***Introduction of HBase***

Subtitle as needed ***(paper subtitle)***

Junjie Liu

Carnegie Mellon University - Australia

Heinz College

junjiel1@andrew.cmu.edu

*Abstract—HBase is a column-family no-relational database, which is also called Hadoop database and runs on the top layer of Hadoop. By taking advantage of the scalability of Hadoop’ Mapreduce programming and HDFS (Hadoop Distributed File System). What’s more, HBase is also a powerful database with strictly consistent reads and writes, automatic failover support between RegionServer for real-time queries and so on.* *This report will introduce the features of HBase, the application of HBase in hadoop and some of its use case.*

Keywords — HBase; column-family database; HDFS; hadoop; nosql

# Emergence of HBase

## Background of the emergence of HBase

Because of the fundamental improvement in mobile devices, data communication quality and the emergence of social media, the production of data booms at a surprising speed. Traditionally, a company would choose a server to store all of these data and install RDBMS to handle operations to these data. However, due to the rapidly increasing amount of data, RDBMS such as Oracle 10g, MySql and MS SQL weren’t able to handle such large amount of data. Furthermore, Scaling up in term of database is expensive. Therefore it is a rigorous task to apply relational database in such situation. Soon afterwards, new technologies like distributed system, Nosql database and relative framework emerged to solve this problem and Hadoop is a popular choice.

At first, the backup storing technology for Hadoop is HDFS, which can only perform batch operation, and data can only be accessed in a sequential way. This situation results in that clients can only do the batch searching, even if they only want to get a single row. Users wanted HDFS to support more needs, which it failed to do, such as fast query and individual lookups. As a result HBase came out and supplements hadoop ecosystem.[1]

## HBase VS. HDFS

When people talk about HBase, they also mention HDFS. HBase is a database that stores its data in a distributed file system. The choice of this file system is typically HDFS, which takes care of backup, redundancy and so on. But it doesn't means that HDFS is the only or best choice. We can think HDFS as local filesystem and HBase as database management system to store data in database in a certain format. Internally HBase performs operations on HDFS such as writes and reads. HDFS divides a big file into blocks (64 MBs), it can handle large chucks of data very well, but not very well for a small amount retrieve of data and low-latency data access. That’s the meaning of HBase. HBase goal is to make use of data distributed across the cluster to achieve online real time access. [2].

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | HDFS | | HBase | |
| Meaning | HDFS is a distributed file system designed for storing large amount of data | | HBase is a column-family database built on the top of HDFS | |
| Look up | HDFS excels at batch processing but doesn't support individual row lookup. | | HBase supports two ways : ‘get’ and ‘scan’ to read data. ‘Get’ is simply a ‘Scan’ limited by the API to one row.[[1]](#endnote-1) | |
| Latency | Batch processing in HDFS features high latency | | HBase supports low latency to single row query from billions of records. | |
| Access to data | Data are scanned sequentially in HDFS. | Because of the adoption of Hash Table and the support of random access, HBase stores data in HDFS files with index to achieve fast queries. | |

Table -1: differences of HDFS and HBase [3]

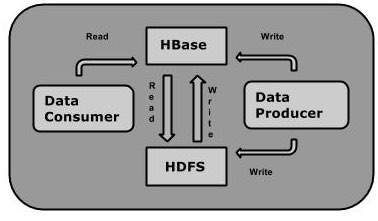


Figure – 1 read and write process in HDFS and HBase[4]

HBase, a column-family data model, was developed from Google’s big table and was designed to provide quick random access to huge amounts of structured data in distributed system. As a key factor to support horizontal scalability, HBase is built on the top of the HDFS in hadoop ecosystem, which provides random real-time read and write access and strictly consistent reads and writes.

What is column-family database? I have had a basic understanding of its structure and some features in the Cassandra case. But HBase has its own features and application in the reality. Data in HBase is identified by its unique Rowid. Like other column-family database, columns in HBase are grouped into column families. All column members of a column family have the same prefix. Column qualifiers are distinct names to the data value, ensuring that users are able to find exact value they want to operate. For instance, the column “Person : student” is a column family, with a column named “student”. [5] Always each row in HBase contain timestamp to identify whether the values in the row is the latest version.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Row id | Column family1 | | | Column family2 | | | Column family3 | | |
| Col1 | Col2 | Col3 | Col1 | Col2 | Col3 | Col1 | Col2 | Col3 |
| 1 |  | | |  | | |  | | |
| 2 |  | | |  | | |  | | |
| 3 |  | | |  | | |  | | |
| 4 |  | | |  | | |  | | |

# HBase distributed model

HBase is often described as a sparse, consistent, distributed, multi-dimensional sorted map. [6] HBase stores structured and semi-structured data in key-value pairs. Each row in HBase is located by a certain style:

<Rowid, Column Family, Column Qualifier, Version>

The purpose of version is to identify the latest version of value in the replications.

