Journal of Theoretical and Applied Information Technology
31st March 2013. Vol. 49 No.3
© 2005 - 2013 JATIT & LLS. All rights reserved·

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

CLOUD DATA MANAGEMENT: SYSTEM INSTANCES AND CURRENT RESEARCH

[1]CHEN ZHIKUN, [1]YANG SHUQIANG, [1]ZHAO HUI, [2]ZHANG GE, [2]YANG HUIYU, [1]TAN SHUANG, [1]HE LI

[1]Department of Computer Science, National University of Defense Technology, Changsha China

[2]Senior Engineer, Beijing Aeronautics Engineering Technology Research Center, Beijing China

E-mail: [1]zkchen@nudt.edu.cn

ABSTRACT

With the development of Internet technology and cloud computing, cloud data management technology has emerged while the technology of traditional database management cannot meet the requirement. In this paper, we will compare the features and analyze the difference among the new cloud data management systems from data model, data partition schema, fault-tolerant mechanism of system, the load balancing mechanism of system and the model of data consistency and availability. And then we analyze and verified the system performance between the two open-source systems. Finally, research statuses of the cloud data management technology are analyzed and the key technologies in the research of cloud data management technology are summarized.

Keywords: *Cloud Computing, Cloud Data Management, NoSQL, Massive Data Storage, Big Data*

1. INTRODUCTION

With the development of information technologies, the amount of data generated by the enterprises or companies growth quickly, and the data size will reach TB or even PB. And how to manage and analyze the massive data is a large challenge for many fields, such as medical, communication and the Internet etc. For example, the data increased by dozens of GB every day in 2007, but with the increase of participants people the data is increased by 2TB every day in 2008. There are some reporter reports that the data is increased by 20TB every day now. As the concept of Cloud Computing [1], the researcher proposes the concept of cloud data management to solve the massive data management.

The definition of cloud data management is the storage management technology for massive data in the cloud environment. So it has to store and organizes the data in reasonable, and it also has to ensure the data is highly fault-tolerant and able to response the queries quickly. There are a lot of researchers and many enterprises did some works in cloud data management and they also developed some system. But there are some researchers have proposed some doubt in the research of cloud data management. They thought that there are not any new substance contents and the research has not any meaning, it is just a new bottle fill old wine. In VLDB2010 Divyakant Agrawal [2] has responded to those questions. Cloud data management technology is proposed under the cloud computer, and it has to base on cloud computer. So in the research of cloud data management will face a lot of challenges which cannot predict in traditional data management. In the industry such as Google, Yahoo! and Facebook have done many research and development, and they all have developed some cloud data management system. And some of them have been used in the actual application. So the research of cloud data management will be an important research branch in the field of Cloud Computer.

In this paper we will analyze the cloud data management systems which are wide used in the actual application and we also compare the differences of those systems, and we also do some investigation and analyze of cloud data management in the academic research.

The rest of paper is structured as follows. We will analyze some cloud data management systems which are wide used in actual application in section 2. In section 3 we will compare those systems and summarize the characteristic of them. Then we will compare the performance of two open source system with some experiments in section 4. In section 5 will analyze the relate research of cloud data management in academic research. Finally, conclusion appears in section 6.

2. THE ANALYZE OF CLOUD DATA MANAGEMENT INSTANCE

The traditional database technology seems weak in massive data store management, therefore industry and academic have research a lot on it and develop some management system for actual application. The main system is: BigTable [3] of Google, HBase [4] of Apache, PNUTS [5] of Yahoo!, Dynamo [6] of Amazon, Cassandra [7][8] of Apache and so on. In this section we will show detailed analysis on those cloud data management system from data model, data partition schema, load balancing strategy, fault-tolerant strategy and data consistent model.

2.1 BigTable

Most of application program of Google need deal with massive structure and half structure data, but the traditional database technology can’t satisfied demand of store and process. BigTable is developed by this drive, and it is the large-scale data management system which has weaker consistency, and the data store by multi-dimensions sequence table. The architecture of servers is Master-Slave. It uses distribute coordinate service-Chubby [9] to implement the fault-tolerant management. In this architecture, the service of store (in GFS [10]) and management are separated, thereby simplify the difficulty of the management, and it is easy to maintenance and man-made controlled. But BigTable is only deployment in cluster, because the underlying storage is based on the distributed file system.

BigTable is a scattered, multi-dimensions and sequential sparse table. The data model of BigTable is shown in Figure 1. The data model includes row, column and timestamp. We can ensure an only value of key with row, column and timestamp. And the value in the table is an unexplained byte array. In BigTable there are some new concepts such as row key, column family and qualifier. The column family is made up by a sequence qualifier with the same properties; column is only defined by prefix of column family and postfix of qualifier, such as column family: qualifier; and row key is the only key that can identification a row in the table; column family and qualifier could amend according to the system requirement. Figure 1 is an example of webtable, the website with inverted order is the row key; Contents, Anchor and mime are the column family. There is not any qualifier of contents and mime, so the column family can represent the column’s name directly. There are two qualifiers for Anchor: cnnsi.com and my.look.ca. So when represent a column we should use column family and qualifier, and divide with colon. Finally, there is new concept of timestamp, which represents the version number of data. Because there is only insert operation but not update in BigTable. Then we have to use the timestamp to represent the version number.

Row Key	Time Stamp	Contents	Anchor		Mime
			cnnsi.com	my.look.ca	
com.cnn.www	T9		CNN		
	T8			CNN.COM	
	T6	<html>……			Text/html
	T5	<html>……			
	T3	<html>……			

Figure 1: Data Model of BigTable

The architecture of BigTable is shown in Figure 2, and the architecture of servers is Master-Slave. The data partition schema of BigTable is range partition of the row key. BigTable partitions a certain range rows of data to a small table which is known as tablet. And the system will allocate the tablet to a server which is called tablet server. Master server will monitor the status of tablet at any time, and Master server is responsible for the remote management and load allocate of tablet server, at last it has to respond the request of Client. BigTable system rely on the distribute task scheduler of underlying cluster and the distribute coordinate service-Chubby. BigTable use Chubby to store the pointer of ROOT table, so the user can get the concrete position of Root table by Chubby lock servers. And then we can get the concrete position of META table from the ROOT table. After that we can get the concrete position of tablet server. Finally we can get the concrete data from the tablet server. At the same time, BigTable can get the active status of every tablet server, so we can detect if there are some nodes failure. The data recovery of failure node is partition into two parts: one is the data that have been durable store in the GFS, it can use the replication strategy to implement fault-tolerant; the other one is the data that store in the memory, which have to redo the log to implement fault-tolerant.

