Design of HBase

Dr. Rajiv Misra

Associate Professor

Dept. of Computer Science & Engg.

Indian Institute of Technology Patna

rajivm@iitp.ac.in

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**

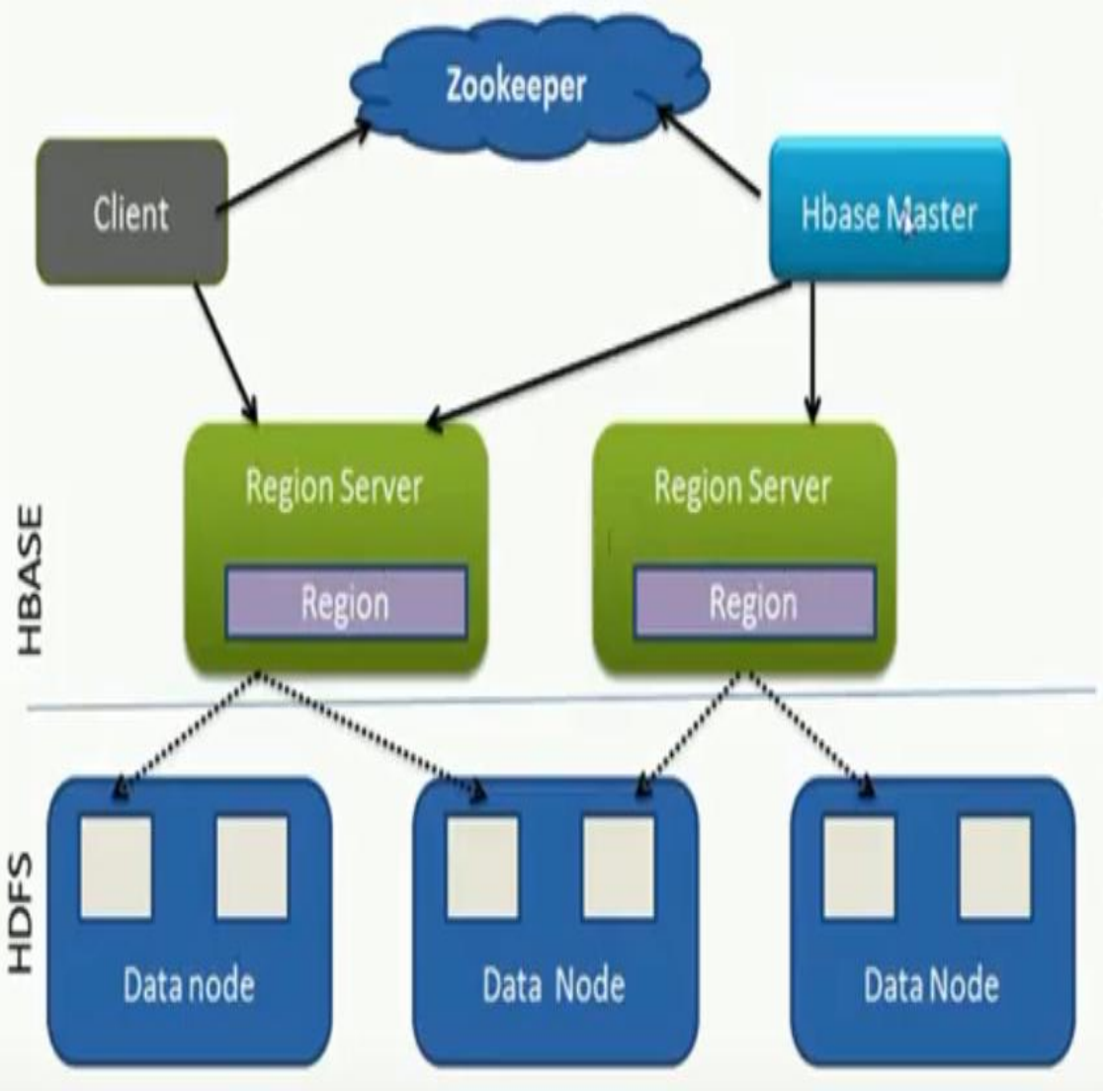
HBase is:

- **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.

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)

HBase Architecture



- Table Split into **regions** and served by region servers.
- Regions vertically divided by column families into "**stores**".
- Stores saved as files on HDFS.
- Hbase utilizes zookeeper for distributed coordination.

