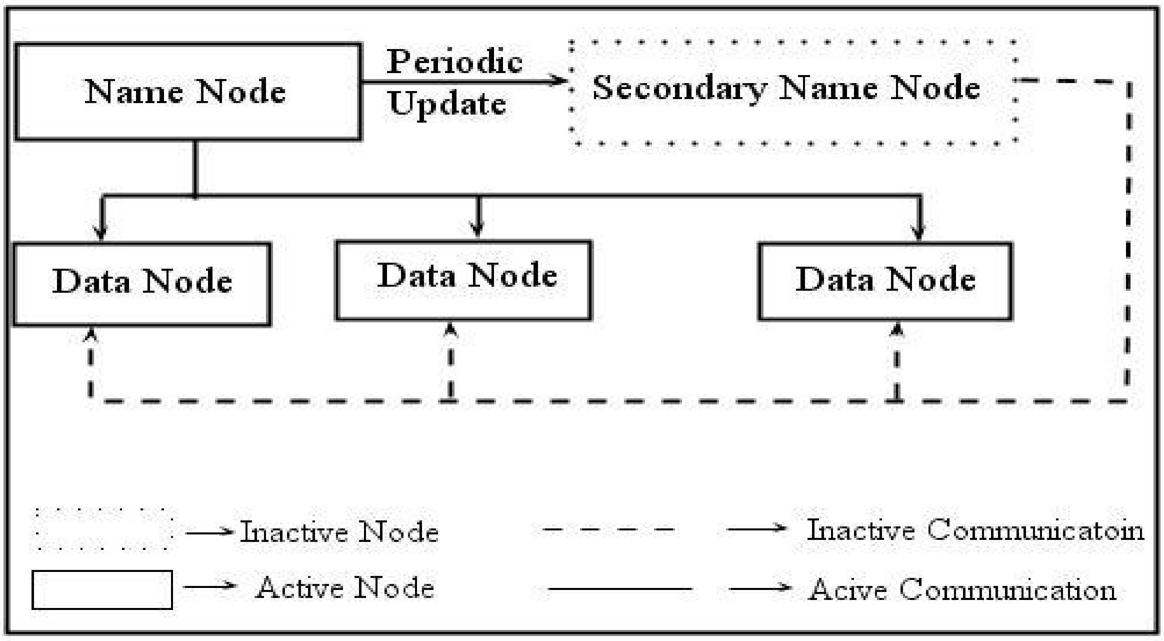

Figure 2: Hadoop File System Architecture

4.1 Hadoop File Systems

Hadoop is a file system designed for storing very large amount files with streaming data access pattern, running on clusters of commodity hardware. HDFS has two components

1. NameNode

It is master of the system which maintains and manages blocks which are present on the DataNodes in the system. It keeps track of how your files are broken down into file blocks, which nodes store those blocks, and the overall health of the distributed file system.
It is a single point of failure for a Hadoop cluster.

2. DataNodes

They are salve nodes which are deployed on each machine and provide the actual storage.

3. Secondary NameNode

It is an assistant process which monitors the state of Hadoop cluster. As we discussed earlier NameNode is a single point of failure for a Hadoop cluster, the secondary NameNode helps to minimize the downtime and loss of data.

4.2 MapReduce

The beauty of Hadoop system lies in MapReduce. It is a programming model for processing large data sets with a parallel, distributed algorithm on a cluster.

The MapReduce term actually refers to the two distinct terms: Map and Reduce. The first is the map job, which takes a set of data and converts it into an- other set of data, where individual elements are bro- ken down into tuples (key/value pairs). The reduce job takes the output from a map as input and com- bines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce job is always performed after the map job.

The master node has a Job Tracker and each slave has Task Tracker which is mainly responsible for MapRe- duce. The Job Tracker daemon keeps tracks of a job that may be assigned to multiple nodes. If the task at any node fails, the Job Tracker reschedules that task at another node. Task Tracker in turn, is responsible for keeping trace of the task that is assigned to individual node.

