Bachelor's thesis

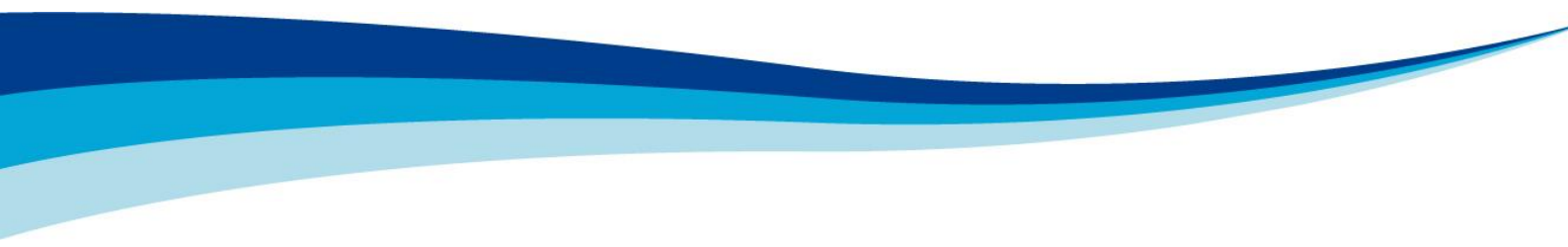
Information Technology

Internet Technology

2015

Shiqi Wu

BIG DATA PROCESSING WITH HADOOP



TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR´S THESIS | ABSTRACT
TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology | Internet Technology

June 2015| 45

Patric Granholm

Shiqi Wu

BIG DATA PROCESSING WITH HADOOP

Computing technology has changed the way we work, study, and live. The distributed data processing technology is one of the popular topics in the IT field. It provides a simple and centralized computing platform by reducing the cost of the hardware. The characteristics of distributed data processing technology have changed the whole industry.

Hadoop, as the open source project of Apache foundation, is the most representative platform of distributed big data processing. The Hadoop distributed framework has provided a safe and rapid big data processing architecture. The users can design the distributed applications without knowing the details in the bottom layer of the system.

This thesis provides a brief introduction to Hadoop. Due to the complexity of Hadoop platform, this thesis only concentrates on the core technologies of the Hadoop, which are the HDFS, MapReduce, and Hbase.

KEYWORDS:

Hadoop, Big Data, HDFS, MapReduce, Hbase, Data Processing

CONTENTS

FIGURES

LIST OF ABBREVIATIONS (OR) SYMBOLS

DDP	DDP, which stands for Distributed Data Processing, is a modern data processing technology.
GFS	GFS, which stands for Google File System, is a file storage system using ordinary personal computers.
SQL	SQL, which stands for Structured Query Language, provides a good analysis method of the relational databases.
HDFS	HDFS, which stands for Hadoop Distributed File System.
POSIX	Portable Operating System Interface
API	Application Programming Interface
RPC	Remote Procedure Call Protocol
RDBMS	Rational Database Management System
JAR	JAR, which stands for Java Archive, is a platform-independent file format. It allows many files be compressed into a zip file.

1 INTRODUCTION TO BIG DATA

The IT industry is always developing new technologies, and big data is the one of them. With the developments of the cloud storage, big data has attracted more and more attention. Due to the emergence of the Internet, the big data technology will accelerate the innovation of the enterprises, lead the revolution of the business mode and create unlimited commercial opportunities.

1.1 Current situation of the big data

In recent years, we have been drowning in the ocean of data that was produced by the development of the Internet, the Mobile Internet, the Internet of Things (loT) and the Social Networks. A photo that uploaded to Instagram is about 1MB; a video that uploaded to YouTube is about dozens of Mega sizes. Chatting online, browsing websites, playing online games, and shopping online will also turn into data that may be stored in any corner in the world. Hence, how much data is there in our daily life? According to a report of IBM, there are 2.5 quintillion bytes of data that we create every day. Ninety percent of that data was created in the recent two years. That means, in one day, the data that appears on the Internet can fill 168 million DVDs; the 294 billion emails we sent equals to the numbers of printed newspaper in United States for recent two years. (Rogell, 2012) By 2012, the volume of data has increased from Terabyte level to Petabyte level. The reducing price of computer hardware and the production of supercomputers make it possible to deal with large and complex data. All the data can be divided into four types: structured data (e.g., stock trading data), semi-structured data (e.g., blogs), unstructured data (e.g., text, audio, video), and multi-structured data.

1.2 The definition of Big Data

Finding a way to define the Big Data is not easy, but authors hold the same view with Ohlhosrt (2012) that Big Data is the large and complex data that is difficult to use the traditional tools to store, manage, and analyze in an acceptable duration. Therefore, the Big Data needs a new processing model which has the better storage, decision-making, and analyzing abilities. This is the reason why the Big Data technology was born. The Big Data technology provides a new way to extract, interact, integrate, and analyze of Big Data. The Big Data strategy is aiming at mining the significant valuable data information behind the Big Data by specialized processing. In other words, if comparing the Big Data to an industry, the key of the industry is to create the data value by increasing the processing capacity of the data. (Snijders et al, 2012)

1.3 The characteristics of Big Data

According to a research report from Gartner (Doug, 2001), the growth of the data is three-dimensional, which is volume, velocity and variety. So far, there are many industries still use the 3Vs model to describe the Big Data. However, the Big Data is not only 3Vs but also has some other characteristics (Geczy, 2014). The first one is the volume. As mentioned, the volume of Big Data has moved from Terabyte level to Petabyte level. The second one is the variety. Compared with the traditional easy to storage structured text data, there is now an increasing amount unstructured data that contains web logs, audio, video, pictures, and locations. Data no longer needs to be stored as traditional tables in databases or data warehouses but also stored as variable data types at the same time. To meet this requirement, it is urgent to improve the data storage abilities. Next is velocity. Velocity is the most significant feature to distinguish the Big Data and the traditional data mining. In the Age of Big Data, the volume of high concurrency access of users and submittion data are huge. Velocity of interactive response occurs when a user submits his or her request to the

server and the corresponding speed should be fast instead of making the users wating for a long time. Taking the Facebook as example, when billions of users submit their request to access to Facebook at the same time, this requires a rapid speed of interactive response to each user. Then is the value. The rate of valuable data is in inverse proportion to the total data volume so that the ratio of valuable data is quite low. For instance, under the continuously monitoring of a one-hour video, the useful data may be a few seconds only. How to combine the business logic and powerful algorithm to mine the treasure is the hardest puzzle of the big data technology. Last but not at least is the online property. Big Data is always online and can be accessed and computed. With the rapid developments of the Internet, the Big Data is not only big but is also getting online. Online data is meaningful when the data connects to the end users or the customers. Taking an example, when users use Internet applications, the users' behavior will be delivered to the developers immediately. These developers will optimize the notifications of the applications by using some methods to analyze the data. Pushing the notifications that users want to see most is also a brilliant way to promote the user experience.

