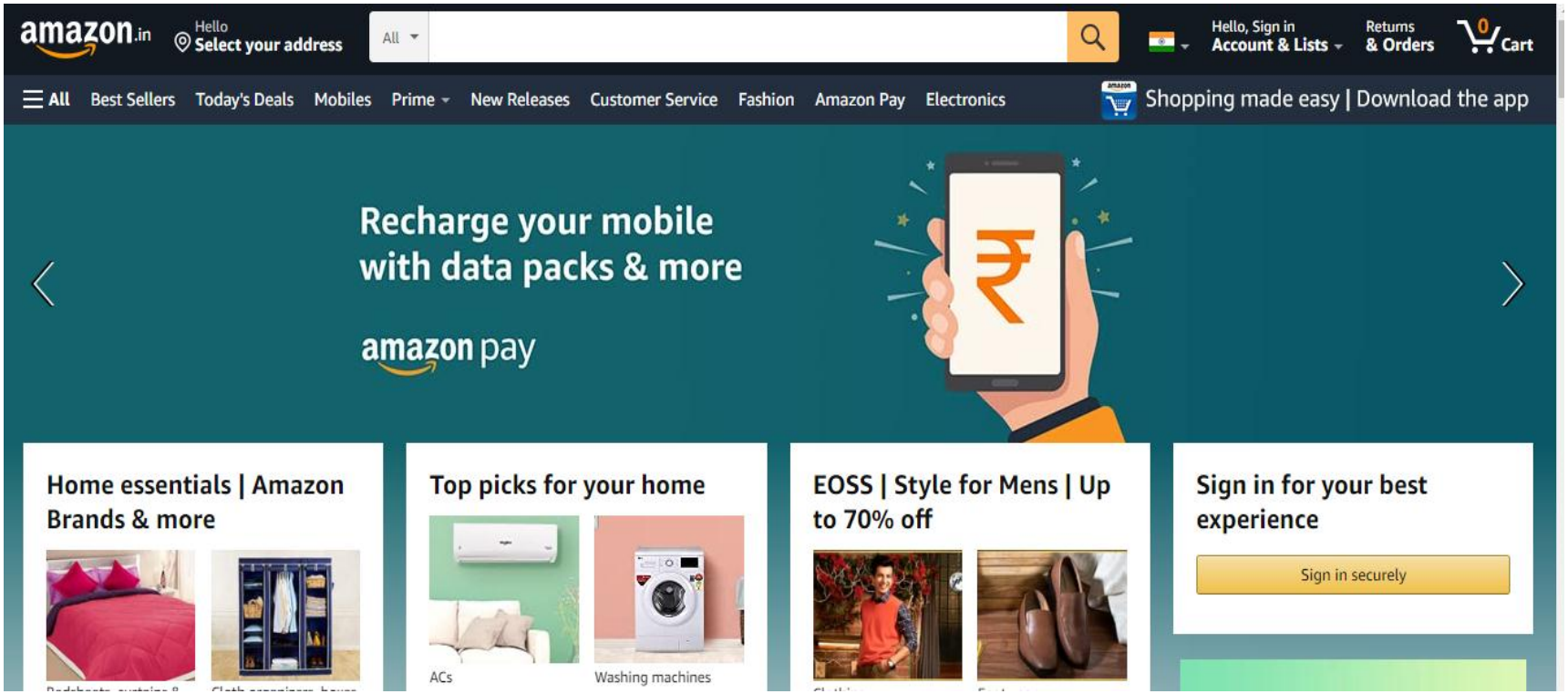
NoSQL and Apache HBase

Problems in real world

Problems in real world



Problems in real world

- Huge Data
- Fast Random access
- Structured / unstructured data
- Variable schema
- Need of compression
- Need of distributed data

How traditional System(RDBMS) SOLVE IT

Users
Id
Name
Gender
age

Connections
User_Id
Connection_id
type

Characteristics of Probable Solution

- Distributed Database
- Sorted Data
- Sparse Data store
- Automatic sharding
- No join
- Faster read and write

NoSQL Landscape

- **NoSQL database** stands for "Not Only SQL" or "Not SQL."
- **NoSQL** Database is a non-relational Data Management System, that does not require a fixed schema.
- It avoids joins, and is easy to scale.
- The major purpose of using a NoSQL database is for distributed data stores with humongous data storage needs.
- NoSQL is used for Big data and real-time web apps.

Features of NoSQL

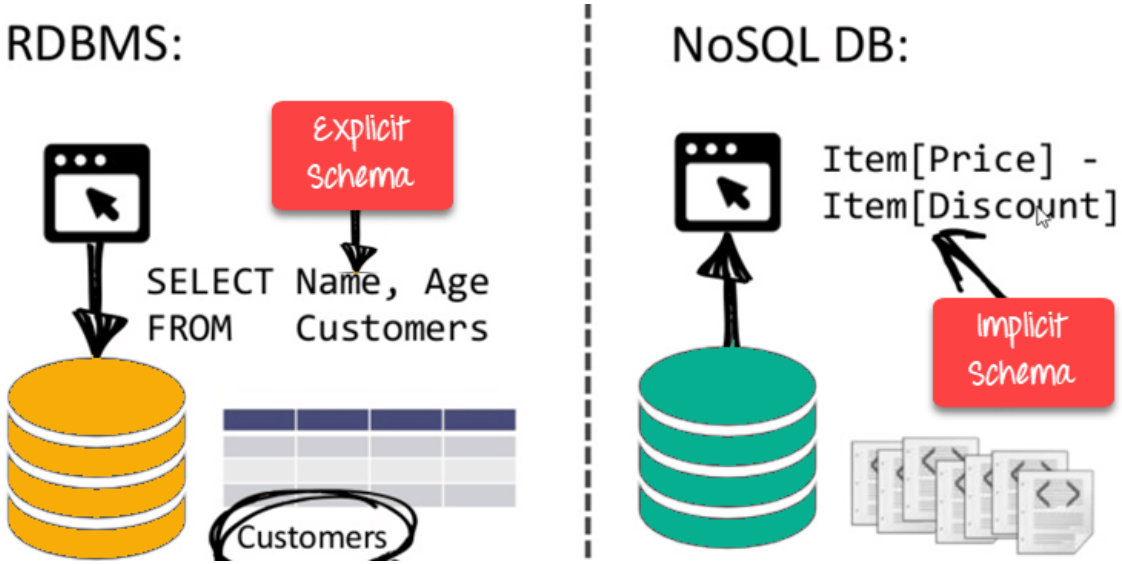
- **Non-relational**
  - NoSQL databases never follow the relational model
  - Never provide tables with flat fixed-column records
  - Work with self-contained aggregates or BLOBs
  - Doesn't require object-relational mapping and data normalization
  - No complex features like query languages, query planners,referential integrity joins, ACID

