**Assignment 9.3:**

Problem Statement:

● Explain the below concepts with an example in brief.

1. **Nosql Databases:**

* NoSQL is an approach to database design that can accomodate a wide variety of data models, including key-value, document, columnar and graph formats. NoSQL, which stand for "not only SQL," is an alternative to traditional relational databases in which data is placed in tables and data schema is carefully designed before the database is built. NoSQL databases are especially useful for working with large sets of distributed data.
* **NoSQL database examples:**Dozens of NoSQL data stores are available; the following are among the most popular:
* **MongoDB**. Open-source document database.
* **CouchDB**. Database that uses JSON for documents, JavaScript for MapReduce queries, and regular HTTP for an API.
* **GemFire**. Distributed data management platform providing dynamic scalability, high performance, and database-like persistence.
* **Redis**. Data structure server wherein keys can contain strings, hashes, lists, sets, and sorted sets.
* **Cassandra**. Database that provides scalability and high availability without compromising performance.
* **memcached**. Open source high-performance, distributed-memory, and object-caching system.
* **Hazelcast**. Open source highly scalable data distribution platform.
* **HBase**. Hadoop database, a distributed and scalable big data store.
* **Mnesia**. Distributed database management system that exhibits soft real-time properties
* **Neo4j**. Open source high-performance, enterprise-grade graph database.

1. **Types of Nosql Databases:**

**There are 4 basic types of NoSQL databases:**

* **Key-Value Store** – It has a Big Hash Table of keys & values {Example- Riak, Amazon S3 (Dynamo)}
* **Document-based Store**- It stores documents made up of tagged elements. {Example- CouchDB}
* **Column-based Store**- Each storage block contains data from only one column, {Example- HBase, Cassandra}
* **Graph-based**-A network database that uses edges and nodes to represent and store data. {Example- Neo4J}

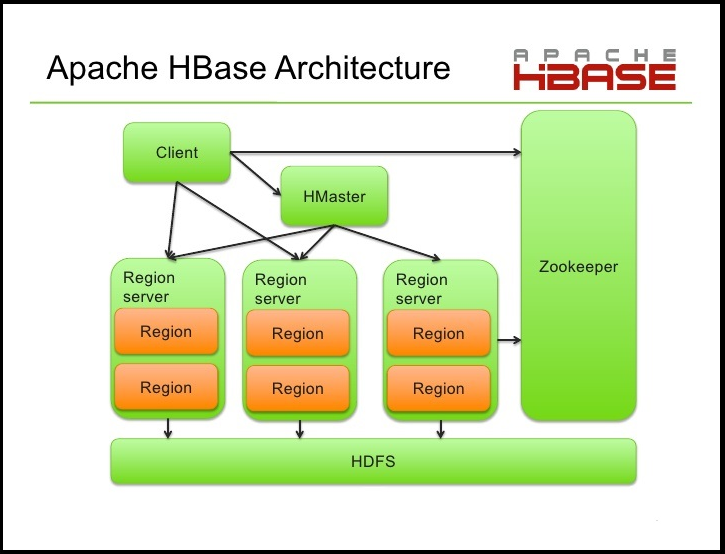
**Q)** **CAP Theorem:**

For any distributed system, CAP Theorem reiterates the need to find balance between Consistency, Availability and Partition tolerance. Consistency means all the nodes see the same data at the same time. Availability implies that every request receives a response about whether it was successful or failed. It’s more of a handshaking mechanism in computer network methodology.

Coming to partition tolerance, the system continues to operate despite arbitrary message loss or failure of part of the system. Systems with partition tolerance feature works well despite physical network partitions.

According to CAP Theorem distributed systems can satisfy any two features at the same time but not all three features. Traditional systems like RDBMS provide consistency and availability. Column oriented databases like MongoDB, Hbase and Big Table provide features consistency and partition tolerance.

