Database Storage Engines (Key-Value Stores or Hash Tables)

Hung-Chang Hsiao (蕭宏章)
Professor
Department of Computer Science and Information Engineering
National Cheng Kung University
Taiwan

Databases in a Word

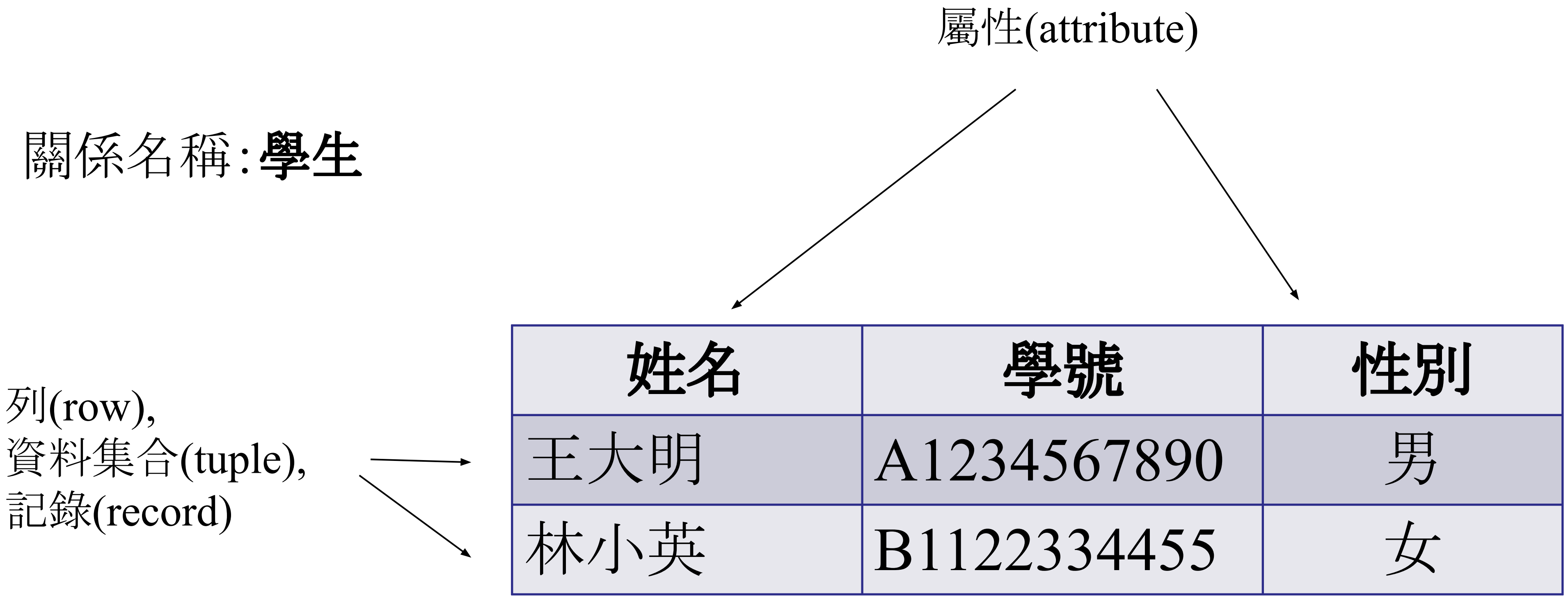
- **SQL compiler + Database storage engine**
  - SQL compiler: 將高階的SQL語言換成低階的storage engine語言
    - e.g., “SELECT * from Table XYZ” -> “SCAN (start key, end key)”
  - Database storage engine: 執行低階的資料操作語言並與檔案系統互動

- **MariaDB (i.e., MySQL) 在給定的SQL compiler下得置換storage engine**

如何表示關係?