HBase Components

- **Client:**

  Finds RegionServers that are serving particular row range of interest

- **Hmaster:**

  Monitoring all RegionServer instances in the cluster

- **Regions:**

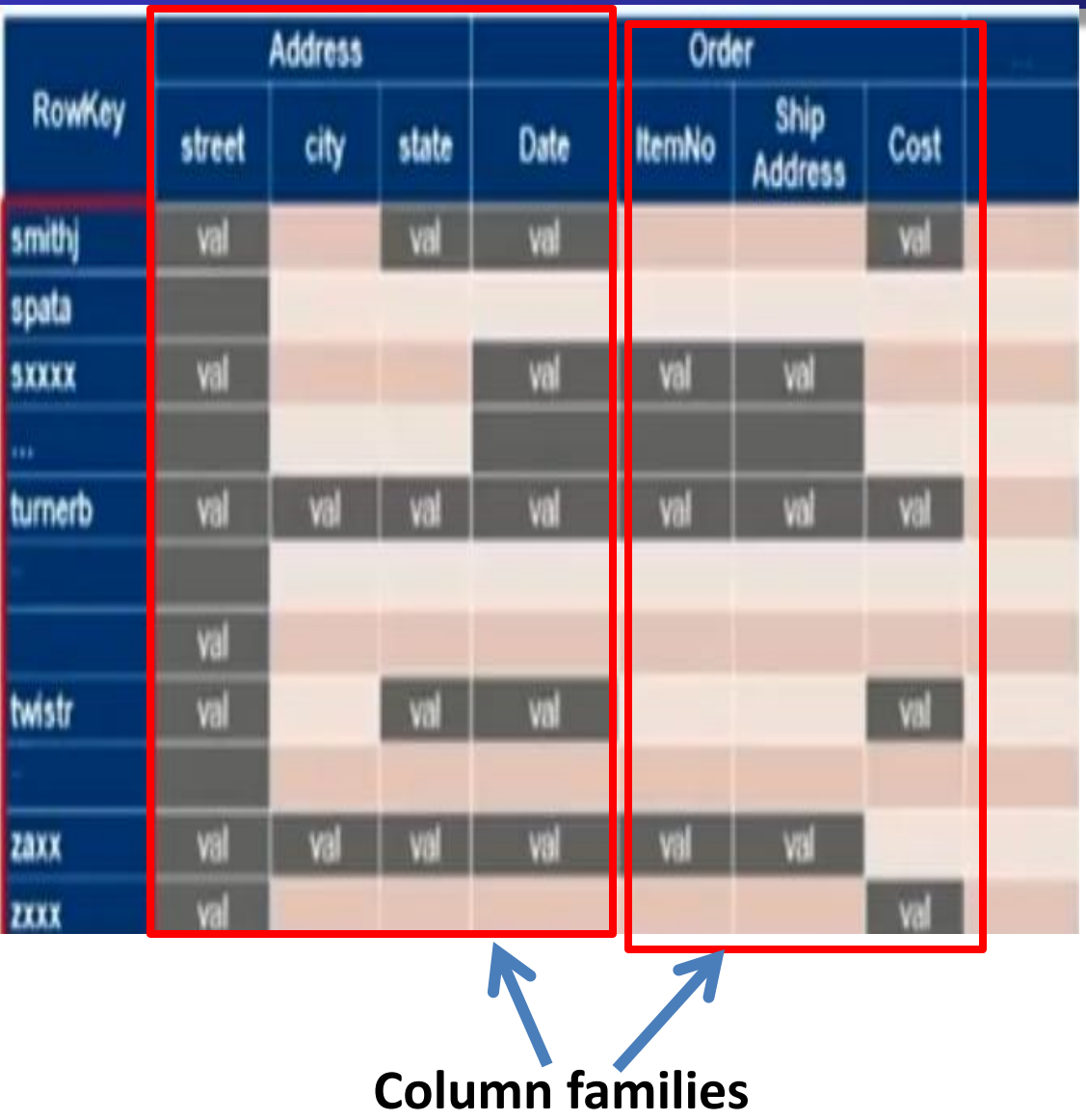
  Basic element of availability and distribution for tables

- **RegionServer:**

  Serving and managing regions

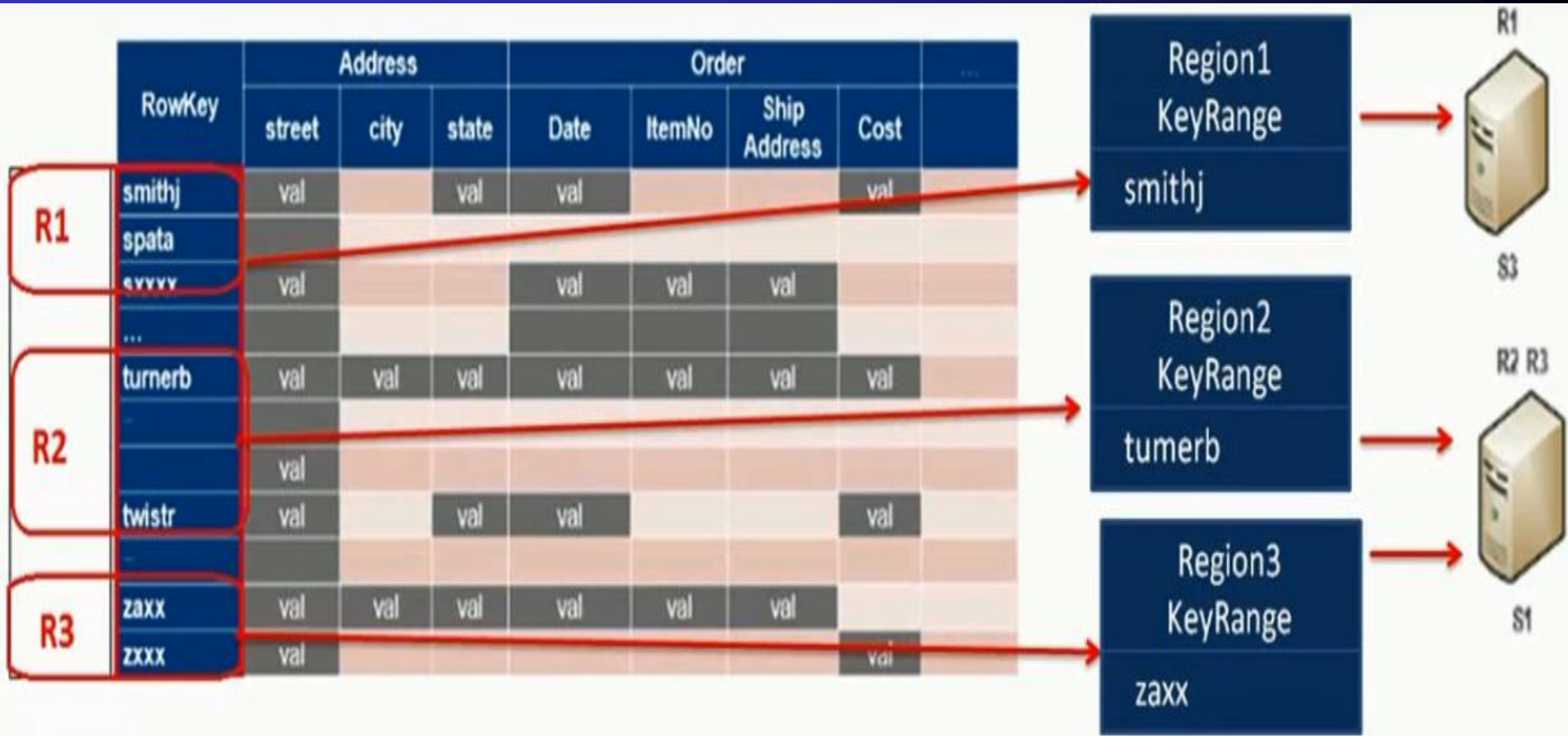
  In a distributed cluster, a RegionServer runs on a DataNode

