Non-structure Data Storage Technology-An Discussion

Shidong Huang[1,2] Lizhi Cai[1,2]
[1]School of Information Science & Engineering
East China University of Science and Technology
Shanghai, China
E-mail: lovehuishouzan@163.com
E-mail: clz@ssc.stn.sh.cn

Zhenyu Liu[2] Yun Hu[2]
[2]Shanghai Key Laboratory of Computer Software
Evaluating & Testing
Shanghai, China
E-mail: lyz@ssc.stn.sh.cn
E-mail: huy@ssc.stn.sh.cn

***Abstract*—The traditional database is designed for the structured data and the complex query. In the environment of the cloud, the scale of data is very large, the data is non-structured, the request of the data is dynamic, these characteristics raise new challenges for the data's storage and administration, in this context, the NoSQL database comes into being. The article compares and analyzes the general structure of the Cassandra, HBase, MongoDB, the data's distribution mechanism in the cluster, the mechanism of the fault tolerance, their support for the programming model and some relevant technologies, in order to provide some references for selecting or building the NoSQL database solution.**

Keywords- Non-structure; NoSQL; key/value;

I. INTRODUCTION

Non-structure data means that the length of the field is variable, and the record's fields can be constituted of the repeatable or non-repeatable subfields. It can not only handle the structured data (such as numbers, symbols and other information) and more suited to deal with the non-structured data (full text, image, sound, video, hypermedia and other information). With the development of Web2.0 technology, large scale and high concurrency SNS represents one new type of cloud computing applications which generate large amounts of unstructured business data and bring the unstructured data processing needs, such as data grids and databases grid, the application of the Web information integration, log analysis and some internet application, multi-tenant *SaaS* applications and so on. The amount of the data that *BaiDu* stored is more than 20PB, the daily added more than 10TB. Faced with such a huge data size and non-structure features, the traditional relational database cannot satisfy the "Three High" application-demands in the new environment: a) High performance (High concurrency read/write requirement on the database). b) Huge storage (the efficient storage and access requirement on the mass data). c) High scalability (High scalability and availability requirement on the database). On the other hand, the traditional relational database's transaction consistency, real-time writing, multi-table queries and some other features are not so important in the new environment. In this context, non-structure data storage technology NoSQL comes into being. In order to solve the problems in the new environment, the CAP, BASE and the eventual consistency theory are proposed. The CAP theory means that one distributed system can not satisfy the three requirements simultaneously, consistency, availability and the partition fault-tolerant, can only meets two of them. BASE model with the core idea that "can work is ok" is completely different from the traditional ACID relational database model, at the expense of high consistency, access the availability and reliability.

All kinds of non-structured data storage management systems, such as Cassandra [1], HBase [2] and MongoDB [3], have taken various measures to address performance issues in order to achieve performance requirements of data storage. Cassandra is a highly reliable, large-scale distributed storage system with a good scalability. HBase is a distributed, column-oriented storage database, use HDFS as the underlying storage, and supports the batch computing model MapReduce and random reads. Its goal is to use ordinary computer to process the big data table with millions of column elements. MongoDB is a powerful, flexible, collection-oriented and scalable data storage system. It has dynamic failover and automatic slice mechanism to support the cloud-level scalability using the efficient storage mode-binary data; with multi-lingual drivers and the built-in support for MapReduce programming model, as well as the geo-spatial index and so on.

The article introduces new features about the Cassandra, HBase and MongoDB, including the overall architecture, data model, the strategy about the data placement, the mechanism of the detection of the node status and fault handling and programming model, to make people have an initial understanding of the NoSQL data storage solutions.

II. OVERALL ARCHITECTURE

There are two different architectures about the current non-structure data storage system: a) master-slave structure. b) peer-to-peer structure. As the form of master-slave, the master node controls all the slave-nodes. The entire system is almost not affected when the slave-node fails, but when the master node fails will brings a great impact on the whole system, so the master node is the bottleneck of the whole system and the gathering point of the faults. In the peer-to-peer cluster, there are no concepts about the master/slave nodes, any one of the cluster fails which will not bring serious impact to the whole system and the system can still work. The system has a good capability of the fault-tolerance. The architecture adopted by the three NoSQL databases and some related properties are shown as the TABLE I.

