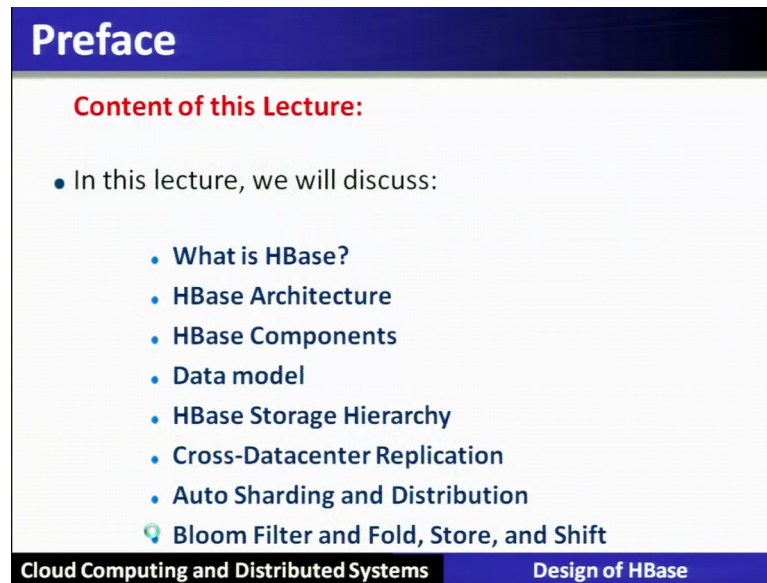


Cloud Computing and Distributed Systems
Dr. Rajiv Misra
Department of Computer Science and Engineering
Indian Institute of Technology, Patna

Lecture – 17
Design of HBase

(Refer Slide Time: 00:16)



Preface

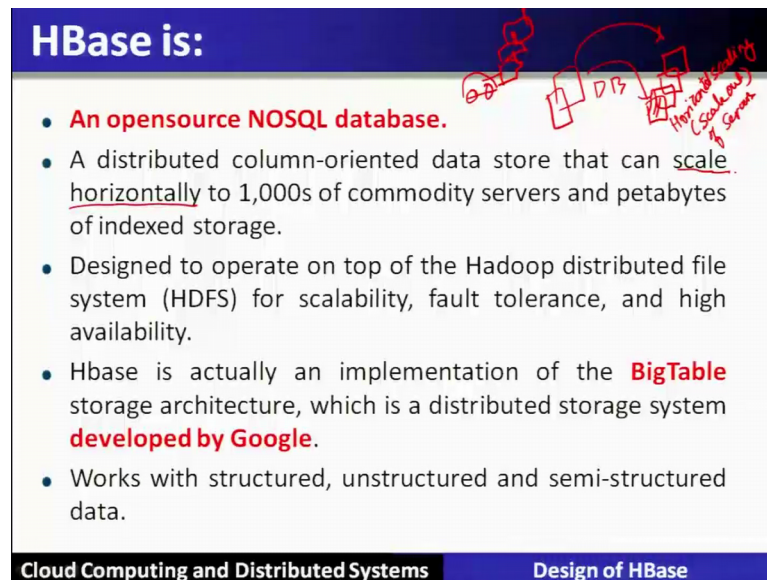
Content of this Lecture:

- In this lecture, we will discuss:
 - What is HBase?
 - HBase Architecture
 - HBase Components
 - Data model
 - HBase Storage Hierarchy
 - Cross-Datacenter Replication
 - Auto Sharding and Distribution
 - Bloom Filter and Fold, Store, and Shift

Cloud Computing and Distributed Systems Design of HBase

Design of a HBase: Preface; content of this lecture. In this lecture we will cover about HBase, its architecture, the components. The data model which is used in HBase and HBase storage hierarchy, Cross-Datacentre Replication, Auto Sharding and Distribution, Bloom Filter and its use in Fold, Store and Shift.

(Refer Slide Time: 00:41)



HBase is:

- **An opensource NOSQL database.**
- A distributed column-oriented data store that can scale horizontally to 1,000s of commodity servers and petabytes of indexed storage.
- Designed to operate on top of the Hadoop distributed file system (HDFS) for scalability, fault tolerance, and high availability.
- Hbase is actually an implementation of the **BigTable** storage architecture, which is a distributed storage system **developed by Google**.
- Works with structured, unstructured and semi-structured data.

Cloud Computing and Distributed Systems Design of HBase

Handwritten notes: @ 00:41, HDFS, Horizontal Scaling (Scale out), 2000s

Let us see what HBase is. HBase is an open source NOSQL database. This particular HBase is a distributed column-oriented data store, that can scale horizontally to 1000s or 2000s of commodity servers and petabytes of indexed storage. Let us understand one term: what do you mean by horizontal scaling. By the meaning of horizontal scaling is that, if let us say this is the database, and this is stored on a particular server. As it grows in size, there are 2 methods, one is called scale up that is scale vertically, the other is called horizontally. Here in the horizontal scaling, we keep on adding the servers as we grow in the size. So, this is called or it is also called scale out of the servers.

So, we keep on adding many servers and so on. So, this particular analogy we can take from a chariot, which is driven by the horse. So, we have to keep on adding the number of horses. That is called horizontal scaling. So, this is a new kind of technology; that means, in today's world we have to scale horizontally so that as we require we can add more number of commodity servers. Hence, the scaling is possible easily in a cost-effective manner. So, this particular HBase uses the scale horizontally or horizontal scaling method so that 1000s of commodity servers and more we can add to it so that it can basically store. The column-oriented data store which is distributed across all these servers. We will be looking up how HBase will do this; that means, it will maintain a distributed database on the cloud.

So, HBase is designed to operate on top of Hadoop distributed file system. So, it assumes that Hadoop distributed file system is available, and it uses HDFS for it is all storage purposes and that HDFS will allow its advantages of scalability fault tolerance and high availability to the HBase. So, HBase is actually an implementation of big table, which is an storage architecture developed by the Google. And based on this HBase is designed and made open source using Apache. So, it works with the structured unstructured and semi structured data.