Data Model



- Data stored in Hbase is located by its "**rowkey**"
- RowKey is like a primary key from a rational database.
- Records in Hbase are stored in sorted order, according to rowkey.
- Data in a row are grouped together as Column Families. Each Column Family has one or more Columns
- These Columns in a family are stored together in a low level storage file known as **HFile**

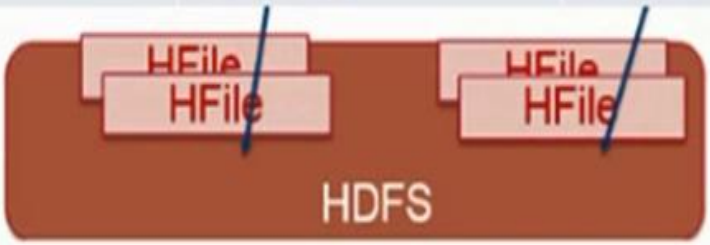
HBase Components



- Tables are divided into sequences of rows, by key range, called **regions.**
- These regions are then assigned to the data nodes in the cluster called **"RegionServers."**

Column Family

Row Key	Personal Data		ProfessionalData	
Emp Id	Name	City	Designation	Salary
101	John	Mumbai	Manager	10L
102	Geetha	New Delhi	Sr.Software Engineer	8L
103	Smita	Pune	Programmer Analyst	4L
104	Ankit	Bangalore	Data Analyst	12L



- A column is identified by a Column Qualifier that consists of the Column Family name concatenated with the Column name using a colon.ex-personaldata:Name
- Column families are mapped to storage files and are stored in separate files, which can also be accesses separately.

Cell in HBase Table

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

- 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.**

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.

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[ ].

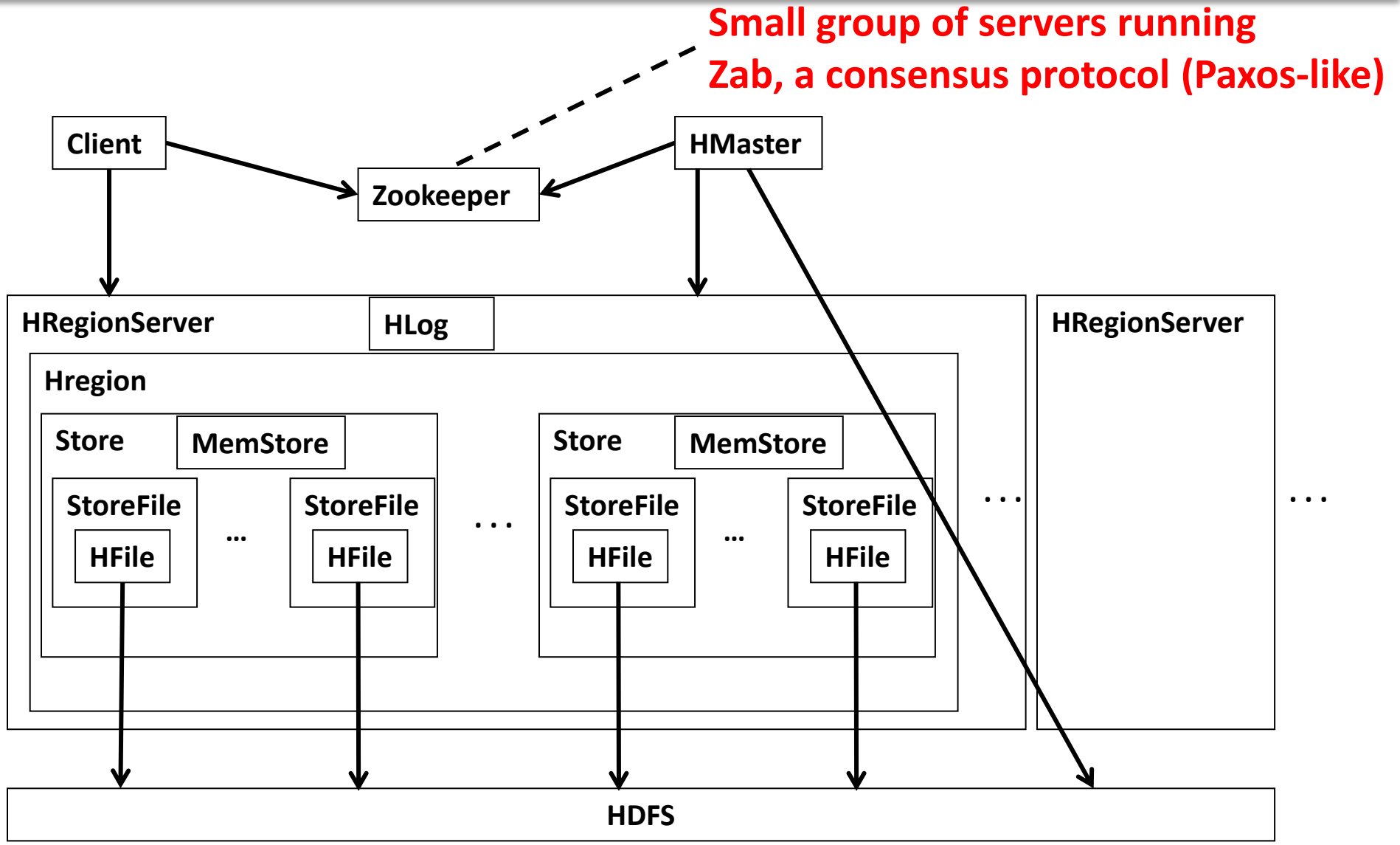
- **Cell:** A combination of row key, column family, and column qualifier uniquely identifies a cell. The data stored in a cell is referred to as that cell's value.