This work is supported by the Foundation of Shanghai Committee of Science and Technology (Grant No. 11511500200, 10DZ2291800, 10DZ1123500, 08DZ501600) and the foundation of 863 project of China (Grand NO. 2009AA012201)

978-0-7695-4694-0/12 $26.00 © 2012 IEEE
DOI 10.1109/ICIS.2012.76

TABLE I. THE COMPARISON OF THE THREE DATABASE

	Cassandra	HBase	MongoDB
architecture type	p2p	M/S	M/S
data model	column-oriented	column-oriented	document-oriented
file system	HDFS/LFS	LFS	LFS
replication placement	improved-hash-algorithm	rack awarness	user's configuration
state detection	Gossip protocol	ZooKeeper	Replica set
programming model	MapReduce	MapReduce	MapReduce
consistency	weak	strong	strong
locate the data	quick	slow	slow

MongoDB and HBase use the master-slave structure. When the master node fails, they also take some appropriate handling mechanisms, to avoid the serious impact on the whole system brought by the single point of failure.

Cassandra cluster uses the peer-to-peer architecture, shown as the Fig.1. The statuses of all the nodes are equal in the cluster. The different nodes take the responsibility for data's backup each other, using the improved consistent hash algorithm to perform the data's allocation among the cluster.

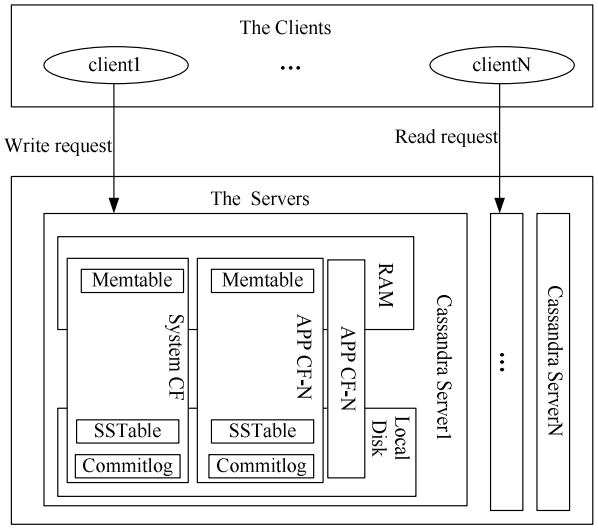

Figure 1. The architecture of the Cassandra.

The architecture of HBase consists of multiple HRegionservers managed by the HMaster, the architecture of the HBase is shown as the Fig.2 [4].With the increment of the number of the records in the table, one table will be spilt into multiple slices which called the regions. Different regions will be assigned to the appropriate HRegionserver for management, and ultimately the data is written to the HDFS(one distributed file system). HMaster is responsible for managing the cluster, HRegionServer is responsible for the data's reading/writing.

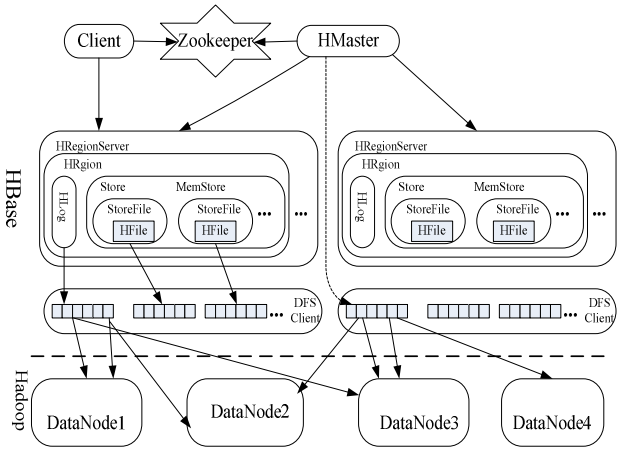

Figure 2. The architecture of the HBase.

