**Study Report of NOSQL**

**Column Family**

**Siyu Zhou**

**23 April, 2017**

**Catalogue**

1. Introduction 3

1.1 How data store in column family database 3

2. Column Family 4

2.1. Bigtable 4

2.1.1. Index 4

2.1.2. Query 4

2.1.3. Characteristic of data store 5

2.1.4. System architecture 6

2.1.5. How to make operations efficient 8

2.1.6. Example 9

2.1.7. Advantege&Disadvantege 10

2.2. HBASE 10

2.2.1. Index and query lanuage 10

2.2.2. Typical application 10

2.2.3. Consistency, replication and partitioning strategy 10

2.2.4. How HBase works efficient 11

2.2.4. Example 14

2.2.5. Advantage & Disadvantage 15

2.3. Hypertable 15

2.3.1. Index 15

2.3.2. Query Routing 16

2.3.3. Query Language 16

2.3.4. System architecture and data store stragety 17

2.3.5. How Hypertable works efficient 18

2.3.6. Example 19

2.3.7. Advantage & Disadvantage 20

2.4. Cassandra 20

2.4.1. Index 20

2.4.2. Query 21

2.4.3. System Architecture and data store strategy 21

2.4.4. How Cassandra works efficient 23

2.4.5. Example 24

2.4.6. Advantage & Disadvantage 24

2.5. ScyllaDB 25

2.5.1. Improvement of Cassandra 25

2.5.2. Example 26

2.5.3. Advantage & Disadvantage 26

3. Conclusion 26

References 27

# 1. Introduction

In this report, I will introduce what I have learn about NOSQL column family including Google Bigtable, HBase, Hypertable, Cassandra and ScyllaDB. I will introduce how these database replicate data, partition data and keep consistency in data store module. Also, I will introduce other characteristics according to the project requirements.

## 1.1 How data store in column family database

In the relational database, every row has the same columns and data is stored according to the row key. In column family database, data is stored according to the columns. To some degree, column family database can be seen as separates data table by column family which is composed of columns. In other words, data table is divided by vertical rather than by horizontal. If there is no data for a specific cell, the cells in column family database will be empty while every cell will have value as a placeholder in the relational database. Different rows have different columns. In column family databases, columns can be added to specific rows rather than every row should add these columns in the relational database [1].  However, if column database is accessing, it is difficult to add new column family which may cause the database inaccessible [1].

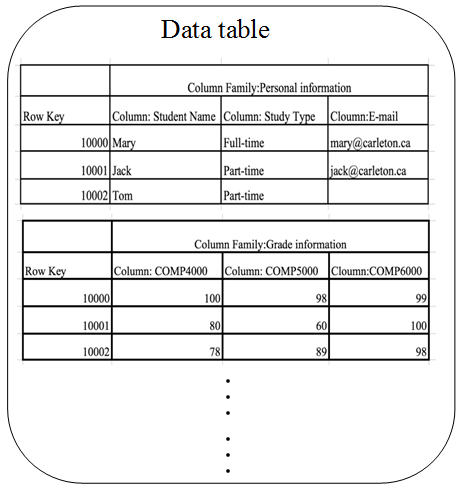


Figure-1: How data store in column family database

# 2. Column Family

## 2.1. Bigtable

Bigtable is a distributed and multidimensional database [2]. Every operation on a single row is an atomic operation.

### 2.1.1. Index

In Bigtable, the index is a string which is composed of row key, column key, and a timestamp [3]. Row key is in lexicographic order. Based on the different ranges of rows, one table can be divided into many tablets according to the balancing load [3]. In this way, small ranges will be more efficient than the whole table when doing operations. Column key has uniform name syntax:" family:qualifier "[4]. Column family contains several columns from the same type. Once column family exists, it can be store data in one specific column in this kind column family. While a table has no constrained about the number of columns, it is better to have at most hundreds of column families and column families not always change when clients do operations [5]. Clients can determine to serve data out of memory or disk through the parameter in column-family level. Every operation will generate a unique timestamp for the cell which has been changed. In order to read the most recent version of the data, different versions of a cell are stored in decreasing timestamp orders [1]. Bigtable use two per-column- family settings to call the cell version garbage collect automatically [1]. In this way, clients can control it should keep last n versions of a cell or versions are changed in specific days.

### 2.1.2. Query

Clients can use API to do operations on Bigtable. When I read the classic Bigtable paper [2], it shows the method to do update in C++ code by RowMutation abstraction. From that paper, there is another example of reading operation in C++ by Scanner abstraction. There are some drawbacks in query method. For example, Bigtable doesn't allow clients to do general operations in different row keys although it has an interface to do writing in different row keys [2]. There is a client supplied script language named Sawzall development by Google can be used to process data [2]. However, Sawzall-based API cannot write data back into Bigtable [2]. I think in this way clients have to use a standard script to do write operations.

### 2.1.3. Characteristic of data store

There are three places to store data: memtable, SSTable, GFS. If clients do update operation, new data will be stored in memtable for temporary [4]. As Time goes, the size of memtable will be increase and exceed the capacity. Then the data in this memtable will be transformed into SSTable [2]. Every SSTable has many blocks. Index of blocks can be used to the navigation [4]. When clients do read operations, the index of blocks will be loaded into memory. Bigtable will do a binary search to find the corresponding block index in memory which has high performance rather than search in a disk. According to the block index, clients can find the block and read data from SSTable.

Finally, Bigtable shifts SSTable into GFS. This process called minor compaction which always creates a new SSTable [2]. Bigtable also has a major compaction which compaction all SSTables into one new SSTable [2]. This SSTable doesn't have the data which already done the delete operation. In this way, some storage can be released and reallocated. What's more, this makes Bigtable become safer if clients want to store sensitive information [2]. If clients did delete operation, these data can be totally disappeared automatically after a specific duration. GFS is a distributed database which has high fault-tolerance. It will replicate data several times and store them in distributed design. If one data cannot be accessed when clients do an operation, don’t worry, there still exist other copy of data. In this way, Bigtable guarantees consistency.