2 BASIC DATA PROCESSING PLATFORM

Distributed Data Processing (DDP) is not only a technical concept but also a logical structure. The concept of DDP is based on the principle that can achieve both centralized and decentralized information service. (Enslow, 1978)

2.1 Capability components of DDP Platform

DDP platforms have different capability components to help it to complete the whole process. Different capability components are responsible for different jobs and aspects. The following sections will introduce the most important capability components of a DDP platform.

a) File Storage

The file storage capability component is the basic unit of data management in the data processing architecture. It aims to provide a fast and reliable access ability to meet the needs of large amount of data computing.

b) Data Storage

The data storage capability component is an advanced unit of data management in the data processing architecture. It aims to store the data according to an organized data model and to provide an independent ability of deleting and modifying data. IBM DB2 is a good example of a data storage capability component.

c) Data Integration

The data integration capability component integrates the different data which has different sources, formats, and characters into units to support the data

input and output between multiple data sources and databases. Oracle Data Integrator is an example of data integration component.

d) Data Computing

The data computing capability component is the core component of the whole platform. It aims to solve the specific problem by using the computing resources of the processing platform. Taking MPI (Message Passing Interface) which is commonly used in parallel computing as an example, it is a typical datacomputing component. In the Big Data environment, the core problem is how to split the task that needs huge computing ability to calculate into a number of small tasks and assign them into specified computing resources to processing.

e) Data analysis

The data analysis capability component is the closest component to the users in the data processing platform. It aims to provide an easy way to support the user to extract the data related to their purpose from the complex information. For instance, as a data analysis component, SQL (Structured Query Language) provides a good analysis method for the relational databases. Data analysis aims at blocking the complex technical details in the bottom layer of the processing platform for the users by abstract data access and analysis. Through the coordinates of data analysis components, the users can do the analysis by using the friendly interfaces rather than concentrate on data storage format, data streaming and file storage.

f) Platform Management

The platform management capability component is the managing component of the data processing. It aims to guarantee the safety and stability for the data processing. In the Big Data processing platform, it may consist of a large

mount of servers that may be distributed in different locations. On this occasion, how to manage these servers' work efficiently to ensure the entire system running is a tremendous challenge.

2.2 Applications of DDP Platform

2.2.1 Google Big Data Platform

Most of the technological breakthroughs come from the actual product needs. Admittedly, the Big Data concept was born in the Google search engine originally. With the explosive growth of the Internet data, in order to meet the requirements of information searching, data storage has become a difficult issue. Based on the considerations of the costs, solving the large quantities of searching data by improving hardware became more and more impractical. As a result, Google came up with a reliable file storage system based on software, which is GFS (Google File System). GFS uses an ordinary PC to support massive storage. Because saving data is worthless, only the data processing can meet the requirements of the actual applications. Then, Google created a new computing model named MapReduce. MapReduce can split the complex calculations into seperate PCs. Obtaining the final results though the summarization of single calculation on every PC so that MapReduce can gain better operation ability by increasing the number of the machines. After GFS and MapReduce were launched, the ability of file storage and computation was solved, but there was a new problem. Because of the poor random I/O ability of GFS, Google needed a new format database to store the data. This is the reason why Google designed the BigTable database system (Google Cloud Platform).

2.2.2 Apache Hadoop

After Google had competed the system, the concepts of the system were published out as papers. Based on these papers, the developers wrote an open source softerware Hadoop in JAVA. Now, the Hadoop is under the Apache Foundation.

3 HADOOP

As mentioned in the Chapter 2, Hadoop is a distributed system infrastructure researched and developed by the Apache Foundation. The users can develop the distributed applications although they do not know the lower distributed layer so that the users can make full use of the power of the cluster to perform the high-speed computing and storage. The core technologies of Hadoop is the Hadoop Distributed File System (HDFS) and the MapReduce. HDFS provides the huge storage ability while MapReduce provides the computing ability of the Big Data. Since HDFS and MapReduce have become open source, their low cost but high processing performance helped them to be adopted by many enterprises and organizations. With the popularity of the Hadoop technologies, there are more tools and technologies which are developed on the basis of the Hadoop framework.

3.1 Relevant Hadoop Tools

Nowadays, Hadoop has developed a collection of many projects. Although the core technologies are HDFS and MapReduce, Chukwa, Hive, Hbase, Pig, Zookeeper and so on are also indispensable. They provide the complementary services and higher level service on the core layer.

- MapReduce is a programming model for parallel computing on the large-scale data sets. The concepts of Mapping and Reducing are referenced to the functional programming languages. The programmers who are not familiar with distributed parallel programming can also run their programs on the distributed system. (Dean&Ghenmawat, 2008)
- HDFS is a distributed file system. Because is is high fault-tolerant, it can be applied on the low-cost hardware. By providing the high throughput access to the data of the applications, it is suitable for applications which contain the huge data sets. (Borthakur, 2008)

- Chukwa is an open source data collection system which is used for data monitoring and analysis. Chukwa is built on the HDFS and MapReduce framework. Chukwa stores the data by HDFS and relies on the MapReduce to manage the data. Chukwa is a flexible and powerful tool to display, monitoring, and analyze the data. (Yang et al., 2008)

- Hive was originally designed by Facebook. It is a data warehouse based on the Hadoop which provides data sets searching, special query, and analysis. Hive is a structured data mechanism which supports the SQL query languages like RDBMS to help those users who are familiar with SQL queries. This kind of query language is called Hive QL. In fact, the traditional MapReduce programmers can query data on the Mapper or Reducer by using Hive QL. The Hive compiler will compile the Hive QL into a MapReduce task. (Thusoo et al., 2009)

- Hbase is a distributed and open source database. As mentioned in Chapter2, the concepts of Hadoop originally came from Google, therefore, Hbase shares the same data model. Because the forms of Hbase are loose and the users can define various columns, Hbase is usually used for Big Data. (George, 2011) This will be discussed further in the Chapter6.

- Pig is a platform which was designed for the analysis and evaluation of the Big Data. The significant advantage of its structure is that it can afford the highly parallel test which is based on the parallel computing. Currently, the bottom layer of the Pig is composed of a complier. When the complier is running, it will generate some program sequences of MapReduce.

- Zookeeper is an open source coordination service for the distributed applications. It is used mainly to provide users' synchronization, configuration management, grouping, and naming services. It can reduce the coordination tasks for the distributed applications. (Hut&etc, 2010)

These are all the related tools with Hadoop, but this thesis will only concentrate on HDFS, MapReduce, and Hbase which are the core technologies of Hadoop.

4 HDFS

HDFS is a distributed system which is suitable for running on the commodity hardware. There are many common characteristics in the existing distributed systems but the differences between them are also obvious. HDFS is a high fault-tolerant system and relaxed the parts of the POSIX constraints to provide high throughput access to the data so that it can be suitable to applying on the Big Data. (Hadoop, 2013)

4.1 HDFS Architecture

HDFS is the master/slave structure. The Namenode is the master node, while the Datanode is the slave node. Documents are stored as data blocks in the Datanode. The default size of a data block is 64M and it cannot be changed. If the files are less than a block data size, HDFS will not take up the whole block storage space. The Namenode and the Datanode normally run as Java programs in the Linux operating system.