The cluster of the MongoDB consists of the configuration server, the data routing servers and the data nodes. The architecture is shown as the Fig.3 [5]. The routing process mongos is responsible for routing all the data's location. The eventual format of the data saved in the disk is BSON. The driver is responsible for the conversion between the native document representation and the BSON format between the client and the server side.

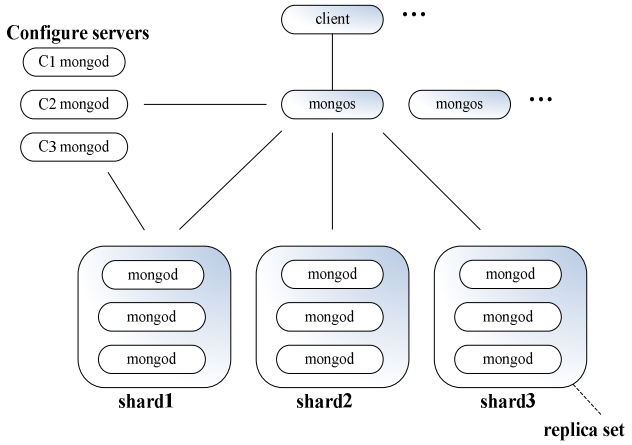

Figure 3. The architecture of the MongoDB.

In the data node, the writing process of Cassandra and HBase, consists of three levels, the log files, memory files, and the persistent files. In the Cassandra, the three kinds of data files are called Commitlog, Memtable, SSTable. In the HBase, the three kinds of data store files are called HLog, MemStore, StoreFile. Cassandra's response procedure of the client's writing request consist of the following three steps. (1) Firstly, update the CommitLog to record the operation of the data. (2) Then the update data is written to the Memtable to cache the data written temporary. (3) When the cached data meets the certain conditions, the updated data is written to the SSTable, and also the update data is written to disk for

the persistent storage. The HBase's process of the data writing is similar to the Cassandra's process.

III. THE CORE SOTRAGE TECHNOLOGIES

A. The Data Model

In order to solve the diversity of the form of data structure, from the point of the data storage model, all the three data storage systems use the key/value to store the data. MongoDB is the document-oriented storage system. The document is the core concept of the MongoDB. The document consists of the multiple keys and the related values. The Fig.4 shows one document consisted of two key/value pairs.

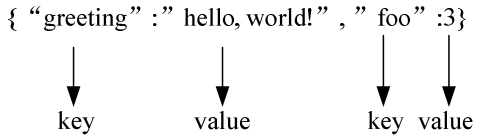

Figure 4. One document consisted of two key/value pairs.

But the Cassandra, HBase are column-oriented storage system, the data are stored in the tables with the tags, the every row of the table consists of one key and many columns, one example of the table is shown as the TABLE II. The table consists of the rows and columns. The cell of the table is the intersection of the row and the column coordination, they all have their version number, by default, the version number of the cell is the timestamp when the cell is inserted assigned by the system automatically. Table cell is a byte array without explanation, the key of the Row of the table is also a byte array without any constraint about the data type. The string, long, binary, serialized data structure can be used as the key of the row. The key of the row of the table is the primary key which is the basis to sort and access the row. Columns of each row are grouped to form the column family. Column family must be defined in advance as part of the table structure, but column family members can also be added based on the actual needs. From the physical point of the view, all the column family members are stored together in the file system.

TABLE II. COLUMN-ORIENTED STORAGE STRUCTURE

Keyspace	Row	Column Family	Column Name	Column Value	Time Stamp
KS-1	Row-1	cfName-1	colName-1	colValue-1	12345678
KS-1	Row-1	cfName-1	colName-2	colValue-2	22345678
KS-1	Row-1	cfName-2	colName-1	colValue-1	12445678
KS-1	Row-2	cfName-1	colName-1	colValue-1	12345678
KS-2	Row-1	cfName-1	colName-1	colValue-1	32345678

B. The Support of the Filesystem

The data of the database is eventually stored in the file system. Some database use the local file system directly, however some work on the distributed file system. MongoDB directly use the common OS' file system to store the data. HBase is structured based on the HDFS [2], so its many mechanisms of management can make use of the HDFS's mechanism, such as the mechanism of the final data's storage and backup. Cassandra can use the local file system and can also make use of the HDFS to store the data through the interface.