(Refer Slide Time: 04:23)

HBase

- **Google's BigTable** was first "blob-based" storage system
- **Yahoo!** Open-sourced it → HBase
- Major Apache project today
- **Facebook** uses HBase internally
- **API functions**
 - Get/Put(row)
 - Scan(row range, filter) – range queries
 - MultiPut
- Unlike Cassandra, HBase prefers consistency (over availability)

Handwritten notes: BASE/ACID, Eventual Consistency

Cloud Computing and Distributed Systems Design of HBase

So, HBase is based on Google's big table, that was the first block based storage system. Yahoo open sourced it and call it as HBase. Now, major apache project today uses HBase. For example, Facebook users HBase internally in one of it is stack. The API which are supported in HBase are quite easy and simple to be used. For example, API such as get and put, the row wise in a table of a database is being supported by get and put.

Similarly, to support the range queries that is range of the rows, let us say the ranges from a to d will also be supported using a can. So, it will fetch all rows which falls in that range to support the range queries in HBase. Multi put option is also available through API's. Here we will see that we have also already seen a Cassandra in a NOSQL store. So, unlike Cassandra HBase prefers consistency over availability.

Now, regions vertically divided by the column families in to the stores. We will see what do you mean by the column families. So, for the time being we just have to understand that what is basically the basic unit of a storage that will be handled through the HBase and stored on the disk. So, that is called column families. So, column families are nothing but the column names; which is having more than one column attributes that is called column families and that will be stored.

So, this is stores and these stores are saved as the files on HDFS. So, these are nothing but the stores or it is also called HFiles which are store on HDFS. We will see in more detail about this part. HBase utilizes the Zookeeper for distributed coordination; that means, the Zookeeper will ensure that all these data nodes are which are basically called as H which are called as h stores are alive and also the master is also available at all points of time.

(Refer Slide Time: 10:16)

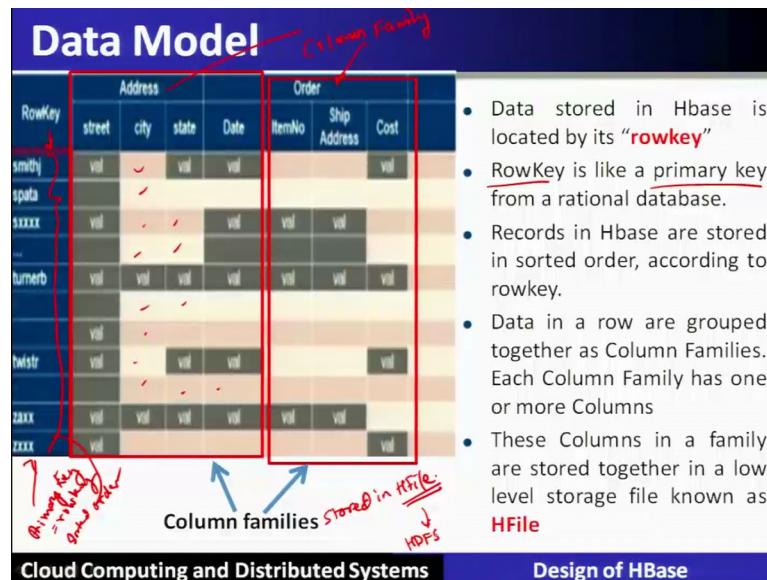
HBase Components

- **Client:**
Finds RegionServers that are serving particular row range of interest
- **Hmaster:**
Monitoring all RegionServer instances in the cluster
- **Regions:**
Basic element of availability and distribution for tables
- **RegionServer:**
Serving and managing regions
In a distributed cluster, a RegionServer runs on a DataNode

Cloud Computing and Distributed Systems Design of HBase

So, HBase components, the client first find the region servers, that are serving particularly row range of interest, then HBase master which is called h master will monitor all the regions server instances in the cluster. So, master is responsible to manage the entire cluster which is used in HBase. Then comes the region servers, this basic element of eligibility and distribution of the table. So, these region servers are serving and managing these regions in a distributed cluster a region server runs on a data node which is maintained by HDFS.

(Refer Slide Time: 11:07)

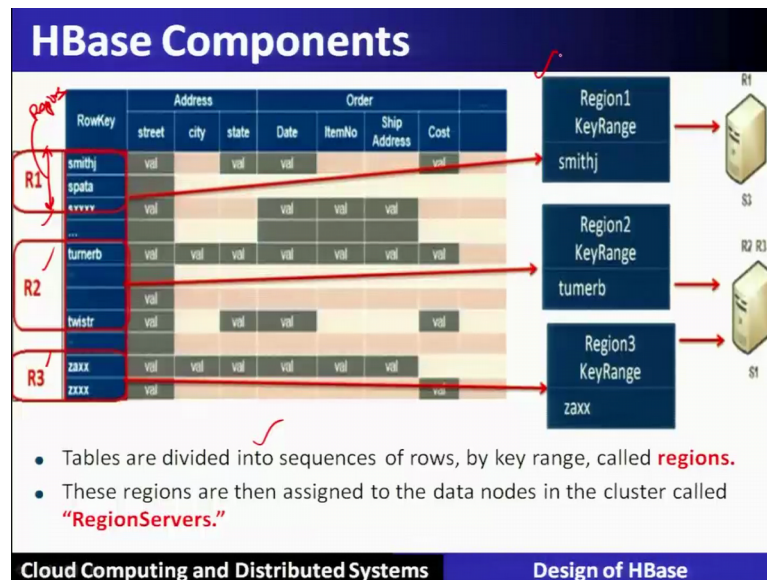


Let us see the data model in more details to understand all these parts, which are managed by HBase. So, data is stored in HBase is located by it is rowkey so, this is the rowkeys. Rowkeys is a primary key; this term is taken from the rational database. So, row key is just like a primary key now records in HBase are stored in the sorted order according to the rookeys. So, all these rockeys are in the sorted order.