Features of NoSQL

- **Non-relational**
  - NoSQL databases never follow the relational model
  - Never provide tables with flat fixed-column records
  - Work with self-contained aggregates or BLOBs
  - Doesn't require object-relational mapping and data normalization
  - No complex features like query languages, query planners,referential integrity joins, ACID

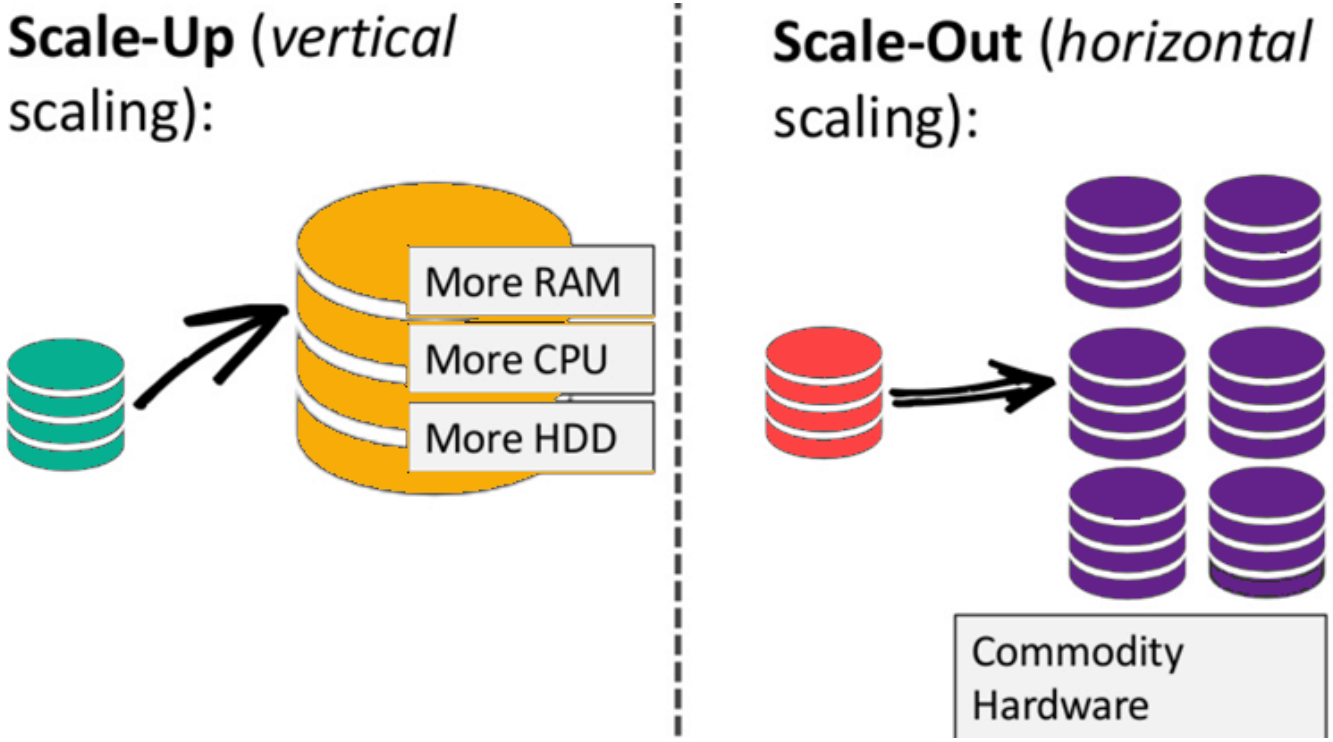
Features of NoSQL

- **Schema-free**
  - NoSQL databases are either schema-free or have relaxed schemas
  - Do not require any sort of definition of the schema of the data
  - Offers heterogeneous structures of data in the same domain



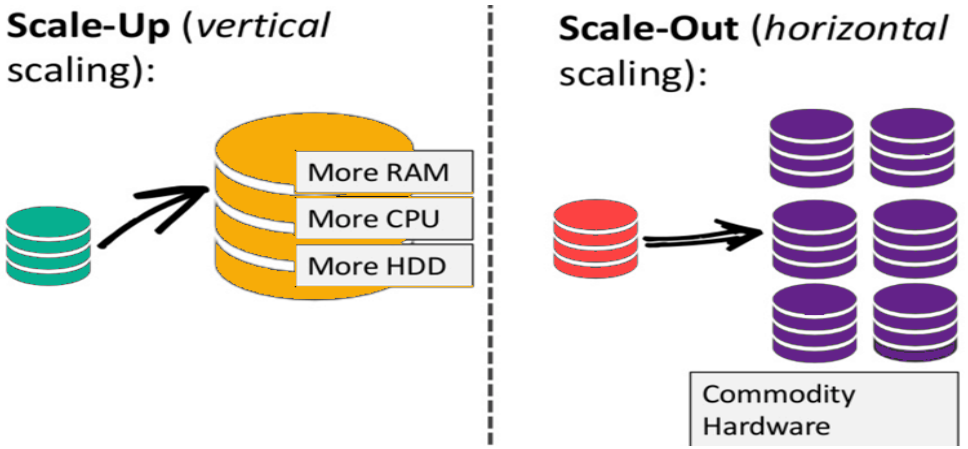
Features of NoSQL

- distribute database
- Distribute database load on multiple hosts whenever the load increases.
- This method is known as "scaling out."



