**Assignment 9**

**NoSQL Databases:**

A NoSQL (originally referring to "non-SQL", "non-relational" or "not only SQL") database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases. Such databases have existed since the late 1960s, but did not obtain the "NoSQL" moniker until a surge of popularity in the early twenty-first century, triggered by the needs of Web 2.0 companies such as Facebook, Google, and Amazon.com. NoSQL databases are increasingly used in big data and real-time web applications. NoSQL systems are also sometimes called "Not only SQL" to emphasize that they may support SQL-like query languages.

Hbase is a column oriented NoSql database for storing a large amount of data on top of Hadoop eco system. Handling tables in Hbase is a very crucial thing because all important functionalities such as Data operations, Data enhancements and Data modeling we can perform it through tables only in HBase.

Handling tables performs the following functions

Creation of tables with column names and rows

Inserting values into tables

Retrieving values from tables

*Example:*

Creating table with column family and columns inside it.

CREATE ‘courses’ ‘hadoop’ ‘programming’

To add columns in each column family the query is enhanced as shown below.

CREATE ‘courses’ ‘hadoop:spark’, ‘programming:java’

**Types of NoSQL Databases:**

There are various NoSQL Databases. Each one uses a different method to store data

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

**Eg: MongoDB, CouchDB**

**Graph stores** are used to store information about networks of data, such as social connections. Graph stores include **Neo4J and Giraph.**

**Key-value stores** are the simplest NoSQL databases. Every single item in the database is stored as an attribute name (or 'key'), together with its value. Examples of key-value stores are **Riak** , **DynamoDB** and **Berkeley DB.** Some key-value stores, such as Redis, allow each value to have a type, such as 'integer', which adds functionality.

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

**CAP Theorem:**

No distributed system is safe from network failures, thus network partitioning generally has to be tolerated. In the presence of a partition, one is then left with two options: consistency or availability. When choosing consistency over availability, the system will return an error or a time-out if information cannot be guaranteed to be up to date due to network partitioning. When choosing availability over consistency, the system will always process the query and try to return the most recent available version of the information, even if it cannot guarantee it is up to date due to network partitioning.

In the absence of network failure – that is, when the distributed system is running normally – both availability and consistency can be satisfied.

CAP is frequently misunderstood as if one had to choose to abandon one of the three guarantees at all times. In fact, the choice is really between consistency and availability only when a network partition or failure happens at all other times, no trade-off has to be made.

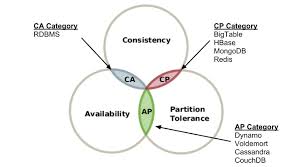
Database systems designed with traditional ACID guarantees in mind such as RDBMS choose consistency over availability, whereas systems designed around the BASE philosophy, common in the NoSQL movement for example, choose availability over consistency.

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.

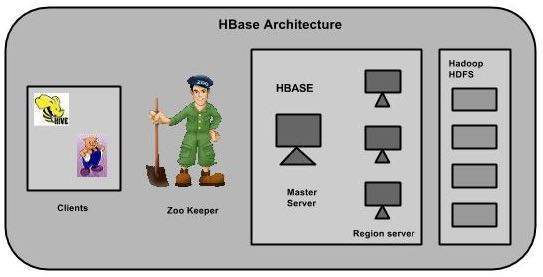
The easiest way to understand CAP is to think of two nodes on opposite sides of a partition. Allowing at least one node to update state will cause the nodes to become inconsistent, thus forfeiting C. Likewise, if the choice is to preserve consistency, one side of the partition must act as if it is unavailable, thus forfeiting A. Only when nodes communicate is it possible to preserve both consistency and availability, thereby forfeiting P. The general belief is that for wide-area systems, designers cannot forfeit P and therefore have a difficult choice between C and A. In some sense, the NoSQL movement is about creating choices that focus on availability first and consistency second; databases that adhere to ACID properties (atomicity, consistency, isolation, and durability) do the opposite.



**HBase Architecture:**

Physically, HBase has three major components: the client library, a master server, and region servers. Region servers can be added or removed as per requirement. Region servers serve data for reads and writes. When accessing data, clients communicate with HBase RegionServers directly. Region assignment, DDL (create, delete tables) operations are handled by the HBase Master process. Zookeeper, which is part of HDFS, maintains a live cluster state.

The Hadoop DataNode stores the data that the Region Server is managing. All HBase data is stored in HDFS files. Region Servers are collocated with the HDFS DataNodes, which enable data locality (putting the data close to where it is needed) for the data served by the RegionServers. HBase data is local when it is written, but when a region is moved, it is not local until compaction.



.

## MasterServer

The master server -

* Assigns regions to the region servers and takes the help of Apache ZooKeeper for this task.
* Handles load balancing of the regions across region servers. It unloads the busy servers and shifts the regions to less occupied servers.
* Maintains the state of the cluster by negotiating the load balancing.
* Is responsible for schema changes and other metadata operations such as creation of tables and column families.

## Regions

Regions are nothing but tables that are split up and spread across the region servers.

### Region server

The region servers have regions that -

* Communicate with the client and handle data-related operations.
* Handle read and write requests for all the regions under it.
* Decide the size of the region by following the region size thresholds.

When we take a deeper look into the region server, it contain regions and stores as shown below:



The store contains memory store and HFiles. Memstore is just like a cache memory. Anything that is entered into the HBase is stored here initially. Later, the data is transferred and saved in Hfiles as blocks and the memstore is flushed.