- **關係是由資料表所構成**

關係名稱:**學生**



屬性(attribute)

列(row),
資料集合(tuple),
記錄(record)

姓名	學號	性別
王大明	A1234567890	男
林小英	B1122334455	女

資料庫綱要 (Database Schema)

- **綱要(Schema)**
  - 關係名稱和屬性所組成
  - 描述資料的形態
  - E.g., Student(name, studentID) 或是 Student(name: string, studentID: string)

資料庫定義語言 (Data-Definition Language, or DDL)

- 用以描述，修改或是刪除資料庫中的資料關係
- 定義欄位、資料型態、資料結構
- E.g.,

```
CREATE TABLE STUDENT(
    name CHAR(10),
    studentID CHAR(15),
    gender CHAR(1),
    PRIMARY KEY (studentID)
)
```

資料庫處理語言
(Data-Management Language, or DML)

- **讓使用者存取或是處理資料庫的資料**
  - 讀取資料
  - 新增資料
  - 修改更新資料
  - 刪除資料

- E.g.,

```
INSERT INTO STUDENT(name, studentID, gender)
VALUES(‘David’, ‘A1231231234’, ‘M’)
```

NoSQL Databases (Storage Engines)

- No relations among DB tables
  - DB relations across tables是透過foreign keys來相互關聯表格的屬性欄位關係
  - Relations正是讓DB engine無法橫向擴充 (scale-out) 的關鍵原因
- APIs
  - GET (key)
  - PUT (key, value)
  - SCAN (start key, end key)
- Highly scalable: distributed DB engines
- SQL-like support, e.g., Apache Hive and Phoenix over HBase
- No transactions
- 經驗:無論SQL或NoSQL, 當對DB執行速度有嚴謹的要求時, 我們會直接操作storage engine而非透過SQL語言

Question?

- **給定大量key-value pairs的資料，我們想查找給定符合某些條件keys的values，請問如何做?**
  - E.g., (1, a), (2, b), (3, c),…, (10, j)
  - 找出 2 < keys < 4的values值

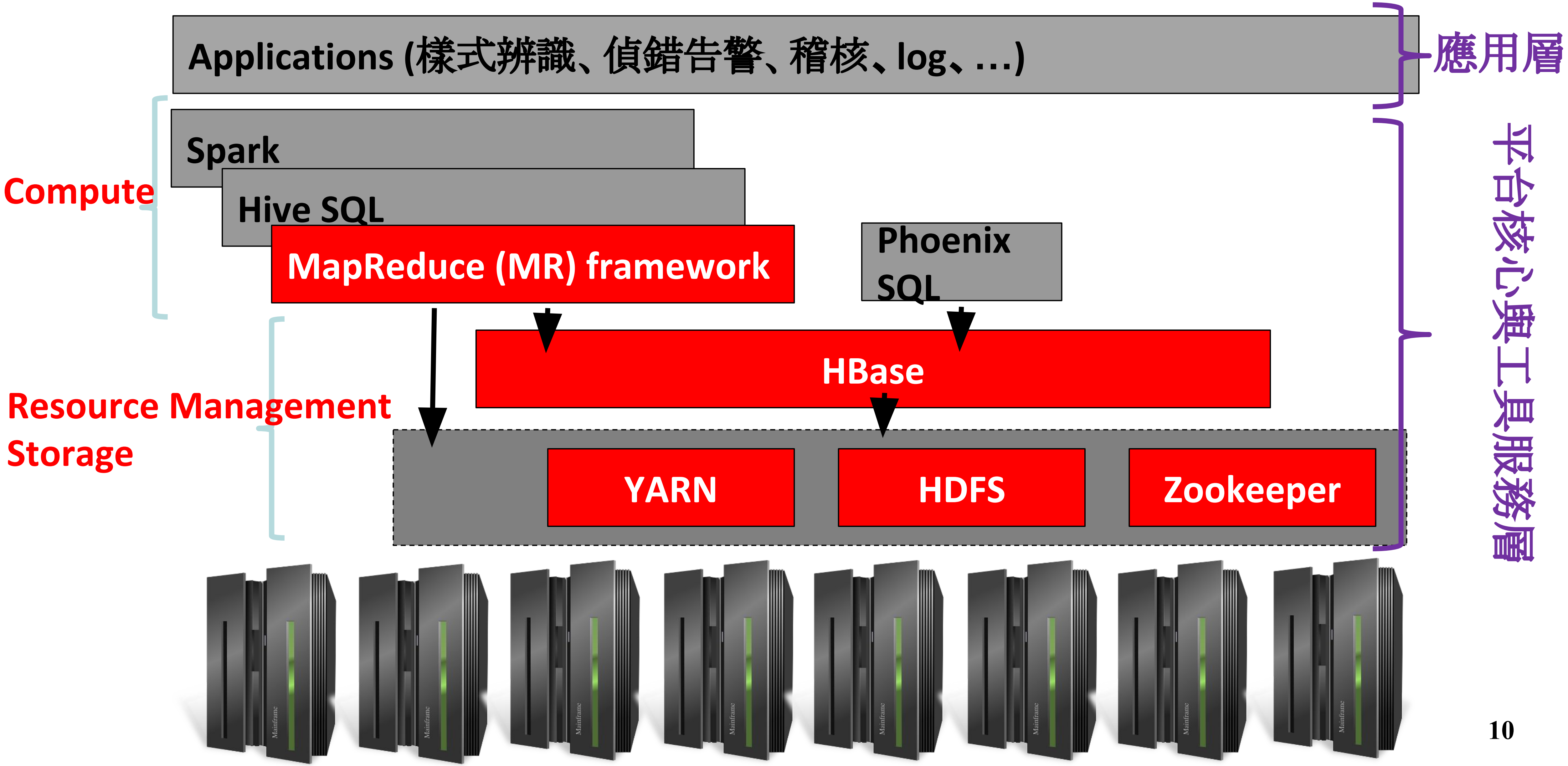
- **要求:我們不能控制key-value pairs資料產生的keys值順序!**