The data in a row are grouped together as the column families. These columns in the family are stored together in a low level storage file known as HFiles. Here the example of column families, are the address and the order, which are shown here in this particular box as Column Family. So, address has various attributes such as street city state date. Similarly order Column Family has various attributes such as item number ship address cost etcetera.

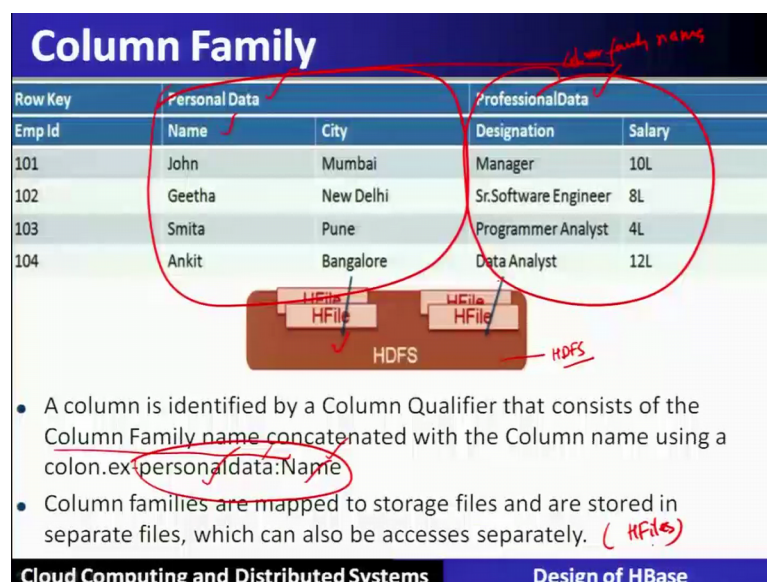
These particular attributes do not have to be essential having the values. They may be null also. So, these columns in a family together they are stored together in a low level storage file which is called HFile and which is maintained by or which is stored by HDFS.

(Refer Slide Time: 13:30)



Here same thing is also shown in this particular figure. That tables are divided into the sequence of rows by the key range and they are called the regions. So, this key range here is called the region. So, this is region 1, 2 and 3. So, each region is stored in the server which is called the region server. These regions are then assign to the data nodes in a cluster which is called the region servers.

(Refer Slide Time: 14:06)



Now as for as column families are concerned, a column is identified by a column qualifier, that consists of the Column Family name for example, here the personal data is

the Column Family name. And professional data is also a Column Family names. So, this Column Family name is concatenated with the column name using colon. And will identify the column qualifier.

For example, if you want to check the name; which is the column qualifier, then it has to be starting with the personal data followed by the name that is personal data. Colon name is an example over here. That is how the column is identified by the column qualifier. So, column qualifier name if you want to check, then it will be starting with the Column Family name and that qualifier that is the name. So, column families are map to the storage files and are stored in the separate files which can be accessed separately sample.

So, for example, here this particular Column Family personal data will be stored in the form of HFile. Similarly, is the professional data is another Column Family this also will be stored in the HFiles which is in the HDFS. So, the column families are the HFiles are map to the storage files. And are stored in a separate files which are access separately that is called HFiles.

(Refer Slide Time: 16:19)

Cell in HBase Table

Row Key	Column Family	Column Qualifier	Timestamp	Value
John	PersonalData	City	123456790123	Mumbai

KEY

VALUE

- Data is stored in **HBASE tables Cells**.
- **Cell is a combination of row, column family, column qualifier and contains a value and a timestamp**
- The key consists of the row key, column name, and timestamp.
- The entire cell, with the added structural information, is called **Key Value**.

Cloud Computing and Distributed Systems Design of HBase

Cell in a HBase table. So, data is stored in HBase tables cells. Cell is a combination of row, Column Family column qualifier, and it contains a value and a timestamp. For sample let us say start with the row key, then comes the Column Family, then comes the column qualifier. And this column qualifier will have a timestamp and it is value. The key in HBase is comprised off row key Column Family column qualifier and timestamp

and this is called a key. And this column qualifier will have its own value. For example, this particular zone with their personal data. It is column qualifier city and that city value is Mumbai. So, this is the data which is being stored in the form of a cell.

So, the key consists of the row key, column name and time stamp that we have already seen here in this example. The entire cell with the added structure information is called the key value. So, this is the key and this qualifier will get the value from here in the cell. And this is how the data is stored.

(Refer Slide Time: 17:55)

HBase Data Model

- **Table:** Hbase organizes data into tables. Table names are Strings and composed of characters that are safe for use in a file system path.
- **Row:** Within a table, data is stored according to its row. Rows are identified uniquely by their row key. Row keys do not have a data type and are always treated as a byte[] (byte array).
- **Column Family:** Data within a row is grouped by column family. Every row in a table has the same column families, although a row need not store data in all its families. Column families are Strings and composed of characters that are safe for use in a file system path.

Cloud Computing and Distributed Systems

Design of HBase

So, HBase data model comprises of the table, HBase organizes data into the table names are the strings, and composed of characters that are safe for use in the file system path. Rows within the table; the data is stored according to the it is rows, rows identified uniquely by the rowkey. Row keys do not have the data type and are always treated as the byte array. Column family's data within a row is grouped by the Column Family. Every row in the table has some column families although a row need not store the data in all it is column families. In that case it will be null. So, column families are the strings and composed of characters that are safe for use in file system path.

(Refer Slide Time: 18:49)

HBase Data Model

- **Column Qualifier:** Data within a column family is addressed via its column qualifier, or simply , column, Column qualifiers need not be specified in advance. Column qualifiers need not be consistent between rows. Like row keys, column qualifiers do not have a data type and are always treated as a byte[].
- **Cell:** A combination of row key, column family, and column qualifier uniquely identifies a cell. The data stored in a cell is referred to as that cell's value.