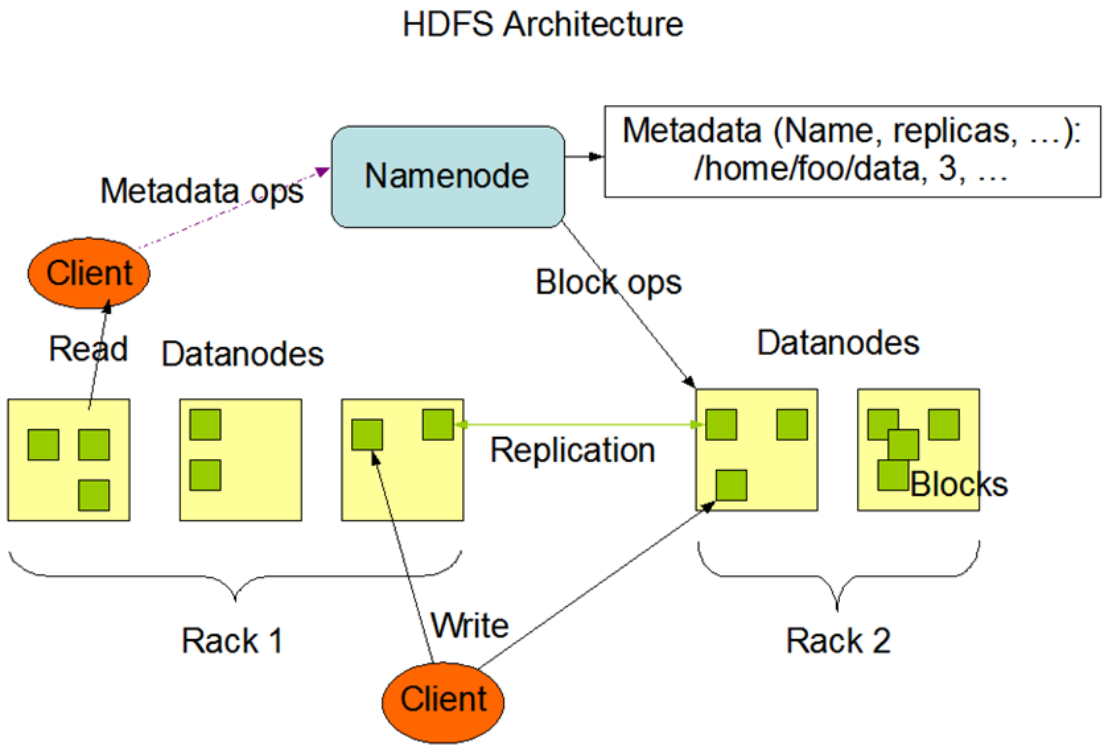


Figure 1. HDFS architecture (Borthakur, 2008)

According to Figure1(Borthakur,2008), the Namenode which is the manager of the HDFS is responsible for the management of the namespace in the file system. It will put all the folders and files metadata into a file system tree which maintains all the metadata of the files directories. At the same time, Namenode also saves the corresponding relations between each file and the location of the data block. Datanode is the place to store the real data in the system. However, all the data is not stored on the hard drives but will be collected when the system starts to find the resource data server of the required documents.

The Secondary Namenode is a backup node for the Namenode. If there is only one Namenode in the Hadoop cluster environment, the Namenode will obviously become the weakest point of the process in the HDFS. Once the failure of the Namenode occurs, it will affect the whole operation of the system. This is the reason why Hadoop designed the Secondary Namenode as the alternative backup. The Secondary Namenode usually runs on a separate physical computer and keeps communication at certain time interval to keep the snapshot of the file system metadata with the Namenode so that it can recovery the data immediately in case some error happens.

The Datanode is the place where the real data is saved and handles most of the fault-tolerant mechanism. The files in HDFS are usually divided into multiple data blocks stored in the form of redundancy backup in the Datanode. The Datanode reports the data storage lists to the Namenode regularly so that the user can obtain the data by directly access to the Datanode.

The Client is the HDFS user. It can read and write the data though calling the API provided by HDFS. While in the read and write process, the client first needs to obtain the metadata information from the Namenode, and then the client can perform the corresponding read and write operations.

4.2 Data Reading Process in HDFS

The data reading process in HDFS is not difficult. It is similar to the programming logic which has created the object, i.e., calling the method and pergorming the execution. The following section will introduce the reading processing of the HDFS.

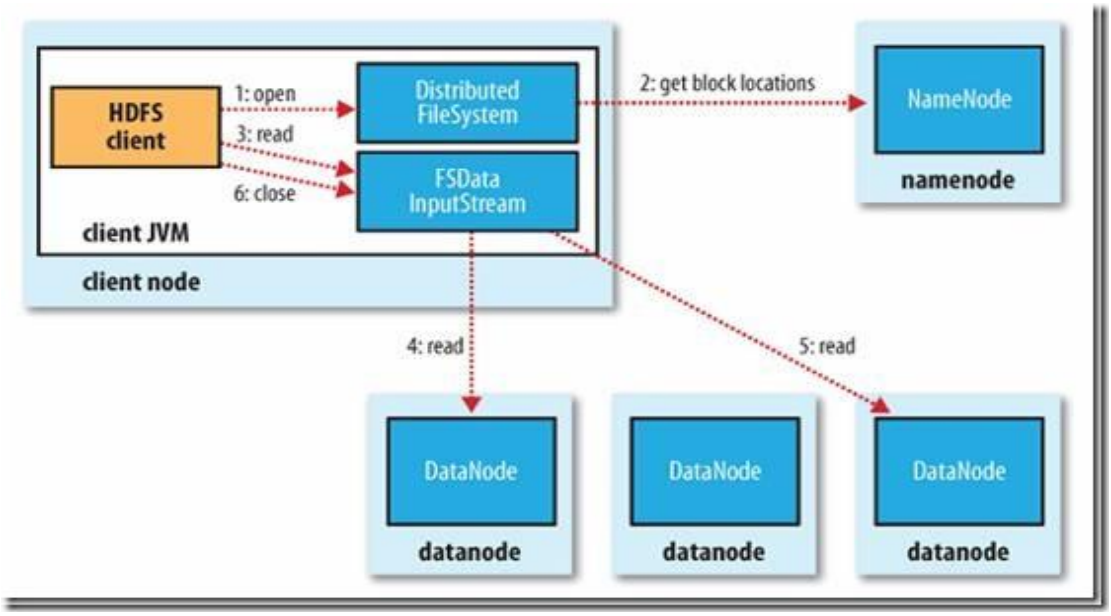


Figure 2. HDFS reading process(White,2009)

According to Figure2, there are six steps when the HDFS has the reading process:

(1) The client will generate an DistributedFileSystem object of the HDFS class library and uses the open() interface to open a file.

(2) DistributedFileSystem sends the reading request to the Namenode by using the Remote Procedure Call Protocol to obtain the location address of the data block. After the calculating and sorting the distance between the client and the Datanode, the Namenode will return the location information of the data block to the DistributedFileSystem.

(3) After, the DistributedFileSystem has already received the distances and address of the data block, it will generate a FSDataInputStream object instance to the client. At the same time, the FSDataInputStream also encapsulates a DFSInputStream object which is responsible for saving the storing data blocks and the address of the Datanode.

(4) When everything get ready, the client will call the read() method.

(5) After receiving the calling method, the encapsulated DFSInputStream of FSDataInputStream will choose the nearest Datanode to read and return the data to the client.

(6) When all the data has been read successfully, the DFSInputStream will be in charge of closing the link between the client and the Datanode.

While the DFSInputStream is reading the data from the Datanode, it is hard to avoid the failure that may be caused by network disconnection or node errors. When this happens, DFSInputStream will give up the failure Datanode and select the nearest Datanode. In the later reading process, the disfunctioning Datanode will not be adopted anymore. It is observed that HDFS separates the index and data reading to the Namenode and Datanode. The Namenode is in charge of the light file index functions while the heavy data reading is accomplished by several distributed Datanodes. This kind of platform can be easily be adapted to the multiple user access and huge data reading.

4.3 Data Writing Process in HDFS

The data writing process in HDFS is the opposite process of the reading but the writing process is more complex. The following section will introduce the writing process in HDFS briefly.