2.2 HBase

HBase is a high-reliability, high performance, column-oriented, scalable, distributed storage system, it is a sub-project of Hadoop [11] in Apache. And its full name is Hadoop Database. The idea of HBase comes from BigTable of Google, and it is the open source implement of BigTable.

The data model of HBase is similar to BigTable, and we will not introduce the data model of HBase. GFS is used to the underlying storage system of data files in BigTable, which HBase uses HDFS

[12] of Hadoop or S3 [13] or EBS [14] of Amazon as the underlying storage system. The Master-Slave architecture is used in the services architecture of HBase, and it used the Zookeeper [15] to implement collaboration service instead of chubby service which is used in BigTable. The architecture of HBase is shown in Figure 3. The data partition schema of HBase is range partition of row key which is used in BigTable too. HBase partitions a certain range rows of data to a small table which is known as Region, and the server which is maintain the information of Region is called RegionServer. The different column families of the same Region is stored in different files of HDFS in HBase, so the HBase is column-oriented storage system. The data access of HBase is depended on the lock services of Zookeeper, and the process of query in HBase is similar to the process of BigTable. When some RegionServer is down the HMaster can found it because every RegionServer will report its active situation to HMaster in a regular interval time. Then the HMaster will get the Regions which are service by the down RegionServer from Zookeeper and the HMaster will redistribute those Regions to the active RegionServers in the system. Finally HMaster will check if there has undone HLog in every Region in the down RegionServer, and then every region will redo the HLog to ensure the complement of data. HBase also provide a Shell query language which is similar to SQL. And user can do some simple operations (such as insert, delete, query etc.) based on row key use this interface.

2.3 PNUTS

PNUTS is a massive parallel data management system which is deployment across data center in Yahoo!. And it is used in some filed of Yahoo! such as user database, social applications, content Meta data, list management and session data etc. The PNUTS center is composed of multiple cross-domain data center. And every data center will use centralized manage architecture which is similar to the architecture of BigTable. The data partition schema of PNUTS can support range partition and Hash partition. And it uses some optimize methods to ensure users have low-latency access service and improve the performance of bulk insert. A table of PNUTS can support tens of thousands to hundreds of millions of records, and the increasing of data record will not affect the performance of the system, so the system has high scalable character. It uses multi-levels (data, Meta data and service component) redundancy measures of data to ensure the high fault-tolerant and availability. The system use asynchronous replication strategy to update the data copies and it can ensure the system has high performance, but it sacrifices the consistency of data. PNUTS provides a simple relation data model, and the data stores in the table as tuples. It can support the standardize data type (such as integer, string, Boolean and so on), but it also support the type of Blob. The tuple structure of PNUTS is flexible. Every property of a tuple does not need a corresponding value, and you can add a new property as you need, but it is not affect the performance of the query or update operation. For the table query in PNUTS, we also need to specify the primary key. The most characteristic of PNUTS is the consistency model of data. The consistency model is between the generic transaction serialization and eventual consistency. It provides a tuple level time consistency which all nodes with the same tuple must be performed in the same order in all update operations.

PNUTS will use range partition of hash partition to split a table, and it will produce many sub tables which are called tablets. All tablets will be decentralized to many servers, and every server will store hundreds to thousands tablets. The allocation strategy of tablet is flexible too, we can migrate the heavier load server ’ s tablet to the lighter load server. So the load of the system will be relative balance. When a server is down, then we can evenly allocate its tablets to the active servers. The architecture of PNUTS is shown in Figure 4. The system is distribution deployed by the domain, but it is not necessarily located in the different geographical areas. As shown in Figure 4 we can know that every domain contains the components of the system and the all data of every table. The storage unit is responsible for storing the tablets of data. Tablet controller is responsible for deciding when to split and move the tablet, and it also store the map table of the data with tablets. Router unit is responsible for storing the partial cache of map information which is map table of data with tablets, and it will get the latest map information from the tablet controller in a regular interval time. So the system will do the following steps to complete a data operation. First, we have to get the correspond tablet of the data from the router unit. If we cannot find the map information from router unit or there is not record range of this data, we will do the second step. Second, the router unit will send a message to tablet controller to get correspond map information. At last, when we get the map information of data with the tablet we will operate the store unit. YMB (Yahoo! Message Broker) is topic-based messages publish/subscribe system, which can record the log of the system and it also provide a reliable message channel to redo log. The asynchronous replication of data copies and the fault recovery of PNUTS are implemented by YMB.

PNUTS provides a simple relation data model, and the data stores in the table as tuples. It can support the standardize data type (such as integer, string, Boolean and so on), but it also support the type of Blob. The tuple structure of PNUTS is flexible. Every property of a tuple does not need a corresponding value, and you can add a new property as you need, but it is not affect the performance of the query or update operation. For the table query in PNUTS, we also need to specify the primary key. The most characteristic of PNUTS is the consistency model of data. The consistency model is between the generic transaction serialization and eventual consistency. It provides a tuple level time consistency which all nodes with the same tuple must be performed in the same order in all update operations.

PNUTS will use range partition of hash partition to split a table, and it will produce many sub tables which are called tablets. All tablets will be decentralized to many servers, and every server will store hundreds to thousands tablets. The allocation strategy of tablet is flexible too, we can migrate the heavier load server’ s tablet to the lighter load server. So the load of the system will be relative balance. When a server is down, then we can evenly allocate its tablets to the active servers. The architecture of PNUTS is shown in Figure 4. The system is distribution deployed by the domain, but it is not necessarily located in the different geographical areas. As shown in Figure 4 we can know that every domain contains the components of the system and the all data of every table. The storage unit is responsible for storing the tablets of data. Tablet controller is responsible for deciding when to split and move the tablet, and it also store the map table of the data with tablets. Router unit is responsible for storing the partial cache of map information which is map table of data with tablets, and it will get the latest map information from the tablet controller in a regular interval time. So the system will do the following steps to complete a data operation. First, we have to get the correspond tablet of the data from the router unit. If we cannot find the map information from router unit or there is not record range of this data, we will do the second step. Second, the router unit will send a message to tablet controller to get correspond map information. At last, when we get the map information of data with the tablet we will operate the store unit. YMB (Yahoo! Message Broker) is topic-based messages publish/subscribe system, which can record the log of the system and it also provide a reliable message channel to redo log. The asynchronous replication of data copies and the fault recovery of PNUTS are implemented by YMB.