Cloud Computing and Distributed Systems Design of HBase

Qualifier; this is the unit where the data is stored on it is also called a value. So, data within the Column Family is addressed via the qualifier or simply the column or a column qualified need not be specified in advance, column qualifier need not be consistent between the rows. Like row keys, column qualifier do not have the data type and are treated as by array. Cell, is a combination of row key Column Family and column qualifier uniquely identify the cell. Data is stored in a cell is referred to the cells value.

(Refer Slide Time: 19:29)

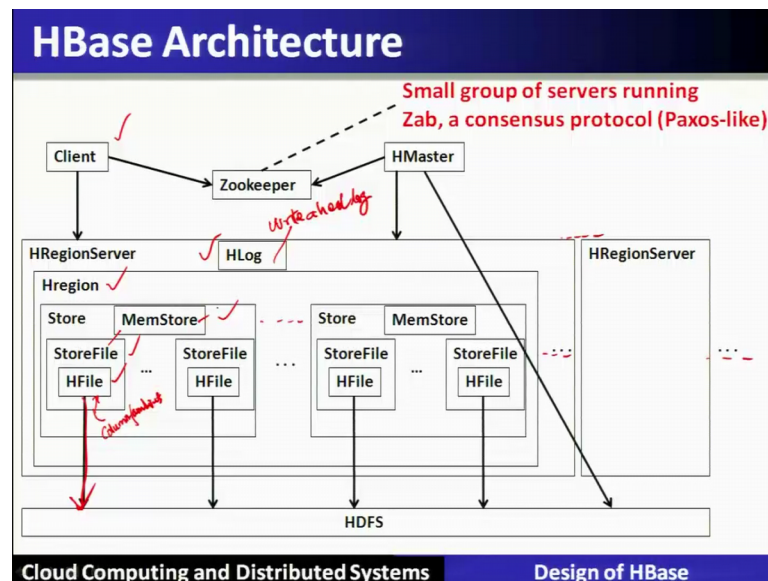
HBase Data Model

- **Timestamp:** Values within a cell are versioned. Versions are identified by their version number, which by default is the timestamp is used.
- If the timestamp is not specified for a read, the latest one is returned. The number of cell value versions retained by Hbase is configured for each column family. The default number of cell versions is three.

Cloud Computing and Distributed Systems Design of HBase

Timestamp this is also very important notion in supported in HBase. So, the values within a cell are versioned. Is versions are identified by their version number which by default is the timestamp is used. If the timestamp is not specified for the read, the latest one is returned the number of cell values versions retained by HBase is configured for each Column Family. The default number of cell versions is 3.

(Refer Slide Time: 20:01)

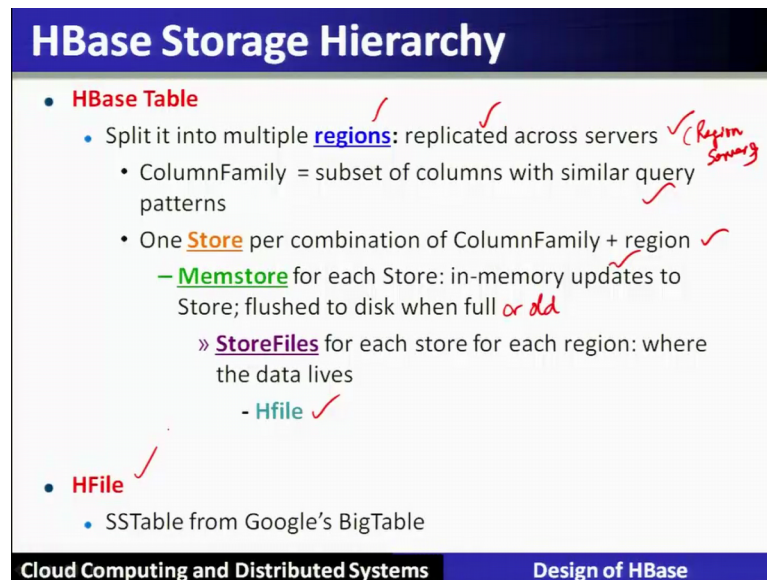


Let us see the HBase architecture in further details. So, HBase has an entity which is called a client. Now client will access the HBase through HRegionServers, which are shown here in this block. There may be more than one HRegionServers HRegionServers has within it called HLog. So, HLog is a write ahead log; that means, before the ride starts that operation is to be logged in HLog, that is called write ahead log. So, HRegionServer has HLog.

So, every HRegionServer maintains several HRegions now each HRegion will in turn maintain different stores. Each store in turn has a mem table and stores the StoreFile. StoreFile in turn has the HFile and HFile is stored in HDFS. We have already seen what is the Hfile so, it is nothing but it will store the column families.

. So, several column families together will comprise the StoreFiles. And these StoreFiles when they are accessed from the desk they will be stored in memory, that is called MemStore. So, this MemStore means as the data is accessed, it will be remain in memory and hence the accesses will be fast thereafter.

(Refer Slide Time: 22:07)