The structure of HDFS reading process is similar to the writing process. There are the following seven steps:

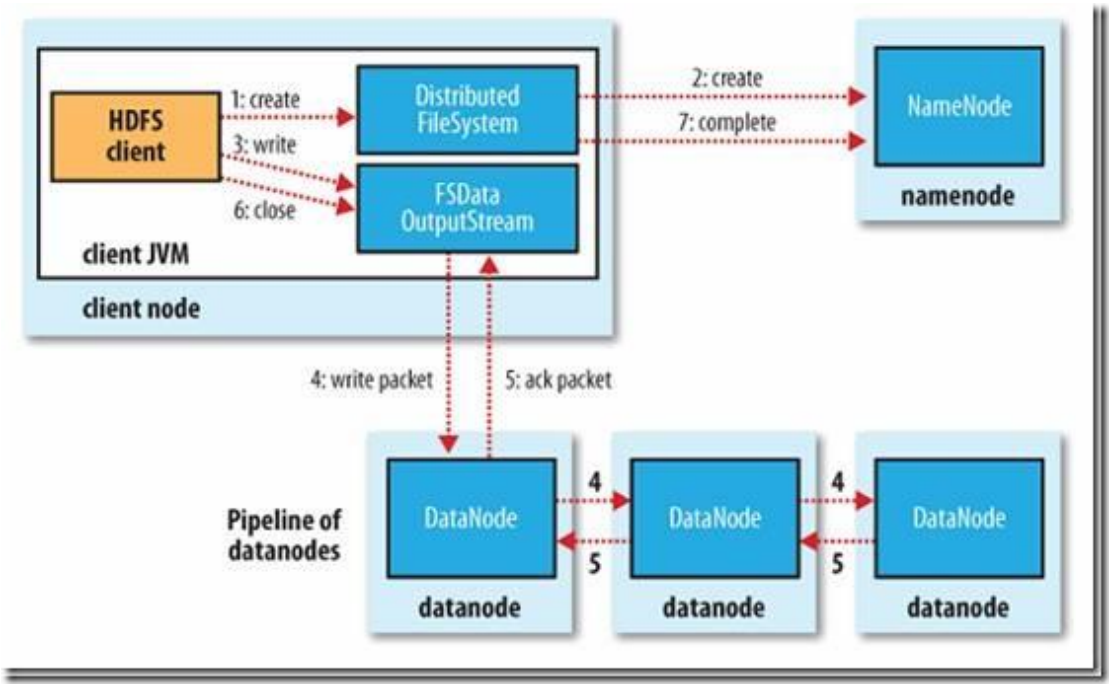


Figure 3. HDFS writing process (White, 2009)

(1) The client generates a DistributedFileSystem object of the HDFS class library and uses the create() interface to open a file.

(2) DistributedFileSystem sends the writing request to the Namenode by using the Remote Procedure Call Protocal (RPC). The Namenode will check if there is a duplicate file name in it. After that, the client with writing authority can create the corresponding records in the namespace. If an error occurrs, the Namenode will return the IOException to the client.

(3) After the DistributedFileSystem has received the successful return message from the Namenode, it will generate a FSDataOutputStream object to the client. In the FSDataOutputStream, there is an encapsulated DFSOutputStream object which is responsible for the writing process. The client calls the write() method and sends the data to the FSDataInputStream.

The DFSOutputStream will put the data into a data queue which is read by the DataStreamer. Before the real writing, the DataStreamer needs to ask for some blocks and the suitable address from the Datanode to store the data.

(4) For each data block, the Namenode will assign several Datanodes to store the data block. For instance, if one block needs to stored in three Datanodes. DataStreamer will write the data block at the first Datanode, then the first Datanode will pass the data block to the second Datanode, and the second one passes to the third one. Finally, it will complete the writing data in the Datanode chain.

(5) After every Datanode has been written, Datanode will report to the DataStreamer. Step4 and Step5 will be repeated until all the data has been written successfully.

(6) When all the data has been written, the client will call the close() method of FSDataInputStream to close the writting operation.

(7) Finally, the Namenode will be informed by the DistributedFileSystem that all the written process has been completed.

In the process of data writing, if one Datanode makes error and causes writing failure, all the links between the DataStreamer and the Datanode will be closed. At the same time, the failure node will be deleted from the Datanode chain. The Namenode will notice the failure by the returned packages and will assign a new Datanode to continue the processing. As long as one Datanode is written successfully, the writing operation will regard the process as completed.

4.4 Authority management of HDFS

HDFS shares a similar authority system to POSIX. Each file or directory has on owner and a group. The authority permissions for the files or the directories

are different to the owner, users in the same group, and other users. On the one hand, for the files, users are required the –r authority to read and the –w authority to write. On the other hand, for the directories, users need the –r authority to list the directory content and –w authority to create or delete. Unlike the POSIX system, there is no sticky, setuid or setgid of directories because there is no concept of executable files in HDFS. (Gang, 2014)

4.5 Limitations of HDFS

HDFS as the open source implementation of GFS is an excellent distributed file system and has many advantages. HDFS was designed to run on the cheap commodity hardware not on expensive machines. This means that the probabilities of node failure are slightly high. To give a full consideration to the design of HDFS, we may find that HDFS has not only advantages but also limits for dealing with some specific problems. These limitations are mainly displayed in the following aspects:

- High Access Latency

HDFS does not fit fort requests which should be applied in a short time. The HDFS was designed for the Big Data storage and it is mainly used for it high throughput abilities. This may cost the high latency instead. (Borthakur, 2007) Because HDFS has only one single Master system, all the file requests need to be processed by the Master. When there is a huge number of requests, there is inevitably has the delay. Currently, there are some additional projects to address this limitation, such as using the Hbase uses the Upper Data Management project to manage the data. The Hbase will be discussed in Chapter 6 of this thesis.

- Poor small files performance

HDFS needs to use the Namenode to manage the metadata of the file system to respond to the client and return the locations so that the limitation of a file

size is determined by the Namenode. In general, each file, folder, and block need to occupy the 150 bytes' space. In other words, if there are one million files and each file occupies one block, it will take 300MB space. Based on the current technology, it is possible to manage millions of files. However, when the files extend to billions, the work pressures on the Namenode is heavier and the time of retrieving data is unacceptable. (Liu et al., 2009)

- Unsupported multiple users write permissions

In HDFS, one file just has one writer because multiple users' writer permissions are not supported yet. The write operations can only be added at the end of the file not at the any positions of the file by using the Append method.

We believe that, with the efforts of the developers of HDFS, HDFS will become more powerful and can meet more requirements of the users.

5 MAPREDUCE

In 2004, Google published a thesis (Dean& Ghemawat, 2004) to introduce MapReduce. When Google published the thesis, the greatest achievement of the MapReduce is that it can rewrite the Google's index file system. Until now, MapReduce is widely used in logs analysis, data sorting, and specific data searching. According to the Google's thesis, Hadoop implements the programming framework of the MapReduce and renders it an open source framework. MapReduce is the core technology of Hadoop. It provides a parallel computing model for the Big Data and supports a set of programming interfaces for the developers.

5.1 Introduction of MapReduce

MapReduce is a standard functional programming model. This kind of model has been used in the early programming languages, such as Lisp. The core of the calculation model is that can pass the function as the parameter to another function. Through multiple concatenations of functions, the data processing can turn into a series of function execution. MapReduce has two stage of processing. The first one is Map and the other one is Reduce. The reason why the MapReduce is popular is that it is very simple, easy to implement, and offers strong expansibility. MapReduce is suitable for processing the Big Data because it can be processed by the multiple hosts at the same time to gain a faster speed.

5.2 MapReduce Architecture

The MapReduce operation architecture includes the following three basic components (Gaizhen, 2011):

- Client: Every job in the Client will be packaged into a JAR file which is stored in HDFS and the client submits the path to the JobTracker.

- JobTracker: The JobTracker is a master service which is responsible for coordinating all the jobs that are executed on the MapReduce. When the software is on, the JobTracker is starting to receive the jobs and monitor them. The functions of MapReduce include designing the job execution plan, assigning the jobs to the TaskTracker, monitoring the tasks, and redistributing the failed tasks.