- **如何儲存該些資料? 若只靠檔案系統 (無論集中或分散)?**
  - 假設資料來的順序是 (3, c), (10,j), (2, b), (1, a),…
  - 我們按資料產生的順序來將資料儲存在數個檔案裡，比如檔案*A*存(3,c)及(10,j)、檔案B存(2,b)及(1,a)，檔案*C*存…

- **如何儲存會大大影響將來的資料查找的效率**

APACHE
HBASE

Big Data Platform: Hadoop Ecosystem



Applications (樣式辨識、偵錯告警、稽核、log、...)

應用層

Spark

Hive SQL

MapReduce (MR) framework

Phoenix SQL

Compute

HBase

YARN

HDFS

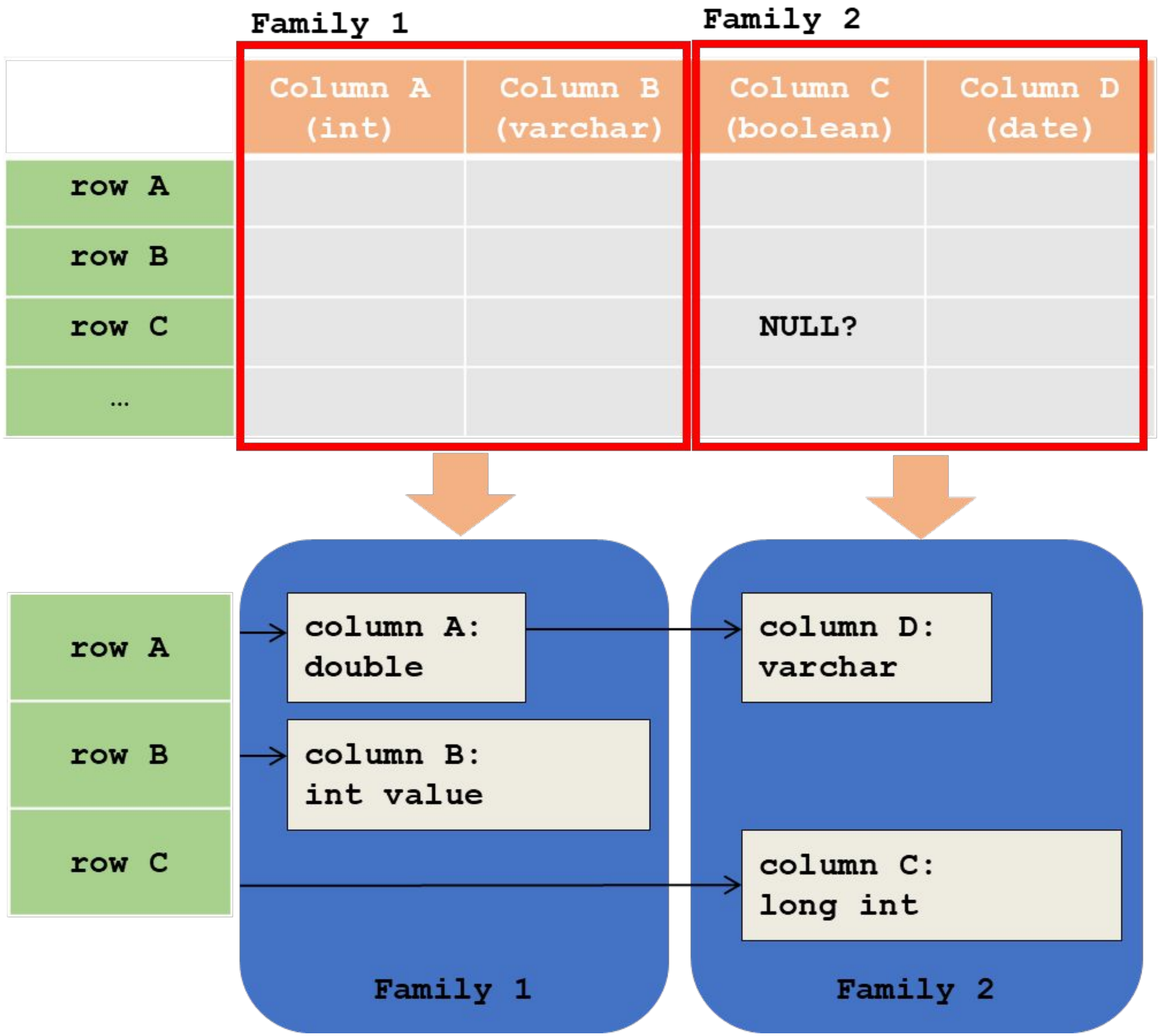
Zookeeper

Resource Management

Storage

平台核心與工具服務層

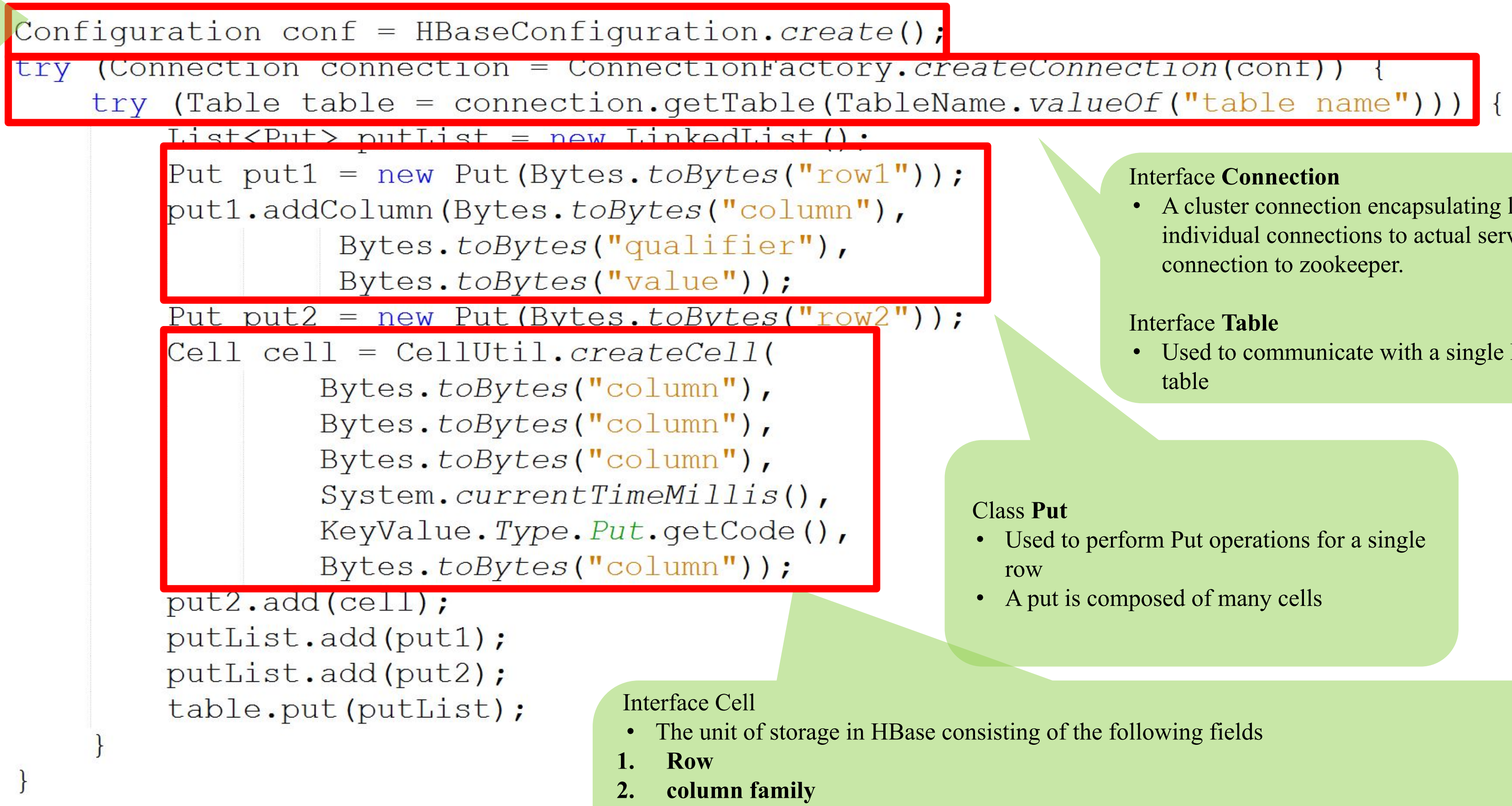
Overview: HBase Table



	Family 1		Family 2	
	Column A (int)	Column B (varchar)	Column C (boolean)	Column D (date)
row A				
row B				
row C			NULL?	
…				

API Overview (1/5): Put Data

Class **Configurations**
- Configurations are specified by resources.
- A resource contains a set of name/value pairs as XML data



```java
Configuration conf = HBaseConfiguration.create();
try (Connection connection = ConnectionFactory.createConnection(conf)) {
    try (Table table = connection.getTable(TableName.valueOf("table name"))) {
        List<Put> putList = new LinkedList();
        Put put1 = new Put(Bytes.toBytes("row1"));
        put1.addColumn(Bytes.toBytes("column"),
                Bytes.toBytes("qualifier"),
                Bytes.toBytes("value"));
        Put put2 = new Put(Bytes.toBytes("row2"));
        Cell cell = CellUtil.createCell(
                Bytes.toBytes("column"),
                Bytes.toBytes("column"),
                Bytes.toBytes("column"),
                System.currentTimeMillis(),
                KeyValue.Type.Put.getCode(),
                Bytes.toBytes("column"));
        put2.add(cell);
        putList.add(put1);
        putList.add(put2);
        table.put(putList);
    }
}
```

Interface **Connection**
- A cluster connection encapsulating lower level individual connections to actual servers and a connection to zookeeper.

