Research on Reliability of Hadoop Distributed File System

Abstract

With the development of cloud computing, more and more enterprises begin to develop their own cloud storage architecture by Hadoop. HDFS, at the bottom of Hadoop framework, stores files on all the nodes in cluster. This article first analyzed the HDFS architecture and put forward a problem single point of failure: There is only one NameNode in HDFS, which has limited the number of metadata storage and is prone to a failure of single node. Once the NameNode fault occurs, the system will break down. It affects the reliability of the cluster directly. Then it introduced Paxos algorithm and improved the structure of HDFS according to its principle. It solved the problem by increasing the number of NameNode and ensuring the synchronization of them by Paxos algorithm. At last, it verified the feasibility of the scheme by an experiment. A failure of metadata server will not interrupt the system so the reliability is improved.

Keywords: *cloud computing, Paxos algorithm, failure of single node, HDFS, reliability*

1. Introduction

In the 21st century, the development of network information has witnessed an astonishing growth. It has an enormous influence on our production, life and learning. According to the statistics of a world famous institution, at the end of 2012, the world's Internet users reached 2.65 billion [1]. With the rapid increase of information, high processing speed is in demand and great changes have taken place in information processing ability." Cloud computing" is a kind of emerging commercial calculation model. It provides a variety of dynamic, elastic and virtual computing resources to users in the form of service. In this way, users needn't understand the internal structure of "cloud" but can get great computing power in "cloud". Cloud computing has caused wide public concern and research in society and the traditional information service is transforming to cloud computing.

Hadoop is an open source distributed computing platform developed by Apache software foundation. It is the most representative product of research and application of cloud computing. It has gotten a good development in the field of Internet since its inception. Hadoop is a distributed software framework of densely data processing and data analysis based on java [2-3].Users can make full use of the enormous resources in Hadoop cluster to develop distributed application programs not knowing the underlying details of the distributed system. On account of the features such as measurement,

extensibility, low cost, high efficiency and reliability, the applications based on Hadoop have flourished everywhere [4]. However, the failure of single node in Hadoop Distributed File System (HDFS) brings inconvenience to users. In order to provide high-quality, more convenient services for users, the research of reliability is very important. To solve the failure caused by the single node, this paper builds a cluster including several metadata servers and ensures the synchronization of them by Paxos algorithm, which greatly improves the reliability of the system.

2. HDFS Architecture and Problems of It

Hadoop including HDFS and MapReduce provides a distributed infrastructure that the underlying details are transparent. The distributed file system, a current research hotspot, is one of the core technologies of cloud computing platform [5]. The advantages of HDFS such as high fault tolerance and high scalability allow users to deploy Hadoop on cheap hardware and form a distributed file system. MapReduce realizes distributed computing and task processing in the cluster. HDFS provides support for file operations and storage in the process of MapReduce processing tasks. MapReduce realizes the distribution, tracking and execution of tasks on the basis of HDFS. In short, MapReduce and HDFS complete the main tasks of Hadoop together.

2.1 HDFS Architecture

HDFS, at the bottom of Hadoop framework, storages files on all the nodes in the cluster. It can achieve mass data management benefited from high transmission, high fault tolerance and data access in the form of stream [6]. Error detection and rapid automatic recovery are the core design goals of HDFS.HDFS can store and process mass data due to good expansion ability. Furthermore, it can be used on cheap business machines, which has greatly reduced the cost of development. HDFS can process data concurrently, maintain copies of data and rearrange computing tasks after the failure automatically [7]. The main characteristics of HDFS are as follows:

(1) It has good fault-tolerant ability. Large files are divided into multiple data chunks stored with its backups in the cluster. It ensures the integrity of data and the failure of a single node will not crash the cluster.

(2) Large files have high storage efficiency because datasets are stored separately in many nodes.

(3) Files are written once and read many times and data will be added to the end of the file. Because the file won't be changed after creation, so the consistency of data is ensured.

(4) The expansion capability of HDFS is very strong. It can dynamically add or delete nodes without affecting the stability of the cluster.

(5) Data is transmitted in the form of small packets between nodes, which improves the throughput of data access and reduces the network congestion caused by large amounts of data transmission.

(6) The platform portability is good. It is developed by Java language, so it can be

deployed on both Linux and Windows operating system.