- TaskTracker: The TaskTracker is a slave service which runs on the multiple nodes. It is in charge of executing the jobs which are assigned by the JobTracker. The TaskTracker receives the tasks through actively communicating with the JobTracker.

5.3 MapReduce Procedure

The MapReduce procedure is complex and smart. This thesis will not discuss the MapReduce procedure in detail but will introduce it briefly based on author's own thoughts.

Usually, MapReduce and HDFS are running in the same group of nodes. This means that the computing nodes and storage nodes are working together. This kind of design allows the framework to schedule the tasks quickly so that the whole cluster will be used efficiently. In brief, the process of MapReduce

can be divided into the following six steps (He, Fang& Luo, 2008):

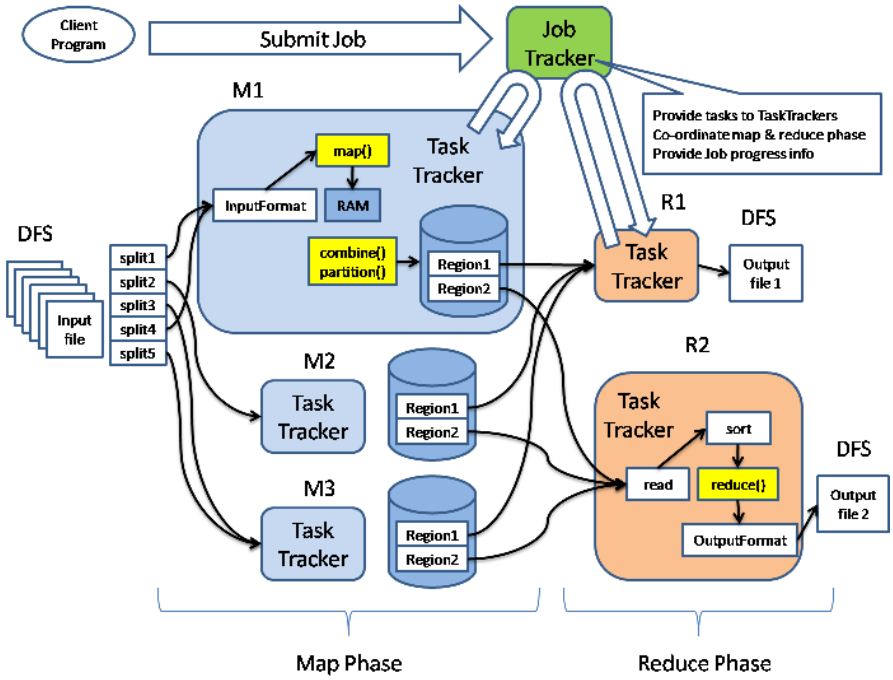


Figure 4. MapReduce procedure(Ricky,2008)

(1) Job Submission

When the user writes a program to create a new JobClient, the JobClient will send the request to JobTracker to obtain a new JobID. Then, the JobClient will check if the input and output directories are correct. After this check, JobClient will store the related resources which contain the configuration files, the number of the input data fragmentations, and Mapper/Reducer JAR files to HDFS. In particular, the JAR files will be stored as multiple backups. After all the preparations have been competed, the JobClient will submit a job request to the JobTracker.

(2) Job Initialization

As the master node of the system, JobTracker will receive several JobClient requests so that JobTracker implements a queue mechanism to deal with these problems. All the requests will be in a queue that is managed by the job

scheduler. When the JobTracker starts to initialize, its job is to create a JobInProgress instance to represent the job. The JobTracker needs to retrieve the input data from HDFS and to decide on the number of the Map tasks. The Reduce tasks and the TaskInProgress are determined by the parameters in the configuration files.

(3) Task Allocation

The task allocation mechanism in the MapReduce is to pull the whole process. Before the task allocation, the TaskTracker which is responsible for Map tasks and Reduce tasks has been already launched. The TaskTracker will send the heartbeat message to the JobTracker to ask if there are any tasks that can be done any time. When the JobTracker job queue is not empty, the TaskTracker will receive the tasks to do. Due to the lack of the TaskTracker computing capability, the tasks that can be done on the TaskTracker are also limited. Each TaskTracker has two fixed task slots which correspond to the Map tasks and Reduce tasks. During the tasks allocation, the JobTracker will use the Map task slot first. Once the Map task slot is empty, it will be assigned to the next Map task. After the Map task slot is full, then the Reduce task slot reveives the tasks to do.

(4) Map Tasks Execution

After the Map TaskTracker has received the Map tasks, there is a series of operations to finish the tasks. Firstly, the Map Task Tracker will create a TaskInProgress object to schedule and monitor the tasks. Secondly, the Map TaskTracker will take out and copy the JAR files and the related parameter configuration files from HDFS to the local working directory. Finally, when all the preparations have been competed, the TaskTracker will create a new TaskRunner to run the Map task. The TaskRunner will launch a separate JVM and will start the MapTask inside to execute the map() function in case the abnormal MapTask affects the normal TaskTracker works. During the process,

the MapTask will communicate with TaskTracker to report the task progress until all the tasks are completed. At that time, all the computing results will be stored in the local disk.

(5) Reduce Tasks Execution

When the part of the Map Tasks completed, the JobTracker will follow a similar mechanism to allocate the tasks to the Reduce TaskTracker. Similar to the process of Map tasks, the Reduce TaskTracker will also execute the reduce() function in the separate JVM. At the same time, the ReduceTask will download the results data files from the Map TaskTracker. Until now, the real Reduce process has not started yet. Only when all the Map tasks have been completed, the JobTracker will inform the Reduce TaskTracker to start to work. Similarly, the ReduceTask will communicate with the TaskTracker about the progress until the tasks are finished.

(6) Job Completion

In the each Reduce execution stage, every ReduceTask will send the result to the temporary files in HDFS. When all the ReduceTasks are competed, all these temporary files will be combined into a final output file. After the JobTracker has received the completion message, it will set the state to show that jobs done. After that, the JobClient will revceive the completion message, then notify the user and display the necessary information.

5.4 Limitations of MapReduce

Although MapReduce is popular all over the world, most people still have realized the limits of the MapReduce. There are following the four main limitations of the MapReduce (Ma&Gu, 2010):

- The bottleneck of JobTracker

From the previous chapters, the JobTracker should be responsible for jobs allocation, management, and scheduling. In addition, it should also communicate with all the nodes to know the processing status. It is obvious that the JobTracker which is unique in the MapReduce, taks too many tasks. If the number of clusters and the submission jobs increase rapidly, it will cause network bandwidth consumption. As a result, the JobTracker will reach bottleneck and this is the core risk of MapReduce.

- The TaskTracker

Because the jobs allocation information is too simple, the TaskTracker might assign a few tasks that need more sources or need a long execution time to the same node. In this situation, it will cause node failure or slow down the processing speed.

- Jobs Delay

Before the MapReduce starts to work, the TaskTracker will report its own resources and operation situation. According to the report, the JobTracker will assign the jobs and then the TaskTracker starts to run. As a consequence, the communication delay may make the JobTracker to wait too long so that the jobs cannot be completed in time.

- Inflexible Framework

Although the MapReduce currently allows the users to define its own functions for different processing stages, the MapReduce framework still limits the programming model and the resources allocation.

5.5 Next generation of MapReduce: YARN

In order to solve above limitations, the designers have put forward the next generation of MapReduce: YARN. Given the limitations of MapReduce, the main purpose of YARN is to divide the tasks for the JobTracker. In YARN, the

resources are managed by the ResourceManager and the jobs are traced by the ApplicationMaster. The TaskTracker has become the NodeManager. Hence, the global ResourceManager and the local NodeManager compose the data computing framework. In YARN, the ResourceManager will be the resources distributor while the ApplicationMaster is responsible for the communication with the ResourceManager and cooperate with the NodeManager to complete the tasks.