## Storage in HBase

HBase is a column-oriented database and rows and comprises a set of tables. There are some concepts in HBase need to be state clear firstly.

**HFiles**: an HFile contains key/values pairs. HFiles are immutable and sorted but can be deleted via compaction or region deletion.

**Regions:** regions are the partitions of table. Each region handles its own reads and writes operations.

**WALs/MemStore**: writes aren’t pushed directly into disk. Otherwise, data are written to the Write Ahead Log (WAL) first, and then stored in-memory until memory reach the limit and triggers a flush to disk. WAL provides a simple way to recover writes not flushed to disk on failure.[7]

Data in tables are sorted by their “Rowkeys”. Users can use “create” command to define the table name and column family names before inserting data into database. Columns families and rows are the units to form the table. In order to realize the read and write consistency, each cell value of the table contains a timestamp. Briefly:

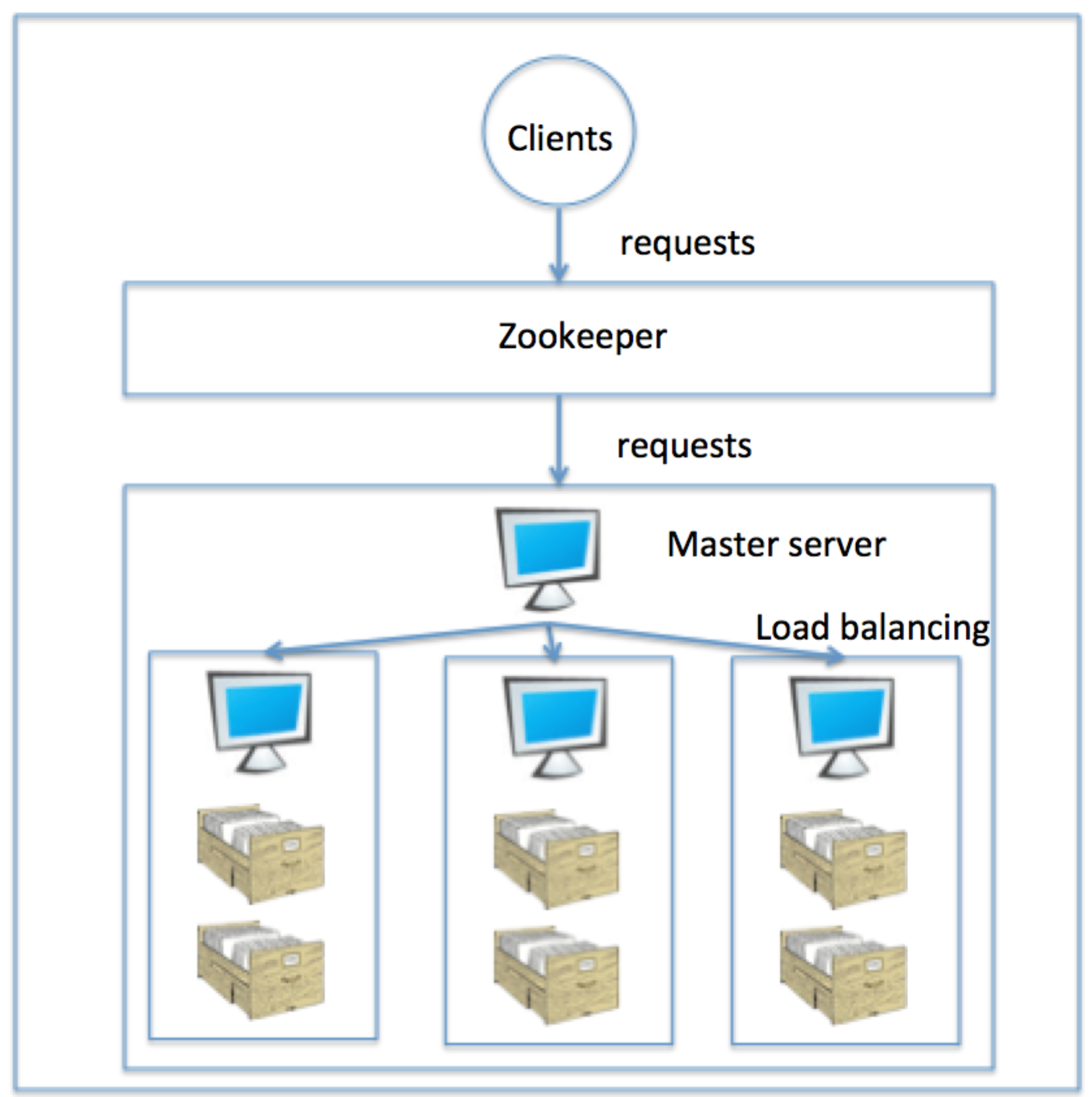
* Table is a collection of rows.
* Row is a collection of columns.
* Columns family is a collection of columns.
* Column is a collection of key-value pairs.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Row id | Personal information | | Department | |
| AndrewID | First Name | Last Name | Depart | Enroll |
| junjiel1 | Junjie | Liu | Heinz | 2015 |
| baibail | BaiBai | Tao | Heinz | 2015 |
| zuoranz | Zhuoran | Zhang | Heinz | 2015 |