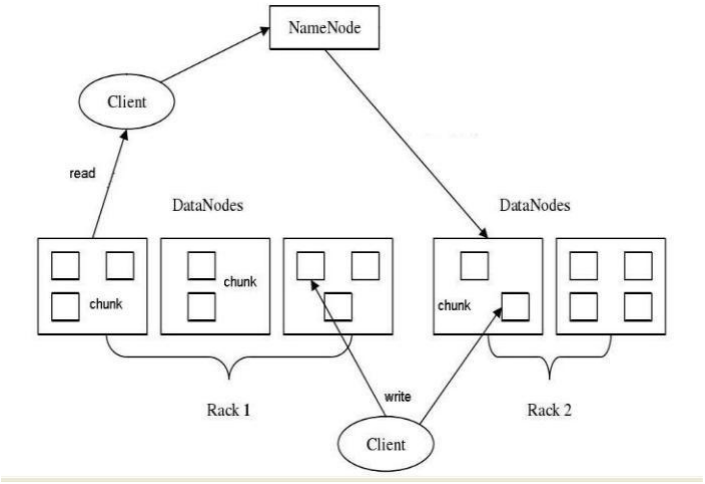

Figure 1. HDFS Architecture

Files in HDFS are divided into multiple chunks. As independent storage units, these chunks are stored in a set of data servers (DataNode) while metadata is stored in metadata servers (NameNode) [8], as is shown in Figure 1. There are two types of nodes in HDFS cluster: a NameNode and multiple DataNodes. It's a kind of master-slave architecture [9].

2.1.1. NameNode

The NameNode is a central server. Its main function is to manage metadata information of user files in HDFS. Metadata information includes the directory tree information of the file system, the corresponding relation between files and data blocks and the location information of blocks' distribution on DataNodes. In addition, the NameNode is also responsible for the management of data blocks. In order to guarantee the reliability of metadata, the metadata information will also be saved to hard disks in the form of image file and edit log. The NameNode needs to deal with requests about metadata operations from clients and DataNodes. The NameNode and DataNode confirm each other's state by heartbeat mechanism [10-11].

2.1.2. DataNode

The DataNode stores the small divided files to local file system. The client gets location information of the data blocks from the NameNode and has interactive communications with the DataNode to read or write data. Considering reliability the DataNode need to report heartbeat information and data block status to the NameNode regularly in order to transfer the data block on fault nodes [12].

2.2. Failure of Single Node

HDFS cluster consists of a single NameNode, multiple DataNodes and multiple clients. Data in a DataNode has backups in other DataNodes, so the data of the whole system will not be lost when one DataNode goes down. HDFS provides only one NameNode to store

namespaces of the entire file system, which limits the number of metadata, especially the number of files [12]. Although a single NameNode server is easy to implement, simple to maintain, the performance of a single NameNode causes the system bottleneck and failure of single node. The NameNode plays a key role in HDFS. Once it fails, the entire system will stop working and bring huge losses. The reliability of NameNode affects the performance of the whole cluster, so it's very important to improve the reliability of the NameNode.

2.2.1. SecondaryNameNode

There's a SecondaryNameNode to update images and logs of the NameNode regularly in version Hadoop - 0.20 x. In this mechanism, the SecondaryNameNode periodically downloads images and log files from the NameNode and merges them into one new image. The NameNode will load the latest image when it restarts. This is a CheckPoint as is shown in Figure 2. In this mechanism, it only saves the metadata before the last CheckPoint and data after the CheckPoint will be lost if the NameNode fails. As a result, it's not an ideal method to solve the problem above [13].