5.5.1 YARN architecture

Compared with the old MapReduce Architecture, it is easy to find out that YARN is more structured and simple. Then, the following section will introduce the YARN architecture.

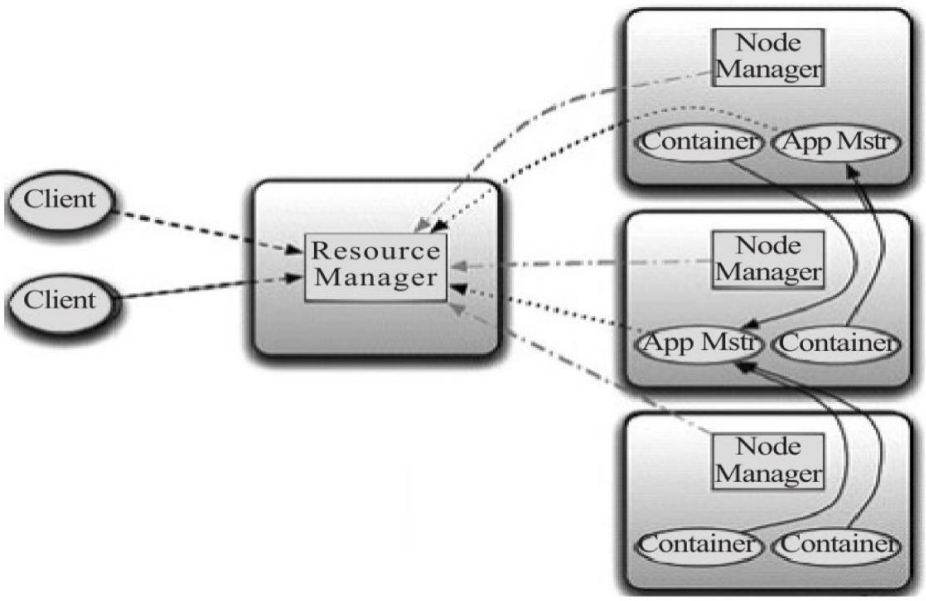


Figure 5. YARN architecture(Lu,2012)

According to Figure 5, there are following four core components of the YARN Architecture:

(1) ResourceManager

According to the different functions of the ResourceManager, it designers has divided it into two lower level components: The Scheduler and the ApplicationManager. On the one hand, the Scheduler assigns the resource to the different running applications based on the cluster size, queues, and resource constraints. The Scheduler is only responsible for the resources allocation but is not responsible for the monitoring the application implementation and task failure. On the other hand, the ApplicationManager is in charge of receiving jobs and redistributing the containers for the failure objects.

(2) NodeManager

The NodeManager is the frame proxy for each node. It is responsible for launching the application container, monitoring the usage of the resource, and reporting all the information to the Scheduler.

(3) ApplicationMaster

The ApplicationMaster is cooperating with the NodeManager to put tasks in the suitable containers to run the tasks and monitor the tasks. When the container has errors, the ApplicationMaster will apply for another resource from the Scheduler to continue the process.

(4) Container

In YARN, the Container is the source unit which is the available node splitting the organization resources. Instead of the Map and Reduce source pools in MapReduce, the ApplicationMaster can apply for any numbers of the Container. Due to the same property Containers, all the Containers can be exchanged in the task execution to improve efficiency.

5.5.2 Advantages of YARN

Compared to the MapReduce, there are many advantages of the YARN framework. There are four main advantages of YARN compared to the MapReudce (Adam, 2014).

- YARN greatly enhances the scalability and availability of the cluster by distributing the tasks to the JobTracker. The ResourceManager and the ApplicationMaster greatly relieves the bottleneck of the JobTracker and the safety problems in the MapReduce.
- In YARN, the ApplicationMaster is a customized component. That means that the users can write their own program based on the programming model. This makes the YARN more flexible and suitable for wide use.
- YARN, on the one hand, supports the program to have a specific checkpoint. It can ensure that the ApplicationMaster can reboot immediately based on the status which was stored on HDFS. On the other hand, it uses the ZooKeeper on the ResourceManager to implement the failover. When the ResourceManager reveives errors, the backup ResourceManager will reboot quickly. These two measures improve the availability of YARN.

The cluster has the same Containers are the Reduce and Map pools in MapReduce. Once there is a request for resources, the Scheduler will assign the available resources in the cluster to the tasks and regard the resource type. It will increase the utilization of the cluster resources.

6 HBASE

6.1 Limitations of the traditional database

With the development of the Internet technology, especially the Web 2.0 websites, like Facebook, and Twitter, the data processing technology has to face the problem of the changes in data amount, data structures, and the processing requirements. All these changes and problems have brought great challenges to the traditional relational database, mainly reflected in three respects (Bloor, 2003). The first one is the tradional databases cannot adapt to the various data structures. In the modern network, there are large amounts of semi-structured and unstructured data, for instance, the emails, webpages, and videos. For the traditional relational databases that are designed for structured data, it is difficult to handle the various data efficiently. The second limitaion is that traditional databases are unable to handle the high concurrent writing operations. In the majority of the new websites, it is common that the websites need to generate dynamic web pages according to the customized features to display the data, like the social updates. At the same time, the users' operations on the website will be stored as the behavior data in the database. There is a huge difference between the traditional static pages and the modern pages. The traditional relational database is not good at the high concurrency writing operation. Last but not at least, the traditional relational databases are unable to manage the rapid changes of the business types and traffic. Under the modern network environment, the business types and traffic may change rapidly in a short time. Take the Facebook as an example, the number of users may increase from millions to the billions in one month. If there are new features launched, the traffic of the website will also increase quickly. These changes need the database to have a powerful extensibility in the underlying hardware and data structure design. It is also one of the weaknesses of the traditional relational databases.

As a result, there is a new database technology, called NoSQL. It needs to be mentioned here that the NoSQL means Not only SQL. In other words, NoSQL does not abandon the traditional relational database and SQL, but it also establishes a faster and extensible database. Hbase is also using the NoSQL technology.

6.2 Introduction of Hbase

Hbase is the Hadoop database which can provide real-time access to the data and powerful scalability. Hbase was designed based on the Bigtable which is a database was lauched by Google. Hbase aims at storing and processing Big Data easily. More specifically, it uses a general hardware configuration to process millions of data. Hbase is an open source, distributed, has multiple versions, and uses the NoSQL database model. It can be applied on the local file systems and on HDFS. In addition, Hbase can use the MapReduce computing model to parallel process Big Data in Hadoop. This is also the core feature of Hbase. It can combine data storage with parallel computing perfectly.

6.3 Architecture of Hbase

Hbase is in the storage layer in the Hadoop. Its underlying storage support is HDFS, using the MapReduce framework to process the data, and cooperate with the ZooKeeper.

