**NoSQL:**

NoSQL, known as Not only SQL database, provides a mechanism for storage and retrieval of data and is the next generation database with schema less.

Most of the NoSQL are open source and it has a capability of horizontal scalability which means that commodity kind of machines could be added.

It is schema free and there is no requirement to design the tables and pushing the data to it.

The crucial factor about NoSQL is that it can handle huge amount of data and can achieve performance by adding more machines to your clusters and can be implemented on commodity hardware.

**NoSQL Database Categories**

**Document Database**- It pairs each key with a complex data structure known as document. It can contain many different key value pairs, or key array pairs  or even nested documents

**Key value stores**- They are the simplest NoSQL databases. Every single item in the database is stored as an attribute  name or key together with its value.

**Graph store**- They are used to store information about networks, such as social connections.Graph stores include Neo4J and HyperGraphDB.

**Wide column stores-** Wide column stores such as Cassandra and HBase are optimized for queries over large datasets, and store columns of data together, instead of rows.

**HBASE:**

Apache HBase provides random, real time access to your data in Hadoop. It was created for hosting very large tables, making it a great choice to store multi-structured or sparse data.

HBase is a distributed column-oriented database built on top of the Hadoop file system. It is an open-source project and is horizontally scalable.

HBase is a data model that is similar to Google’s big table designed to provide quick random access to huge amounts of structured data. It leverages the fault tolerance provided by the Hadoop File System (HDFS).

**HBase and HDFS**

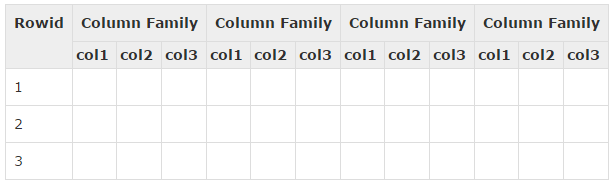
|  |  |
| --- | --- |
| **HDFS** | **HBase** |
| HDFS is a distributed file system suitable for storing large files. | HBase is a database built on top of the HDFS. |
| HDFS does not support fast individual record lookups. | HBase provides fast lookups for larger tables. |
| It provides high latency batch processing | It provides low latency access to single rows from billions of records (Random access). |
| It provides only sequential access of data. | HBase internally uses Hash tables and provides random access, and it stores the data in indexed HDFS files for faster lookups. |

**Storage Mechanism in HBase**

HBase is a **column-oriented database** and the tables in it are sorted by row. The table schema defines only column families, which are the key value pairs. A table have multiple column families and each column family can have any number of columns. Subsequent column values are stored contiguously on the disk. Each cell value of the table has a timestamp. In short, in an HBase:

* Table is a collection of rows.
* Row is a collection of column families.
* Column family is a collection of columns.
* Column is a collection of key value pairs.

Example:



**HBase and RDBMS:**

|  |  |
| --- | --- |
| HBase | RDBMS |
| Column oriented | Row oriented |
| It is built for wide tables. HBase is horizontally scalable. | It is thin and built for small tables. Hard to scale. |
| HBase is schema-less, it doesn't have the concept of fixed columns schema; defines only column families. | An RDBMS is governed by its schema, which describes the whole structure of tables. |
| It has de-normalized data. | It will have normalized data. |
| It is good for semi-structured as well as structured data. | It is good for structured data. |

**HBase components:**

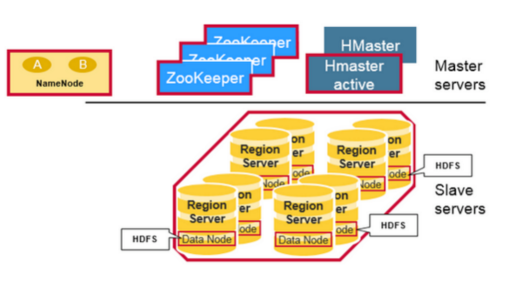
Table made of regions

Region – range of rows stored together

Region Server – serves one or more regions

Master server – It’s responsible for managing HBase cluster

Zookeeper – Coordinates between Master and Salves.