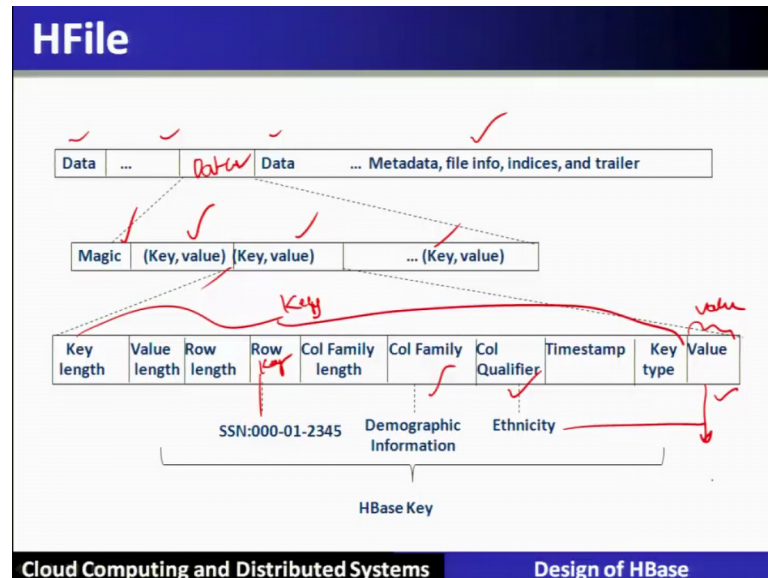
Now let us see this HBase storage hierarchy, HBase table we is split into the multiple regions, that we have already seen these regions are nothing but the range of rows, they are called regions. This is splitted rows which are called regions are now replicated across different servers and they are called as. So, the table is stored after the splitting into the rain server. Why this way the table is stored? If the table is very big which cannot be stored in one server multiple servers are used to store the tables so that not only big table, but it can be so efficiently if it is stored in multiple region servers.

This also regions are replicated across the servers. So, replication if we see as far as HDFS is concerned by default the replication number is 3. So, these are replicated so that it will be accessible very efficiently. So, the column families are nothing but a subset of the columns with similar query pattern. So, column families will support the query pattern and they are together stored. So, one store per combination Column Family and the region is there in HBase.

So, there is also a concept we have seen about the MemStore for each store. So, MemStore for each store is nothing but an in memory updates to the store. So, that will be retained, and when the disk is full or this become old then only it will be flush to the disk. This ensures the number of iOS are to be reduced in this particular manner by using the MemStore, and also the accesses are going to be fast. Now comes the StoreFile. StoreFiles for each store for each region where the data lives is maintained or is stored in

the form of Hfile. Now H it is file you know that it is stored into HDFS. So, as file we have seen the SSTable, use this SSTable is also there in Google big table design.

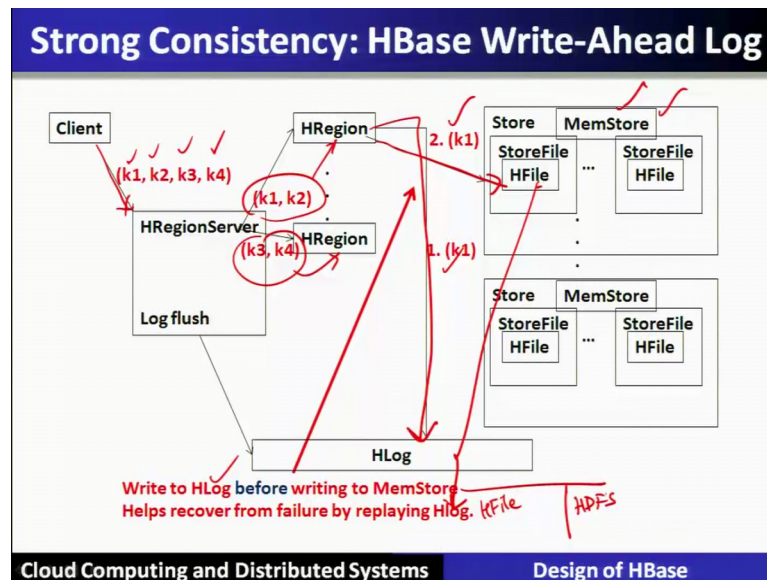
(Refer Slide Time: 25:22)



Let us go and see the details of HFile which is stored into HDFS and which is the unit of the data which is stored in HBase. HFile comprises of data, data and large amount of data followed by the meta data, file information indices and trailers. So, let us see what is there inside a data which is stored in HFile. So, data file is starting with the magic number which will uniquely identify a particular data followed by the key value pairs, several key value pairs. Let us see: what is there inside key value pair which we have already seen, but for the sake of completeness let us understand it.

. So, key comprises of; so, it has 2 parts key part and the value part. So, key comprises of key length, value length, row and ColumnFamily length, ColumnFamily, column qualifier, timestamp and the key type, then followed by it is value. The example which we have seen earlier here also we can see that the row key is nothing but a primary key over here. And then followed by a ColumnFamily name and then column qualifier: that means, the ethnicity of a particular person here in this example and this value will be about that particular name. So, this is called HBase key. And it is corresponding value which is being stored into the data part of HFile.

(Refer Slide Time: 27:31)



HBase supports strong consistency. For that it uses HLog which is called a write ahead log. Let us understand this aspect of HBase. Now client contacts to the HRegionServer for the keys k 1, k 2, k 3 and 4. Now, this HRegionServer will divide this particular keys as per their availability in different servers. So, k 1, k 2 is maintained here in a HRegion, and k 3 and k 4 is maintained another HRegion. So, as far as h is concerned for the keys k 1 what it will do? It will first go and write into HLog.

This is the first operation it will do so; that means, write to HLog before writing to the MemStore helps to recover from the failure by replaying an HLog. So, it will be first log and then it will send for writing that is the second step. Second step is to write to the MemStore. So, this values are written for the write operation into the MemStore. And once the MemStore is full, the disk is full or it is old then only that corresponding file will be placed into HFile which is there in HDFS.

(Refer Slide Time: 29:32)

Log Replay

- **After recovery from failure, or upon bootup (HRegionServer/HMaster)**
 - Replay any stale logs (use timestamps to find out where the database is with respect to the logs)
 - Replay: add edits to the MemStore

Cloud Computing and Distributed Systems Design of HBase

So, log replay, after the recovery from the failure and upon boot up the HRegionServer with the master they will replay any stale logs. According to the timestamp to find out where the database is with respect to the logs. So, using replay it will be constant after the failure. That particular entire operations which were not complete which were not persistent. So, replay will add edits to the MemStore also.