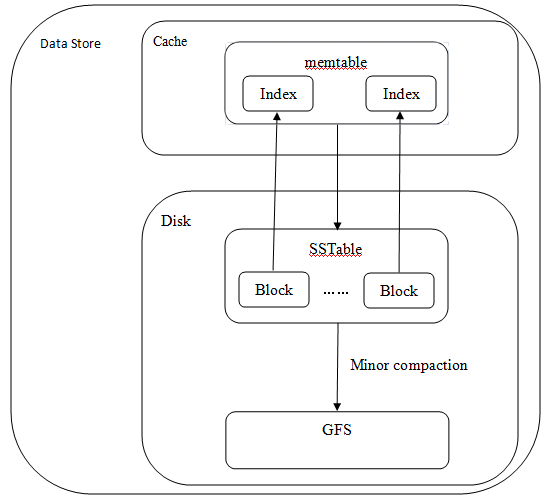


Figure 2.1-1: Data store

### 2.1.4. System architecture

Bigtable is not only relying on GFS but also depend on Chubby lock service. Chubby lock service is another google technology based on Paxos algorithm which helps Bigtable work consistency and manages tablets.

Bigtable uses a method which is similar to B+ tree to store the location information of tablets [2]. Bigtable store the location information of all tablets into one metadata table which is divided into many metadata tablets. The first level is the root tablet’s location which is stored in Chubby. The second level of is root tablet which contains the first metadata tablets and cannot be split. The third level stores other metadata tablets which is pointer to usertable. I think the first metadata tablet should contain the information of other metadata tablets rather than just a regular tablet from one table. In this way, it can be hierarchy structure to find tablet location. However, from the paper which is written by the designer of Bigtable, they didn't clearly say this. The second and third level of B+ tree stored in memory rather than disk

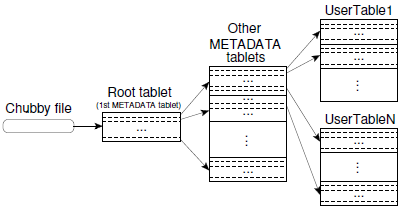


Figure 2.1-2: Query route [3]

Bigtable manage his work through the cluster. The cluster has three parts as following: master server, tablet servers, and tablets [2]. There is just one master server in a cluster while many tablet servers and tablets. Chubby lock service can guarantee this master server is available at any time. In this way, Chubby helps Bigtable to keep consistency. Bigtable uses the master server to keep the balance of tablet servers' workload which has ability to add or remove tablets servers and assign tablets to tablet servers. If there is one tablet unassigned, master server will send an assignment requirement to a tablet server which has sufficient capacity. If clients want to do any operations, they just need to communicate with tablets servers rather than master servers, which release the burden of the master server. If clients’ operations make the size of one tablet exceed a threshold, tablets servers will split this tablet and get one new tablet.

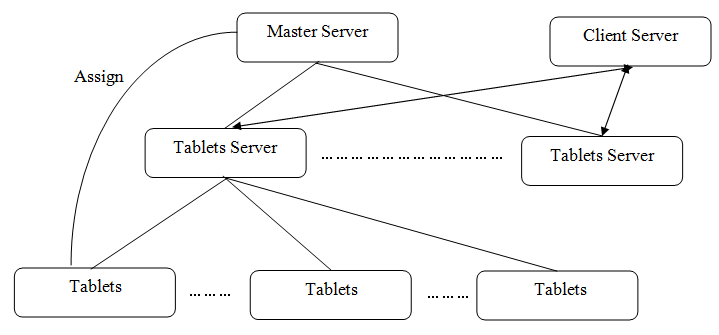


Figure 2.1-3: System component

### 2.1.5. How to make operations efficient

Bigtable use Locality groups, compression, caching, bloom filters, commit-log implementation, speeding up tablet recovery, exploiting immutability to improve operations efficiency [2].

Some column-families who are usually accessed together will be aggregated into same locality group [2]. Every locality-group in every tablet has an independent SSTable. In this way, clients will get as much as possibly related data values from one SSTable rather than read several SSTables. If there are the small size of data always being accessed, clients can store locality group in memory. Once clients request the data in locality group, the data in corresponding SSTables will be loaded into memory. So, clients can get data from in-memory rather access them from disk.

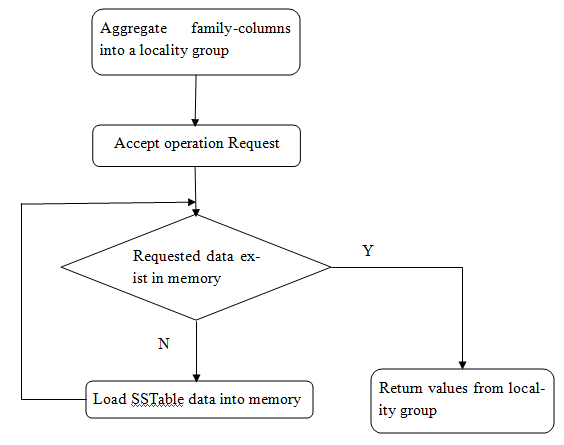


Figure 2.1-4: How locality group works

Clients can do compression to SSTable which contains locality group. In order to get high compression ratios, clients can give similar row keys to similar data tuples. Compressing similar data tuples will output a smaller size file rather than compress totally different data tuples.

In order to improve read performance, Bigtable has two kinds of catch. If clients always access same data, scan cache will provide a higher speed [2]. However, if clients always access data in sequential order or in locality-group, block scan will give a higher speed [2].

In databases, every operation will be written into logs which means is it also important to improve the performance of reading and write log. Bigtable gives one log file for several tablets in order to decrease write operations in GFS and access times of disks. This kind of log file named co-mingling log [2]. However, if clients want to recovery one specific tablets, co-mingling log will be difficult to use. In order to solve this problem, Bigtable have the ability to sort commit log entries. Logs from same tables will be stored in sequence. Tablet server prepares two threads to write log. The writing operation will choose the higher performance thread. In this way, log writing operation will be improved.