2.4 Dynamo

Dynamo is a decentralized massive data management system which is based on distributes hash. And its consistency model of data is eventual consistency. The data in Dynamo is organized by key-value, and it is mainly store the raw data. The overlay of servers is P2P architecture. Under this architecture every node of the system can know each other and they have the ability to self-management, and there is no single point failure. So it has the characteristics of high availability, high scalability and performance.

The consistent hash algorithm (CHA) [16] is used to partition the data in Dynamo. CHA uses a hash function to partition the data base on the key value to N segments, and then connect the range of hash function end to end to form a ring. For example, there are S servers, and then we can know there are N/S data segments stored in every server. Data segments in the ring are allocate to the node in the order cycle of nodes. We can take an example such as Figure 5. The data is partitioned to 12 segments, and there are 3 servers in the cluster. So we can know every server will responsible for some range of key value. When there is some operation (read, insert or update) request, we need computer the hash value of the key first, and then the first node in the ring will be the node to complete the request. When there are new nodes insert into the cluster, and does the data need to redistribute? The data redistribute result is show in Figure 6. We can know that there is only need move 1/4 data to the new node, and we do not need to redistribute all data. So the advantages of using consistent hashing algorithm for data partitioning are as follows.

- We can locate the node through the hash value of key, so it can partition data automatic.
- The system is high scalable, it can reduce the amount of data that have been redistributed when there are node expansion.

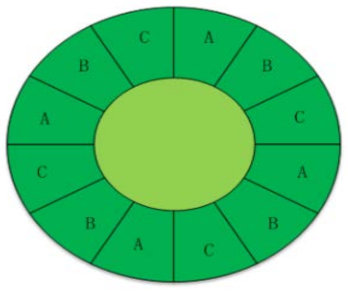


Figure 5: Data Split Method of Dynamo

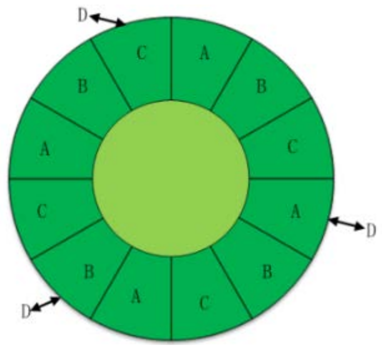


Figure 6: Insert one Node to Dynamo Cluster

Dynamo use the technology of virtual node to solve the load imbalance which is produced by node heterogeneous and the randomness access by user. The technology of virtual node split a physical node into several virtual nodes, and then maps every virtual node to the hash ring. The technology is expanding by CHA. The stronger machine will split into more virtual nodes in general. And the system will allocate the heavier load virtual node to the physical machine which has stronger performance. So the load of system will be balance as possible.

Dynamo implements the concurrent data access and high availability through redundant storage of data. The data copy of Dynamo is asynchronous replication through gossip protocol. And it implements the data consistent through Quorum algorithm in the client of users. Quorum algorithm is the core algorithm of Dynamo, which is called <N, R, W> model. The N is represent the replication number of data, R represents how many replications need to be read a least in a successful read operation, W is represents how many replications need to be write a least in a successful write operation. Finally in order to ensure the consistent of data the sum of R and W must more than N (R+W>N). So that there will some intersection between read and write, and then we can get the latest data through the version of data. We can know that this model is a tradeoff between the efficiency of read and write. If you want to get a high efficiency of read you should set a low value of R, otherwise we want to get a high efficiency of write you should set a low value of W. The parameters of this strategy are controlled by user, so it can get a high performance by the application requirement of users.

Every node in Dynamo can get other nodes active situation through gossip protocol, then to detect if there is any node down. Dynamo use different recovery strategy in different failure (temporary failure or durable failure). In the situation of temporary failure, the system will write the data to a temporary table to an available node. The data of temporary table will be written back to the destination node which is recovered from the failure. In the situation of durable failure, we can implement the data recovery through the copy of replication. When a node is failure the Synchronous of replication is implemented by Merkel tree [20]. Every node will maintain an independent Merkel tree with its key range. When there is Inconsistent of data, they will compare those two Merkel tree to implement the synchronous of data.

2.5 Cassandra

Facebook is the largest social networking platform in the world, it has tens of thousands of servers in various data centers around the world, and it need to provide services to hundreds of millions of users in the peak. The user of Facebook in growing quickly and the data is increasing rapid too, so in order to provide high quality, reliable and efficient service to users, Facebook must face up to massive data processing requirements. And Facebook needs to develop a high degree of extensibility system to solve the problem which is produce by the growing number of users and data. The exception is perfect normal in such large system; there are some failure of servers and network component too. The system must use some failure-tolerance schema to process those failure instead of process it as the system exception. In order to solve those problems, Facebook develop the Cassandra system. The system can provide a high reliable and scalable service for the social network platform. The system is contributed to the open source community-Apache in 2008.

The data model of Cassandra is similar to BigTable. It uses the concept of class family, but it does some expansion. The model adds the concept of super column to expand the model of BigTable. Every table has a primary key in Cassandra, but the primary key is a string which does not limit the size of string. The table of Cassandra is a distribution multi-dimensions chart which is indexed by primary key. The data model of Cassandra is similar to the column family model of BigTable. They will combine multi columns to form a column set which is called column family. But Cassandra expands the concept of column family; it provides two type of column family: simple column family and super column family. The simple column family is similar to the concept of column family, but the super column family is the set of column family. When we access a table we should know the following value: primary key, column family, column and timestamp. But if the table has super column family we should specify the super column family too.

The data partition schema of Cassandra is similar to Dynamo-Consistent Hash Algorithm. It can evenly distribute the data to servers to ensure the system’s load is balance to avoid the problem of hot spot. It also make the system high scalable. The nodes are heterogamous in Cassandra, so the system’s load will be imbalance. In Dynamo will use the technology of virtual node to solve this problem. But in Cassandra we will analyze the load information in the ring of system, and then move the position of node which has lighter load to rebalance the system’s load. The detail we can find in [17]. We use the data replication to ensure the data’s reliable in Cassandra, and we will use the Gossip protocol to asynchronous replication of data copies. We use the Accrual Failure Detector [18] base on Gossip protocol to implement failure detect. The detect schema does not produce a Boolean value to represent the active situation of a node; otherwise this component will produce a suspect level value to represent the failure ratio of the node. The system’s accuracy and speed are very good which are proved in the experiment, and they are also well adapted to different network and server load environment.