(Refer Slide Time: 30:06)

Cross-Datacenter Replication

- Single **"Master"** cluster
- Other **"Slave"** clusters replicate the same tables
- Master cluster synchronously sends HLogs over to slave clusters
- Coordination among clusters is via Zookeeper
- Zookeeper can be used like a file system to store control information

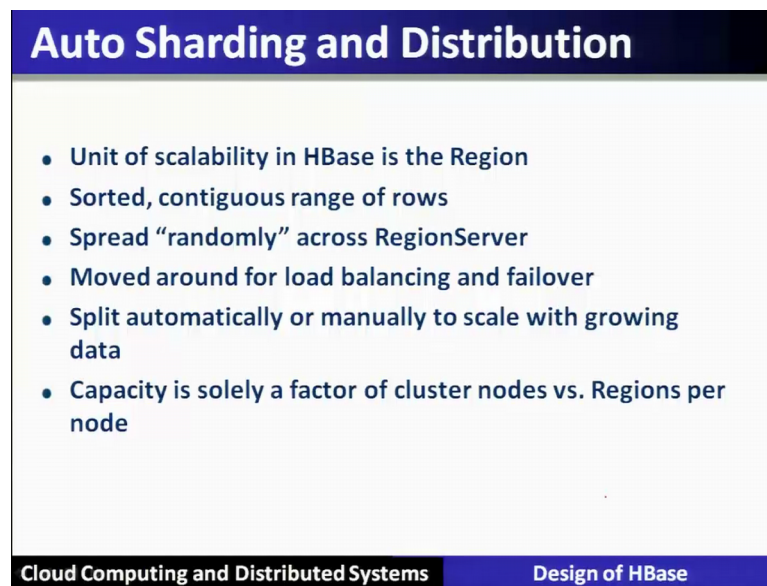
1. `/hbase/replication/state`
2. `/hbase/replication/peers/<peer cluster number>`
3. `/hbase/replication/rs/<hlog>`

Cloud Computing and Distributed Systems Design of HBase

Now we will see the cross data centre application. Now here we have a single master, which manages the cluster and there are other slave clusters which replicates the tables.

Master cluster synchronously sends HLogs over to the slave cluster so that they can store all that logs which are created for the write operations. Co-ordination among the clusters is carried out via Zookeeper. So, Zookeeper can be used like a file system to store control information also, which are listed over here, state information than pure cluster number HLog all this particular information is stored control information is stored in the form of a file system.

(Refer Slide Time: 31:16)



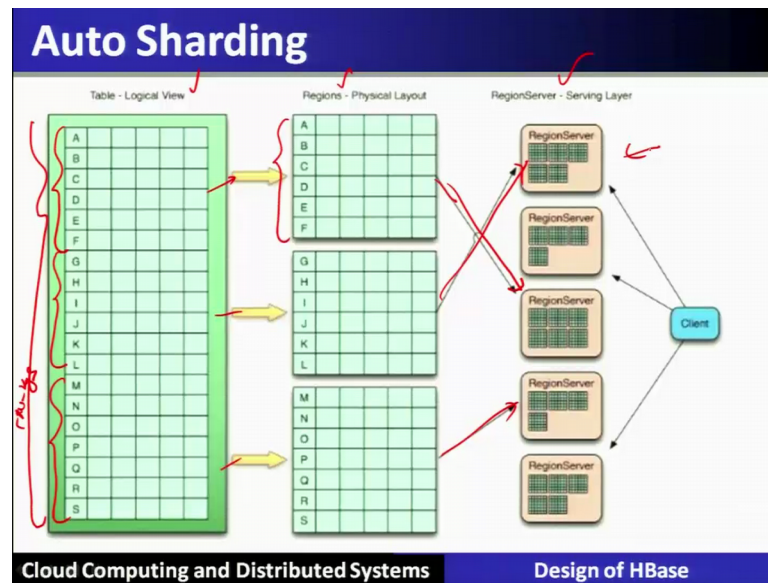
Auto Sharding and Distribution

- Unit of scalability in HBase is the Region
- Sorted, contiguous range of rows
- Spread “randomly” across RegionServer
- Moved around for load balancing and failover
- Split automatically or manually to scale with growing data
- Capacity is solely a factor of cluster nodes vs. Regions per node

Cloud Computing and Distributed Systems Design of HBase

Now there is also a process which is called auto sharding and distribution. So, that means, as we told that it supports horizontally scaling and that is provided using auto sharding. So, unit of scalability in HBase is the region, region is nothing but range of row keys, they are sorted continuous range of rows spread randomly across RegionServer moved around for load balancing and failover. So, splitted automatically or manually to scale growing the data with the growing data; so, capacity is solely the factor of cluster nodes versus the region per node.

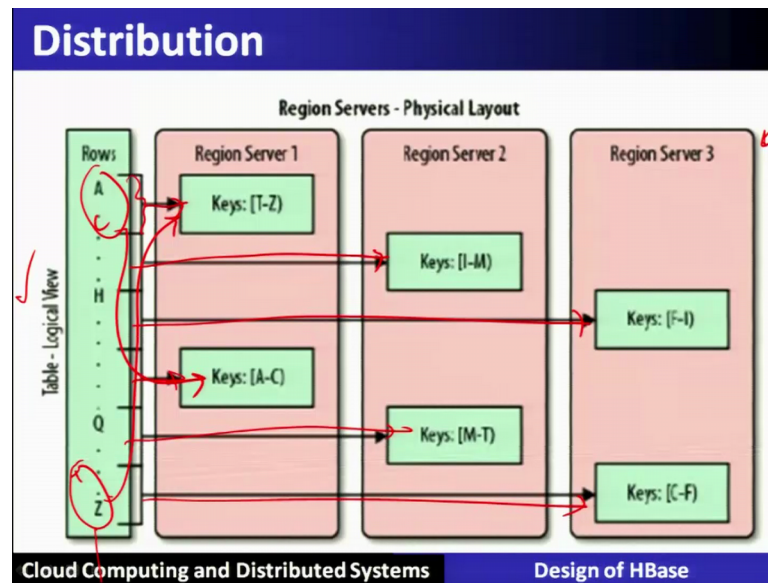
(Refer Slide Time: 32:09)



Let us see how this horizontally scaling is supported using auto sharding. So, this is the logical view of a particular table, which are sorted order in the form of rowkeys. This particular row keys are the splitted into the regions. That is in the first split up to A to F will be put in the first region. From G to L will be split into the second region. And from M to S it will go into the third split. These splits are stored these splits are called region. And these regions are stored in the region server; so, the client with the help of HRegionServer.