### 2.1.6. Example

In that classic paper [2], it provides several examples of Bigtable. Here I will do a summary of Google Earth from that paper. Google Earth uses Bigtable to provide data to clients [2]. There are three types of tables in Bigtable for Google Earth. The first type is just one table which is used for preprocessing data [2]. This table stores the raw data from imagery which is already compressed. There are approximately 70 terabytes data stored in this one table [2]. Because of the data size, it has to store in disks rather than in memory. The second type is tables used for store data after preprocessing [2]. Every key value in the row of this type tables is related to a geographic segment. Similar to locality group, clients can give adjacent geographic segments related row key which can ensure these data are store closely. Every table uses a column family to store the information of every segment's source data. The third type is one table store the index for data in GFS [2]. Since this table works with huge queries per seconds, it has low latency. This table can work with many tablets servers. In this way, clients can query index first and then locate to the tablets in second types table. What's more, in memory of column families contained in this table can speed up this process.

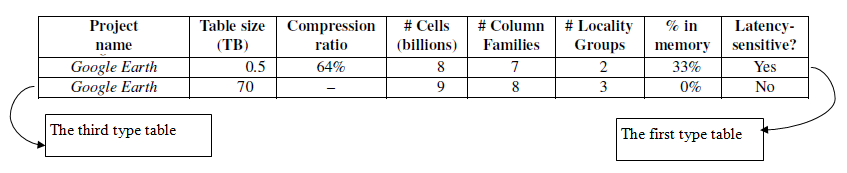


Figure 2.1-5: Example of Bigtable[2]

### 2.1.7. Advantege&Disadvantege

Bigtable is based on distributed system, it is not necessary to provide high-performance machine to store data which is helpful for decrease low cost. Like I said above, Bigtable can guarantee consistency. Bigtable also is a great choice to store sensitive data which can be disappeared from the database after specific duration if clients did delete operation. Based in GFS, Bigtable has high fault-tolerance. API for query is not flexible, some clients supplied scripts cannot be supported in Bigtable.

## 2.2. HBASE

HBase was developed based on the classical google Bigtable paper which means HBase has many similar characteristics with Bigtable. HBase can support large tables for Hadoop.

### 2.2.1. Index and query lanuage

Hbase has same index with Bigtable. The index is a string which is composed of row key, column family:qualifier, and a timestamp[6]. Timestamp will be assigned automatically when an operation happened on a cell. Similarly, clients can choose to keep last n versions of a cell or versions are changed in specific days.

HBase supports 3 types query language as following: API-based, REST-based queries and languages supported [7].

### 2.2.2. Typical application

HBase can be used to store the data which has the version requirement in queries [8]. HBase can work as a scalable data warehouse in Hadoop.

### 2.2.3. Consistency, replication and partitioning strategy

HBase store data in HDFS which is similar to GFS. HDFS can help HBase become consistency. HDFS can break files into blocks and contact namenode to get the location information [6]. Then block will be stored in one datanode and replicated into other two nodes. HDFS has two namenode. If first one doesn’t work, it will use the secondary namenode automatically. In this way, HDFS can not only handle hardware failures but also keep consistency and fault-tolerance.

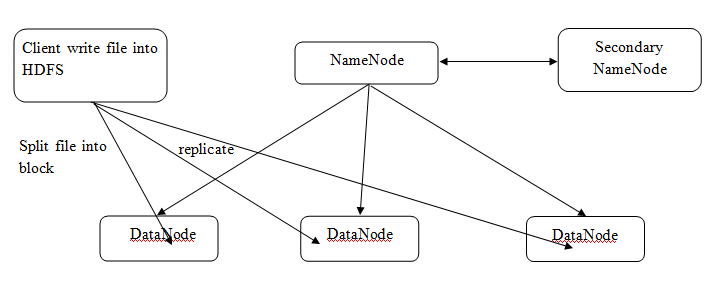


Figure 2.2-1: How HDFS works

HBase also supports replicate data from one cluster into other clusters, which is helpful for the communication between clusters in different locations. HBase does this kind of replication asynchronously [7]. The cluster provides source data called master cluster while the cluster accepts data called slave cluster. In the master cluster, there are Hlogs which contain source data and stored in HDFS. HRegionServer in the master cluster will keep the current replication position in ZooKeeper [7]. When the slave cluster accepts data, replication position will be updated. If replication is stopped because of some reasons, it can be recovered from the latest replication position if ZooKeeper still work.

Graph

Figure 2.2-2: Replication between clusters

In Bigtable, a table is divided into many tablets. In HBase, a table is also divided into small segments according to the row key. These segments are called regions. At first, every table has just one region. If the size of one region exceeds a threshold, it will be split into two regions [6]. By default, region always has 1GB store size [6].

### 2.2.4. How HBase works efficient

HBase also needs a ZooKeeper cluster [8]. In HBase, Master Server called HMaster which runs on namenode. A cluster have has two HMasters, in order to guarantee at least one will work even accident happens [8]. Even HMaster stops for a little while, cluster can still run if HMaster can restart soon [6]. HMaster has the ability to add or move regions in order to keep a balanced workload.

Region Server called HRegionServer, which runs on datanode, has the ability to manage regions. It can check whether regions need to split and manage minor and major compactions. HRegionServer has BlockCache which is very helpful for reading operation [6].  If some data being read frequently, it will be store in catch which will improve the efficiency of reading operation. HRegionSever also has MemStore which is very helpful for writing operation. If the data hasn't been stored in disks, it will be stored in MemStore and be ordered. In every region, there will be one MemStore for one column-family. Since data is sorted before writing into the disk, it will be more easily to write sorted catch data into sorted disk. HRegionServer includes WAL (write Ahead Log). Before system writes data into MemStore, all the data will be write into this log [6]. If HRegionServer doesn’t work suddenly, client can read data from this log file and redo the writing operation. This will keep consistency and fault-tolerance of HBase.