Features of NoSQL

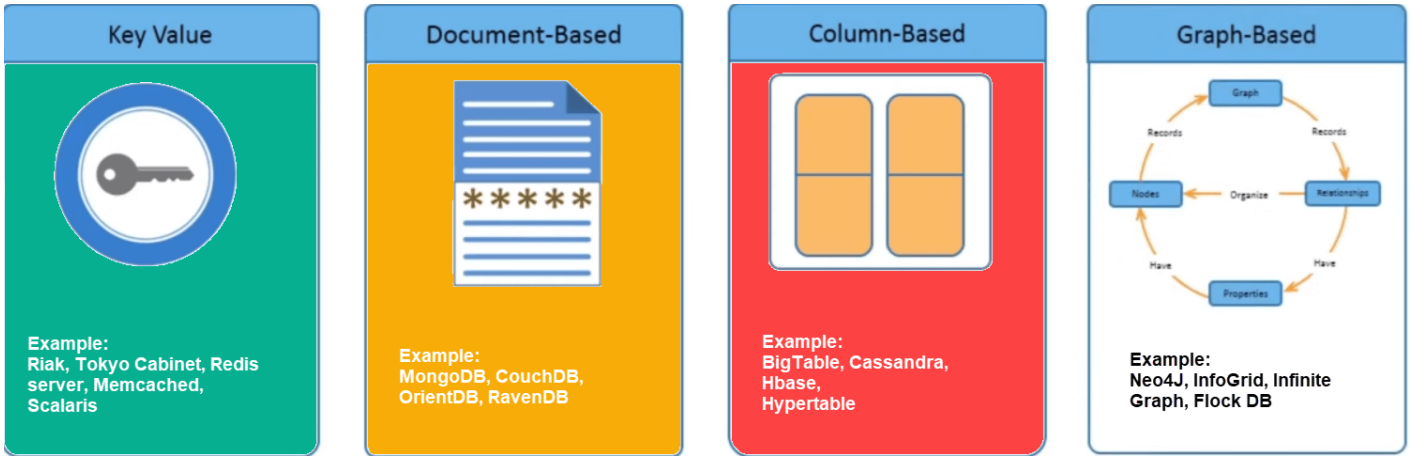
- Distribute database
- Multiple NoSQL databases can be executed in a distributed fashion
- Offers auto-scaling and fail-over capabilities
- Mostly no synchronous replication between distributed nodes Asynchronous Multi-Master Replication, peer-to-peer, HDFS Replication
- Only providing eventual consistency
- Shared Nothing Architecture. This enables less coordination and higher distribution.
-



NoSQL Landscape

The NoSQL databases can be classified as follows:

- **Key-Value Stores** – Dynamo (Amazon), Voldemort (LinkedIn), Citrusleaf, Membasae, Riak, Tokyo Cabinet, etc.
- **Wide Column Family Database** – BigTable(Google), Cassandra, HBase, Hypertable, etc.
- **Document Database** – CouchOne, MongoDB, Terrastore, OrientDB etc.
- **Graph Databases** – FlockDB (Twitter), AllegroGraph, DEX, InfoGRid, Neo4J, Sones, etc



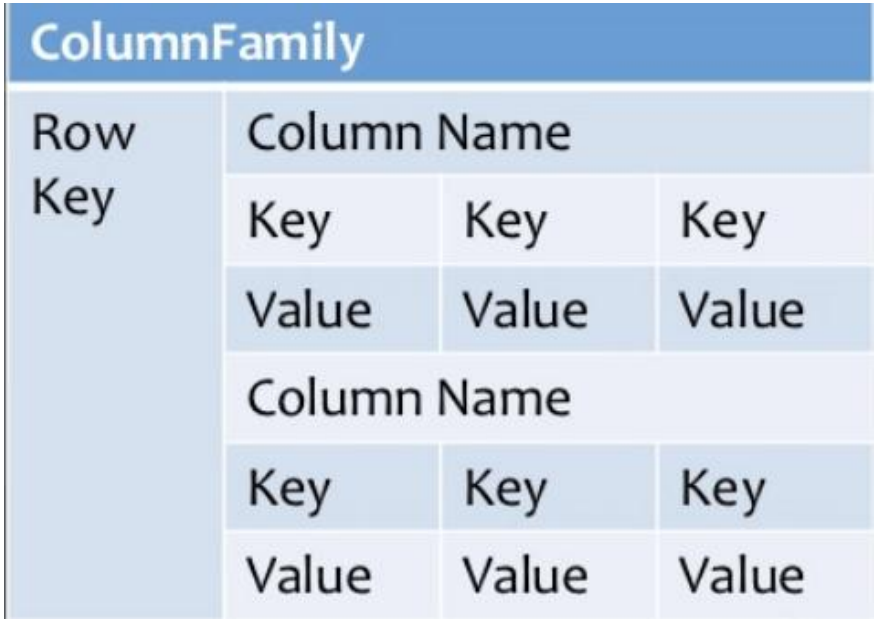
Key Value Pair Based

- Data is stored in key/value pairs.
- Key-value pair storage databases store data as a hash table where each key is unique, and the value can be a JSON, BLOB(Binary Large Objects), string, etc.
- For example, a key-value pair may contain a key like "Website" associated with a value like “linkedin".
- It is one of the most basic NoSQL database example. This kind of NoSQL database is used as a collection, dictionaries, associative arrays, etc. Key value stores help the developer to store schema-less data. They work best for shopping cart contents.