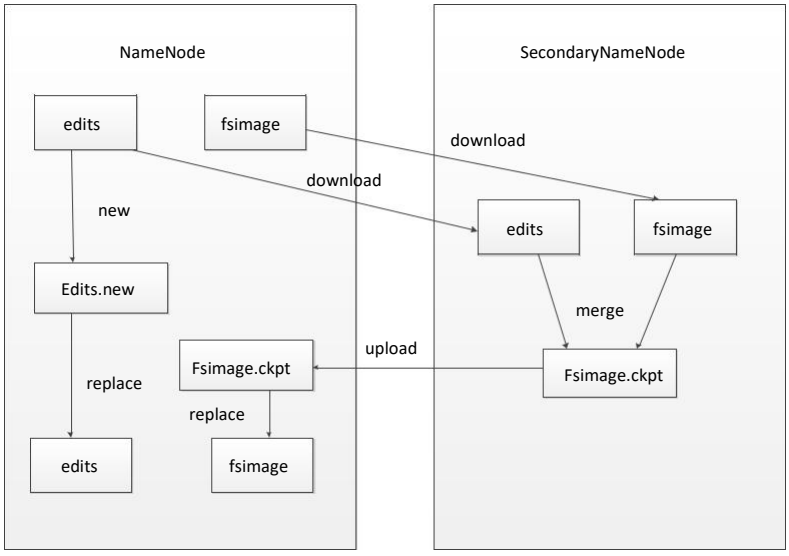

Figure 2. CheckPoint

2.2.2. Avatar

FaceBook put forward Avatar mechanism to solve the problem of NameNode. Avatar mechanism provides a StandbyNameNode as the hot backup node to back-up all the data in PrimaryNameNode. In addition, another node NFS (Network File System) is used to store the logs of the PrimaryNameNode. StandbyNameNode continuously reads logs from NFS to keep synchronization of namespaces and does CheckPoint as well. The DataNode interacts with both PrimaryNameNode and StandbyNameNode and sends heartbeat information and data information to them at the same time, as is shown in Figure 3. When the PrimaryNameNode is down, the StandbyNameNode can replace it immediately.

Virtual IP is introduced into Avatar mechanism but it isn't stable. The NFS server is a new single node and concurrent access is not ensured [14].

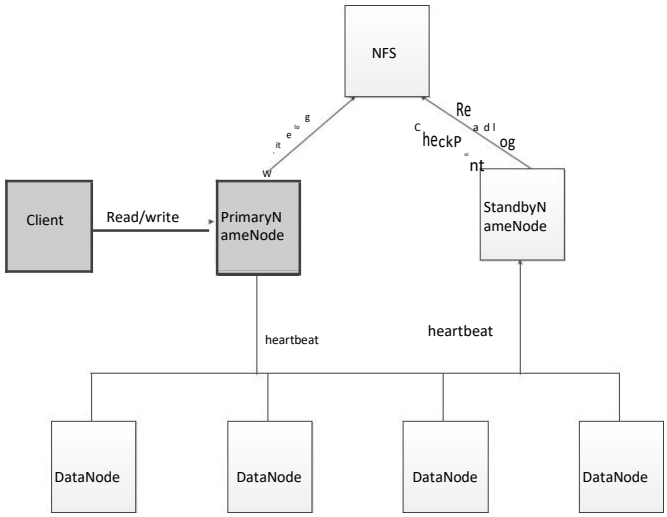

Figure 3. Avatar Architecture

The existing solutions of NameNode still have defects: Lost of data, long switching time and manual switching. In a word, the NameNode is still single node and is likely to cause system failure.

3. Improvement of HDFS Based on Paxos Algorithm

The NameNode is an important part of HDFS. This part proposes a new architecture of HDFS. It solves the problem of single node by adding multiple metadata servers into the original system. As a result, when a metadata server fails, another can replace it to work. The system will continue working and the data will not be lost [12].It uses paxos algorithm to achieve consistency of the data in different NameNodes.

3.1. Paxos Algorithm

Paxos algorithm is a consistency algorithm to solve node failure in distributed system. It was put forward by Lesile Lamport. In recent years it's widely used in distributed computing. For example, Google's Chubby and Apache's ZooKeeper are implemented based on its theory. There's a premise in Paxos: no Byzantine General. That is to say, Paxos can only be used in a credible computing environment and the environment is not damaged by invasion [10].A typical scenario is that in a distributed database system, if the initial state of each node has no difference, they get the same state after executing the same command sequences. Each command needs to execute a "consistency algorithm" in order to ensure the consistency [15]. There are two kinds of models of node communication: memory sharing and message passing. Paxos algorithm is based on message passing model.

Suppose a collection contains a lot of processes, and they are able to put forward

In

proposals. Paxos algorithm can ensure that in these proposals only one can be selected. The safety requirements of consistency are as follows:

1) In all the proposals only one can be selected.

2) An approver can but approve one proposal and only the proposals approved by the majority come into force.

3) A proposal can be learned only after it is selected [12].

There are two phases to pass a proposal:

A. Preparing

1) The proposer chooses a number n and sends preparing request to the majority of the approvers;

2) After receiving preparing message, the approver replies to the proposer with the last approval if the proposal number is greater than that of the messages it answered. And the approver promises that it will no longer approve the proposal whose number is smaller than n.

B. Approval

1) When a proposer has received the reply of the majority of the approvers, it's the approval stage. It sends approval request including the number n and value to the approvers that gave replies.

2) The approver approves the request if it doesn't go against the promise to other proposers.