**HBase Write Path:**

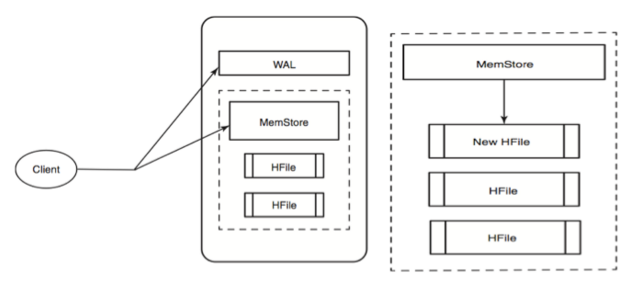
The client doesn’t write data directly into HFile on HDFS

Firstly it writes data to WAL(Write Ahead Log), and Secondly, writes to MemStore shared by a HStore in memory.

**MemStore** is a write buffer(64MB by default). When the data in MemStore accumulates its threshold, data will be flush to a new HFile on HDFS persistently.

Each Column Family can have many HFiles, but each HFile only belongs to one Column Family.

**WAL** is for data reliability, WAL is persistent on HDFS and each Region Server has only on WAL. When the Region Server is down before MemStore flush, HBase can replay WAL to restore data on a new Region Server.

**http://i58.tinypic.com/20i77l0.jpgHBase Read Path**

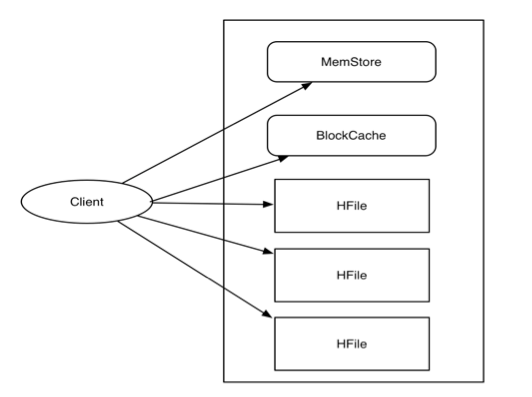
Client will query the MemStore in memory, if it has the target row.

When MemStore query failed, client will hit the BlockCache.

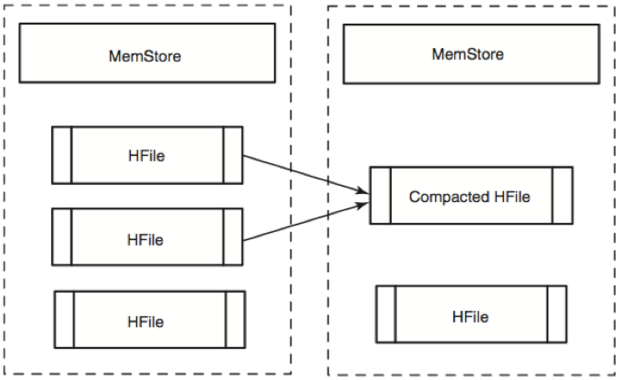
After the MemStore and BlockCache query failed, HBase will load HFiles into memory which may contain the target row info.

The MemStore and BlockCache is the mechanism for real time data access for distributed large data.

**BlockCache is a LRU(Lease Recently Used) priority cache.** Each RegionServer has a single BlockCache. It keeps frequently accessed data from HFile in memory to reduce disk data reads.



**HBase Housekeeping: HFile Compaction:**



Minor Compaction: Merging small HFiles into large HFile and cleaning small HFiles.

Major Compaction:Major Compaction compact all HFiles in a HStore(Column Family) into one HFile. It is the only chance to delete records permanently.

When HBase client send delete request, the record will be marked “tombstone”, it is a “predicate deletion”

Therefore, HBase adopts major compaction to clean up deleted or expired records.

**Locating a Row-Key: Which Region Server is Responsible?**

To put or get a row clients don’t have to contact the master, clients can directly contact the Region Server that handles the specified row, or in case of a client scan, can directly contact the set of Region Servers responsible for handling the set of keys:

To identify the Region Server, the client does a query on the META table.

META is a system table used to keep track of regions. It contains the server name and a region identifier comprising a table name and the start row-key. By looking at the start-key and the next region start-key clients are able to identify the range of rows contained in aa particular region.