Key	Value
Name	Joe Bloggs
Age	42
Occupation	Stunt Double
Height	175cm
Weight	77kg

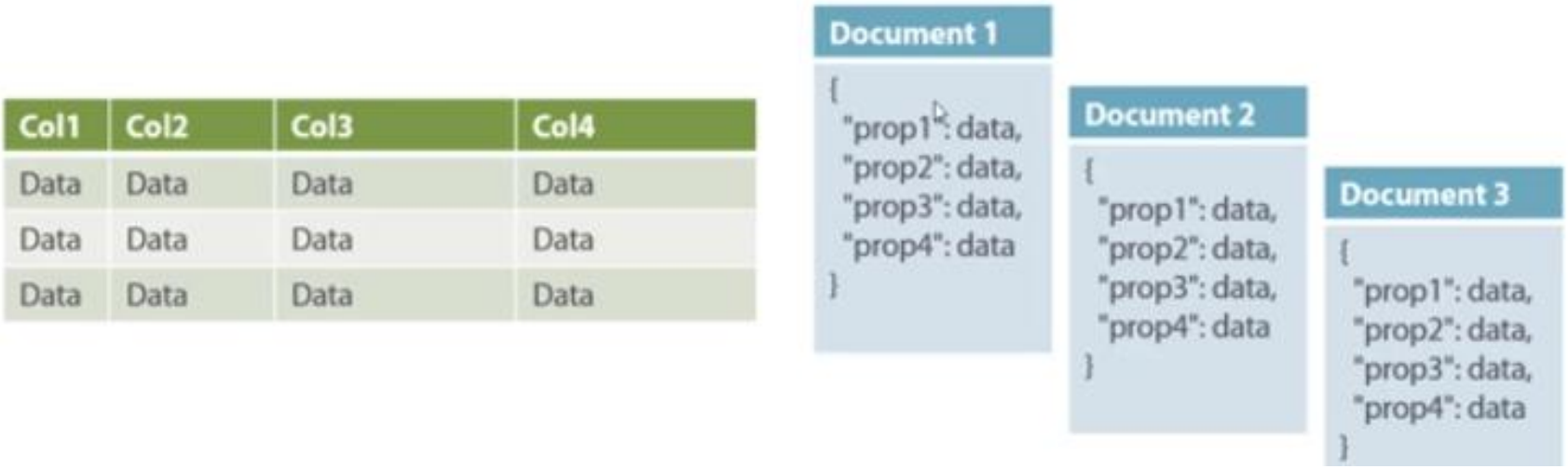
Column-based

- Column-oriented databases work on columns and are based on BigTable paper by Google. Every column is treated separately. Values of single column databases are stored contiguously.
- They deliver high performance on aggregation queries like SUM, COUNT, AVG, MIN etc. as the data is readily available in a column.
- Column-based NoSQL databases are widely used to manage data warehouses, business intelligence, CRM, Library card catalogs,



Document-Oriented based

- Document-Oriented NoSQL DB stores and retrieves data as a key value pair but the value part is stored as a document.
- The document is stored in JSON or XML formats. The value is understood by the DB and can be queried.
- The document type is mostly used for CMS systems, blogging platforms, real-time analytics & e-commerce applications

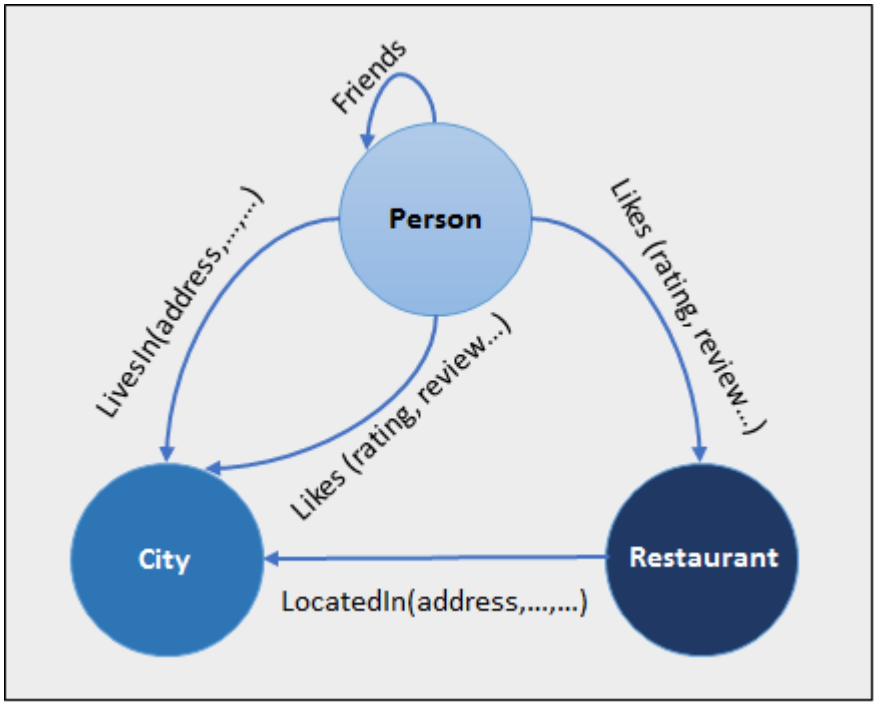


Graph based

- A graph type database stores entities as well the relations amongst those entities. The entity is stored as a node with the relationship as edges. An edge gives a relationship between nodes. Every node and edge has a unique identifier.

- Graph database is a multi-relational in nature. Traversing relationship is fast as they are already captured into the DB, and there is no need to calculate them.