C. The Strategy of the Data Placement

The strategy of the data placement in the cluster differs from each other among the three storage system. Cassandra uses the virtual data node to implement the distribution of the data; as to HBase, the strategy of the data distribution determined by the HMaster and the HRegionServer, but the data's distribution among the MongoDB is determined by the user's configuration. Cassandra is structured based on the Bigtable [6] and Dynamo [7]. The strategy of the data distribution among the cluster uses the improved consistency hash algorithm shown in the Fig.5 [1]. It introduces the concept of virtual nodes in the course of the implementation [8].

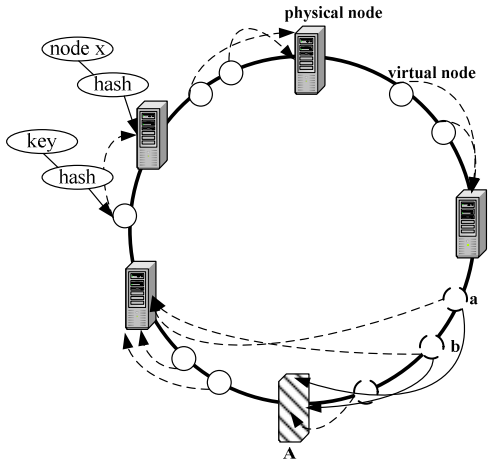

Figure 5. The virtual node of the Cassandra.

When the cluster needs to add or reduce the server, all servers in the cluster do not have to make large-scale changes about the existing data's distribution, only need to change a small amount of the data's distribution in the cluster.

The strategy of the data allocation introducing the virtual nodes is shown as the Fig.5. Every physical node can have many virtual nodes, and every virtual node belongs to one physical node. The data are distributed based on the virtual nodes. How much virtual nodes one physical node has is based on the performance of the physical node, the allocation scheme solves the problem that basic hash algorithm [9] does not consider the differences of physical node's performance. Before the node *A* is added in the cluster, how the virtual nodes are distributed among the physical nodes is shown as the dotted line in the figure5; After the node *A* is added in the cluster, only need to move the data belonged to the virtual nodes *a* and *b* to the physical node *A*(shown as the solid line), the other data on the other virtual nodes does not need movement, the way avoid the large-scale change of the data distribution among the cluster.

In HBase, the HRegionServer is mainly responsible for the data's reading/writing. HRegionServer manage a series of HRegion objects, every HRegion corresponded with one Table's HRegion consists of many HStore. Every HStore acts as one central storage unit corresponded with one Column Family. In the process of the designation of the data model, it's better to put the columns which have the common features of I/O in one column family to improve the I/O efficiency. HStore is composed of MemStore and StoreFiles. The data written by the user firstly is put into the Memstore, when the Memstore file is full, it will be flushed into a StoreFile; when the number of the StoreFlie up to a certain threshold, it will trigger the merge operation. After the merge operation of the StoreFile, the size of the StoreFile will increase; When the size of one StoreFile up to a certain threshold, it will trigger the split operation; at the same time, one Region will be spilt into two Regions, the parent Region will disappear and the two new Regions will be distributed to the corresponding HRegionServer to make the burden on the former one region divided to the two new regions. The eventual data's location is determined by the HDFS.

The distribution of the data in the MongoDB is determined by the user's configuration of the cluster, the writing of the data, the replication of the data, the shard of the data are all determined by the user's configuration. When the user use the single node to save the data without replication, use the process mongos to connect one process instance MongoDB, create the database, then write data to the database. The location of data is determined by the configuration parameter *dbpath*. When the user use the replication mechanism provided by the MongoDB to backup the data, the location of the replication is specified by the user. When use the shard mechanism to store the data, the location of the data is determined by the configuration of the cluster, the user can configure the personalized storage solutions based on the need.