So, client using the master or the Zookeeper, it will directly go and access this HRegionServer and can access this particular table data. So, Zookeeper is basically the coordination service, and HBase for the client will give this information to contact the HRegionServers and directly the client accesses the data through HRegions.

(Refer Slide Time: 33:46)



Let us see the distribution again in more detail. The rows are now distributed; that means, sharding is that is the split, and how these particular splits are distributed across the region server is shown in this particular figure. So, if this is the rows according to their row keys are sorted order.

So, these different splits are maintained in the random order to different RegionServer, that we have seen here. So, for example, the first A to C key is now stored in the region server 1. Similarly, from T to Z range of keys they are also stored in the in the region server 1. So, that way these splits are organized in the region servers. So, as this particular row increases, this the number of servers are also increasing if it is, then the new particular regions are being supported in a new servers or they may be balancing with each other. And this information is maintained by the master.

(Refer Slide Time: 35:30)

Bloom Filter

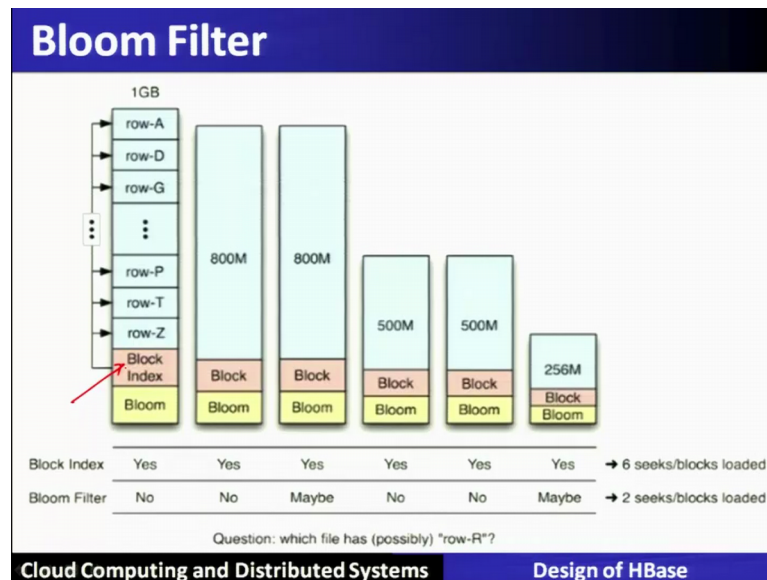
- Bloom Filters are generated when HFile is persisted
 - Stored at the end of each HFile
 - Loaded into memory
- Allows check on row or row + column level
- Can filter entire store files from reads
 - Useful when data is grouped
- Also useful when many misses are expected during reads (non existing keys)

Cloud Computing and Distributed Systems Design of HBase

Now, here there is the use of bloom filter, let us see what is the use of bloom filter here in HBase. So, bloom filters are generated when the HFile is persisted; that means, as long as the file is in Mem table. This is not done and as soon as the file is written on HDFS, the bloom filter index is are generated. So, these are stored at the end of each HFile and also loaded into the memory so that is whenever that file is preferred or accessed by the client; the bloom filter can hash it. And can get that information about the rows and the column families and it will access in a fast manner.

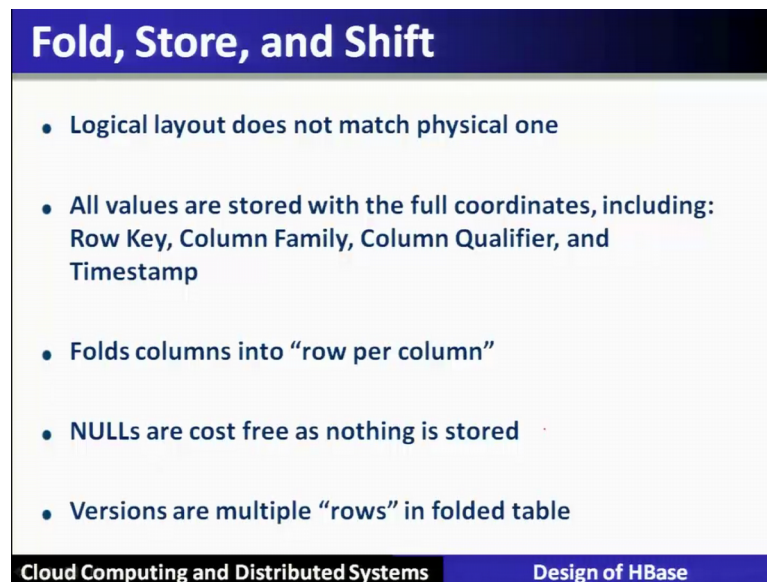
Now, bloom filter will allow to check on the rows basis. So, that indexes on the rows are done in the form of bloom filters. It can also be possible that rows and column level both together are indexed in the form of bloom filter, then an extra overhead will be added up in case rows and columns also are being indexed. So, either in the form of rows then bloom filter will create index or if let us say that rows and column level if it is together done then other indexes are created. Now, this bloom filter can filter the entire StoreFiles from the reads it is useful, when the data is grouped also useful when many misses are expected during the reeds.

(Refer Slide Time: 37:19)



So, here there is an example that at the end of this table these block indexes are created, and being maintained to access this particular files. So, these are to be maintained at the region servers. So, that whenever there is an access whenever an access to a HFile then basically through the bloom filter indexes it can be made efficient access.