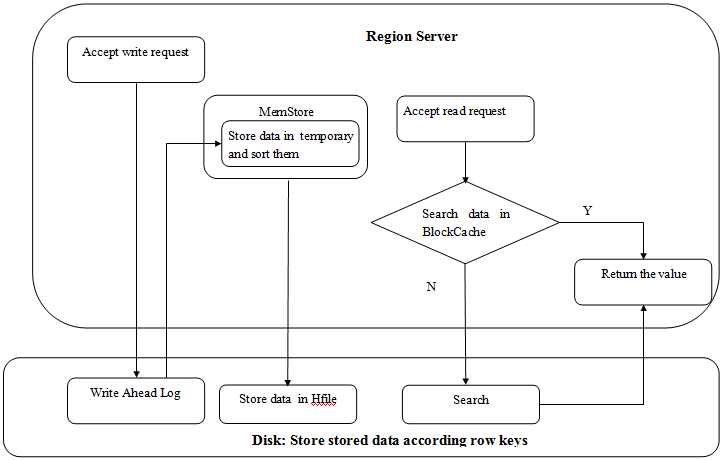


Figure 2.2-3 Read and write operation

ZooKeeper is similar to Chubby in Bigtable, which can track HMaster and HRegionServer[6]. If the communication between ZooKeeper and Region Servers is broken, Zookeeper will redistribute regions from these region servers. If the communication between ZooKeeper and Master Servers is broken, Zookeeper will select a new HMaster to this cluster [8].

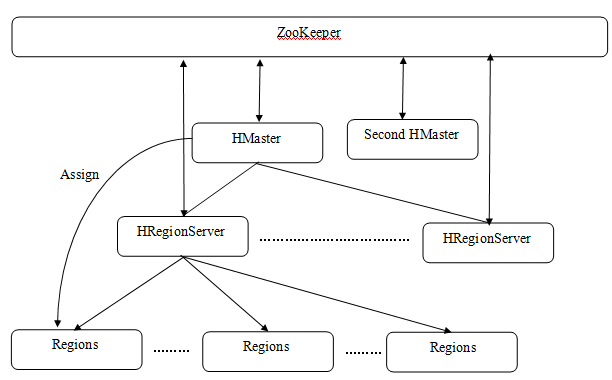


Figure 2.2-4: System component

HBase uses a method similar to B tree to store the location information of regions [6]. HBase store the location information of all regions into one metadata table. The second level of is root tablet which contains just one region to store the location information of META table [6]. This root tablet is stored in ZooKeeper which works as the first level B tree. META table contains the location information of all regions and works as the third level in B tree. Then through HRegionServers, META table can point to the corresponding regions. In this way, clients can find the data in region. Once META location was accessed, clients will store that in catch. Next time, clients can use this information directly rather than do query again if there is nothing change in regions [6]. As a result, this makes operations more efficient.

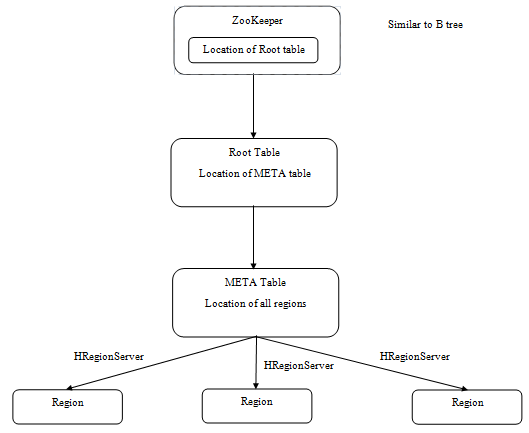


Figure 2.2-5: Query route

### 2.2.4. Example

There is an example about the application of HBase from reference [23]. Facebook used HBase to store data from messages, chat, email, SMS[23]. These data are in a denormalized schema which is easy to store data as column family. These data produce at a massive scale every day. In HBase, this will be multiple cells for saving data in one row. At least 5 racks composing one cluster, four of these racks replicate from the other one [23]. For every rack, there are 20 HRegionServer [23].

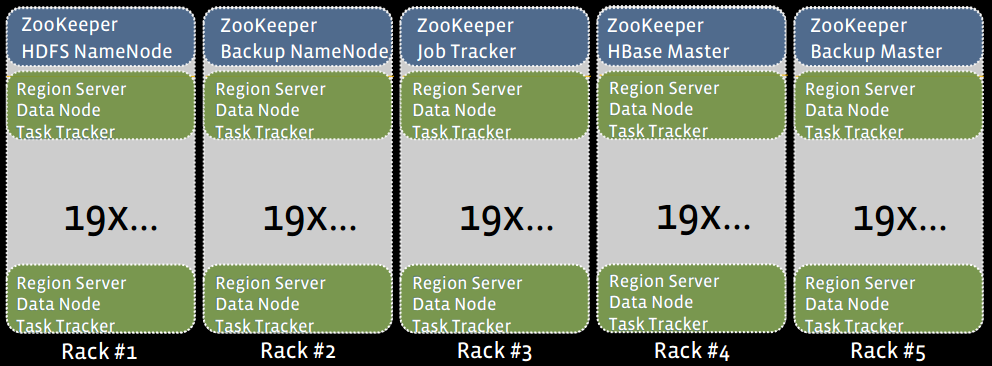


Figure 2.2-6: Example of HBase [23]

When write operation happens, it will first write key and value in to Write Ahead Log which is located on disk. Secondly, these data will be written in to memory and sorted. Finally, these data will be stored into disk.

### 2.2.5. Advantage & Disadvantage

HBase is very suitable to clients who need random real-time read and write operation. API in HBase can be implemented by Java, Rest, Thrift and Avro [9]. HBase is a very important part for Hadoop to deal with non-structure data which Hive has the ability to work with relational data. It makes hadoop ecosystem more powerful. HBase just can do query according to row key which makes it less flexible. Replicating performance between clusters is not very high which makes a long time for HBase recovering from breaking [6].

## 2.3. Hypertable

Hypertable is also based on Google Bigtbale and DFS [10]. It is an open soured datadase for scalable data and written in C++ with high performance [11].

### 2.3.1. Index

Hypertable also has same index with Bigtable. The index is a string which is composed of row key, column family:qualifier, and a timestamp. Every row contains up to 255 column-families which no constraints about the number of qualifiers can be included in every column-family [10]. In other words, every row can have many column attributes by up to 255 categories. Like Bigtable, column data is stored in a sparse format which means empty cells will not be stored [10]. Every cell can have different versions because of the timestamp. Table's schema has the ability to determine how many versions can be kept simultaneous for every cell. When clients do a query by default, the last version will be returned as result. As an example in Figure 2.3-1, data is stored as key-value list while key value is sorted according to lexicographic order. I note the component of index in this picture [10].