The shard can be a single mongod server and can also be a replication set. The Fig.6 shows the cluster with two shards. The example has two shards and each shard has three servers, but there is only one master server among the three servers, the other servers keep the same replication to avoid the single point of failure. Once the shards function of the database is activated, the data belongs to the different collection will be stored in the different shards, but the data of the same collection is still stored in one shard. Storing the data of one collection in different shards can also be achieved through some special operations. How the data are distributed among the shards are determined by the inner mechanism of the MongoDB, the macro location of the data is determined by the configuration of the cluster.

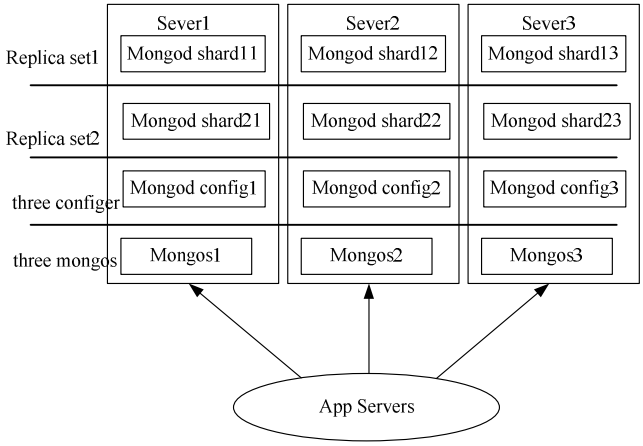

Figure 6. The cluster with two shards.

D. *The Strategy of the Replication*

There are three core goals of the distribute system: Consistency, Availability, Partition tolerance. The three core goals exist in a special relationship that we can't achieve all the three goals at the same time. We can only achieve two of them. This is the CAP theory proposed by the Brewer [10]. As to the distributed database, faced with the same situation, they have their own implementations. In order to achieve the high fault tolerance they all use the replication technology.

Cassandra provides three strategies of the replication placement: simple strategy, old network topology strategy, new network topology strategy [11]. The simple strategy means that according to the specified token to find n nodes in the hash ring in the clockwise direction which will store the replication. The old internet strategy is similar to the simple strategy according to the specified token to find *n* nodes in the hash ring in the clockwise direction. The difference is that when find the second node will choose one node in a different data center from the first node; when find the third node will choose one node in the same data center with the second node, but in the different rack with the second node. All the following nodes were found based on the simple strategy. The new internet strategy can specify the number of the replication one data center should store compared with the old internet strategy.

The mechanism of the fault tolerance of the HBase is dependent on the HDFS' fault tolerance to some extent. The

mechanism of the fault tolerance of the HDFS is implemented through the replication technology. Assume the factor of the replication is *m*, when one block was broken (maybe caused by one node broken) leading to the factor of the replication can't reach the value *m*, the HDFS will copy one from the other available node to make the factor of the replication reach *m* again. Use this way to ensure the availability of the data. At the same time, HDFS use the rack-awareness strategy to improve the reliability of the data, availability of the data, and network bandwidth utilization. The NameNode access the rack id of the each DataNode via the process outlined in Hadoop Rack Awareness [12]. A simple but non-optimal policy is to place replicas on unique racks. This prevents losing data when an entire rack fails and allows use of bandwidth from multiple racks when reading data. This policy evenly distributes replicas in the cluster which makes it easy to balance load on component failure. However, this policy increases the cost of writes because a write needs to transfer blocks to multiple racks. To minimize global bandwidth consumption and read latency, HDFS tries to satisfy a read request from a replica that is closest to the reader. The strategy of placement of replicas is very important for one distributed data storage system. To achieve one better strategy of the placement of the replicas needs lots of tuning and experience.

MongoDB stores all the data in the data directory. By create a copy of all the data in the data directory to implement the backup dealing with failover and data integration. The copy can be used as the extension of the reading, the hot backup, and the source data of the batch operation. The most common way of copy used by the MongoDB is the master-slave copy. The most basic way is to set up a master node and multiple slave nodes. Configure the master/slave node manually, the slave node should know the address of the master node, all the slave nodes copy data form the master node to realize the replication of the data. But there are some problems in the situation, when there are too much slave nodes in the system requesting the master node, the master node is likely to be the bottleneck of the system performance. Therefore, when configure the cluster consist of the master/slave nodes, the scale of the cluster should not be too large. MongoDB currently don't support the copy mechanism from the slave nodes, the future version will support this mechanism.