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.

HBase Architecture

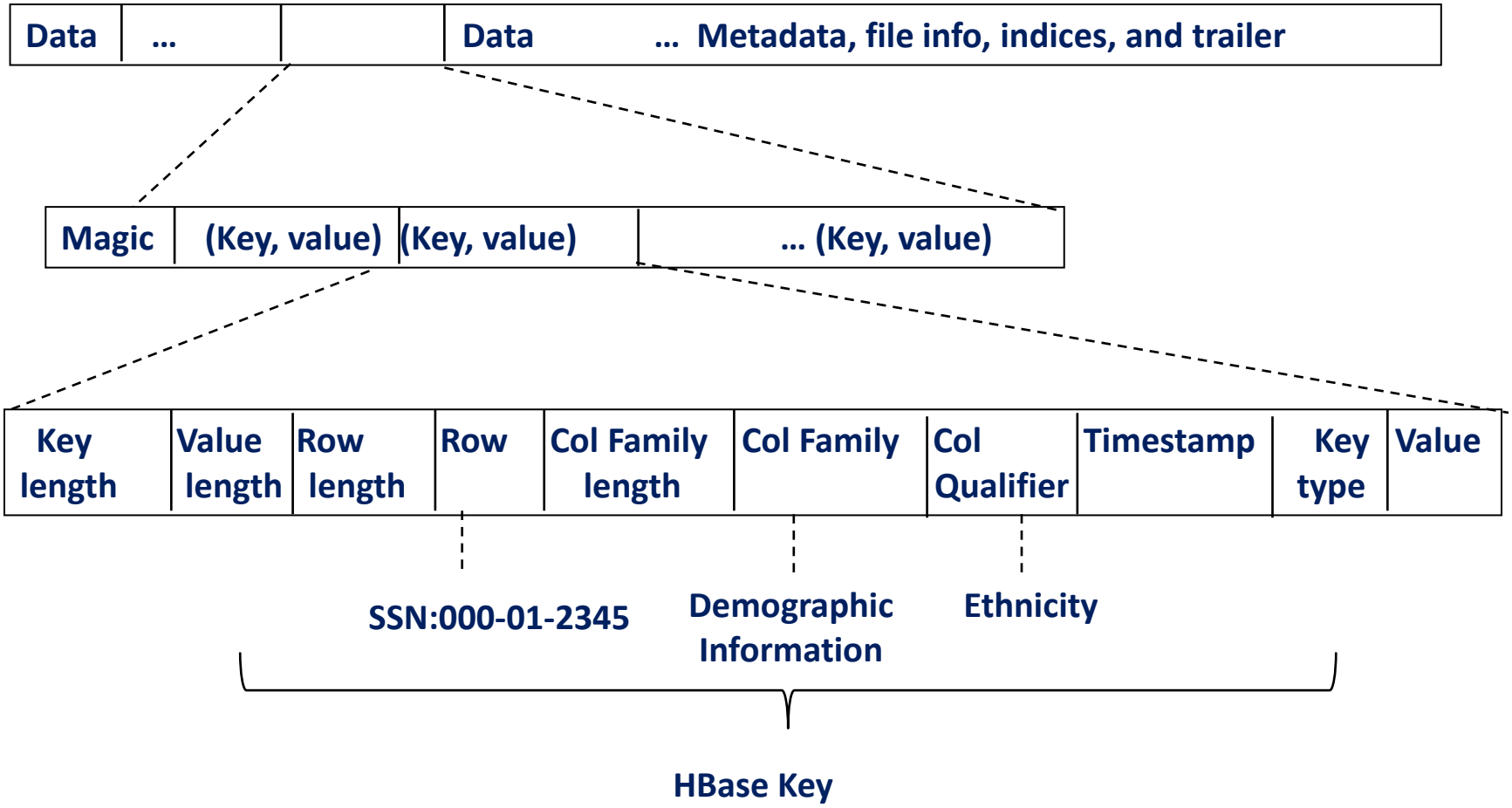


HBase Storage Hierarchy

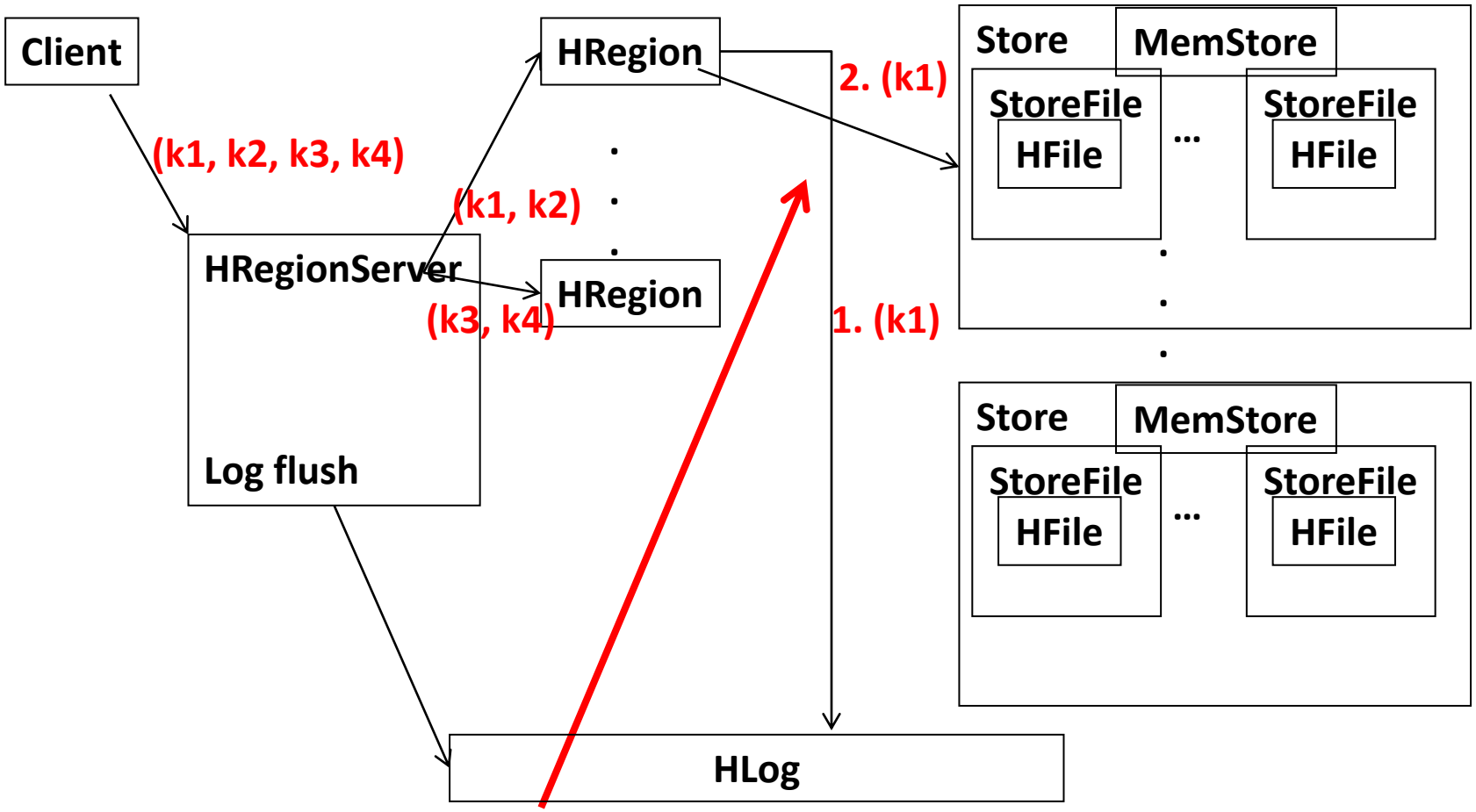
- **HBase Table**
  - Split it into multiple **regions**: replicated across servers
    - ColumnFamily = subset of columns with similar query patterns
    - One **Store** per combination of ColumnFamily + region
      - **Memstore** for each Store: in-memory updates to Store; flushed to disk when full
        - » **StoreFiles** for each store for each region: where the data lives
          - **Hfile**

- **HFile**
  - SSTable from Google's BigTable

HFile


Data
...
Data
... Metadata, file info, indices, and trailer
Magic
(Key, value)
(Key, value)
... (Key, value)
Key length
Value length
Row length
Row
Col Family length
Col Family
Col Qualifier
Timestamp
Key type
Value
SSN:000-01-2345
Demographic Information
Ethnicity
HBase Key

Strong Consistency: HBase Write-Ahead Log



Write to HLog before writing to MemStore

Helps recover from failure by replaying Hlog.