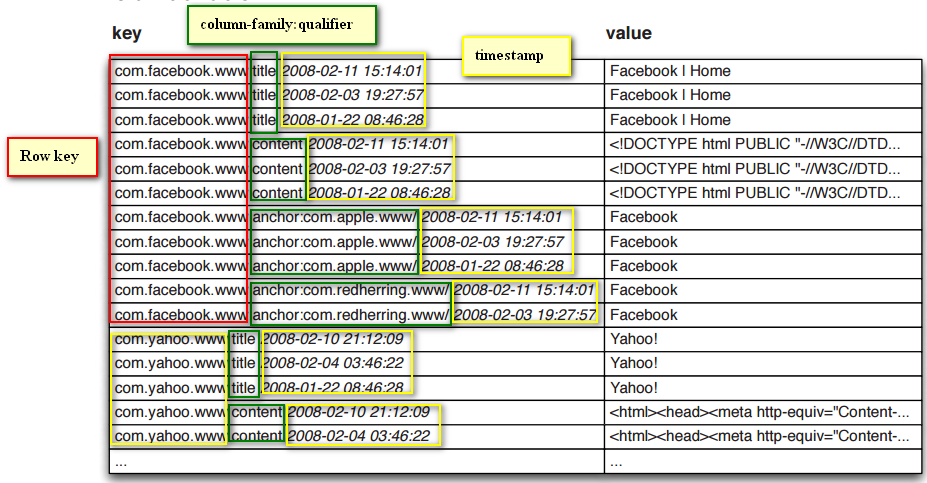


Figure 2.3-1: Index

### 2.3.2. Query Routing

Hypertable has the same three level structures with Bigtable’s, except root table’s location information is store in Hyperspace [10]. Hypertable support finding the routes in public clouds.

### 2.3.3. Query Language

There is a query language called HQL for Hypertable which has similar syntax with SQL to do select, delete ad so on. Here I will write commands [12] for delete the first row and insert one row in Figure 2.3-2 as an example.

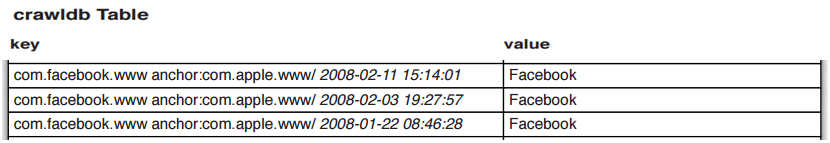


Figure 2.3-2: Crawldb Table[10]

Hypertable> DELETE "anchor:com.apple.www/"

FROM crawldb

WHERE ROW= ‘com.facebook.www’ VERSION “2008-02-11 15:14:01”

Hypertable>INSERT INTO crawldb VALUES ("2008-02-11 15:14:01", " com.facebook.www", "anchor:com.apple.www/", " Facebook " )

### 2.3.4. System architecture and data store stragety

Hypertable uses Hyperspace to work as Chubby in Bigtable. Hyperspace worked on a single server which means if this server was broke, Hyperspace and Hypertable will stop [13]. Chubby works on a cluster which can guarantee consistency.

There is also Master exist in Hypertable [10]. It can be used to create and delete tables. It can check the workload of every Range Server and balance it. If there is a Range Server broke, Master has the ability to find it and assign the workload to other Range Servers or reassign to a new Range Server. Since there is no data transform through Master, it is possible for Master to stop work for a while without cause damage to the system.

There is Range Server in Hypertable which is similar Tablets Server in Bigtable[10]. In Hypertable, a table is also divided into small segments according to the row key. These segments are called Ranges. Range Servers have the ability to manage up to thousands of ranges. All of clients’ operation on data will be managed by Range Server. It is kind of like bridge between operations and data.

Data replication happened in file system. DFS Broker makes Hypertable become possible to work on many kinds of Distributed File System [10]. This design makes Hypertable more flexible than Google Bigtable which is just can work with GFS and HBase which is just can work with HDFS. DFS Broker provides an interface to convert standard filesystem requests to special filesystem requests. According to the Hypertable website, it has the ability to work with HDFS, MapS, Ceph, KFS and local filesystem [10].

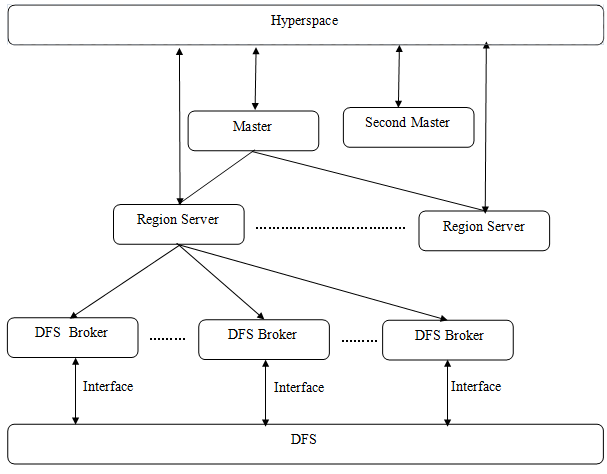


Figure 2.3-3 System architecture

### 2.3.5. How Hypertable works efficient

Access group helps Hypertable to manage data in disks from physical store perspective [10]. It will store the related data together as one access group. When clients do an operation, the disk will just work on this access group rather than work on all columns. For example, this table has 5 column family and two of them are always accessed together. Hypertable makes these two column families as one access group on disk. The disk will only transfer data in this access group if any column family in this group be needed [10]. In this way, it optimizes the I/O of disk and makes queries more efficient.

Before introducing the details of reading and writing operations, I will introduce a concept CellStore. In Ranger Server, there is in-memory cache named CellCahe which is similar to MemStore in HBase [10]. When the size of CellCache exceeds the threshold, the data will be transformed into CellStore. CellStore is a storage name in disk. In CellStore, key/value pairs are stored in blocks while blocks are compressed in CellStore. Bloom Filter statistics the frequency of keys in the block [10]. If the wanted key doesn't exist in the block, Ranger Server will not do block transfer or decompression. In this way, Hypertable will improve the query efficiency. Cell Store also has Block Index and Trailer to work together [10].