E. The Detection of the Nodes' State and Fault Handling

As to a distributed database, it is inevitable that there will be a dynamic node joining and leaving, so it must provide a viable state detection and fault handling mechanism to ensure the health of the entire system. Cassandra use the mechanism similar to the Gossip Protocol. HBase's mechanism of the membership detection and the state control is implemented through ZooKeeper. MongoDB uses the replica set to manage the set of the data's backup and implement the node's fault recovery.

In order to translate the information more quickly, the every node in the cluster of Cassandra must keep the routing information of the other nodes. The member's joining and leaving often occurs, in order to the membership information owned by the every node is the newest, all the nodes every fixed time interval will use the Gossip Protocol [5] to communicate to the other one node. If the link is established successfully, they exchange the membership information including the data state and the routing information. In order to discovery the newly node timely, Cassandra introduce the concept of the speed node. When the node fails, the strategy of Cassandra's error detection is very simple, once find the node without response, regard the node fail down, and then select the node to communicate with. Regularly send message to the failed node, if the failed node can response, then they will communicate with it again.

HBase's strategy of the membership detection and the state control is implemented through ZooKeeper. ZooKeeper is one open source service based on the protocol Zab[13] for the distribute coordination, such as the service of the synchronization, the maintenance of information about the configuration, the service of naming etc. ZooKeeper instance is mainly responsible for storing the information of HBase's schema, watching the state of HRegionServer whether it can work well, storing the entry address of the Region. The address of the ROOT table and the HMaster are stored in the Zookeeper Quorum, the HRegionServer also register itself to the Zookeeper, so as to the HMaster can sense the state of the HRegionServer's health. In addition to, the Zookeeper avoids the single point failure of the HMaster, HBase can start more than one HMaster nodes, then use the Master Election mechanism of Zookeeper to make sure that at least one master node can work well.

MongoDB use the replica set to manage the set of the data's backup and realize the fault recovery of the cluster when some nodes fails to make the cluster work well. After assigning the server, the driver will find the server automatically, being responsible for the nodes' state detection and failure recovery when some nodes fail [6]. The difference between the set of the replicas and the master-slave set is that there is no fixed master node in the set of the replicas, the set will select one active node, when the active node fails the set will select one new node acting as the active node providing services, the electoral process initiated by any one non-active node. Whenever the electoral process occurred, the data of the new active node is regarded the newest version and do rollback on the other nodes. The switching of the failed nodes and the procedure of the active node's election is shown as the Fig7 [14]: Every node in the set of the replicas has one value about the priority, in the example, the replica set have five nodes with different values about the priority, if the active node *b* fails, the set will select one new active node which has the biggest value about the priority and the newest data. So the node *c* will be selected as the new active node, the active node use the heartbeat mechanism to detect how many nodes in the cluster can communicate with it, if less than half nodes in the cluster can communicate with it, the active node will be automatically reduced to a backup node. This strategy avoids the phenomenon that the active node will not give up the right when the network failure makes the node apart from the cluster.

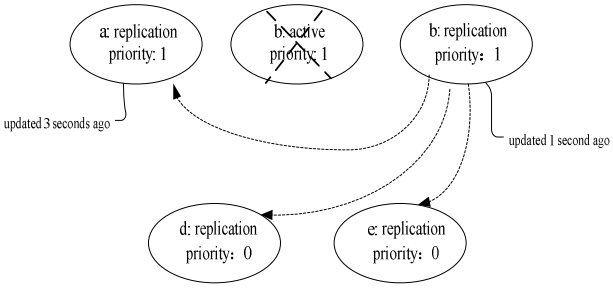

Figure 7. The selection of the active node.