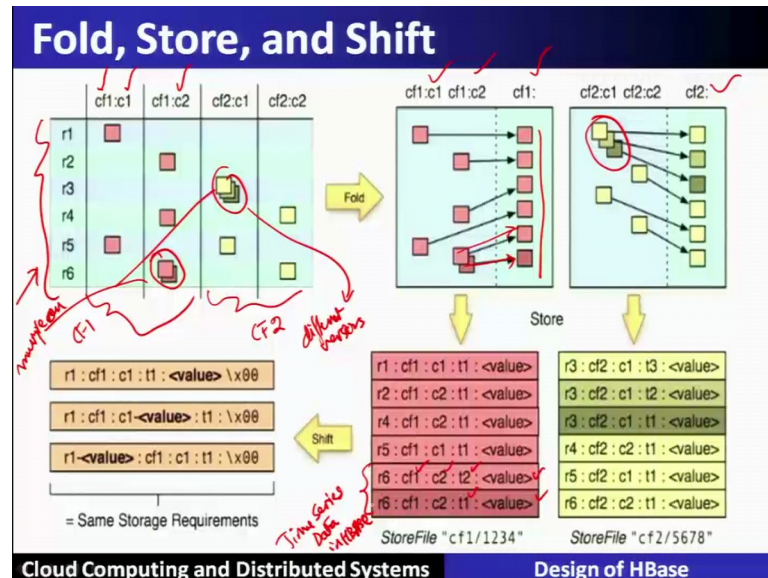
(Refer Slide Time: 37:48)



So, fold store and shift logical layout does not match the physical one all the values are stored with full coordinate's row key Column Family, Column Qualifier and Timestamp

that we have seen. And it will fold columns into the row per C column nulls are cost free, as there are nothing is stored and versions are the multiple rows in the folder table.

(Refer Slide Time: 38:12)



Let us see this example that how these h information or a data is stored in HFile. Now here this is the typical table which is nothing but sorted order in the form of a row keys, r 1 to r 6 it is shown. And it is having Column Family cf 1 with the column reference. So, that is Column Family one with having a particular attribute c 1 ColumnFamily 1 will have another attributes cf 2. So, this is one Column Family which is shown as the red.

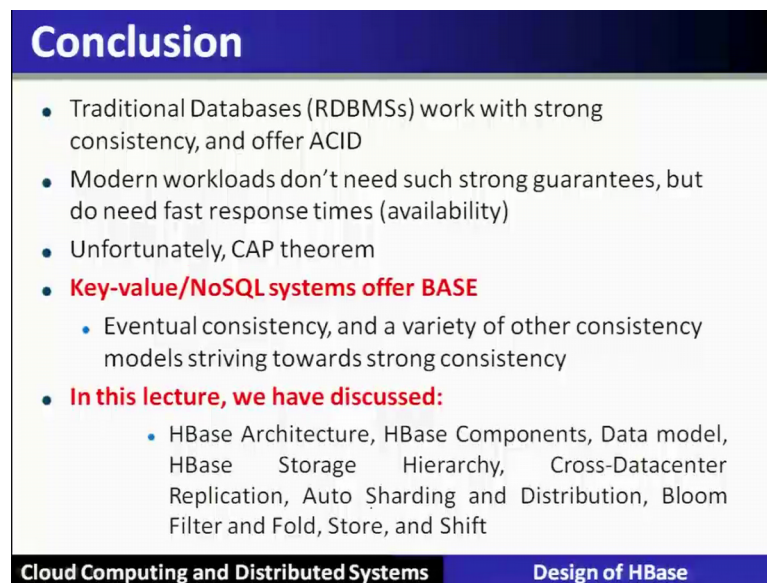
Similarly, there is another ColumnFamily cf 2, which is shown as the yellow color. Now this particular cell or this particular attribute that is called c 2 is having the multiple entries so, also here. This and this means that it has different versions, which is stored here this is folded and stored into the HFile.

Let us see how this is all being carried out. So, when this particular table is folded, let us say that this particular table that is called cf 1 is folded. So, all this particular data is stored as far as ColumnFamily one is concerned. So, multiple versions of that particular column attribute is also stored. Similarly, here also you can see that this is having multiple versions, and all are restored in the column families when it is folded up.

Here we can see that this particular entry which is r 6, having multiple entries that is r 6 with the ColumnFamily 1 C 2 is basically the ColumnFamily well attribute. It is having 2

timestamps; that means, at 2 timestamp 2 versions, these 2 different values are there. So, that means, we can also store the time series data in HBase. So, HBase supports this unstructured data, and it also supports this concept so that time series and multiple versions can also be stored with their timestamps.

(Refer Slide Time: 41:13)



Conclusion

- Traditional Databases (RDBMSs) work with strong consistency, and offer ACID
- Modern workloads don't need such strong guarantees, but do need fast response times (availability)
- Unfortunately, CAP theorem
- **Key-value/NoSQL systems offer BASE**
 - Eventual consistency, and a variety of other consistency models striving towards strong consistency
- **In this lecture, we have discussed:**
 - HBase Architecture, HBase Components, Data model, HBase Storage Hierarchy, Cross-Datcenter Replication, Auto Sharding and Distribution, Bloom Filter and Fold, Store, and Shift

Cloud Computing and Distributed Systems Design of HBase

Conclusion; traditional database is like RDBMS work with strong consistency and offer acid properties; whereas the modern workloads do not need such a strong guarantees, but do need fast response times that is in terms of high availability. Unfortunately, there is a cap theorem, which says that any 2 out of 3 we have to choose.

So, as far as Cassandra is concerned we have seen the Casandra has chosen availability with partitioning, and has compromised with the consistency that is called eventual consistency. This particular database which is HBase prefers consistency over availability with partition. So, this key values database that is NOSQL database offers n base property instead of acid property. This means basically available storage eventual consistency in this lecture. We have covered HBase architecture HBase components data model. HBase storage cross data centre replication, auto sharding and the use of bloom filter, and fold store and shift for HFiles.

Thank you.