“Rowkey” is a unique identifier for a row, and in the table above, “Personal information” and “Department” are column families. “First Name” and “Last Name” are a collection of key value pairs for column family “Personal information”, “Depart” and “Enroll” are a collection of key value pairs for column family “Department”.

## HBase structure

HBase uses master/master and cyclic replication as well as replication to multiple slaves. [8] Everyday, millions requests sent from clients are received by Zookeeper. Via using the nodes availability information stored in zookeeper, MasterServer deals with load balancing across Regions to ensure the traffic for each node. RegionServer responses to requests. This is the common process of how HBase deal with operations sent by users. Furthermore, zookeeper, MasterServer and RegionServer have more duties in the ecosystem.



Slave server

Figure : mechanism of HBase

**Zookeeper**

1. Clients communicate with RegionServer through Zookeeper.

2. Zookeeper stores the configuration information, offers distributed synchronization.

3. Zookeeper provides important information to MasterServer to find the available RegionServer in the cluster.

4. In addition to availability, the nodes are also used to track server failures or network partitions.

**MasterServer:**

1. MasterServer keeps a list of availableRegionServers in the HBase cluster.

2. MasterServer handles the operations to metadata like attribute of tables.

3. MasterServer automatically assigned regions to RegionServers, manage RegionServer failover and balancing workload among RegionServers.

**Region server**

1. RegionServer stores data file and handles the requests from clients.

2. RegionServer monitors the size of regions stored in it and react to situations when the size of the region exceeds the size thresholds. [9]

## Colume-family in HBase

A column family is a collection of HFiles in HBase. The directory structure of a table in HBase is like this:

/table1/region-id1/column-family1/[list of HFiles]

/table1/region-id1/column-family2/[list of HFiles] [10]

Data from a single column family for a single row need not to be stored in the same HFile. But within an HFile, data from a row’s column family should be stored close together. There is a rowkey and all data in one HFile. Where clients need to add some new data, they will be stored in memory. Once memory is full, HBase will store these data in a new HFile. Therefore, column qualifiers are necessary in two HFiles. If someone wants to read values related to that rowkey, HBase needs to search two files. [11]

These HFiles are immutable [12] (someone can’t delete the value by simply remove the key-value pair, instead, a delete marker is written to indicate this value is removed), and sorted. When reading, the Scanner (which reads the data) guarantees that it will scan all HFiles to get the wanted column family data.

# Features [13]

## Eventually consistent

The critical feature of HBase is that it **isn’t** an “Eventually consistent” database, which makes it suitable for heavy application and for situation where high degree of consistency of reads and writes are required.

## Automatic sharding

Another feature of HBase is “automatic sharding”; when regions become too large after keeping writing to it, the region is split into two from the middle key.

When a table is created initially, HBase only allocates one region for the table, which means all operations are handled by a single RegionServer. Therefore, at the initial phase of loading data into an empty table, the whole capacity of the cluster can’t be fully utilized. Once the amount of data of this region reach the threshold, it is automatically split at the middle key, creating two regions by half. [[2]](#endnote-2)

There are other features: HBase supports massively parallelized processing via MapReduce, which uses HBase as both source and sink.(MapReduce). HBase provides Java API for programmatic access. Last, HBase supports a Block Cache and Bloom Filters for high volume query optimization.

