**Explain the below concepts with an example in brief.**

**● Nosql Databases**

Non relational and largely distributed database system that enables rapid, ad-hoc organization and analysis of extremely high-volume, disparate data types.

A very flexible and schema-less data model, horizontal scalability, distributed architectures, and the use of languages and interfaces that are “not only” SQL typically characterize this technology.

**Features of nosql db:**

* + Generic data model :

Heterogeneous containers, including sets, maps, and arrays.

* Dynamic type discovery and conversion :

NoSQL analytics systems support runtime type identification and conversion so that custom business logic can be used to dictate analytic treatment of variation.

* Non-relational and De-normalised :

Data is stored in single tables as compared to joining multiple tables.

* Commodity hardware :

Adding more of the economical servers allows NoSQL databases to scale to handle more data.

* Highly distributable :

Distributed databases can store and process a set of information on more than one device.

**● Types of Nosql Databases**

* **Graph database** – Based on [graph theory](https://web.archive.org/web/20140922065727/http:/en.wikipedia.org/wiki/Graph_theory), these databases are designed for data whose relations are well represented as a graph and has elements which are interconnected, with an undetermined number of relations between them. Examples include: Neo4j and Titan.
* **Key-Value store** – we start with this type of database because these are some of the least complex NoSQL options. These databases are designed for storing data in a schema-less way. In a key-value store, all of the data within consists of an indexed key and a value, hence the name. Examples of this type of database include: [Cassandra](https://web.archive.org/web/20140922065727/http:/www.datastax.com/resources/tutorials/cassandra-overview), DyanmoDB
* **Column store** – (also known as wide-column stores) instead of storing data in rows, these databases are designed for storing data tables as sections of columns of data, rather than as rows of data. Examples include: HBase, BigTable
* **Document databases** - also called document stores, store semi-structured data and descriptions of that data in document format.

Document databases are used for content management and mobile application data handling. Couchbase Server, [CouchDB](http://searchdatamanagement.techtarget.com/definition/CouchDB), DocumentDB, MarkLogic and [MongoDB](http://searchdatamanagement.techtarget.com/definition/MongoDB) are examples of document databases.

**● CAP Theorem**

Concerns of consistency (C), availability (A), and partition tolerance (P) across distributed systems make up what Eric Brewer coined as the CAP Theorem.

Simply put, the CAP theorem demonstrates that any distributed system cannot guaranty C, A, and P simultaneously, rather, trade-offs must be made at a point-in-time to achieve the level of performance and availability required for a specific task.

• Consistency - This means that the data in the database remains consistent after the execution of an operation. For example after an update operation, all clients see the same data.

• Availability - This means that the system is always on (service guarantee availability), no downtime.

• Partition Tolerance - This means that the system continues to function even if the communication among the servers is unreliable, i.e. the servers may be partitioned into multiple groups that cannot communicate with one another.

If two sections of system cannot talk to each other, can they make forward progress on their own?

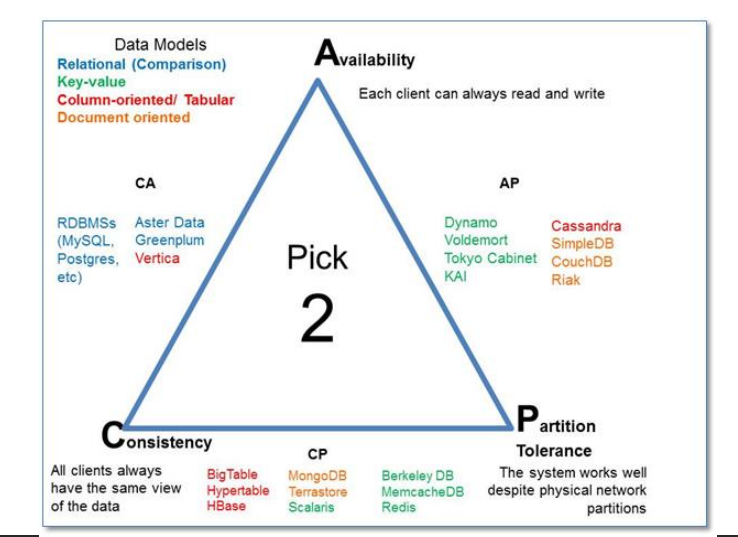
If not then availability goes down.

If yes then consistency goes down.

Duplicate Copy of same data is maintained on Multiple Machines. This increases availability, but decreases consistency.

If data on one machine changes, the update propagates to the other Machine, system is inconsistent, but will become eventually consistent.

If duplicate copy of same data is not maintained, consistency is superior, But availability decreases.



**● HBase Architecture**

HBase is composed of three types of servers in a master slave type of architecture.

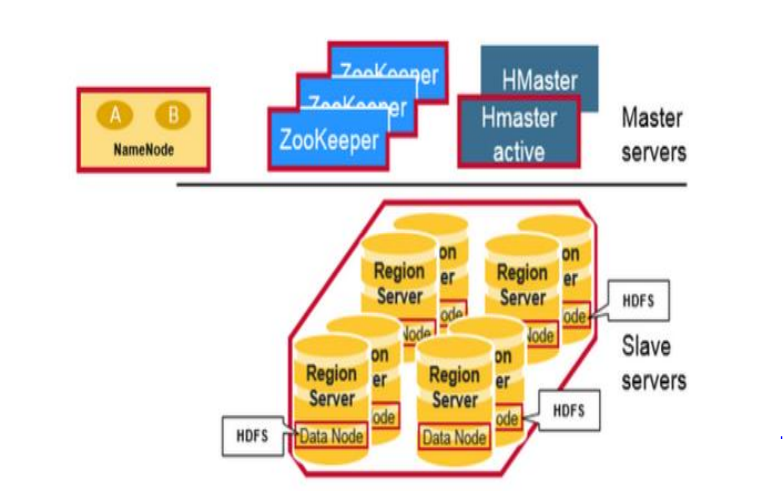
• Region servers serve data for reads and writes.