F. Distributed Programming Model-MapReduce

The data storage management system is designed to facilitate people to manage and use the data. Whether can support the convenient programming model is one important aspect of evaluating the performance of the system. The non-structure data storage management system regards the support of the distributed programming model as one of the goals pursued. Cassandra, HBase and MongoDB all support the distributed programming model-MapReduce [15]. The execution model of the MapReduce is shown as the Fig.8 [8]. One map function will do some specified operation on the raw data, the every map operation will work on the different raw data, the relation among the maps are mutually independent, which makes them be fully parallelized. The Reduce will do some merge operations based on the output of the above map operations, the outputs of the map operation processed by the Reduce are not intersectional, do some simple operations based on the outputs of the Reduce can get the complete results, so the Reduce procedure can also be parallelized.

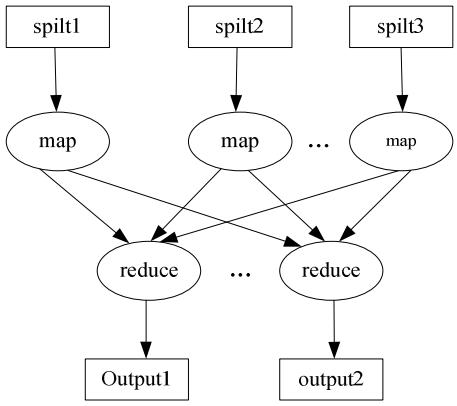

Figure 8. The execution model of the MapReduce

IV. SUMMARY

With the continuous development of cloud computing, the technology of the non-structure data storage and management also develops rapidly. The article simply introduces new features such as the data model about the non-structured data, the mechanism of the data's replication, the mechanism of the fault handling of the cluster, different implement mechanism serves for the different application requirements, the author expect the results of the article's analysis can provide some references for building the non-structured data storage solutions and the selection of the non-structured data storage solutions for the cloud application.

REFERENCES

[1] Guopeng, Cassandra Practice (in Chinese). Mechanical Industry Press, 2011, pp. 2-3.

[2] Tom White, Hadoop: The Definitive Guide (in Chinese). Tsinghua University Press, 2010., pp.13-14.

[3] http://www.mongodb.org.

[4] http://www.searchtb.com/2011/01/understanding-hbase.html.

[5] http://www.tech126.com/category/mongodb-2/

[6] Fay Chang, Jeffety Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E.Gruber. Bigtable: A Distributed Storage System for Structured Data. ACM 2 Penn Plaza, Suite 701 New York NY USA. 2008.

[7] Giuseppe DeCandia, Deniz Hashtorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilichin, Peter Vosshall, Werner Vogels. Swaminathan Sivasubramanian. Dynamo: Amazon's Highly Available Key-value Store. SOSP'07, October 14-27, 2007.

[8] LiuPeng. Cloud Computing (in Chinese), Electronics Industry Press, 2010, pp: 72-73.

[9] Kager,D,Lehman,T.,Panigraphy,R,Levine,M.,and Lewin, D.Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the World Wide Web. In Proceedings of the Twenty-Ninth Annual ACM Symposium on theory of Computing. STOC'97. ACM Press, New York, NY, 654-663.

[10] Seth Gibert. Nancy Lynch. Brewer's Conjecture and the Feasibility of Consistent, Avaliable, Partition-Tolearant Web Services. ACM SIGACT News. Volume 33 Issue 2, June 2002. ACM New York, NY, USA.

[11] Eben Hewitt. Cassandra: The Definitive Guide. Oreilly,2010, pp:103-107

[12] http://hadoop.apache.org/common/docs/cur-rent/hdfs/_design.html.

[13] Benjamin Reed, Flavio P. Junqueira. A simple totally ordered broadcast protocol. In LADIS '08 Proceedings of the 2nd Workshop on Large-Scale Distributed Systems and Middleware ACM New York, NY, USA .2008.

[14] Kristina Cbodorow &Micbael Dirolf. MongoDB: The Definitive Guide,People Post Press, 2011, pp.128-129.

[15] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simple Data Processing on Large Clusters. Communications of the ACM. Volume 51 Issue 1, January 2008.