# CAP for HBase

## Consistency [14]

HBase replication supports to copy the data from one node to another node; this mechanism is designed for data recovery. When read replicas are active, HMaster replicates read-only copies of regions (replicas) to different RegionServers. One of these RegionServers acts as the master node, the only server that can handle write requests. If the master RegionServer is down, writes will fail.[15]

Other RegionServers act as the slave replicas, only handle read requests, receive updated data from master nodes. The slave nodes are read-only, and are unable to service write requests. The secondary replicas can keep up to date by reading the master node's HFiles at a set interval. In this situation, slave nodes aren’t able to see the latest updates that the master node has not flushed from the memstore to HDFS. If the client receives the read response from a slave node, this value may be stale. Clients can check whether the response is stale and then deal with it.

## Availability [16]

Typically, HBase adopted 2 methods to maintain availability: firstly, HBase automatically partitions data and replicates the partitions across nodes. When a node in cluster crashes, following requests to this node will be guided to other nodes, which has same replications of it. Secondly, HBase uses HDFS to store its file. When data is written in HDFS from memstore in master node, one duplication is stored locally, and then it is replicated to a secondary node and a third copy. This lets HBase automatically distribute data from crushed servers across nodes that are still active.

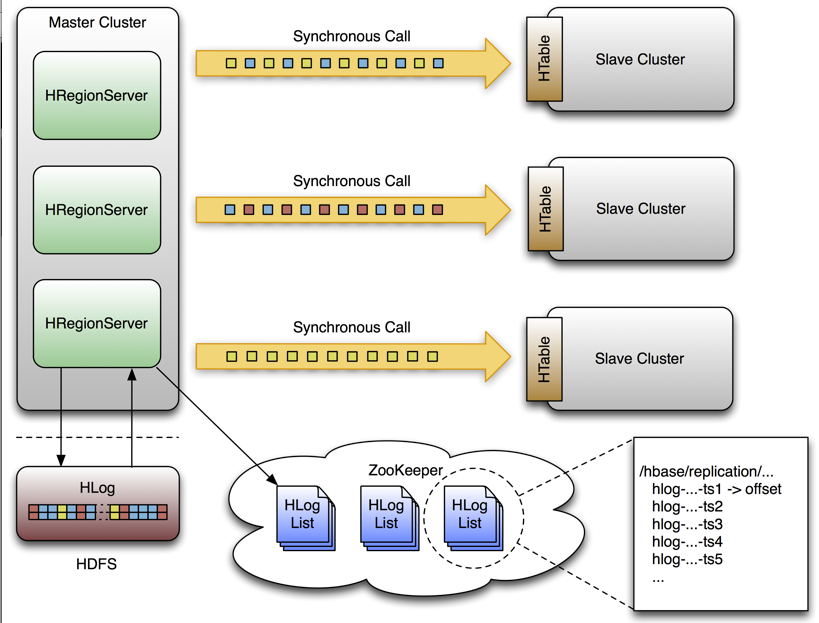


Figure:Hbase replication model[17]

The WAL file and the Hfiles are stored safely on disk and are copied across the cluster, so how does HBase recover the MemStore updates not persisted to HFiles? WAL files contain a list of edits. Edits are written chronologically, so following writes are added to the end of the WAL file. To recover the WAL which hasn't flushed to disk from the crushed HMaster, The HMaster splits the WAL into small parts and stores these file in the other RegionServers. Each Region Server then sends the replicated WAL split to the new MasterServer, therefore the new MasterServer can restore the lost WAL.

## Partition tolerance

Slave nodes aren’t able to communicate with each, but they share information via master node. Typically, Nosql database always sacrifices consistency for availability, but HBase is trying to balance the two parts in CAP theory.

For a read request, the system can response to the request quickly even the response comes from a slave node. If consistency is more important than speed, HBase can ensure that the request is serviced by the master RegionServer. This allows system to adjust the weight of consistency and availability to fit in the need of the application by using Timeline Consistency semantics. [18]

Time consistency is a new feature of HBase and offers more flexible consistency than HBase’s default setting: eventual consistency. Clients are able to claim a level of consistency, so the response to the get and scan commands can meet their requirements. The default consistency in Time consistency is “Strong”, which means that all the read requests are sent to RegionServer. Another setting option “TIMELINE” This is the same behavior as when read replicas are not used. The other possibility, TIMELINE, specified requests are sent to all nodes in cluster. After receiving the response, clients are able to check where the response came from. If it was sent by a RegionServer, client can choose to check whether it is the latest data.