3.2. Improvement of HDFS Architecture

In the architecture of multiple metadata servers, each server is written separately so the metadata needs to be copied to ensure high availability and reliability. However, it's difficult to ensure the consistency of metedata. The NameNode, as the central server, is responsible for the management of the namespaces of file system and client's access to files. The metadata servers in cluster communicate with each other by Paxos algorithm. The algorithm solves the problem of data consistency in distributed computing. The cluster of metadata servers is composed of multiple NameNodes where one is Master, others are Slave [16]. In order to reduce the load of Master, Slave can reply to the client directly without informing Maser when the client sends a read and query request, as is shown in Figure 4.

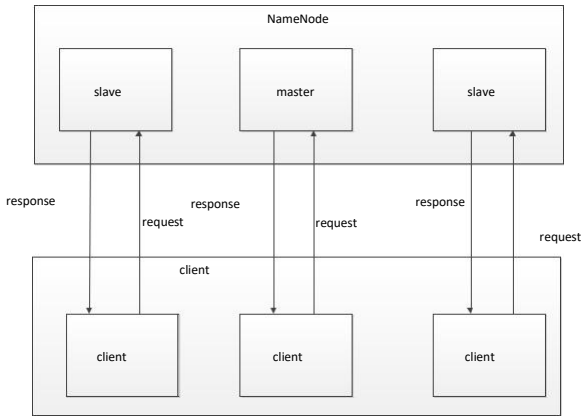

Figure 4. Reading

The client must interact with Master when performing write operation [14]. The Slaves perform the same operations after they receive orders from Master to keep synchronized with Master, as is shown in Figure 5. The steps of write operation of the client are as follows:

1) The client sends write operation request to the Master/Slave.

2) If it is a slave who receives the request, it sends the request to the Master.

3) The Master sends a write operation proposal to all the Slaves.

4) The Slaves received the proposal respond to it.

5) The Master sends write operation orders to all the Slaves when N / 2 + 1 Slaves accept the proposal.

6) The Slaves respond to the Master when they finish the operation.

7) The Master responds to the client after it receives the replies from all the Slaves.

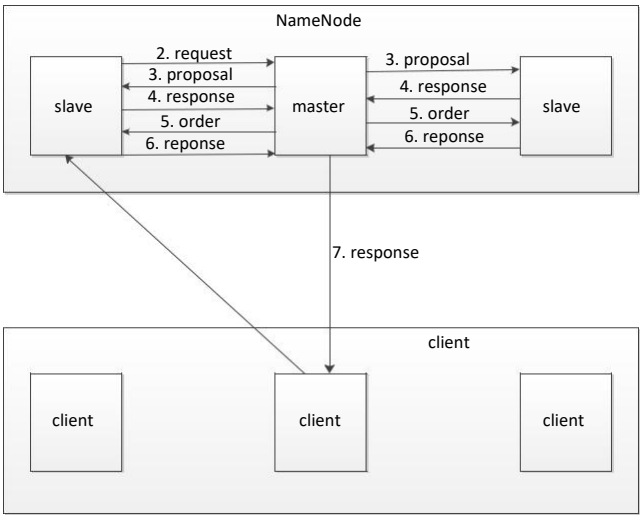

Figure 5. Writing

The Slaves can accept the requests from the client and participate in the confirmation of the proposal. The Master can also accept the requests from the client and deal with the

operations overall. The Master and Slaves may switch to each other when the Master is down [17]. At this time, the Slaves in the cluster can elect a Slave as the new Master. And the failed Master will join the cluster as a Slave. In a cluster, there is only one Master and multiple Slaves except the election process.

4. Experiments and Analysis

4.1. Experiment

There are four servers as DataNodes, three servers as NameNodes (one Master and two Slaves), three servers as clients in the test system. The structure diagram of the system is as follows:

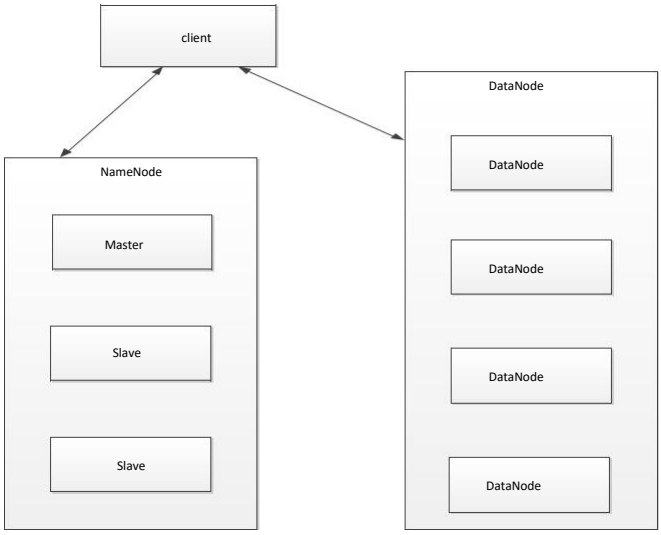

Figure 6. Architecture of Test System

The test system is set up by version Hadoop - 0.21.0. The configurations of hardware and software are shown in Table 1.