• HBase Master process handles the Region assignment, DDL (create, delete tables) operations

• Zookeeper maintains a live cluster state.

• All HBase data is stored in HDFS files.

• The NameNode maintains metadata information for all the physical data blocks that comprise the files.



**Regions**:

HBase Tables are divided horizontally by row key range into “Regions.”

• A region contains all rows in the table between the region’s start key and end key.

• Regions are assigned to the nodes in the cluster, called “Region Servers,” and these serve data for reads and writes.

• A region server can serve about 1,000 regions.

**HMaster**:

Region assignment, DDL (create, delete tables) operations are handled by the HBase Master. A master is responsible for:

• Coordinating the region servers

• Assigning regions on startup

• Re-assigning regions for recovery or load balancing

• Monitoring all RegionServer instances in the cluster (listens for notifications from zookeeper)

**Zookeeper**:

HBase uses ZooKeeper as a distributed coordination service to maintain server state in the cluster.

Zookeeper maintains which servers are alive and available, and provides server failure notification.

**Meta Table for read:**

• There is a special HBase Catalog table called the META table, which holds the location of the regions in the cluster.

• ZooKeeper stores the location of the META table.

• The client gets the Region server that hosts the META table from ZooKeeper.

• The client will query the .META. server to get the region server corresponding to the row key it wants to access. The client caches this information along with the META table location.

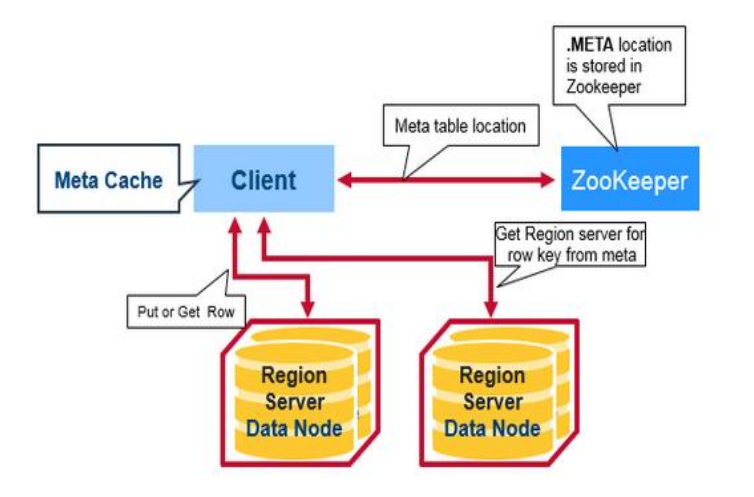
• It will get the row from the corresponding Region Server.

• For future reads, the client uses the cache to retrieve the META location and previously read row keys.

• Over time, it does not need to query the META table, unless there is a miss because a region has moved; then it will re-query and update the cache.

• The .META. table is like a b tree.

• The .META. table structure is as follows: Key: region start key, region id Values: RegionServer



**Region Server:**

* WAL

Write Ahead Log is a file on the distributed file system. The WAL is used to store new data that hasn't yet been persisted to permanent storage; it is used for recovery in the case of failure.

* BlockCache

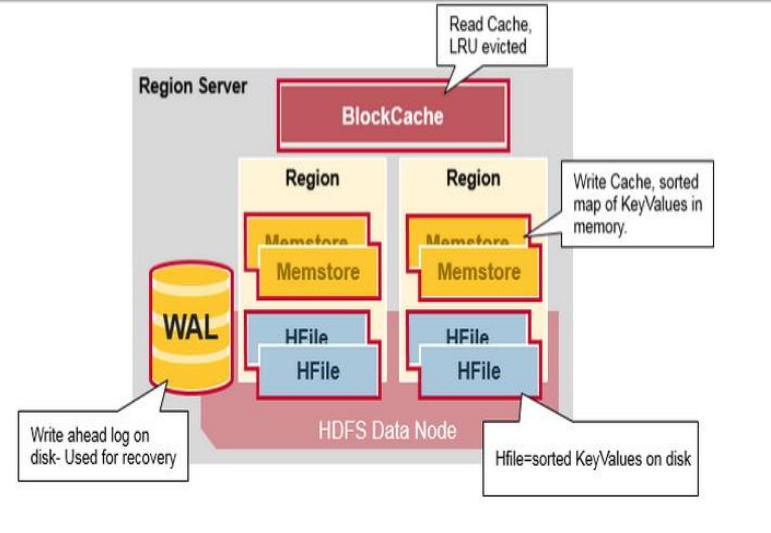
It is the read cache. It stores frequently read data in memory. Least Recently Used data is evicted when full.

* MemStore

It is the write cache. It stores new data which has not yet been written to disk. It is sorted before writing to disk. There is one MemStore per column family per region.

* Hfiles

They store the rows as sorted KeyValues on disk



IMP link:

https://mapr.com/blog/in-depth-look-hbase-architecture/

**● HBase vs RDBMS**

|  |  |
| --- | --- |
| **RDBMS** | **HBASE** |
| Row oriented DB | Distributed, column oriented data storage system |
| Tables have fixed schema | Tables do not have fixed schema |
| Tables guarantee acid properties | Tables guarantee consistency and partition tolerance |
| Uses SQL to query data | Uses java client API and Jruby |
| It is thin and build for small tables. Hard to scale. | Build for wide tables. Horizontal scalable. |
| Transactional | Not transactional |
| It will have normalized data | It has de-normalized data |
| Good for structured data | Good for semi-structured as well as for structured data. |