In short, Cassandra is a distributed storage system which is designed for massive data storage, data reliable and the query requirement based on key-value. It cannot support the complex relation query, but the query performance and response time of simple query based on key is much better than relation database. In today's popular online communities, B2B as well as B2C Web site will face the challenge of massive data and a large amount of information throughput per day. The key-value distributed storage system such as Cassandra can solve those challenges easier than relation database, and it will gradually be more widely applied in the future.

3. COMPARE THE CHARACTERISTICS OF CLOUD DATA MANAGEMENT SYSTEMS

We discussed and analyzed the detail of some more popular massive data management database in section1, and we can know they are all high scalability, high availability and high fault-tolerant. But they will have different solution when they face those problems; because of they have different application requirements. In this section we will compare those system from consistent model, data management method, data model, data partition schema, the solution of data available, load balance, failure detect and failure recovery and so on. The result of compare is shown in table 1 which is based on the result of [19].

Eric Brewer proposed the famous theory of CAP [21]. Which C represents the Consistency, A represents Availability and P represents Tolerance of network Partition. The theory point out that we cannot get all CAP in a system, we have to sacrifice one of them in a system. We can know from table1 that the popular cloud data management system are all do some sacrifice in CAP. For example, the Dynamo system improves the A and P, but it sacrifices the C. The NRW strategy is the tradeoff between C and A. Once the value of W is increased then the Consistent of data will be enhanced and the Availability of data will be less. Otherwise, once the value of W is reduced then the Consistent of data will be less and the Availability of data will be enhanced. The Consistent model of systems such as BigTable and HBase is weak consistency, but the availability is sacrifice much more than the consistent. So those systems do more work on consistency and tolerance of network partition but sacrifice the availability of system. In fact, these three characteristics are relative, they just emphasis more on a feature but do not completely abandon other features. In [3] have mentioned that the average cannot available of "some" data time is about 0.0047%. And we know that the system is high availability, but these three characteristics in comparison availability abandon some more. In [22] has summarized the location of existing popular cloud data management systems and relation database products in the CAP connection. In Figure 7 we can know the tradeoff of CAP in every product.

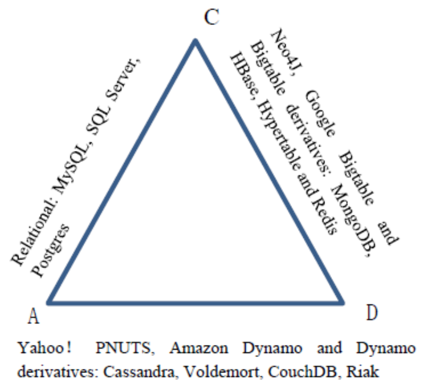

C
Relational: MySQL, SQL Server, Postgres
Neo4J, Google Bigtable and Bigtable derivatives: MongoDB, HBase, Hypertable and Redis
A
D
Yahoo! PNUTS, Amazon Dynamo and Dynamo derivatives: Cassandra, Voldemort, CouchDB, Riak

Figure 7: The position of Database System in CAP

4. EXPERIMENTS AND RESULTS ANALYSIS

Based on the analysis of the above, we will compare the performance (read, write etc.) of the system through some experiments. BigTable, PNUTS and Dynamo are not open source, so we just compare the two open source systems (HBase and Cassandra) in this section.

4.1 Experiments Environment

The experiments were constructed HBase cluster and Cassandra cluster, which are all deployed in Ubuntu system and there are 12 nodes in each cluster. And the CPU of each node is Intel dual-core 2.4G, and the RAM is 4G, the capability of each disk is 500G. In the HBase cluster, one node will be the HMaster and the other 11 nodes will all be RegionServer. Three of them will be the Zookeeper node too. The Hadoop and HBase will in the same cluster. The Cassandra system is decentralized, so we will deploy the Cassandra system in those 12 nodes.

In the experiments we will use the YCSB [23] Benchmark which is provided by Yahoo! to measure the performance of those two systems. There are some workloads which provided by YCSB to measure performance of cloud data management system. But the detail of the architecture of YCSB and the type of workload are described in [23]. In our experiments we just do workload A and workload B. The difference between workload A and workload B is just the ratio of read and write. The ratio of workload A is 1:1, but the ratio of workload B is 19:1. And we will generate 100G data insert into those two clusters before start our experiments. We will do 5,000,000 operations in every experiment. We will measure the throughput and read/write latency when change the degree of concurrency of operation.

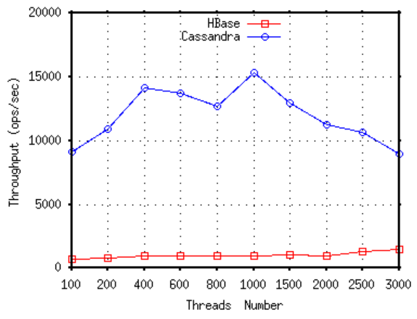
4.2 Experiment Results and Analysis

In the YCSB measures the read/write latency through change the target throughput, and it also measures the scalability of the system through change the number of servers. In this paper we will measure the performance of the system from another view. We will measure the relationship between the degree of concurrency and throughput of the system. And we also measure the read/write latency in the particular degree of concurrency. The experiment of scalability will not redo in this paper, and the detail of experiment results can see in Reference [23].

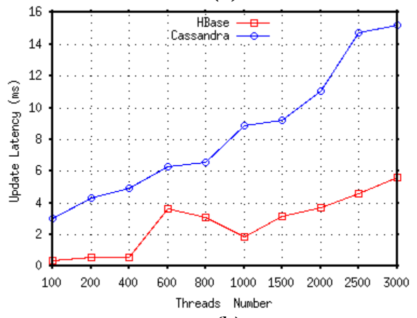
A) Workload A-Update Heavy

The ratio of read and update in workload A is 1:1 and it is called updated heavy workload. The experiment result is shown in Figure. 8. The relation between throughput and degree of concurrency is shown in Figure 8-(a). We can know that when the degree of concurrency is increasing then the throughput of both HBase and Cassandra will increase. But the variant of HBase is small and the throughput can be in a steady range. The relation between throughput and degree of concurrency in Cassandra is complicate. There are tradeoff relation between throughput and degree of concurrency. And it is not the relation that you increase the degree of concurrency then you can get a higher throughput. There is threshold in the concurrency. If the concurrency does not over this threshold, then you can get greater throughput when increases the concurrency. If the concurrency overs this threshold, then you can get lower throughput when increases the concurrency. The throughput of Cassandra is higher than HBase in workload A, because of the read latency of HBase is larger than Cassandra. The cache hit ratio of read operation in update heavy workload of HBase result in the large read latency, and it is shown in Figure 8-(c). The update performances of HBase and Cassandra are shown in Figure 8-(b). The update operation of HBase is completed in MemCache, so the update latency of HBase is lower than Cassandra. If there are massive concurrency write in HBase then the write performance of the system will reduce. Because in the process of writing required to maintain the consistent of log and the splitting of Region. But the write performance of Cassandra is very well too, because the consistency model of Cassandra is eventual consistency.