Table1. Configurations of Test System

Server	CPU	RAM	Network card	OS
Client	Intel(R)Xeon(R) 2.0GHz	1GB	Gigabit ethernet card	Redhat Enterprise Linux 4
NameNode	Intel(R)Xeon(R) 2.4GHz	2GB	Gigabit ethernet card	Redhat Enterprise Linux 4
DataNode	Intel(R)Xeon(R) 2.0GHz	512MB	Gigabit ethernet card	Redhat Enterprise Linux 4

Because (N - 1) / 2 failures are allowed in the test system, one of the three NameNodes can break down. Now the following cases will be tested:

1) State A: A Slave fails and then goes well.

2) State B: The Master fails and a new one is selected from the two Slaves.

3) State C: Two of the three servers break down.

4) State D: All the servers work properly. Figure 7 is the results made by TestDFSIO.

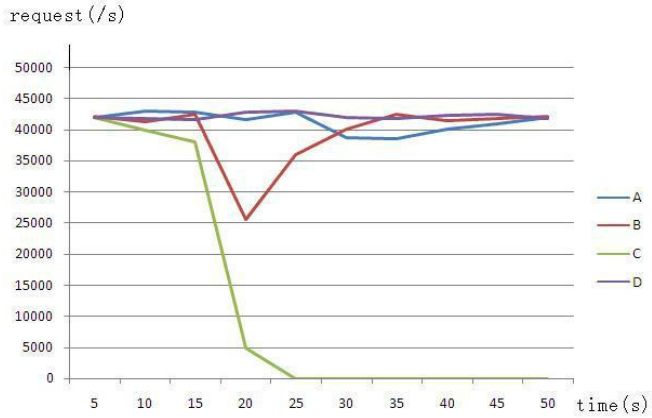

Figure 7. Test Results

4.2 Analysis of Reliability

The analysis of test results is as follows:

1) State A: There are still two servers (not less than N / 2 + 1) in the system, so the system will continue running. However, the recovery of the Slave need synchronize with the Master, so it will influence the performance of the system.

2) State B: When the Master fails, a new Master need be selected to replace the original one by electoral system. During this period, the system can't work. After the new Master begins to work, the system will recover.

3) State C: At this time because only one (less than N / 2 + 1) server in the system can work properly, the system will break down. It proved that in a system of N servers, it can run properly only if more than N / 2 + 1 servers can work.

4) State D: All the servers are in good condition and the system is running properly.

Usually, we use MTTF(Mean Time to Failures) to evaluate the system reliability, MTTR(Mean Time to Repair) to evaluate the system maintainability, MTTF and MTTR to evaluate the system availability. In the cluster of metadata servers, the architecture based on Paxos algorithm allows some (less than a half) servers to break down. Assume that the failure and recovery of probability of metadata servers in the cluster obey negative exponential distribution, the system failure rate λ, repair rate ξ, availability of single node Aare expressed as follows:

$\lambda=1/M_{MTTF}$ (1)$\xi=1/M_{MTTR}$(2) $A=\xi/(\xi+\lambda)$(3)In the test system, P_A, P_B, P_C, P_D are the steady probability of State A, State B, State C and State D. The reliability of the system is: $R=P_A+P_B+P_D$= C(2,1)*A*A*(1-A)+ A*A*(1-A)+C(3,3)*A*A*A(4) MTTF of the

nodes: M_{MTTF}>=2000h, failure rate: $\lambda=1/M_{MTTF}$ =0.0005;

MTTR of the nodes: M_{MTTR} =1h, repair rate: $\xi=1/M_{MTTR}$ =1

To substitute them into form(4), we can get the reliability: R =99.9995%. It shows that the scheme has high reliability.

5. Conclusion

This paper introduces Hadoop Distributed File System in detail and optimizes the architecture of it based on Paxos algorithm. As a result, it solves the problem of single node. When there are N metadata servers in the system, it will continue working as long as N / 2 + 1 metadata servers are in good condition. That is to say, the system can tolerate (N - 1) / 2 servers' failure. However, all the requests of write operation are sent to the Master, so it increases the communication overhead and reduces the processing speed. Furthermore, it takes time to select a new Master when it breaks down and the system can't work during this time. These problems need more researches and settlements in the future.