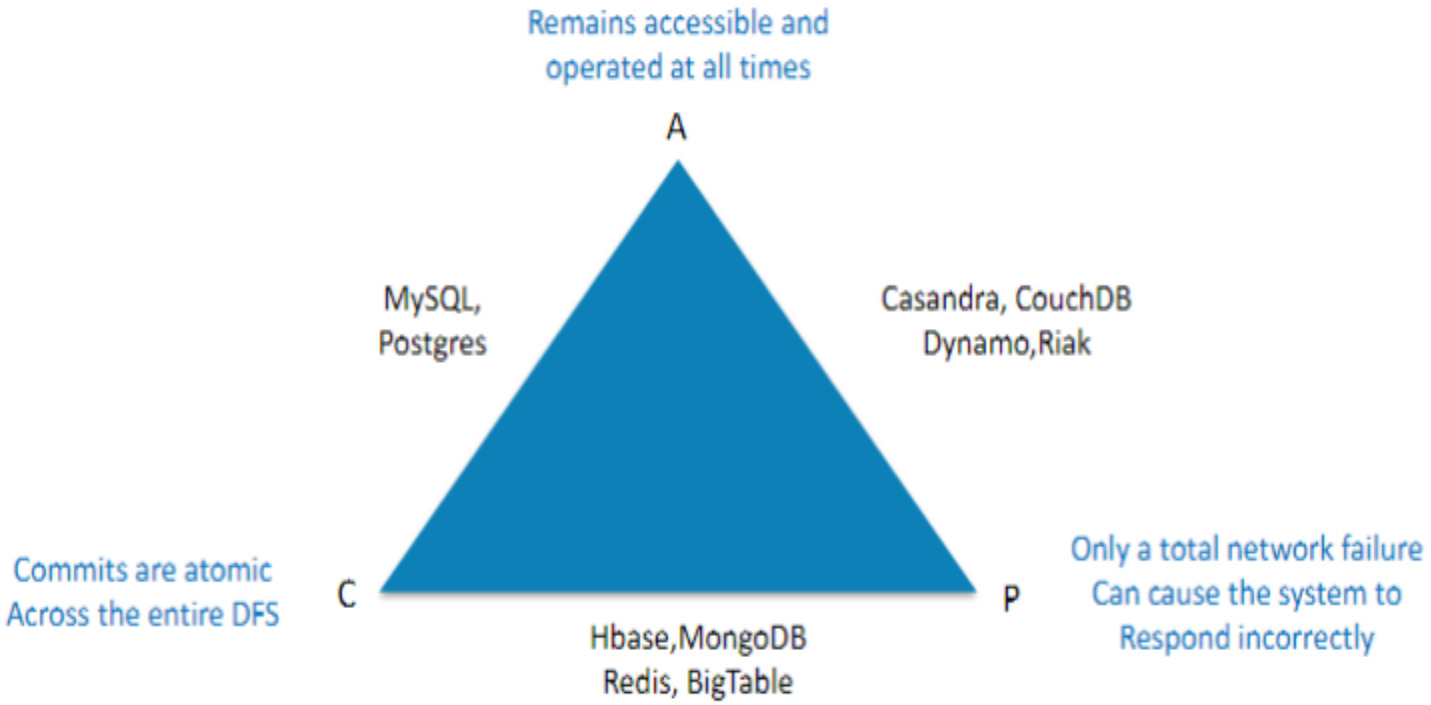
- Graph base database mostly used for social networks, logistics, spatial data.



CAP Theorem?

- CAP theorem is also called brewer's theorem.
- It states that is impossible for a distributed data store to offer more than two out of three guarantees
  - Consistency
  - Availability
  - Partition Tolerance

CAP Theorem

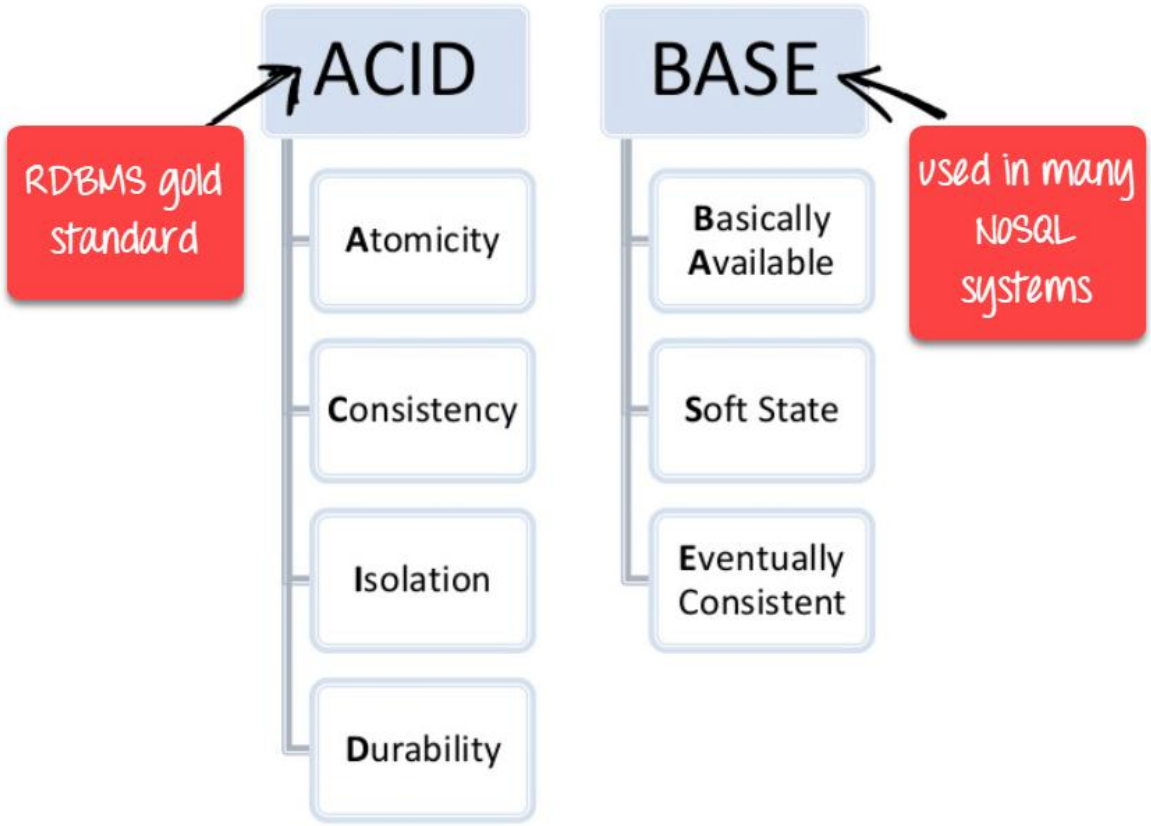

Remains accessible and
operated at all times
A
MySQL,
Postgres
Casandra, CouchDB
Dynamo,Riak
Commits are atomic
Across the entire DFS
C
P
Only a total network failure
Can cause the system to
Respond incorrectly
Hbase,MongoDB
Redis, BigTable

Eventual Consistency

- The term "eventual consistency" means to have copies of data on multiple machines to get high availability and scalability.
- Thus, changes made to any data item on one machine has to be propagated to other replicas.
- Data replication may not be instantaneous as some copies will be updated immediately while others in due course of time. These copies may be mutually, but in due course of time, they become consistent. Hence, the name eventual consistency.