Log Replay

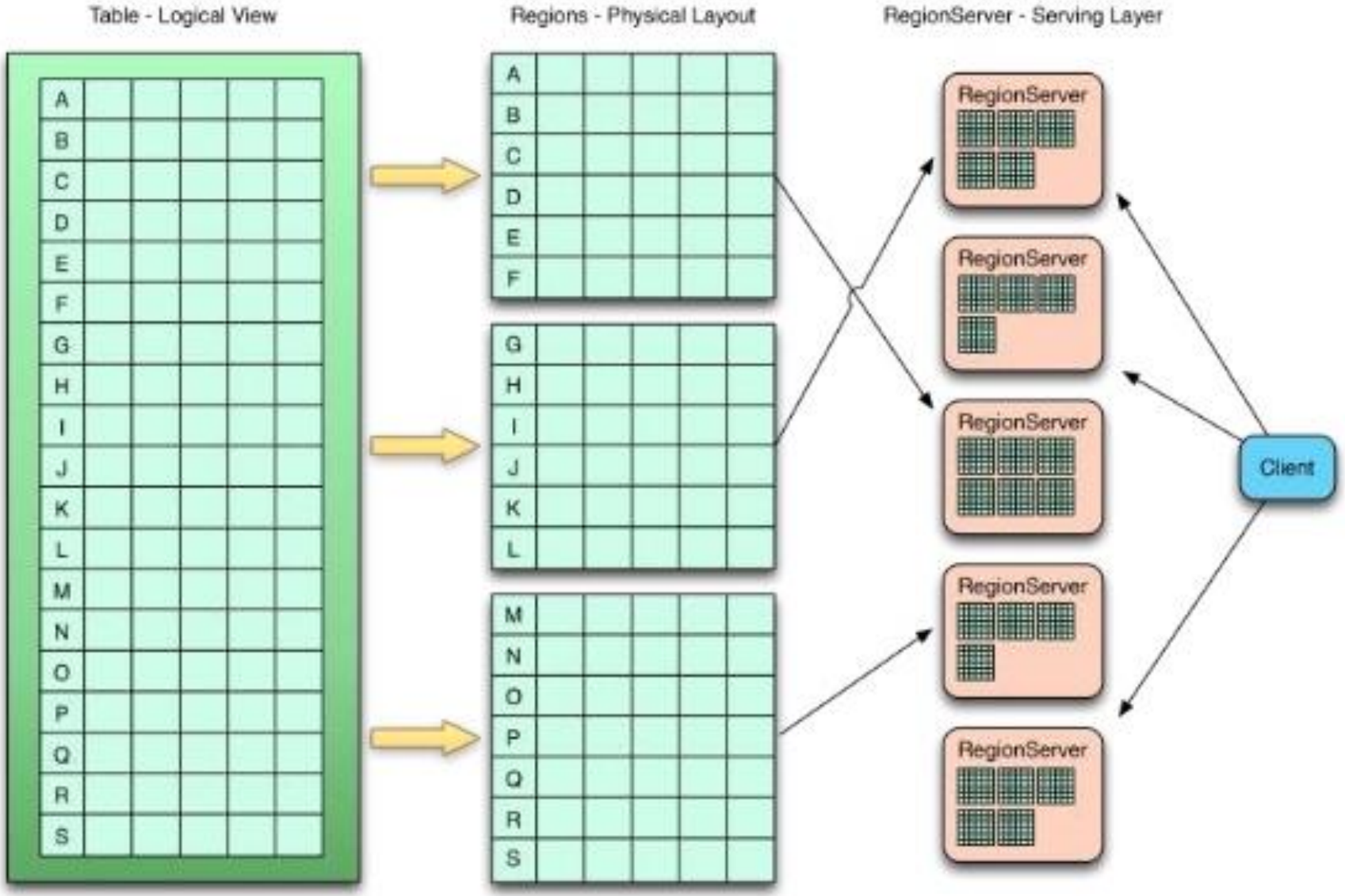
- **After recovery from failure, or upon bootup (HRegionServer/HMaster)**
  - Replay any stale logs (use timestamps to find out where the database is with respect to the logs)
  - Replay: add edits to the MemStore