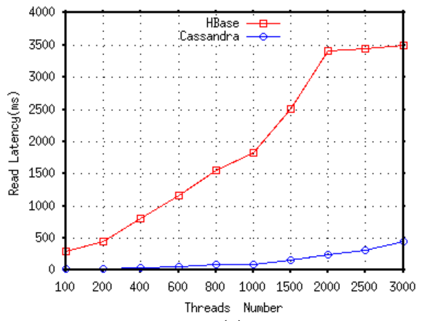
B) Workload B-Read Heavy



(a)



(b)

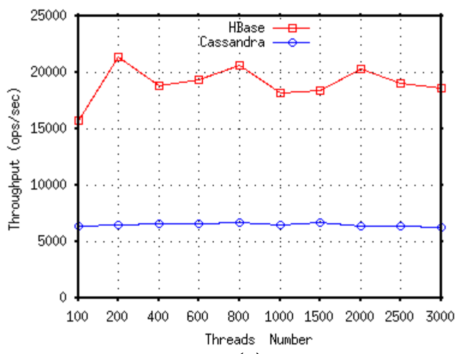


(c)

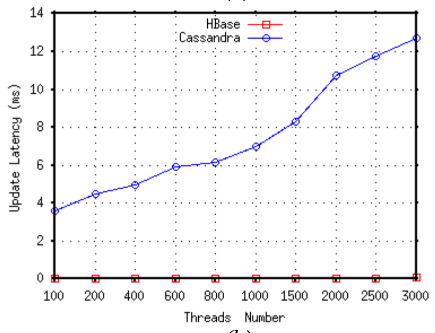
Figure 8: Workload A (Update heavy): (a) Throughput, (b) Update Latency, (c) Read Latency

The ratio of read and update in workload B is 19:1 and it is called read heavy workload. The experiment result of workload B is shown in Figure 9. We can know that the read latency of Cassandra is larger than HBase from Figure 9-(c). The reason is that Cassandra has to read more than one replication to determine the value of particular key. The read operation of HBase is only need to three times location to find the RegionServer. And the distribution of operation keys is a Zipfan distribution, and then the hit rate of the cache of RegionServer position will be very high. So the read latency of HBase will be very low. The update latency has discussed in workload A and the ratio of update in workload B is very low, so the update latency is low too, which is shown in Figure 9-(b). We can get result that the throughput of HBase is higher than Cassandra, which is concluded from the read/update latency of two systems. The relation between throughput and concurrency is shown in Figure 9-(a).

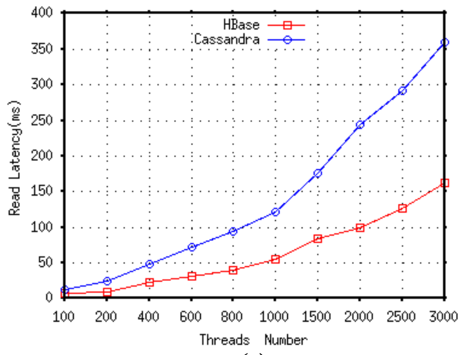
From the experiments of workload A and Workload B, we can know that the performance of HBase is better than Cassandra in the update heavy load, otherwise Cassandra will better than HBase when the workload is update heavy. So we have some conclusion from the experiments. First, HBase on support for concurrent load is better than Cassandra. Second, the read performance of HBase is better than Cassandra when the distribution of request key is Zipfan distribution. Third, if there are massive write operations then the performance of Cassandra is better than HBase. So when we select a system to actual application should base on the request of the application.



(a)



(b)



(c)

Figure 9: Workload B (Read heavy): (a) Throughput, (b) Update Latency, (c) Read Latency

5. CURRENT RESEARCH OF CLOUD DATA MANAGEMENT

With the development of cloud computing and requirements of massive data processing on large-scale application platform in the Internet, it is necessary to research a system which can reasonable store and manage the massive data. The system can respond quickly to users' requests and ensure that the system has features of high scalability, high availability and high fault-tolerant. Therefore, the researching of cloud data management is a popular research direction in the field of cloud computing. The researching and progress of cloud data management in academic will be discussed and analyzed in this section.

Google proposed BigTable data management system in 2006 whenever the cloud computing technology has not really been put forward. The system is mainly for massive distributed data storage management and the system has characteristics of high scalability, high availability and high fault-tolerant. In 2011, Google proposed Megastore [24] system which is mainly designed to face the needs of online interactive services. It can respond quickly to user ’ s requests. Megastore system is based on BigTable and draws on the scalability of the NoSQL and the convenience of the traditional RDBMS, so that the system has a strong consistency guarantees and high availability. The system also concerns on consistency (C) and availability (A). Some other large Internet companies such as the Amazon, Yahoo! also developed a corresponding system, Dynamo and PNUTS. Driven by the prototype which these companies to developed, there are some companies develop the open-source system based on the concept of these systems, such as HBase and HyperTable [25] are open source implementation of BigTable and Cassandra is an open source implementation of Dynamo.