Eventual Consistency

- **BASE**: Basically Availabl e, Soft state, Eventual consistency
- Basically, available means DB is available all the time as per CAP theorem
- Soft state means even without an input; the system state may change
- Eventual consistency means that the system will become consistent over time



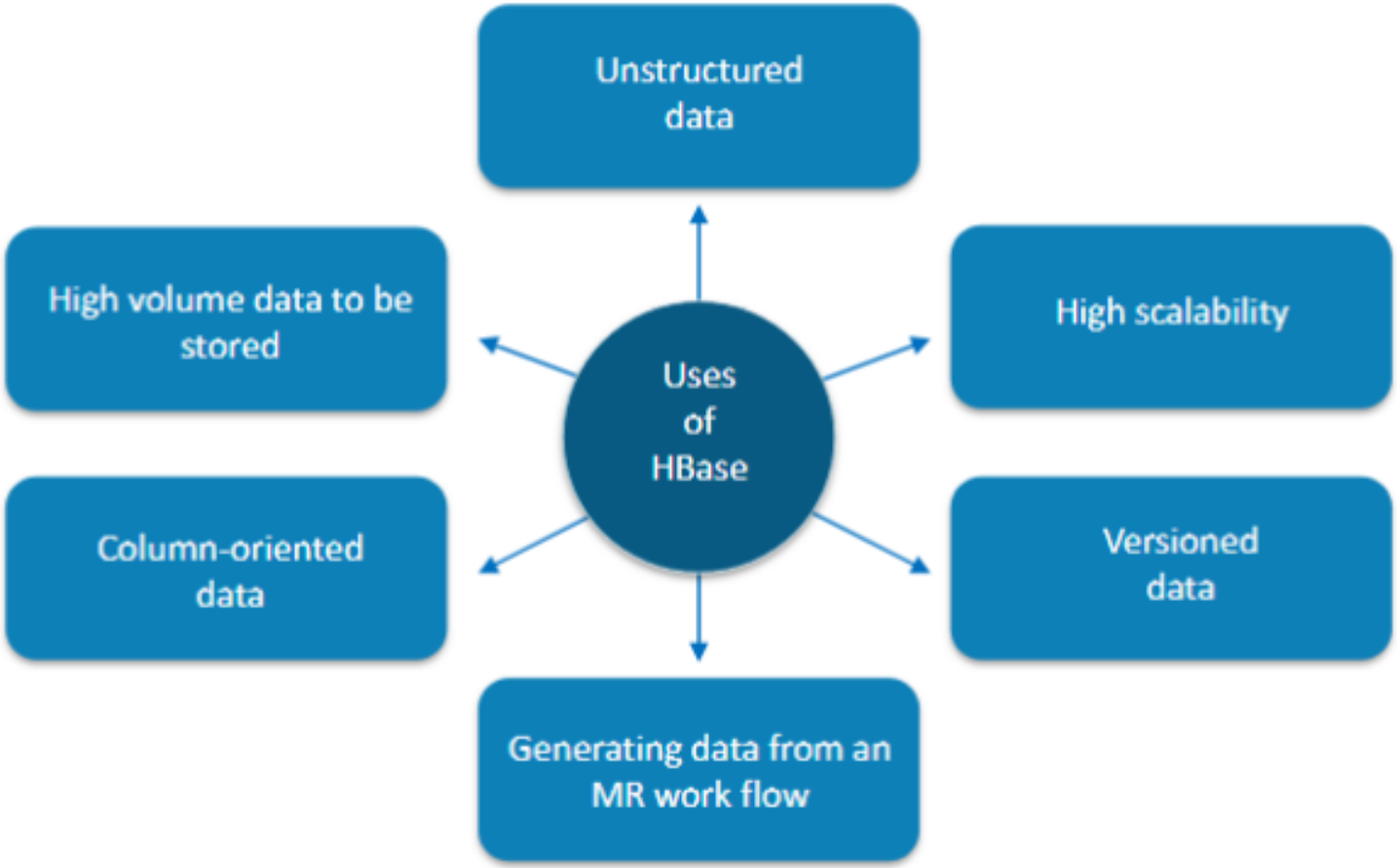
What is HBase

- Hbase is a NoSQL database that runs on top of HDFS
- HBase is a column-oriented data storage.
- Hbase is …
  - highly available
  - Fault tolerant
  - Very scalable
  - Can handle high throughput
  - Able to handle massive tables with ease
  - Well suited to sparse rows where the number of columns varies
  - An open source, Apache project

What differentiate HBase

- Hbase helps solve data access issues where random access is required
- Hbase scales easily, making it ideal for Big Data storage and processing
- Columns in an HBase table are defined dynamically, as required
- High Capacity
- High read and write throughput
- Large amount of stored data , but queries often access a small subset
- No penalty for sparse columns
- Scalable in-memory caching

When to use HBase



When not to use HBase

- When only few thousands/ million of rows
- L:acks RDBMS commands

HBase Use Cases

- Facebook : to power their message infrastructure
- Yahoo : to store document fingerprint for detecting near-duplication.
- Twitter : a number of application like people search rely on Hbase.
- Trend Micro : uses Hbase as a foundation for cloud scale storage for a variety of applications
- explorys : to store billions of anormalized clinical records
- ebay, pinterest, SalesForce
- StumbleUpon, FINRA etc

HBase Concept

- Node
- Cluster
- Master Node
- Slave/Worker node
- Daemon

HBase Concept

- Hbase stores data in tables
- Table data is stored on the HDFS
- Data is split into HDFS blocks and stored on multiple nodes

Table

- Hbase tables are comprised of rows, columns and columnfamily
- Every row has a row key for fast lookup
- Columns hold the data for the table
- Each column belong to a particular column family
- A table has one or more column families
- Hbase is essentially a distributed, sorted map
  - Distributed
  - Sorted Map

Rows

- Hbase tables are comprised of rows, columns and columnfamily
- Every row has a row key.
- A row key is analogous to a primary key in a traditional RDBMS
- Rows are stored sorted by row key to enable speedy retrieval

Columns

- Columns hold the data for the table
- Columns can be created on fly
- A column exists for a particular ro only if the row has data in that column
- Columns are the different attributes of a dataset.
- Each RowKey can have unlimited columns.