# Transaction [19]

What is the key of transaction? ACID!

* Atomic: entire transaction is committed as one. Users cant see any intermediate steps of the transaction.
* Consistent: No partial change is saved in the database.
* Isolation: transaction is only visible after committed.
* Durable: committed transaction will persist in the database.

HBase isn’t able to provide full support of transaction, but it do provide some degree of transaction:

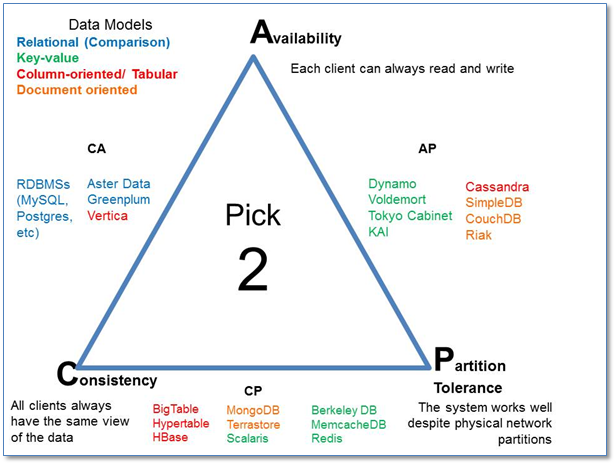
* Atomic operation on cell level.
* Atomic batching operations on rows within a region.
* No support for atomic operations cross regions.
* No support for atomic operations cross tables.

# Cassandra VS. HBase

Either of the two database superior to the other. They are different systems, with their own strengths and weaknesses. They can be used together in supplement of one another in the same infrastructure. One of the criticisms of NoSQL databases is that they do not log data before it is written. However, HBase and Cassandra do use a write before logger to ensure that data can be written to disk safely. This feature guarantee that HBase and Cassandra support to recover when it suffer from downtime.

Cassandra provides the Availability and Partition Tolerance properties and adopt peer-to-peer communication model.

HBase provides the Consistency and Partition Tolerance properties and use master-slave communication model.



Figrue: CAP theory[20]

HBase is built on top of HDFS, so when the system uses hadoop framework, necessarily, HBase is better choice. HBase is also supported by Cloudera. But it doesn't mean that HBase is the only choice for Hadoop, Cassandra also can be used with hadoop. Data generated from the Hadoop tadk can be stored into Cassandra cluster and Cassandra-provided storage file (for example BulkOutputFormat).

# Use cases [21]

There are hundreds of famous companies, which adopt HBase, such as Facebook, Taobao and IBM.



Figure: Companies which use Hbase[22]

HBase is not designed for applications, which require relational analytics. It is also not an absolute substitute for HDFS when doing large batch MapReduce. Although, HBase has SQL-like command, it doesn't mean that it has relational features, such as cross record transactions or joins.

If the applications have a variable schema where each row is slightly different, HBase can be a good option. Hbase is designed and implemented when applications have real-time needs. HBase is good at scaling faster. Reads are faster as a result of some features like schema-like structure, which was built along with timestamp to check version and every piece of datum is a key-value pair.

## Facebook case [23]

Facebook has used HBase to support it Social Messaging business for a long time. Due to the increase of Facebook users, traditional Relational database management system can’t satisfy the rapid increase of the contents. Facebook used cellular to support the data requests. Largely static cellular can provide the ability to analyze data query, but it isn’t able to handle rapid changing data, which push cellular model did not meet the needs of the new social network of Facebook messaging . At last, Facebook chose to adopt hadoop and Hive and HBase to handle the access to data stored in Facebook’ data center. Hive is a data analysis tool on the establishment of Hadoop HDFS and HBase. Facebook uses Hbase to achieve the rapid read and write to data. Because of the adoption of HIVE, HBase, HDFS and Hadoop, Facebook is able to design a larger landscape for its future.

# Future of HBase

HBase has come of age and is used by a prominent customer like Facebook, Twitter, Taobao and so on. It would take a large amount of capital to transform from HBase to other Database in the near future.