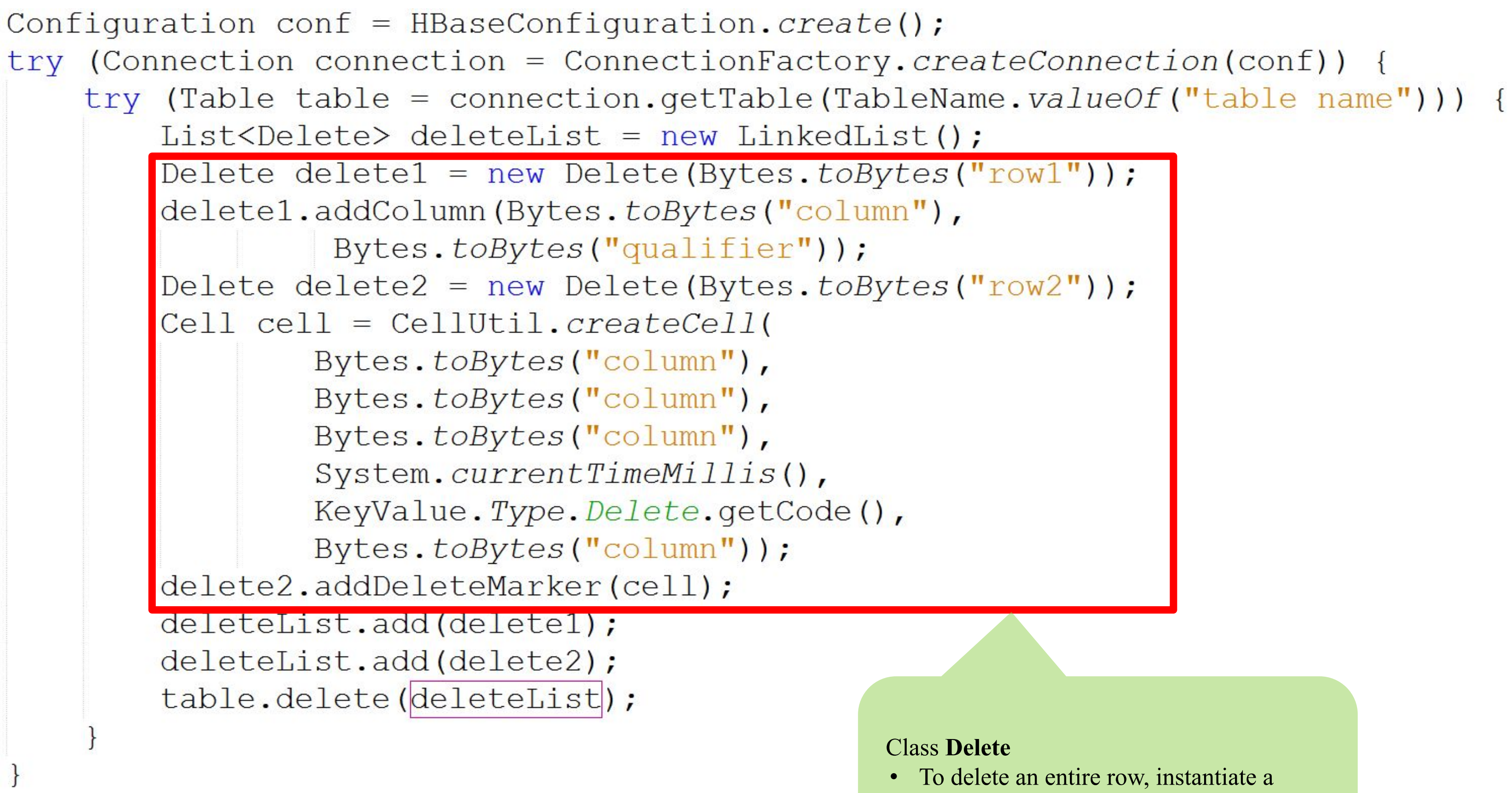
Interface **Table**
- Used to communicate with a single HBase table

Class **Put**
- Used to perform Put operations for a single row
- A put is composed of many cells

Interface Cell
- The unit of storage in HBase consisting of the following fields

1. **Row**
2. **column family**
3. **column qualifier**
4. **Timestamp**
5. **Type**
6. **MVCC version (set by server)**
7. **value**

