Problem Statement:

● Explain the below concepts with an example in brief.

● Nosql Databases

NoSQL is an approach to databases that represents a shift away from traditional relational database management systems (RDBMS). To define NoSQL, it is helpful to start by describing SQL, which is a query language used by RDBMS. Relational databases rely on tables, columns, rows, or schemas to organize and retrieve data. In contrast, NoSQL databases do not rely on these structures and use more flexible data models. NoSQL can mean “not SQL” or “not only SQL

● Types of Nosql Databases

Several different varieties of NoSQL databases have been created to support specific needs and use cases. These fall into four main categories:

Key-value data stores

Document stores

Wide-column stores

Graph stores

● CAP Theorem

The CAP theorem is a tool used to makes system designers aware of the trade-offs while designing networked shared-data systems. CAP has influenced the design of many distributed data systems. It made designers aware of a wide range of tradeoffs to consider while designing distributed data systems. Over the years, the CAP theorem has been a **widely misunderstood tool** used to categorize databases.

* The theorem states that networked shared-data systems can only guarantee/strongly support two of the following three properties:
* Consistency - A guarantee that every node in a distributed cluster returns the same, most recent, successful write. Consistency refers to every client having the same view of the data. There are various types of consistency models. Consistency in CAP (used to prove the theorem) refers to linearizability or sequential consistency, a very strong form of consistency.
* Availability - Every non-failing node returns a response for all read and write requests in a reasonable amount of time. The key word here is every. To be available, every node on (either side of a network partition) must be able to respond in a reasonable amount of time.
* Partition Tolerant - The system continues to function and upholds its consistency guarantees in spite of network partitions. Network partitions are a fact of life. Distributed systems guaranteeing partition tolerance can gracefully recover from partitions once the partition heals.

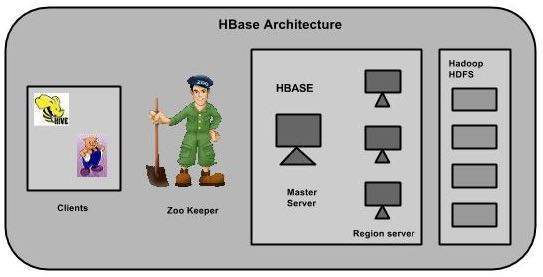
The CAP theorem categorizes systems into three categories:

* CP (Consistent and Partition Tolerant) - At first glance, the CP category is confusing, i.e., a system that is consistent and partition tolerant but never available. CP is referring to a category of systems where availability is sacrificed only in the case of a network partition.
* CA (Consistent and Available) - CA systems are consistent and available systems in the absence of any network partition. Often a single node's DB servers are categorized as CA systems. Single node DB servers do not need to deal with partition tolerance and are thus considered CA systems. The only hole in this theory is that single node DB systems are not a network of shared data systems and thus do not fall under the preview of CAP. [^11]
* AP (Available and Partition Tolerant) - These are systems that are available and partition tolerant but cannot guarantee consistency.

● HBase Architecture

In HBase, tables are split into regions and are served by the region servers. Regions are vertically divided by column families into “Stores”. Stores are saved as files in HDFS. Shown below is the architecture of HBase.

Note: The term ‘store’ is used for regions to explain the storage structure.



HBase has three major components: the client library, a master server, and region servers. Region servers can be added or removed as per requirement.

MasterServer

he 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.

## Zookeeper

* Zookeeper is an open-source project that provides services like maintaining configuration information, naming, providing distributed synchronization, etc.
* Zookeeper has ephemeral nodes representing different region servers. Master servers use these nodes to discover available servers.
* In addition to availability, the nodes are also used to track server failures or network partitions.
* Clients communicate with region servers via zookeeper.
* In pseudo and standalone modes, HBase itself will take care of zookeeper.

● HBase vs RDBMS

|  |  |
| --- | --- |
| HBase | RDBMS |
| Column-oriented | Row-Oriented(mostly) |
| Flexible Schema,add columns on the fly | Fixed schema. |
| Good with sparse tables | Not Optimized for sparse tables. |
| Joins using MR -not optimized | Optimized for joins. |
| Tight integration with MR | Not really |
| Horizontal scalable -just add hardware | Hard to shard and scale |
| Good for semi structured data as well as unstructured datan | Good for structured data |