Under the promotion of Google, Yahoo!, Amazon and other companies as well as the open source systems, domestic and foreign academic circles conducted the appropriate research on the cloud data management system and have formed their own system. Reference [26] proposed the Epic system, which is mainly used for storage and processing of data in the cloud platform. It is able to handle data-intensive OLAP and OLTP tasks. Sudipto proposed a flexible transaction data storage system in the cloud environment-ElasTras in [27]. They proposed a scalable data management system in a cloud environment to support the Multi-key transaction-GStore [28] which is based on ElasTras. A data management system mixed with MapReduce clouds and the DBMS-HadoopDB is proposed when the research team from Yale University Daniel J. Abadi is on the basis of taking into account the advantages and disadvantages of the MapReduce architecture and parallel DBMS in [29][30]. Thereby, it makes the system able to give full play to the advantages of MapReduce and the DBMS system to achieve the highest performance. Reference [31] proposed the Starfish system for the requirements for the need for timely analysis and lower cost of large data processing. The system is implemented on the basis of the Hadoop project. It is a data management system that has a rapid analysis of large data and the ability to self-adjust. The cloud data management project team of Renmin University of China which is led by Xiaofeng Meng has developed a TaiJiDB [32] which is a dual-core cloud database management system. Its architecture mixes the Master-Slave architecture of cloud storage and peer-to-peer architecture. So it can use both advantage of those two architectures. It supports the SQL language to manage the massive data in the cloud database system. Achievements of academic are not only that, there are many cloud data management systems. CouchDB [33], MongoDB [34] are the cloud data management system for document type; Ceph [35], Sinfonia [36] are designed for the storage object, and their goal is to get higher performance in the object-based query so as to replace the collection-based query.

When the Cloud data management platform is established, the user's query will be faced with massive data. How to improve query efficiency and response quickly to users’ queries also need to be solved in the cloud data management. By the inspiration of the relational database to improve query efficiency, we can know that index mechanisms of Cloud data management system can be established to speed up query performance. Index mechanism in the cloud data management system also has a lot of research in academia, and there are some representative research achievements. References [37][38][39] establish an one-dimensional index in the cloud data management system to speed up query performance. They establish a local index on the physical node which is the actual storage of data and establish a global index in the main system service node. When a user query request, it can locate the physical node by the global index to quickly find the corresponding data from the physical node on the local index, so as to accelerate the efficiency of query. The index mechanisms of [38][39] are based on Epic System [26]. The research team proposed RT-CAN indexing mechanism for multidimensional queries in the Epic system in literature [40], and the index is based on the achievement of the above researches. It establishes the R-Tree index for local data on each node. Thus, it selects the best node in the R-Tree to publish according to a query cost-aware algorithm. Then save frequently connected nodes and maintain a global multi-dimensional search index. So users can find the node containing the results through the least number of hops, and the index model can be extended according to the expansion of the size of the amount of data and the number of compute nodes. The index mechanism has been experimented in the Amazon EC2, and the experiments results showed that RT-CAN indexing mechanism is strong, effective and scalable. Reference [41] proposed a multidimensional index structure for the cloud data management system. The index structure is designed for Master-slave architecture management platform. It establishes the KD-Tree index for the local data on each slave node. Master node will establish an R-Tree index for Slave nodes ' index. The literature also proposed an update strategy based on the cost of index to update the index structure effective and improve update efficiency, and the effectiveness of the index mechanism has been verified by experiments. Reference [42] proposed a similar DBMS index framework contains a similar distribution of B+-Tree index, distributed multidimensional index mechanism. In order to speed up the efficiency of data query, which users only need to select the appropriate index mechanism to establish for the data. And the index framework is scalability and effectiveness which have been varied in Amazon EC2.

The research of cloud data management is not only concern on the system and index mechanism, there are a lot of related technologies need to be researched. Such as the expansion of the data model, system load balancing strategies, query processing research and data security and privacy issues in the cloud data management. Now, both academia and industry do a lot of researches to establish the system, data storage and data index, while little for security issues in query processing and data management in the cloud. This requires a lot of researches on the future research of cloud data management. So the cloud data management technology is more and more mature, it will be widely used in the future.

6. CONCLUSIONS

This paper mainly discuss the forward position technology of cloud data management, the actual instances of concrete cloud data management system through the data model, data partition schema, fault-tolerant mechanism of system, load balancing technology of system, data consistency, availability model and other aspects are analyzed and studied. The paper also Contrast the performance of two open source system detailed by experiment. Finally, the current research situation of cloud data management is investigated. With the development of cloud compute technology and internet technology, we believe that the research of cloud data management technology will be a focus in the industry and academic.

ACKNOWLEDGEMENTS

This work is supported by Natural Science Foundation of China (60933005, 91124002); National High Technology Research and Development Plan of China ("863" plan)(2010AA012505, 2011AA010702, 2012AA01A401, 2012AA01A402); The National Key Technology Research and Development Program of China (2012BAH38B04, 2012BAH38B06); National 242 Information Security Program of China (2011A010).

REFRENCES:

[1] Sims K. IBM introduces ready-to-use cloud computing collaboration services get clients started with cloud computing [OL]. http://www-03.ibm.com/press/us/en/pressrelease/22613.wss.

[2] Divyakant Agrawal, Sudipto Das, Amr El Abbadi, Big Data and Cloud Computing: New Wine or just New Bottles? [J], *VLDB Endowment*, 3(1-2), 2010, pp. 1647-1648.

[3] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, et al, “BigTable: A Distributed Storage System for Structured Data” [C], *Proc of the 7th OSDI*, ACM, 2006, pp. 205–218.

[4] HBase Development Team. “HBase: BigTable-like structured storage for Hadoop HDFS” [OL]. http://wiki.apache.org/hadoop/Hbase/.

[5] Brian F. Cooper, Raghu Ramakrishnan, Utkarsh Srivastava, et al, “PNUTS: Yahoo!'s Hosted Data Serving Platform” [J], *VLDB Endowment*, 1(2), 2008, 1277-1288.

[6] G DeCandia, D Hastorun, M Jampani, et al, “Dynamo: Amazon’s highly available key-value store”, *Proc of SOSP:* ACM, 2007, pp. 205-220.

[7] A. Lakshman, P. Malik, K. Ranganathan, “Cassandra: A structured storage system on a p2p network” [C], *Proc of SPAA*: ACM, 2009, pp. 47-47.

[8] A. Lakshman, P. Malik, “Cassandra: a decentralized structured storage system” [C], *Proc of ACM SIGOPS Operating Systems Review*, 44(2), 2010, pp. 35-40.

[9] Burrows M, “The chubby lock service for loosely-coupled distributed systems” [C], *Proc of the 7th OSDI*: USENIX, 2006, pp. 335−350.

[10] Sanjay Ghemawat, Howard Gobioff, Shun-Tak Leung, “The Google File System” [C], *Proc of SOSP:* ACM, 2003: 29-43.