When clients want to write data into DFS, it will write data into commit log first [10]. Then, system Hypertable sends a message to filesystem. If there are many write operations paused at this stage, all of these operations will write into log at one time [10]. In this way, it makes Hypertable more efficient. Secondly, the data will be written into CellCache. Finally, when there is no capacity in CellCache, the data will be store in CellStore and clean up CellCache which can be used for a new write operation [10]. When clients want to read data, Range Server will create a unified view of data through MergeScanner object. It will combine data in CellCache and CellStores and sorted data before doing the query.

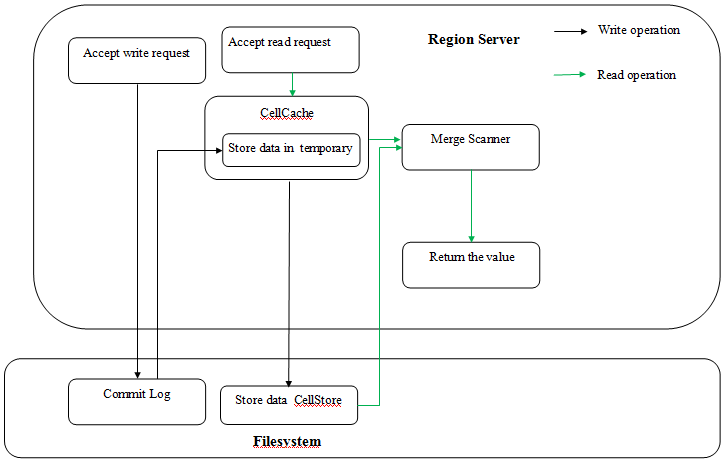


Figure 2.3-4 Read and write operation

### 2.3.6. Example

Serhrch provides an example of application of Hypertable. Sehrch.com is a powerful search engine which is not only supporting structure data but unstructured data [14]. It uses Hypertable to store unstructured data which is in Resource Description Framework (RDF) [14]. Figure 2.3-5 shows how to transform RDF data into Hypertable data format. Since RDF is triples, it makes the transform become easier and data already become the atomic object. Developers in Sehrch.com use Java to write a tool which can work with Thrift API and import RDF into Hypertable [14]. Then data will be worked as what I stated before.

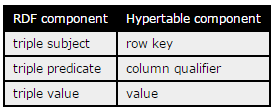


Figure 2.3-5: Example of Hypertable[14]

### 2.3.7. Advantage & Disadvantage

Hypertable which is written by C++ has high performance than HBase. It is also can work with many kinds of filesystem which is more flexible than other databases. Since Hyperspace is just single machine rather than a cluster, this drawback makes Hypertable become dangerous if this single machine broken. Hypertable just support basic query and doesn’t support transaction query [10].

## 2.4. Cassandra

Cassandra is an opensource database based on Google Bigtable and Amazon Dynamo [15]. Cassandra is not based on distributed filesystem which Bigtable and HBase use. It just uses data from local store [15].

### 2.4.1. Index

Cassandra has a different index with Bigtable. The index (Figure 2.4-1) is a string which is composed of row key, column family, super column and a timestamp. Like in Bigtale, it doesn’t have spectacular constraints for row keys. Every operation for one row key is an atomic operation. Super column in nested in column family. In Cassandra, it is possible for clients for to sort columns according to the time or name. Although there is also exist timestamp, Cassandra doesn't keep the different cell values at the same time which means it just keep the latest version. Cassandra supports secondary indexes which can improve the speed of operation.



Figure 2.4-1: Index

### 2.4.2. Query

There is a query language called CQL [16] for Cassandra which has similar syntax with SQL to do select, delete ad so on. Here I will write commands for update an example.

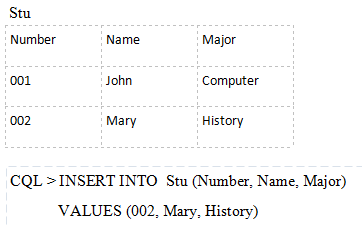


Figure 2.4-2: Query

### 2.4.3. System Architecture and data store strategy

Unlike Bigtbale, Cassandra doesn’t have master node. It uses peer to peer design which means all nodes have same features. In one cluster, all of the nodes work like a ring [15]. This design is very suitable to scalable data. If node capacity is not enough, Cassandra can easily add in a cluster. Since HBase has two master nodes, if they both died HBase will stop work. However, this will not happen in Cassandra. If one node broken in Cassandra, there are many other nodes can replace it. In this way, Cassandra has higher fault-tolerance and scalability than HBase.

In a cluster, Cassandra uses consistent hashing to partition data [17]. By using gossip protocol to communication with each other, nodes switch information every second and they know other nodes’ situations. The key of every data item will be hashed and then find the corresponding node to set data [17]. There is just one node called coordinate for this key while this node can work with clients [17]. Based on hashing, Cassandra assigns data into different nodes and every node takes responsibility of his region. Because of hashing, if Cassandra remove or add one node, it will just affect this node’s nearly neighbor nodes. In order to balance the workload of every node, Cassandra chooses to move lightly loaded nodes near to heavily loaded nodes [17].

Every node in a cluster has one keyspace which can be used for defining replication methods [17]. If a cluster has N nodes, every node will be replicate N-1 times. Coordinate nodes manage every node replication in correct ranges. Cassandra works with Zookeeper to select a node as leader node [17]. When a new node adds into a cluster, leader node will tell new node the information about replicates. Leader node also has the ability to keep every node has other N-1 nodes' replications. In this way, replication makes Cassandra more consistency and fault-tolerance.

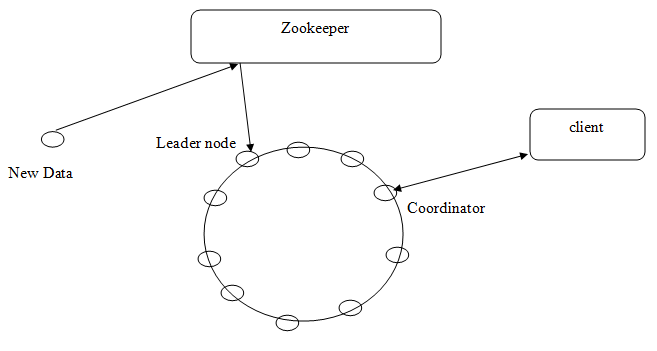


Figure2.4-3 Data partition and replication

Cassandra replication only supports one cluster but also can work with many clusters [15]. For example, clients have many data centers in different cities, data can be replicated across different centers and keep all data centers can be synchronized [15]. In this way, Cassandra can work with distributed filesystem in some degree.