API Overview (2/5): Delete Data

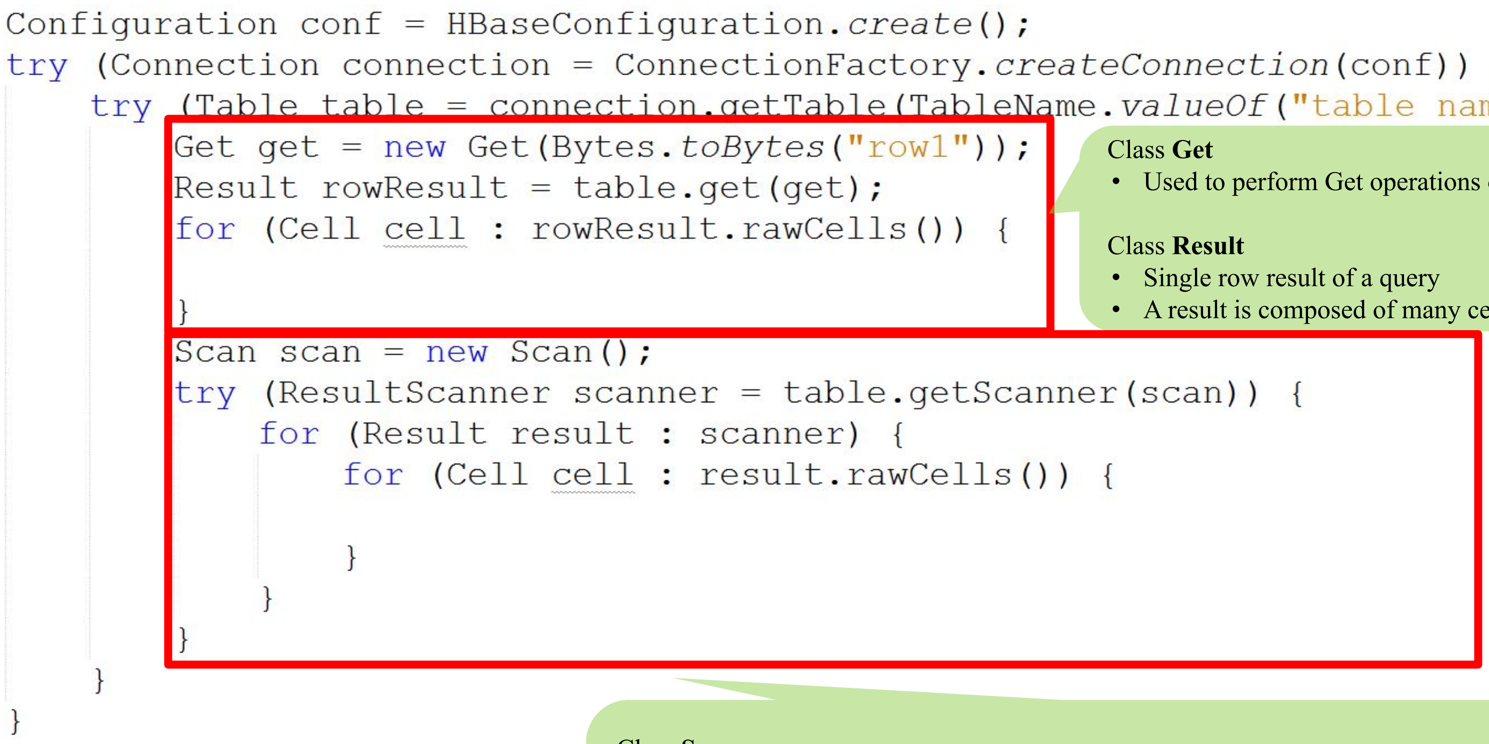


```java
Configuration conf = HBaseConfiguration.create();
try (Connection connection = ConnectionFactory.createConnection(conf)) {
    try (Table table = connection.getTable(TableName.valueOf("table name"))) {
        List<Delete> deleteList = new LinkedList();
        Delete delete1 = new Delete(Bytes.toBytes("row1"));
        delete1.addColumn(Bytes.toBytes("column"),
                 Bytes.toBytes("qualifier"));
        Delete delete2 = new Delete(Bytes.toBytes("row2"));
        Cell cell = CellUtil.createCell(
                Bytes.toBytes("column"),
                Bytes.toBytes("column"),
                Bytes.toBytes("column"),
                System.currentTimeMillis(),
                KeyValue.Type.Delete.getCode(),
                Bytes.toBytes("column"));
        delete2.addDeleteMarker(cell);
        deleteList.add(delete1);
        deleteList.add(delete2);
        table.delete(deleteList);
    }
}
```

Class **Delete**

- To delete an entire row, instantiate a Delete object with the row to delete.
- To define the scope of what to delete