[11]Apache Hadoop [OL], http://hadoop.apache.org/.
[12]Apache HDFS [OL], http://hadoop.apache.org/hdfs/.
[13]Amazon S3 [OL], http://en.wikipedia.org/wiki/Amazon_S3.
[14]Amazon EBS [OL], http://aws.amazon.com/ebs/.
[15]Apache Zookeeper [OL], http://zookeeper.apache.org/.
[16]Karger. D, Lehman. E, Leighton. T, et al, "Consistent hashing and random trees" [C], *Proc of the 29th STOC*: ACM, 1997, pp. 654-663.
[17]Ion Stoica, Robert Morris, David Liben-nowell, et al, "Chord: a scalable peer-to-peer lookup protocol for internet applications" [J], *IEEE/ACM Transactions on Networking*, 11(1), 2003, pp. 17-32.
[18]Xavier Defago, Peter Urban, Naohiro Hayashibara, et al, "The φ accrual failure detector" [C], *Proc of the 23rd International Symposium on Reliable Distributed Systems:* IEEE, 2004, pp. 66-78.
[19]Sijie Guo, Hongfei Jia, Jing Xiong, "The Technology Survey of Massive Data Storage and Processing in Internet" [J], *Information Technology Letter*, 7(5), 2009, pp. 1-29.
[20]R. Merkle, "A digital signature based on a conventional encryption function" [C], *Proc of the 7th Crypt.*: Springer, 1987, pp. 369-378.
[21]Eric A. Brewer, "Towards robust distributed systems" (Invited Talk) [C], *Proc of the 19th Principles of Distributed Computing:* ACM, 2000.
[22]Eben Hewitt, "Cassandra: The Definitive Guide" [M], *O'Reilly Media*, 2011, pp. 20-22.
[23]Brian F. Cooper, Adam Silberstein, Erwin Tam, et al, "Benchmarking Cloud Serving Systems with YCSB" [C], *Proc of the 1st SOCC*: ACM, 2010, pp. 143-154.
[24]Jason Baker, Chris Bond, James C. Corbett, et al, "Megastore: Providing Scalable, Highly Available Storage for Interactive Services" [C], *Proc of the 5th CIDR:* ACM, 2011, pp. 223-234.
[25]HyperTable [OL], http://hypertable.org/.
[26]Chun Chen, Gang Chen, Dawei Jiang, et al, "Providing Scalable Database Services on the Cloud" [C], *Proc of the 11th Wise*: Springer, 2010, pp. 1-19.
[27]Sudipto Das, Divyakant Agrawal, Amr El Abbadi, "ElasTraS: An Elastic Transactional Data Store in the Cloud" [R], *UCSB*, 2010.
[28]Sudipto Das, Divyakant Agrawal, Amr El Abbadi, "G-Store: A Scalable Data Store for Transactional Multi key Access in the Cloud" [C], *Proc of the 1st SoCC:* ACM, 2010, pp. 163-174.
[29]Azza Abouzeid, Kamil BajdaPawlikowski, Daniel Abadi, et al, "HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads" [C], *Proc of VLDB:* ACM, 2009, pp. 922-933.
[30]Azza Abouzied, Kamil Bajda-Pawlikowski, Jiewen Huang, et al, "HadoopDB in Action: Building Real World Applications" [C], *Proc of SIGMOD:* ACM, 2010, pp. 1111-1114.
[31]Herodotos Herodotou, Harold Lim, Gang Luo, et al, "Starfish: A Self-tuning System for Big Data Analytics" [C], *Proc of 5th CIDR: ACM*, 2011, pp. 261-272.
[32]X. Hu, J. Zhao, X. Meng, et al, "TaijiDB: A Dual-Core Cloud-based Database System" [C], *Journal of Computer Research and Development, Proc of NDBC*, 2010, pp. 433-437.
[33]J. Chris Anderson, Jan Lehnardt, and Noah Slater, "CouchDB: The Definitive Guide" [M]. *O'Reilly Media*，2010.
[34]K. Chodorow, M. Dirolf, "MongoDB: The Definitive Guide" [M]. *O'Reilly Media*, 2010.
[35]S. A.Weil, S. A.Brandt, E. L.Miller, et al, "Ceph: A scalable, high-performance distributed file system" [C], *Proc of the 7th OSDI*: ACM, 2006, pp. 307-320.
[36]M. K. Aguilera, A. Merchant, M. Shah, et al, "Sinfonia: A new paradigm for building scalable distributed systems" [C], *Proc of 21st SOSP*: ACM, 2007, pp. 159-174.
[37]M. K. Aguilera, W. Golab, M.A.Shah, "A Practical Scalable Distributed B-Tree" [J], *VLDB Endowment*, 1 (1), 2008, pp. 598-609.
[38]S.Wu, K.-L. Wu, "An Indexing Framework for Efficient Retrieval on the Cloud" [J], *IEEE Data Engineering Bulletin*, 32(1), 2009, pp. 77–84.
[39]S. Wu, D. Jiang, B. C. Ooi , et al, "CG-Index: A Scalable B+-tree Based Indexing Scheme for Cloud Data Management Systems" [J], *VLDB Endowment*, 3(1-2), 2010, pp. 1207-1218.
[40]Jinbao Wang, Sai Wu, Hong Gao, et al, "Indexing Multi-dimensional Data in a Cloud System" [C], *Proc of SIGMOD:* ACM, 2010, pp. 591-602.

[41]X. Zhang, Z. Wang, J. Ai, et al, “An Efficient Multi-Dimensional Index for Cloud Data Management” [C], *Proc of CloudDB*: ACM, 2009, pp. 17-24.

[42]Gang Chen, Hoang Tam Vo, Sai Wu, et al, “A Framework for Supporting DBMS like Indexes in the Cloud” [J], *VLDB Endowment*, 4(11), 2011, pp. 702-713.