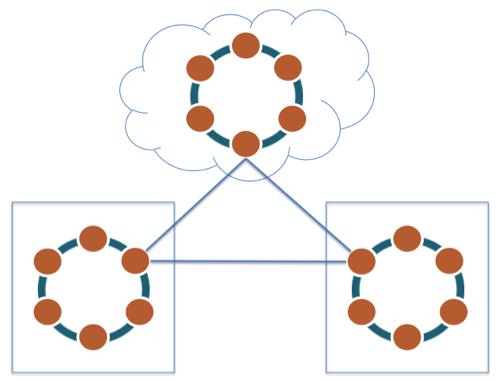


Figure 2.4-4: Cassandra supports multi-data-center and cloud deployments [15]

### 2.4.4. How Cassandra works efficient

Cassandra has similar read and writes process with Hypertable. When clients do write operation, data will be written into commit log first [15]. Then data will be written into memtable. When the size of memtable exceeds the threshold, all the data in memtable will be transferred into SSTable and store in the disk [15]. In this way, data is transferred at one time which will improve writing performance rather than many times in a short period. For reading operation, Cassandra will use Bloom filter in memory to check the probability of whether current SSTable have the requested data [15]. If yes, Cassandra will check memory data and partition key cache to find the location of required data. If no, Cassandra will move to next SSTable and use Bloom filter.

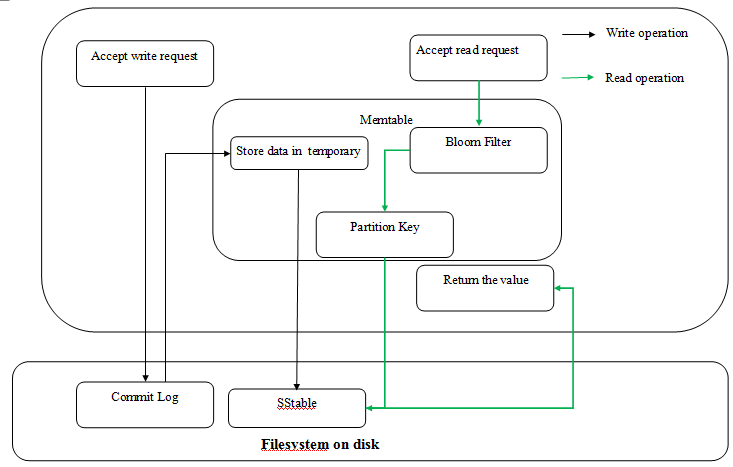


Figure 2.4-5 Read and write operation

### 2.4.5. Example

Facebook Inbox Search use Cassandra as database [17]. It contains message data from senders and recipients. There is a typical search in Inbox Search as following: given a person name and return all message which is sent or accepted by this person [17]. In Cassandra, user's id is row key, recipients' id is super columns and individual message is column [17]. Cassandra also does an improvement by using caching [17]. When a user clicks search bar, the catch will prepare data related to this user. When the user really does query, data is already prepared.

### 2.4.6. Advantage & Disadvantage

Cassandra has high fault-tolerance and several failure nodes don't affect the whole system. It is more convenient to add nodes, which makes system salability. All of the data store in Cassandra storage system, it will be difficult for MapReduce or other tools to work with data. Compare with HBase, it doesn't have a mature ecosystem.

## 2.5. ScyllaDB

To some degree, ScyllaDB is the second generation of Cassandra. ScyllaDB rewrites Cassandra by using C++ [18]. As a result, it completely compatible Cassandra while it improves the writing and reading performance around 10 times [18]. ScyllaDB uses similar method to replicate data, partition data and keep consistency. It also uses similar way to keep efficient when reading and writing data. So, I am not going to repeat this.

### 2.5.1. Improvement of Cassandra

ScyllaDB has the same data structure and data store with Cassandra. By avoiding loading JVM, ScyllaDB reduces the usage of CPU [18]. Cassandra has to allocate memory to take care of garbage collection which is a super cost process when clients do writing and reading operations in high frequency. In some situation, this huge memory will take up all the available memory in several seconds and then Cassandra has to stop until finish garbage collection [19]. SCylla run multiple engines, each one has his own core, CPU and memory which guarantee the low latency for query [22].

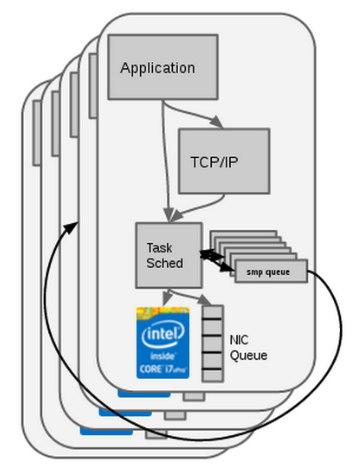


Figure 2.4-6: Modern shared-nothing approach keeps low latency [22]

ScyllaDB uses a kind of cache system called Seastar [20] memory which makes mixed read and write operations more efficient rather than using a complexity data model to solve this problem in Cassandra.  Also because of written in asynchronous [21] programming model, it makes ScyllaDB can run in high throughout.

### 2.5.2. Example

Mogujie is a e-commence company which need low latency database for shopping transaction [23]. They tried to set up ScyllaDB on six machines and it is capable for ScyllaDB deliver 10000TPS [23]. They tried to change disk into SSD, update data schema. Then they inserted 360 data items per seconds and 100 million data items per minute [23]. ScyllaDB still work and doesn’t break. And the network card restricts the performance rather than ScyllaDB [23].

### 2.5.3. Advantage & Disadvantage

ScyllaDB has 10 times throughput than Cassandra which significantly reduces latency. But it still doesn’t solve the problem about store the data in local filesystem which is difficult to work with data analysis tool. Since ScyllaDB is new to world, it is not very stable as HBase.