Column Family

- Column families are a combination of several columns.
- A single request to read a column family gives access to all the columns in that family, making it quicker and easier to read data
- Each column in an Hbase table belongs to a particular column family
- A table has one or more column families.
- Tuning and storage settings can be specific to each column family
- A column family can have any number of columns
- Contactinformation:fname

Column Qualifiers

- Column qualifiers are like column titles or attribute names in a normal table.
- Contactinformation:fname

Cell

- It is a row-column tuple that is identified using RowKey and column qualifiers
- Cells are arbitrary arrays of bytes

Timestamp

- Whenever a data is stored in the HBase data model, it is stored with a timestamp.

HBase Table : Conceptual View

Row Key	contactinfo		profilephoto
	fname	lname	image
jdupont	Jean	Dupont	
jsmith	John	Smith	<smith.jpg>
mrossi	Mario	Rossi	<mario.jpg>

Column Families

Column Names

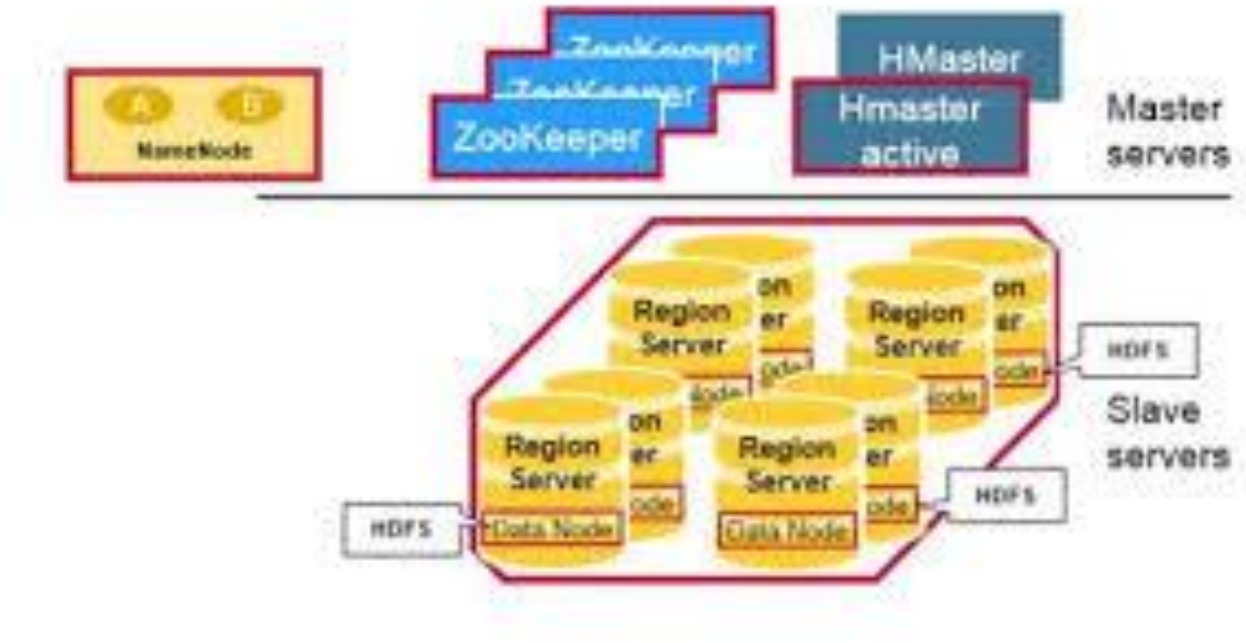
Rows

Hbase Vs RDBMS

HBase	RDBMS
Column-oriented	Row oriented
Flexible schema, added on fly	Fixed schema
Good for sparse data	Not good for sparse data
Not optimized for join	Optimized for join
Query centric	Data centric
Horizontal scale	Vertical scale
Good for semi-structured data	Good for structured data

HMaster

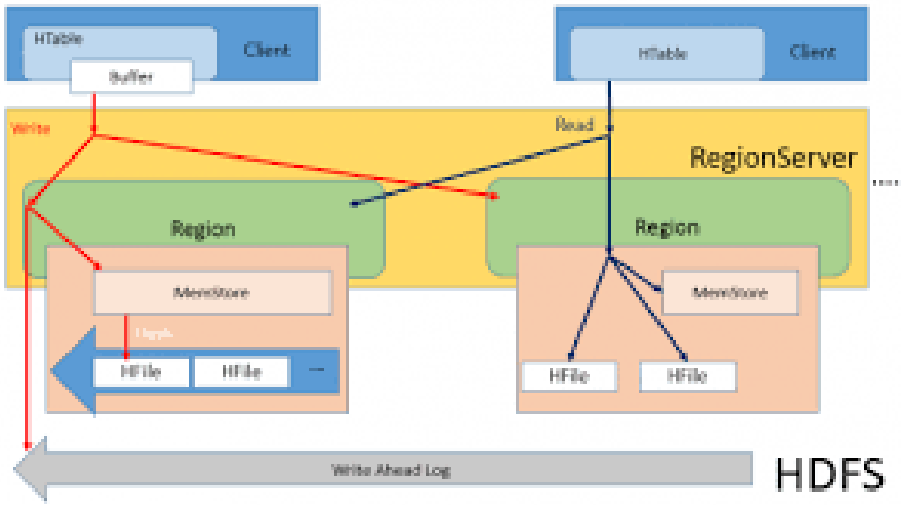
- The HBase architecture comprises three major components,
  - HMaster,
  - Region Server, and
  - ZooKeeper.
  - Hbase