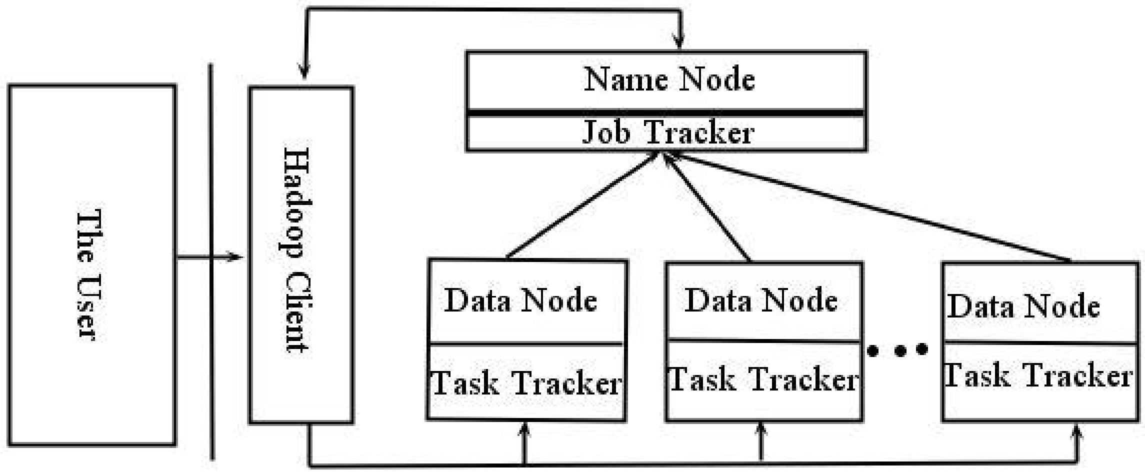

Figure 3: The Hadoop System

4.3 Features

- Hadoop is designed to operate on very large sized file typically of the range of megabytes, gigabytes, terabytes or petabytes.
- Data is accessed in the form of byte stream.
- Hadoop does not require expensive hardware. Commonly available hardware is sufficient.

4.4 Drawback

- Hadoop provides high throughput of data with the cost of latency and hence low latency data access will not work with Hadoop.
- Hadoop is not designed to work with small sized files. In Hadoop architecture metadata is separated and stored on another hard disk. So number of files that can be stored in Hadoop file system does not depend upon the total capacity of available hard disks. It only depends upon the size of hard disk that holds the metadata.

5. Conclusion

The classical centralized system was a great solution to the problem of uniform accessibility of the information which was otherwise scattered on different network node. The information is stored on a set of hard disks connected to a single large mainframe machine. But it failed to solve many complex problems like

Big Data, and frequent failure of the central system. Now a day's paradigm is shifting from centralized system to distributed system. The distributed system provides a better solution in that area normally where the centralized system got failed. But this shifting is not sudden. There is a complete mathematical result which provides a way for this shifting. The probability theory helps us to formalize the basic tree like architecture of distributed file system if we try to solve the problem like Big Data. The Hadoop system which is currently the most acceptable system used in distributed area is also following the same basic architecture with slight modification. The modification is made for the solution of failure of root of the file system. The Hadoop system not only solves the Big Data problem but also provides an API for solving problem in parallel and monitoring the progress.

References

Lam Chuck. Hadoop in Action (1st ed) (2013). Dreamtech Publication, (Chapter 1 and 2). ISBN- 978-81-7722-813-7.

White Tom. Hadoop-The Definitive Guide (3rd ed) (2012). O'reilly Publication, (Chapter 1, 2 and 3). ISBN- 978-93-5023-756-4

Hadoop Documentation, [Online] Available: http://hadoop.apache.org/docs/r0.18.3/hdfs_design.html (January 7, 2014).

Gantz F John, Chute Christopher et al. The Diverse and Exploding Digital Universe, [Online] Available: http://www.emc.com/collateral/analyst-reports/diverse-exploding-digital-universe.pdf, (December 21, 2013).

Schroeder Stan. Facebook Statistics. [Online] Available: http://mashable.com/2008/10/15/facebook-10-billion-photos/, (December 20, 2013).

Haddad Diana. Genealogy Insider (2009). The family tree magazine. [Online] Available: http://blog.familytreemagazine.com/insider/Inside+Ancestrycoms+TopSecret+Data+Center.aspx. (January 1, 2014).

Papoulis Athanasios and Pillai Unnikrishna S. Probability, Random Variables and Stochastic Processes (4th ed.) (2010) Tata McGraw-Hill Publication (Chapter 1 to 3). ISBN- 978-0-07-048658-4.

Bach J. M. The Design of the UNIX Operating System (1986). PHI Publication (Chapter 4) ISBN-978-81-203-0516-8.