Cross-Datacenter Replication

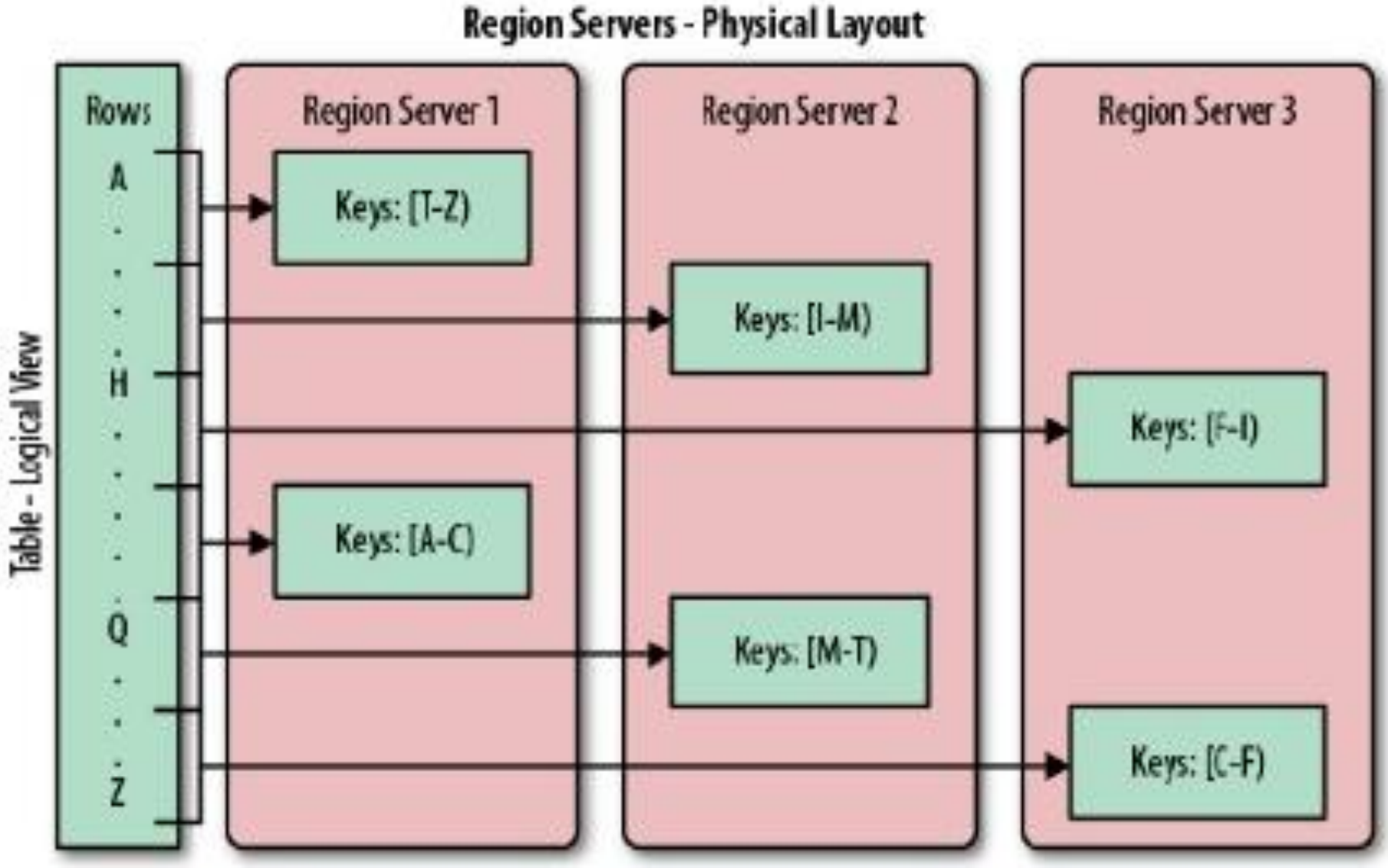
- Single "**Master**" cluster
- Other "**Slave**" clusters replicate the same tables
- Master cluster synchronously sends HLogs over to slave clusters
- Coordination among clusters is via Zookeeper
- Zookeeper can be used like a file system to store control information