Furthermore, HBase in changing and provides more and more functions and features such as timeline consistency solution to solve its nature disadvantage.

In my opinion, HBase will still be one of the popular NoSql databases within 5 years.

1. Tutorialpoint; “HBase - Overview”; [Viewed by 2016/04/13]; http://www.tutorialspoint.com/hbase/hbase\_overview.htm
2. Clint, “The differences between Pig, Hive and HBase”, [cloudera][08-20-2015 01:39 PM]: https://community.cloudera.com/t5/Apache-Hadoop-Concepts-and/The-Differences-between-Pig-Hive-and-HBase/ta-p/31007, viewed by 2016-04-09
3. Tom, “Difference between hadoop/hdfs and hbase”; [viewed by 2016/04/13];http://stackoverflow.com/questions/16929832/difference-between-hbase-and-hadoop-hdfs
4. Tutorialpoint; “HBase - Overview”; [Viewed by 2016/04/13]; http://www.tutorialspoint.com/hbase/hbase\_overview.htm
5. Apache; “Chaptere 5 Data Model”; [viewed by 2016/04/13] ; (http://hbase.apache.org/0.94/book/datamodel.html)
6. Enis Sortutor, “Apache HBase Region Splitting and Merging”, [h ortonworks] http://hortonworks.com/blog/apache-hbase-region-splitting-and-merging/, viewed by 2016-04-0
7. Cloudera; “Hbase write path”; [viewed by 2016/04/13]; https://blog.cloudera.com/blog/2012/06/hbase-write-path/
8. Hadoop, http://hadoop-hbase.blogspot.com.au/2011/12/introduction-to-hbase.html, 2014-03-13.
9. Apache; “Replication”; [viewed by 2016/04/13]; https://hbase.apache.org/0.94/replication.html
10. Tutorialspoint; “HBase archetecture”; [viewed by 2016/04/13]; http://www.tutorialspoint.com/hbase/hbase\_architecture.htm
11. StackOverFlow; “habase and hfiles”; [viewed by 2016/04/13]; http://stackoverflow.com/questions/22732179/hbase-and-hfiles-how-does-it-store-the-columns-family
12. StackOverFlow; “How data stored in HBase”; [viewed by 2016/04/13]; http://stackoverflow.com/questions/22732179/hbase-and-hfiles-how-does-it-store-the-columns-family
13. Hadooptpoint; “Key feature in HBase”; [viewed by 2016/04/13]; http://www.hadooptpoint.com/key-features-in-hbase/
14. Cloudera; “3 learning of HBase”; [viewed by 2016/04/13]; http://www.slideshare.net/cloudera/3-learning-h-base-internals-lars-hofhansl-salesforce-final
15. Apache; “Replication”; [viewed by 2016/04/13]; https://hbase.apache.org/0.94/replication.html
16. Mapr; “In Depth Look Hbase Archeitecture”; [viewed by 2016/04/13]; https://www.mapr.com/blog/in-depth-look-hbase-architecture
17. Apache; replications; [viewed by 2016/04/13]; https://hbase.apache.org/0.94/replication.html
18. Enissoz; “HBase high availability for reads with time”; [viewed by 2016/04/13];http://www.slideshare.net/enissoz/hbase-high-availability-for-reads-with-time
19. Blogspot; “Acid in HBase”; [viewed by 2016/04/13]; http://hadoop-hbase.blogspot.com.au/2012/03/acid-in-hbase.html
20. Google; “Cap Theory”; [viewed by 2016/04/13]; https://www.google.com.au/search?q=cap+theorem&espv=2&biw=1275&bih=651&source=lnms&tbm=isch&sa=X&ved=0ahUKEwjf8MPBoIzMAhXFHJQKHVkLBh4Q\_AUIBigB#imgrc=Td5kzgfL5qfJfM%3A
21. Guru99; “HBase architecture data flow usecase”; [viewed by 2016/04/13];http://www.guru99.com/hbase-architecture-data-flow-usecases.html
22. Linkedin; “Introduction to Hbase”; [viewed by 2016/04/13]; http://www.slideshare.net/alexbaranau/intro-to-hbase
23. Hortonworks; “How facebook uses hadoop and HBase”; [viewed by 2016/04/13];http://hortonworks.com/big-data-insights/how-facebook-uses-hadoop-and-hbase/

1. [↑](#endnote-ref-1)
2. [↑](#endnote-ref-2)