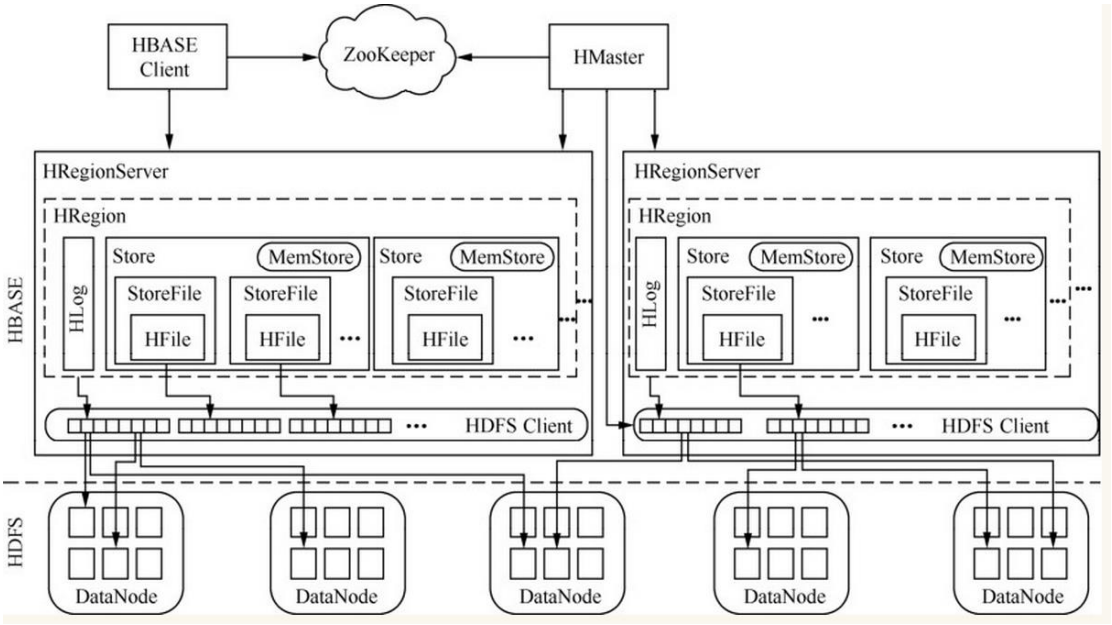


Figure 6. Hbase Architecture(Liu,2013)

According to Figure 6, there are following four key components:

- Hbase Client: The client is the user of the Hbase. It takes part in the manage operations with HMaster and read/write operations with HRegionServer.

- ZooKeeper: ZooKeeper is the collaborative management node of Hbase. It can provide distributed collaboration, distributed synchronization, and configuration functions. The ZooKeeper coordinates all the clusters of Hbase by using data which contains the HMaster address and HRegionServer status information.

- HMaster: HMaster is the controller of the Hbase. It is responsible for adding, deleting, and quering the data. It adjusts the HRegionServer load balance and the Region distribution to ensure that the Region will move to the next Region when the HRegionServer suffers failure. An Hbase environment can launch multiple HMaster to avoid failure. At the same time, there is always a Master Election mechanism working in case of the node failure.

- HRegionServer: HRegionServer is the core component of Hbase. It is responsible for handling the reading and writing requests for the users and performing the corresponding operations on HDFS.

6.4 Comparison of RDBMS and Hbase

Hbase, as the representative database, is often compared with the traditional RDBMS. The design target, implementation mechanism, and running performance are different. Due to the reason that the Hbase and RDBMS can replace each other in some special situations, it is inevitable to compare RDBMS with Hbase. As mentioned before, Hbase is a distributed database system and the underlying physical storage uses the Hadoop distributed file system. It does not have particularly strict requirements on the hardware platform. However, RDBMS is a fixed structure database system. The difference between their design goals makes them have the greatest difference in the implementation mechanism. These can be compared in the following aspects (wikiDifference):

- The hardware requirements: RDBMS organizes the data in rows so that it needs to read the entire line data even though the users just need a few columns of data. This means that RDBMS needs a large amount or expensive high performance hardware to meet the requirements. Hence, RDBMS is a typical IO bottleneck system. When an organization adopting the RDBMS to build a Big Data processing platform, the cost may be too high. On the contrary, Hbase, as a typical new database is based on columns, which facilitates easier access to the data with same attributes resulting in improved access speed to the data. Compared with RDBMS, Hbase has the higher processing efficiency due to its columns based design. At the same time, at the beginning of design the Hbase, the costs of implementing the wholes system have been considered. Through the underlying distributed file system, Hbase can run on a large number of

low-cost hardware clusters and maitain a high concurrent throughput performance.

- The extensibility: An excellent database system should be able to extend continuously with the growth of data. Although RDBMS can have a limited extensibility by using some technologies like Memcached, its technical architecture does not support it to improve the extensibility by simply adding the nodes. However, Hbase has taken the extensibility in the Big Data environment into account at the beginning of the design. Based on the parallel processing capability on HDFS, Hbase can increase the extensibility by simply adding the RegionServer.

- The reliability: The storage nodes failure in RDBMS usually means disastrous data loss or system shut down. Although RDBMS can have a certain degree of reliability through the master-slave replication mechanism, it can also improve the fault-tolerance ability by using the standby hot node mechanism but it often requires multiple times hardware costs to achieve.

- Difficulty in use: On the one hand, RDBMS has gone through many years of practical applications so that it is not difficult for the regular SQL users or senior application developers. On the other hand, the applications that developed on the RDBMS are not difficult, because the row oriented database design is easier to accept and understand. Compared to RMDBS, Hbase and the development mode of MapReduce are still at the early promotion stage and the advanced developers are relatively rare so that the difficulty of Hbase development is high. Nevertheless, with the developments of the Hadoop technology, the inherent advantages of Hbase in data processing and architecture will contribute to the popularity of Hbase.

- The Maturity. It is obvious that the maturity of the RDBMS is higher than the Hbase so that if the data is not as huge as RDBMS cannot manage, RDBMS is still the first choice in the majority cases. Compared with the Hbase, RDBMS is more stable and the technology is more mature. For Hbase, there are some deficiencies in some key features and optimization support.
- Processing features: RDBMS is more suitable for real-time analysis while Hbase is more suitable for non-real-time big data processing.

Based on the comparison above, it is clear that the RDBMS is suitable for the majority of small-scale data management conditions. Only when the potential requirements of data processing have reached the hundreds of millions level, Hbase should be still considered as the best choice.

7 THE APPLICATIONS OF HADOOP

Nowadays, with the rapid growth of the data volume, the storage and processing of Big Data has become the most pressing needs of the enterprises. Hadoop as the open source distributed computing platform has become a brilliant choice for the business. The users can develop their own distributed applications on Hadoop and processing Big Data even if they do not know the bottom-level details of the system. Due to the high performance of Hadoop, it has been widely used in many companies.

7.1 Hadoop in Yahoo!

Yahoo! is the leader in Hadoop technology research and applications. It applies Hadoop on various products, which include the data analysis, content optimization, anti-span email system, and advertising optimization. Hadoop has also been fully used in user interests' prediction, searching ranking, and advertising location (Yahoo official website, 2015).

In the Yahoo! home page personalization, the real-time service system will read the data from the database to the interest mapping through the Apache. Every 5 minutes, the system will rearrange the contents based on Hadoop cluster and update the contents every 7 minutes.

Concerning span emails, Yahoo! uses the Hadoop cluster to score the emails. Every couple of hours, the Yahoo! will improve the anti-spam email model in the Hadoop clusters and the clusters will push 5 billion times of emails' delivery every day (Baldeschwieler, 2010).

At present, the largest application of the Hadoop is the Search Webmap of Yahoo!. It has been run on more than 10 000 Linux cluster machines.

7.2 Hadoop in Facebook

It is known that Facebook is the largest social network in the world. From 2004 to 2009, Facebook has over 800 million active users. The data created everyday is huge. This means that Facebook is facing the problem with big data processing which contains content maintainance, photos sharing, comments, and users access histories. These data are not easy to process so Facebook has adopted the Hadoop and Hbase to handle it (Joydeep, 2008).