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

Auto Sharding



Distribution



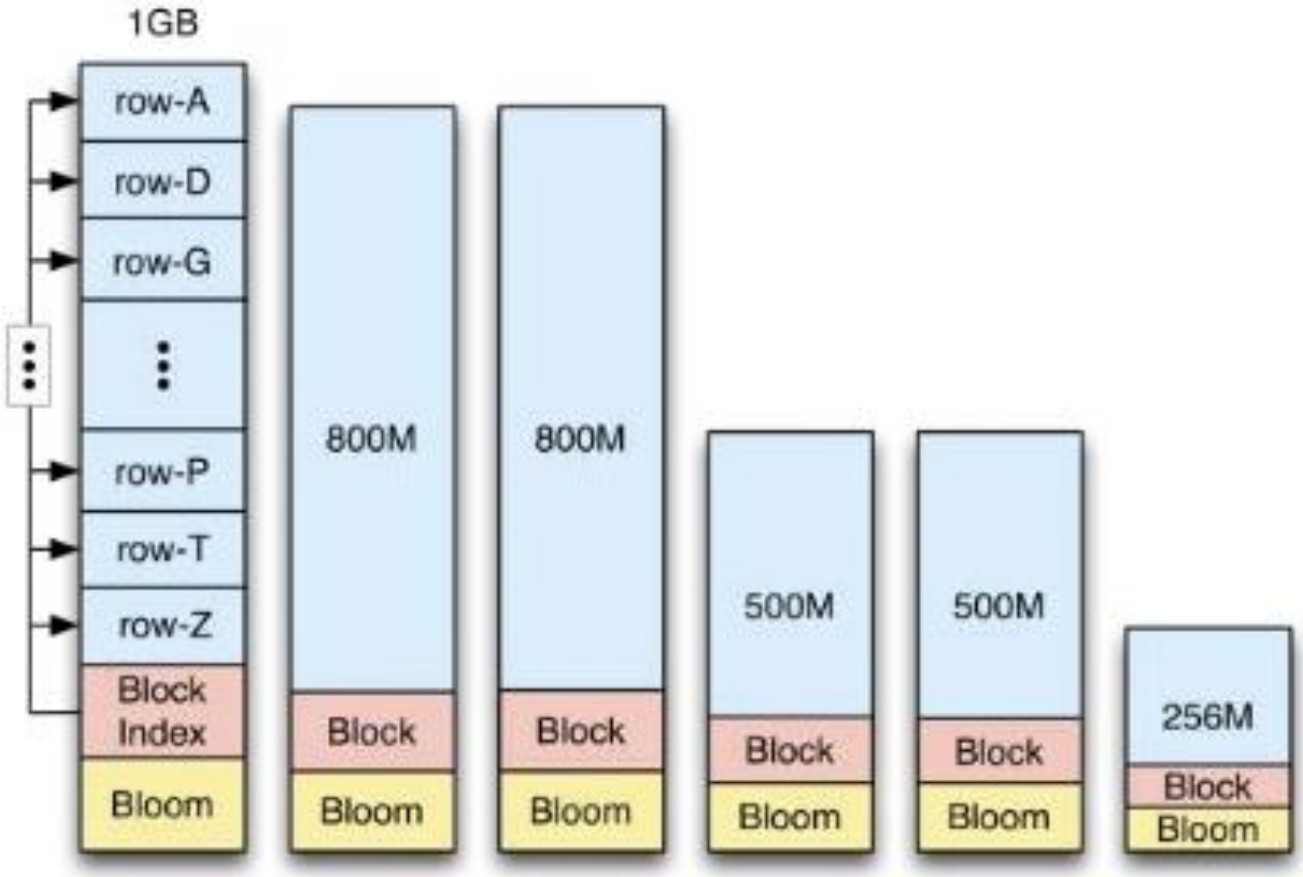
Auto Sharding and Distribution

- 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

Bloom Filter

- Bloom Filters are generated when HFile is persisted
  - Stored at the end of each HFile
  - Loaded into memory
- Allows check on 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)

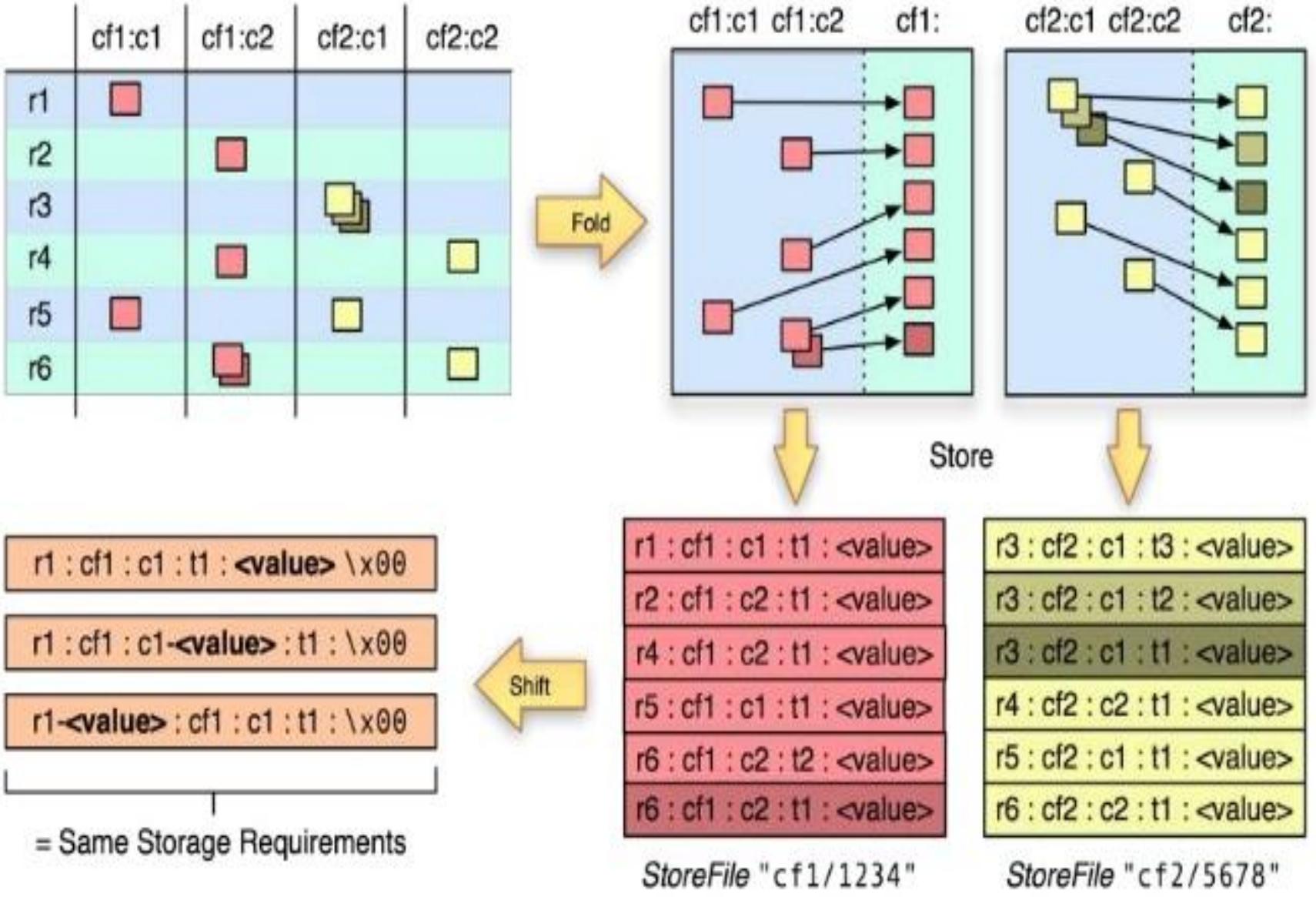
Bloom Filter



Block Index	Yes	Yes	Yes	Yes	Yes	Yes	→ 6 seeks/blocks loaded
Bloom Filter	No	No	Maybe	No	No	Maybe	→ 2 seeks/blocks loaded

Question: which file has (possibly) "row-R"?

Fold, Store, and Shift



Fold, Store, and Shift

- Logical layout does not match physical one
- All values are stored with the full coordinates, including: Row Key, Column Family, Column Qualifier, and Timestamp
- Folds columns into “row per column”
- NULLs are cost free as nothing is stored
- Versions are multiple “rows” in folded table

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-Datacenter Replication, Auto Sharding and Distribution, Bloom Filter and Fold, Store, and Shift