**HBase Vs RDBMS**

|  |  |
| --- | --- |
| **H Base** | **RDBMS** |
| 1. Column-oriented | 1. 1.Row-oriented(mostly) |
| 2. Flexible schema, add columns on the Fly | 2. Fixed schema |
| 3. Good with sparse tables. | 3. Not optimized for sparse tables. |
| 4. No query language | 4. SQL |
| 5. Wide tables | 5. Narrow tables |
| 6. Joins using MR – not optimized | 6. optimized for Joins(small, fast ones) |
| 7. Tight – Integration with MR | 7. Not really |
| 8. De-normalize your data. | 8. Normalize as you can |
| 9. Horizontal scalability | 9. Vertical scalability |
| 10. Consistent | 10. Consistent |
| 11. No transactions. | 11. transactional |
| 12. Good for semi-structured data as well as structured data. | 12. Good for structured data. |

**SQL vs NoSQL: High-Level Differences**

SQL databases are primarily called as Relational Databases (RDBMS); whereas NoSQL database are primarily called as non-relational or distributed database. SQL databases are table based databases whereas NoSQL databases are document based, key-value pairs, graph databases or wide-column stores. This means that SQL databases represent data in form of tables which consists of n number of rows of data whereas NoSQL databases are the collection of key-value pair, documents, graph databases or wide-column stores which do not have standard schema definitions which it needs to adhered to. SQL databases have predefined schema whereas NoSQL databases have dynamic schema for unstructured data. SQL databases are vertically scalable whereas the NoSQL databases are horizontally scalable. SQL databases are scaled by increasing the horse-power of the hardware. NoSQL databases are scaled by increasing the databases servers in the pool of resources to reduce the load. SQL databases uses SQL (structured query language ) for defining and manipulating the data, which is very powerful. In NoSQL database, queries are focused on collection of documents. Sometimes it is also called as UnQL (Unstructured Query Language). The syntax of using UnQL varies from database to database.

SQL database examples: MySql, Oracle, Sqlite, Postgres and MS-SQL. NoSQL database examples: MongoDB, BigTable, Redis, RavenDb, Cassandra, Hbase, Neo4j and CouchDb

For complex queries: SQL databases are good fit for the complex query intensive environment whereas NoSQL databases are not good fit for complex queries. On a high-level, NoSQL don’t have standard interfaces to perform complex queries, and the queries themselves in NoSQL are not as powerful as SQL query language.

For the type of data to be stored: SQL databases are not best fit for hierarchical data storage. But, NoSQL database fits better for the hierarchical data storage as it follows the key-value pair way of storing data similar to JSON data. NoSQL database are highly preferred for large data set (i.e for big data). Hbase is an example for this purpose.

For scalability: In most typical situations, SQL databases are vertically scalable. You can manage increasing load by increasing the CPU, RAM, SSD, etc, on a single server. On the other hand, NoSQL databases are horizontally scalable. You can just add few more servers easily in your NoSQL database infrastructure to handle the large traffic.

For high transactional based application: SQL databases are best fit for heavy duty transactional type applications, as it is more stable and promises the atomicity as well as integrity of the data. While you can use NoSQL for transactions purpose, it is still not comparable and sable enough in high load and for complex transactional applications.

For support: Excellent support are available for all SQL database from their vendors. There are also lot of independent consultations who can help you with SQL database for a very large scale deployments. For some NoSQL database you still have to rely on community support, and only limited outside experts are available for you to setup and deploy your large scale NoSQL deployments.

For properties: SQL databases emphasizes on ACID properties ( Atomicity, Consistency, Isolation and Durability) whereas the NoSQL database follows the Brewers CAP theorem ( Consistency, Availability and Partition tolerance )

For DB types: On a high-level, we can classify SQL databases as either open-source or close-sourced from commercial vendors. NoSQL databases can be classified on the basis of way of storing data as graph databases, key-value store databases, document store databases, column store database and XML databases.

### **SQL Database Examples**

1. MS-SQL Server Express Edition

It is a powerful and user friendly database which has good stability, reliability and scalability with support from Microsoft. The following are some of MS-SQL benefits and strengths:

Integrated Development Environment: Microsoft visual studio, Sql Server Management Studio and Visual Developer tools provide a very helpful way for development and increase the developers productivity.

Disaster Recovery: It has good disaster recovery mechanism including database mirroring, fail over clustering and RAID partitioning.

Cloud back-up: Microsoft also provides cloud storage when you perform a cloud-backup of your database.

**NoSQL Database Examples:**

1. MongoDB

MongoDB is one of the most popular document based NoSQL database as it stores data in JSON like documents. It is non-relational database with dynamic schema. It has been developed by the founders of DoubleClick, written in C++ and is currently being used by some big companies like The New York Times, Craigslist, MTV Networks. The following are some of MongoDB benefits and strengths:

*Speed:* For simple queries, it gives good performance, as all the related data are in single document which eliminates the join operations.

*Scalability:* It is horizontally scalable i.e. you can reduce the workload by increasing the number of servers in your resource pool instead of relying on a standalone resource.

*Manageable:* It is easy to use for both developers and administrators. This also gives the ability to shard database

*Dynamic Schema:* Its gives you the flexibility to evolve your data schema without modifying the existing data.