7.2.1 Why Facebook has chosen Hadoop

As Facebook is developing, it discovered that MySQL cannot meet all its requirements. After long-term research and experiments, Facebook finally chose Hadoop and Hbase as the data processing system. The reason why Facebook chose the Hadoop and Hbase has the two aspects(Klint,2011). On the one hand, Hbase meets the requirements of Facebook. Hbase can support the rapid access to the data. Although Hbase does not support the traditional outer form operations, the Hbase column oriented storage model brings high flexibility search in the inner form. Hbase is also a good choice for intensive data. It is able to maintain huge data, support the complex index with the flexible scalability and guarantee the speed of data access. On the other hand, Facebook has the confidence to solve the Hadoop problems in real use. For now, Hbase has already been able to provide high consistency and high throughput key-value storage but the Namenode as the only manager node in the HDFS may become the bottleneck of the system. Then, Facebook has designed a high availability Namenode, called AvatarNode to solve this problem. In the aspect of the fault tolerance, HDFS can tolerate and isolate faults in the subsystem of the disk. The failures of the whole clusters of Hbase and HDFS are part of fault tolerance system.

Overall, according to the improvements by the Facebook, Hadoop can meet the Facebook most requirements and can provide a stable, efficient, and safe service for the Facebook users.

8 CONCLUSION

The thesis has given a brief introduction to the core thechnology of Hadoop but there are still many applications and projects developed on Hadoop. In conclusion, the Hadoop, which is based on the Hadoop HDFS and MapReduce has provided a distributed data processing platform. The high fault tolerance and high scalability allow its users to apply Hadoop on cheap hardware. The MapReduce distributed programming mode allows the users to develop their own applications without the users having to know the bottom layer of the MapReduce. Because of the advantages of Hadoop, the users can easily manage the computer resources and build their own distributed data processing platform.

Above all, it is obvious to notice the convenience that the Hadoop has brought in Big Data processing. It also should be pointed out that since Google published the first paper on the distributed file system till now, the history of Hadoop is only 10-year old. With the advancement of the computer science and the Internet technology, Hadoop has rapidly solved key problems and been widely used in real life. In spite of this, there are still some problems in facing the rapid changes and the ever increasing demand of analysis. To solve these problems, Internet companies, such as Google also introduced the newer technologies. It is predictable that with the key problems being solved, Big Data processing based on Hadoop will have a wider application prospect.

REFERENCES

Apache Hadoop Org. (2013). *HDFS architecture guide*, available at http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html#Introduction [Accessed: 15 May 2015].

Baldeschsieler, E. (2010), *How Yahoo Spawned Hadoop, the Future of Big Data*. Available at http://www.wired.com/2011/10/how-yahoo-spawned-hadoop/ [Accessed: 21 May 2015].

Bloor, B. (2003). *The failure of relational database, the rise of object technology and the need for the hybrid database.* Arlington: Baroudi Bloor International Inc.

Borthakur, D. (2007). *The hadoop distributed file system: Architecture and design.* Hadoop Project Research. Apache Orgnization.

Borthakur, D. (2008). *HDFS architecture guide*. Available at: http://hadoop.apache.org/common/docs/current/hdfs design.pdf. [Accessed: 14 May 2015].

Boulon, J., Konwinski, A., Qi, R., Rabkin, A., Yang, E., and Yang, M. (2008). Chukwa, a large-scale monitoring system. Company Report. Yahoo!, inc.

Dean, J. and Ghemawat, S. (2008). *MapReduce: simplified data processing on large clusters*. Communications of the ACM, 51(1), 113.

Enslow, P. (1978). 'What is a "Distributed" Data Processing System?' Computer, 11(1), pp.13-21.

Gang, L. (2014). *Applications and development of Hadoop*. Beijing: Zhangtu Information Technology, Inc.

Geczy, P. (2014). *Big Data Characteristics*. Bachelor. National Institute of Advanced Industrial Science and Technology (AIST), Japan.

George, L. (2011). *HBase: the definitive guide.* Sebastopol, CA: O'Reilly.

Google Developers, (2015). *Google Cloud Computing, Hosting Services & Cloud Support.* [Online] Available at: https://cloud.google.com/ [Accessed: 13 May 2015].

Hunt, P., Konar, M., Junqueira, F. P., and Reed, B. (2010). *ZooKeeper: Wait-free Coordination for Internet-scale Systems*. In USENIX Annual Technical Conference (Vol. 8, p. 9).

He, B., Fang, W., Luo, Q., Govindaraju, N. K., & Wang, T. (2008, October). *Mars: a MapReduce framework on graphics processors*. In Proceedings of the 17th international conference on Parallel architectures and compilation techniques (pp. 260-269). ACM.

Joydeeo, S. (2008) *Hadoop.* Available at https://www.facebook.com/notes/facebook-engineering/hadoop/16121578919 [Accessed: 23 May 2015].

Kawa, A. (2014) *Introduction to YARN*. Available at http://www.ibm.com/developerworks/library/bd-yarn-intro/ [Accessed: 20 May 2015].

Klint F., (2011). *Why Facebook use Apache Hadoop and Hbase.* Available at http://readwrite.com/2011/05/20/why-facebook-uses-apache-hadoo [Accessed: 25 May 2015].

Laney, D. (2001). *3D data management: Controlling data colume, velocity and variety.* [Online] Available at: http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf [Accessed: 20 May 2015].

Liu, X., Han, J., Zhong, Y., Han, C., and He, X. (2009). 'Implementing WebGIS on Hadoop: A case study of improving small file I/O performance on HDFS'. In Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on (pp. 1-8). IEEE.

Lu J. (2012). *Hadoop in action*. Beijing: Hua Zhang Tu Wen Inc.

Liu, J. (2013) *The Big Data Processing*. Beijing: People Posts& Telecom Press Publisher.

Ma, Z. and Lin G. (2010).*The Limitation of MapReduce: A Probing Case and a Lightweight Solution*. Bachelor's thesis. The Hong Kong University of Science and Technology.

Ohlhorst, F. (2013). *Big data analytics*. Hoboken, N.J.: John Wiley & Sons.

Ricky, H. (2008). *Hadoop Map/Reduce implementation*, available at http://horicky.blogspot.fi/2008/11/hadoop-mapreduce-implementation.html [Accessed: 20 May 2015].

Rogell, E. (2015). '532 Million Status Updates, 864,000 Hours of Video: A Typical Day on the Internet (Infographic) | TechSpy'. [Online] Available at: http://techspy.com/news/956851/532-million-status-updates-864-000-hours-of-video-a-typical-day-on-the-internet-infographic [Accessed: 10 May 2015].

Snijders, C., Matzat, U., & Reips, U. D. (2012). 'Big data: Big gaps of knowledge in the field of internet science'. *International Journal of Internet Science*, 7(1), 1-5.

Thusoo, A., Sarma, J. S., Jain, N., Shao, Z., Chakka, P., Anthony, S., and Murthy, R. (2009). *Hive: a warehousing solution over a map-reduce framework. Proceedings of the VLDB Endowment*, 2(2), 1626-1629.

White, T. (2009). *Hadoop: the definitive guide: the definitive guide*. Sebastopol, CA: O'Reilly

WikiDifference, *Difference between Hadoop and RDBMS*. Available at http://www.wikidifference.com/difference-between-hadoop-and-rdbms/ [Accessed: 20 May 2015].

Yahoo! official website. *Hadoop at Yahoo!.* Available at https://developer.yahoo.com/hadoop/ [Accessed: 22 May 2015].

Yang, G. (2011). The application of MapReduce in the cloud computing.'In Intelligence Information Processing and Trusted Computing (IPTC). 2011 2nd International Symposium on. Hubei, RPC, 22-23 Oct. 2011. IEEE