**Q) HBase Architecture:**



Hbase architecture consists of mainly HMaster, HRegionserver, HRegions and Zookeeper. Zookeeper is a centralized monitoring server which maintains configuration information and provides distributed synchronization. If the client wants to communicate with regions servers, client has to approach Zookeeper.

**HMaster**

HMaster in Hbase plays vital role in terms of performance and maintaining nodes in the cluster. It provides admin performance and distributes services to different region servers. HMaster assigns regions to region servers.

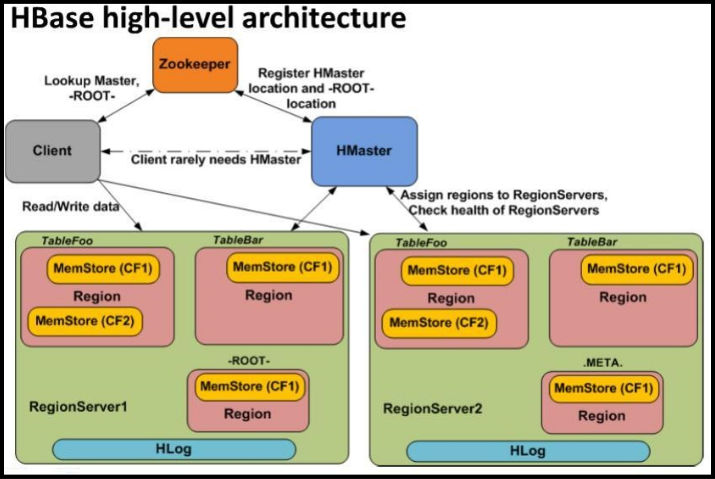
The HMaster has the features like controlling load balancing and failover to handle the load over nodes present in the cluster. When client wants to change any schema and to change any Meta data operations, HMaster takes responsibility for these operations.

**HRegions Servers**

It will perform the following functions in communication with HMaster and Zookeeper.

* + Hosting and managing regions.
  + Splitting regions automatically.
  + Handling read and writes requests.
  + Communicating with clients directly.

**HRegions**

It contains multiple stores, one for each column family. It consists of mainly two components, which are Memstore and Hfile. The Memstore holds in-memory modifications to the store. 

**Data Flow**

The client communicates in a bi-directional way with both Zoo keeper and HMaster. To read and write operations, it directly contacts with HRegion servers. HMaster assigns regions to region servers and in turn check the health status of region servers. In entire architecture, we have multiple regional servers. Hlog present in region servers will be used to store all the log files.

**Hbase Use Cases**

In this Hbase use case, we have to take some parameters into consideration like amount of data, speed at data flows and scalability. If the client wants to access a single row details from billions of records Hbase will be used. Hbase permits high compression rates due to few distinct values in the column.

Telecom Industry Use case - Storing billions of mobile call records and providing real time access to the call records and billing information to customers. Traditional storage/database systems couldn't scale to the loads and provide a cost effective solution.

The solution to this use case HBase is used to store billions of rows of call record details. 20TB of data is added monthly. To handle large amount of data in this use case Hbase gives the best solution in telecom industry.

**Q) HBase vs RDBMS:**

|  |  |
| --- | --- |
| HBASE | RDBMS |
| Schema-less in database. | Having fixed schema in database. |
| Column oriented database. | Row oriented data store. |
| Designed to store De-normalized data. | Designed to store Normalized data. |
| Wide and sparsely populated tables present in Hbase. | Contains thin tables in database. |
| Supports automatic partitioning. | Has no built in support for partitioning. |
| Well suited for OLAP systems. | Well suited for OLTP systems. |
| Read only relevant data from database. | To retrieve one row at a time and hence could read unnecessary data if only some of the data in a row is required. |
| Structured and semi structure data can be stored and processed using Hbase. | Structured data can be stored and processed using an RDBMS. |
| Enables aggregation over many rows and columns. | Aggregation is an expensive operation. |