HBase Architecture

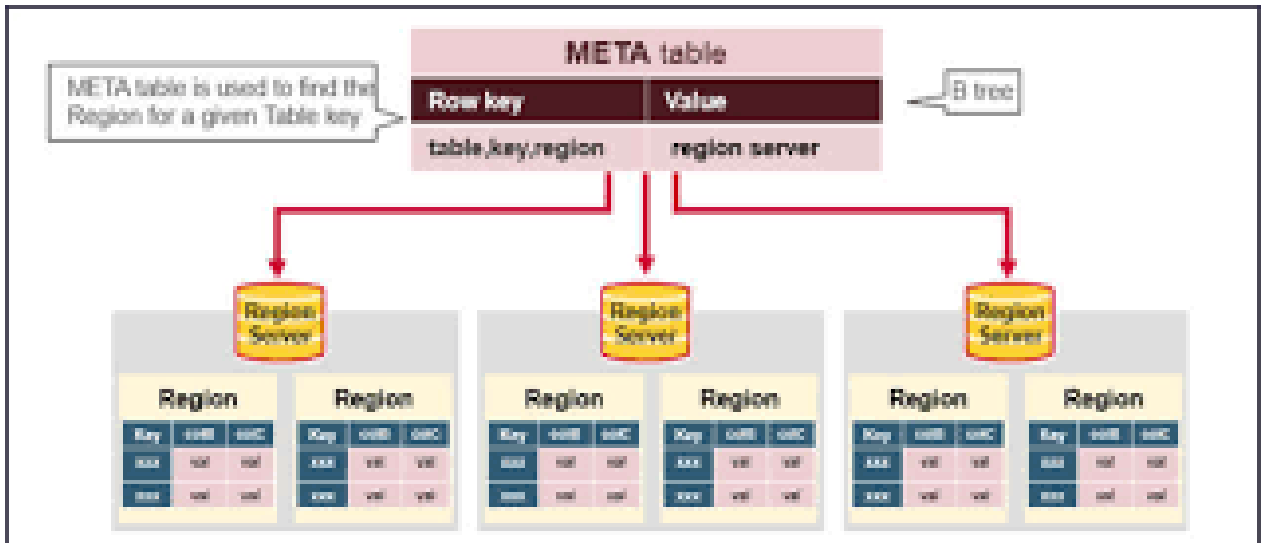
- HMaster operates similar to its name. It is the master that assigns regions to Region Server (slave). HBase architecture uses an Auto Sharding process to maintain data. In this process, whenever an HBase table becomes too long, it is distributed by the system with the help of HMaster. Some of the typical responsibilities of HMaster includes:
- Control the failover
- Manage the Region Server and Hadoop cluster
- Handle the DDL operations such as creating and deleting tables

- Manage changes in metadata operations
- Manage and assign regions to Region Servers
- Accept requests and sends it to the relevant Region Server



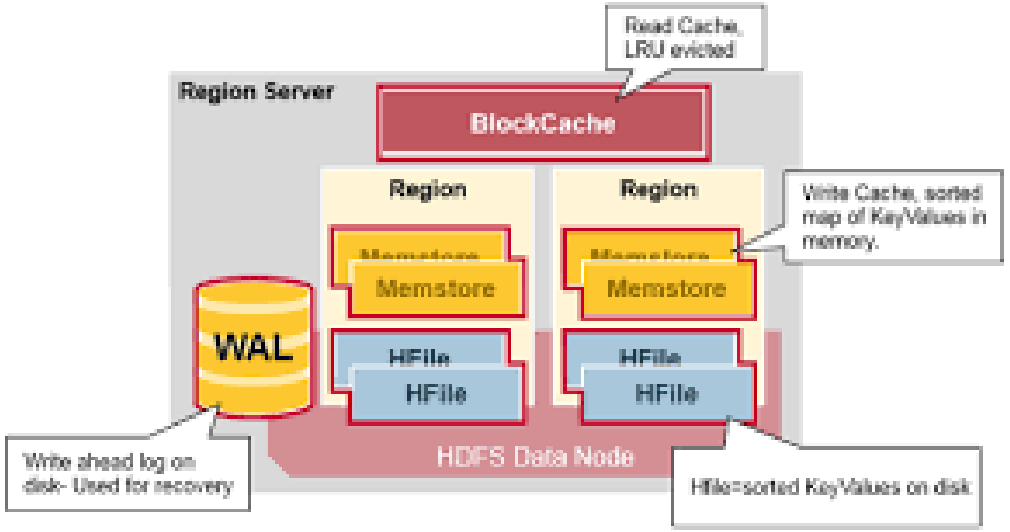
Region Server

- Region Servers are the end nodes that handle all user requests. Several regions are combined within a single Region Server.
- These regions contain all the rows between specified keys.
- Handling user requests is a complex task to execute, and hence
- Region Servers are further divided into four different components to make managing requests seamless.



Region Server

- **Write-Ahead Log (WAL):** WAL is attached to every Region Server and stores sort of temporary data that is not yet committed to the drive.
- **Block Cache**: It is a read request cache; all the recently read data is stored in block cache. Data that is not used often is automatically removed from the stock when it is full.
- **MemStore**: It is a write cache responsible for storing data not written to the disk yet.
- **HFile**: The HFile stores all the actual data after the commitment.



ZooKeeper

- ZooKeeper acts as the bridge across the communication of the HBase architecture.
- It is responsible for keeping track of all the Region Servers and the regions that are within them.
- Monitoring which Region Servers and HMaster are active and which have failed is also a part of ZooKeeper’s duties.
- When it finds that a Server Region has failed, it triggers the HMaster to take necessary actions.
- On the other hand, if the HMaster itself fails, it triggers the inactive HMaster that becomes active after the alert.
- Every user and even the HMaster need to go through ZooKeeper to access Region Servers and the data within.

ZooKeeper

- ZooKeeper stores a.Meta file, which contains a list of all the Region Servers. ZooKeeper's responsibilities include:
- Establishing communication across the Hadoop cluster
- Maintaining configuration information
- Tracking Region Server and HMaster failure
- Maintaining Region Server information