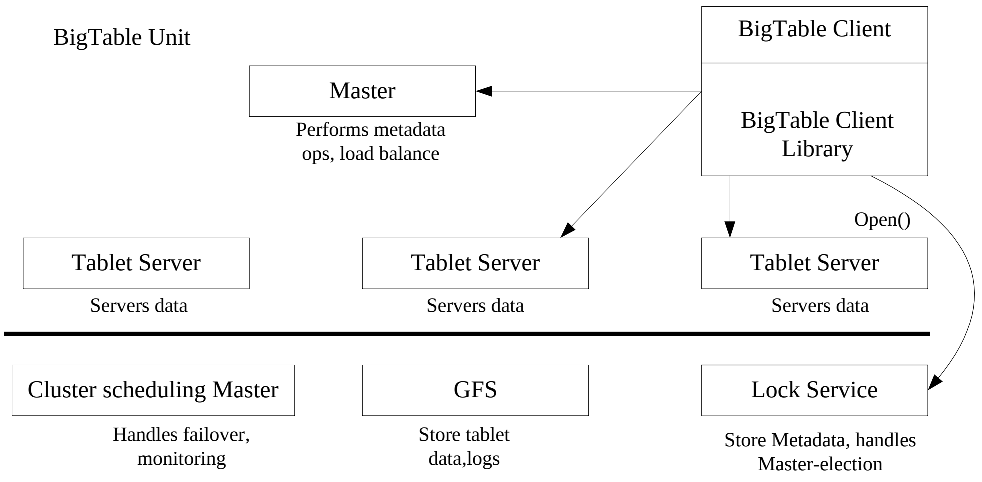


Figure 2: The Architecture of BigTable

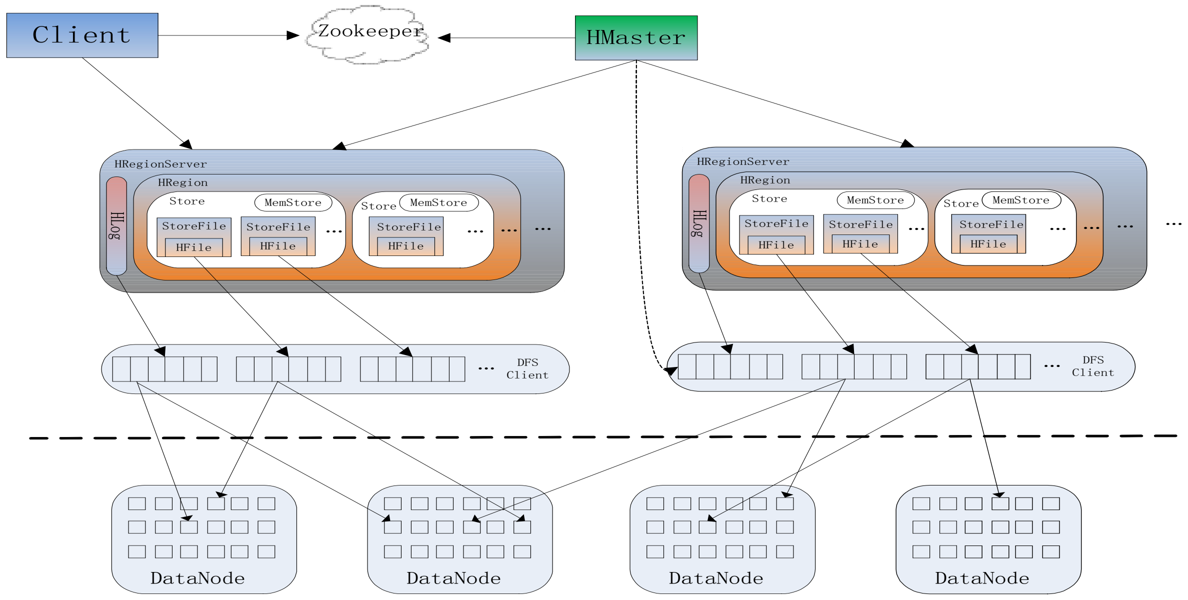


Figure 3: The Architecture of Hbase

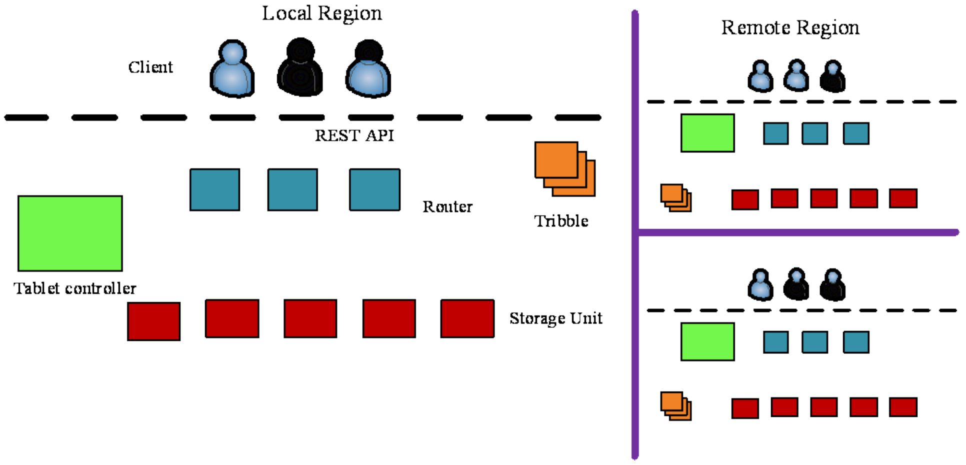


Figure 4: The Architecture of PNUTS

Table 1: Features Comparison among Cloud Data Management Systems

System / Featur	BigTable	HBase	PNUTS	Dynamo	Cassandra
Data consistency model	Weak consistency	Weak consistency	Record eventual time consistency	Eventual consistency	Eventual consistency
Data management	Centralized management	Centralized management	Centralized management	Decentralized management	Decentralized management
Data model	Multi-dimensions table	Multi-dimensions table	Relation table	Raw data, key-value	Multi-dimensions table
Data partition	Range partition	Range partition	Range and hash partition	Consistent Hash partition	Consistent Hash partition
Data high availability	Data log and data replication and replication Synchronous	Data log and data replication and replication Synchronous	Data replication and Master-slave replication strategy of record-level	Data replication and asynchronous replica strategy through the Gossip protocol	Data replication and asynchronous replica strategy through the Gossip protocol
Load balance	Master node schedule	Master node schedule	Master node schedule	Virtual node to reduce load	Chord protocol to reduce load
Failure detection	Chubby lock service and master node monitor the tablet server's status through "heartbeat"	Zookeeper service and master node monitor the RegionServers' status through "heartbeat"	master node monitor every node's status through "heartbeat"	Get every node's status by Gossip schema	Get every node's status by Gossip schema
Failure recovery	Redo log and GFS' Fault-tolerant strategy	Redo log and HDFS' Fault-tolerant strategy	Redo log and remote replication copy strategy	Redo log failure recovery by Merkel tree	Redo log failure recovery by Merkel tree
CAP	CP	CP	AP	AP	AP
Open Source	No	Yes	No	No	Yes