# 3. Conclusion

In this report, I studied five databases which have different strengths and drawbacks. Google Bigtable is the basic of other four databases. But query API and scripts are not flexible which makes Bigtable cannot be popular in nowadays. Hypertable has the flexible filesystem and high performance but the single node in Hyperspace makes it very dangerous. Cassandra has higher fault-tolerance than Bigtable, HBase and Hypertable but local filesystem makes it difficult to work with data analysis tools. ScyllaDB has very low latency and high performance but still have the same problem with Cassandra. Although, HBase don't have the highest performance in this five databases but it doesn't have fatal drawbacks. And It has a mature ecosystem which makes it easier to be used.

# References

[1] Kim, S. (2014, May). *Better Motivation, Different Thinking*. Retrived April 23, 2017 from http://sungsoo.github.io/2014/05/19/column-family-stores.html

[2] Burrows, M., Chang, F., Chandra, T., Dean, J., Epstein M., Fikes, a., . . . Wallach, D. (2008, June). Bigtable: A Distributed Storage System for Structured Data. ACM Transactions on Computer Systems (TOCS), 26. Retrieved April 3, 2017 from: http://dl.acm.org/citation.cfm?id=1365816

[3] Mira, G. (2016, Februrary 4). NoSQL 6 Example BigTable [Video file]. Retrieved April 3, 2017 from: https://www.youtube.com/watch?v=7p8OXGbOqOg&t=8s

[4] Federico Conte. (2014, November 8). BigTable: A Distributed Structured Storage System [Video file]. Retrieved April 3, 2017 from: https://www.youtube.com/watch?v=2cXBNQClehA&t=61s

[5] Kuznetsov, S.D. & Poskonin, A.V. (2014, November). NoSQL data management systems. Programming and Computer Software, 40. Retrieved April 3, 2017 from: <http://link.springer.com/article/10.1134/S0361768814060152>

[6] McDonald, C. (2015, August). An In-Depth Look at the HBase Architecture. Retrieved April 3, 2017 from: https://mapr.com/blog/in-depth-look-hbase-architecture/

[7] QuABaseBD. (n.d.). HBase Query Language Feature. Retrieved April 23,2017 from https://quabase.sei.cmu.edu/mediawiki/index.php/HBase\_Query\_Language\_Features

[8] JavaOne. (2015, June 2). Apache HBase 101: How HBase Can Help You Build Scalable, Distributed Java Applications [Video file]. Retrieved April 3, 2017 from:https://www.youtube.com/watch?v=KZps2dzr\_u4&t=672s

[9] Apache HBase Team. (n.d.). Apache HBase ™ Reference Guide. Retrieved April 3, 2017 from: http://hbase.apache.org/book.html#arch.overview

[10] HYPERTABLE. (2012, May). Hypertable Architecture Overview [PDF document]. Retrieved April 3, 2017 from: http://www.hypertable.com/collateral/whitepaper-hypertable-architecture.pdf

[11] Neha Kumari. (2015, Februrary 4). Hypertable (introduction and data structures) [Video file]. Retrieved April 3, 2017 from: https://www.youtube.com/watch?v=i2h\_wY4YgzQ

[12] HQL REFERENCE. (n.d.). Retrieved April 3, 2017 from: http://www.hypertable.com/documentation/reference\_manual/hql#insert

[13] Ganesh, Vinay. & Khetrapal, Anhur. (2008). HBase and Hypertable for large scale distributed storage systems A Performance evaluation for Open Source BigTable Implementations [PDF document]. Retrieved April 3, 2017 from: http://cloud.pubs.dbs.uni-leipzig.de/sites/cloud.pubs.dbs.uni-leipzig.de/files/Khetrapal2008HBaseandHypertableforlargescaledistributedstorage.pdf

[14] HYPERTABLE. (n.d.). Sehrch.com: A Structured Search Engine Powered By Hypertable. Retrieved April 3, 2017 from: http://www.hypertable.com/case\_studies/sehrch.html

[15] Data Academy. (n.d.). A Brief Introduction to Apache Cassandra. Retrieved April 3, 2017 from: https://academy.datastax.com/resources/brief-introduction-apache-cassandra

[16] Berryman, J. (2013, July). Understanding How CQL3 Maps to Cassandra’s Internal Data Structure. Retrieved April 3, 2017 from: http://opensourceconnections.com/blog/2013/07/24/understanding-how-cql3-maps-to-cassandras-internal-data-structure/

[17] Malik, P. & Lakshman, A. (2009). Cassandra – A Decentralized Structured Storage System. Retrieved April 3, 2017 from: http://www.cl.cam.ac.uk/~ey204/teaching/ACS/R212\_2014\_2015/papers/lakshman\_ladis\_2009.pdf

[18] Alonso, M,. & Mouron, T. (2015, December). ScyllaDB vs Cassandra: towards a new myth? Retrieved April 3, 2017 from: http://blog.octo.com/en/scylladb-vs-cassandra-towards-a-new-myth/

[19] Qura. (2016, May). What are the main differences between Cassandra and Scylla? Retrieved April 3, 2017 from: <https://www.quora.com/What-are-the-main-differences-between-Cassandra-and-Scylla>

[20] Scylla. (n.d.). Architecture. Retrieved April 3, 2017 from: http://www.scylladb.com/technology/memory/

[21] Lahav, O. (2016, March). We are testing ScyllaDB – live blogging #1. Retrieved April 3, 2017 from: http://techblog.outbrain.com/2016/03/we-are-testing-scylladb-live-blogging-1/

[22] Scylla. (n.d.). Architecture. Retrieved April 3, 2017 from: http://www.scylladb.com/technology/architecture/

[23] Scylla. (n.d.). Building Data Store to Support 60 Million Online Shoppers. Retrieved April 3, 2017 from: http://www.scylladb.com/case-study-mogujie-builds-data-store-to-support-60-million-online-shoppers/

[24] Borthakur, D. (2011, October). Realtime Apache Hadoop at Facebook [PDF document]. Retrieved April 3, 2017 from: <https://people.eecs.berkeley.edu/~istoica/classes/cs294/11/notes/11-dhruba-hbase.pdf>