API Overview (3/5): Get Data



```
Configuration conf = HBaseConfiguration.create();
try (Connection connection = ConnectionFactory.createConnection(conf)) {
    try (Table table = connection.getTable(TableName.valueOf("table name"))) {
        Get get = new Get(Bytes.toBytes("row1"));
        Result rowResult = table.get(get);
        for (Cell cell : rowResult.rawCells()) {

        }
        Scan scan = new Scan();
        try (ResultScanner scanner = table.getScanner(scan)) {
            for (Result result : scanner) {
                for (Cell cell : result.rawCells()) {

                }
            }
        }
    }
}
```

Class **Get**

- Used to perform Get operations on a single row.

Class **Result**

- Single row result of a query
- A result is composed of many cells

Class **Scan**

- Used to perform Scan operations.
- Rather than specifying a single row, an optional **startRow** and **stopRow** may be defined

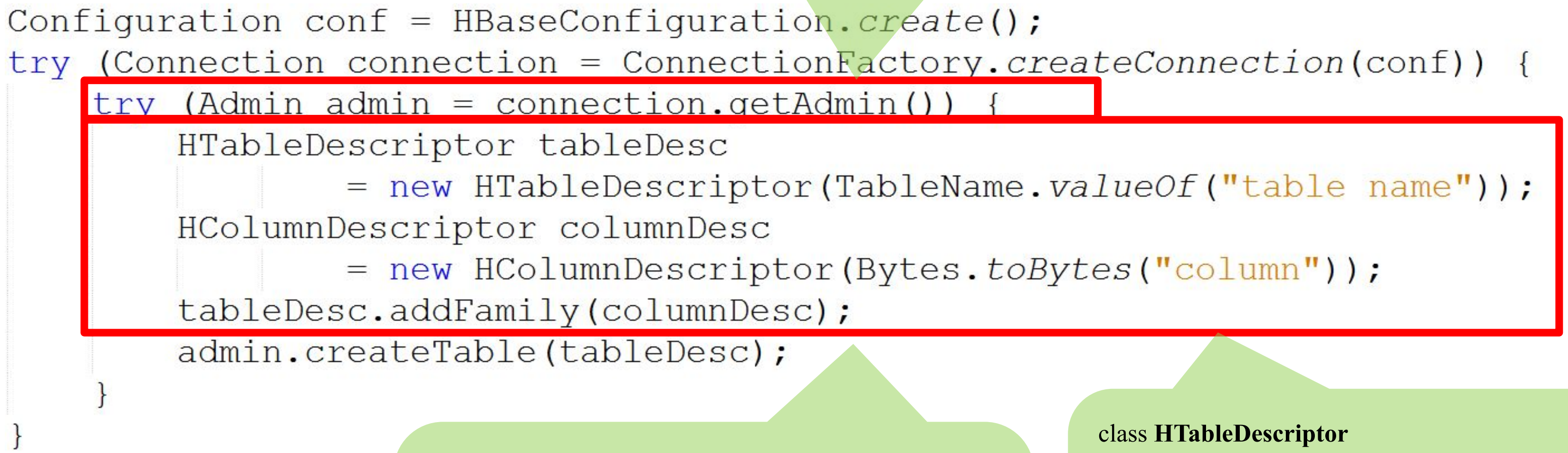
Interface ResultScanner

- iterate over all rows.

API Overview (4/5): Create a Table

Interface **Admin**

- The administrative API for HBase



```
Configuration conf = HBaseConfiguration.create();
try (Connection connection = ConnectionFactory.createConnection(conf)) {
    try (Admin admin = connection.getAdmin()) {
        HTableDescriptor tableDesc
                = new HTableDescriptor(TableName.valueOf("table name"));
        HColumnDescriptor columnDesc
                = new HColumnDescriptor(Bytes.toBytes("column"));
        tableDesc.addFamily(columnDesc);
        admin.createTable(tableDesc);
    }
}
```

class **HColumnDescriptor**

- contains information about a column family

1. Set block cache
2. Set bloom filter
3. Compression
4. Data block encoding
5. HDFS block size
6. Max/min version

class **HTableDescriptor**

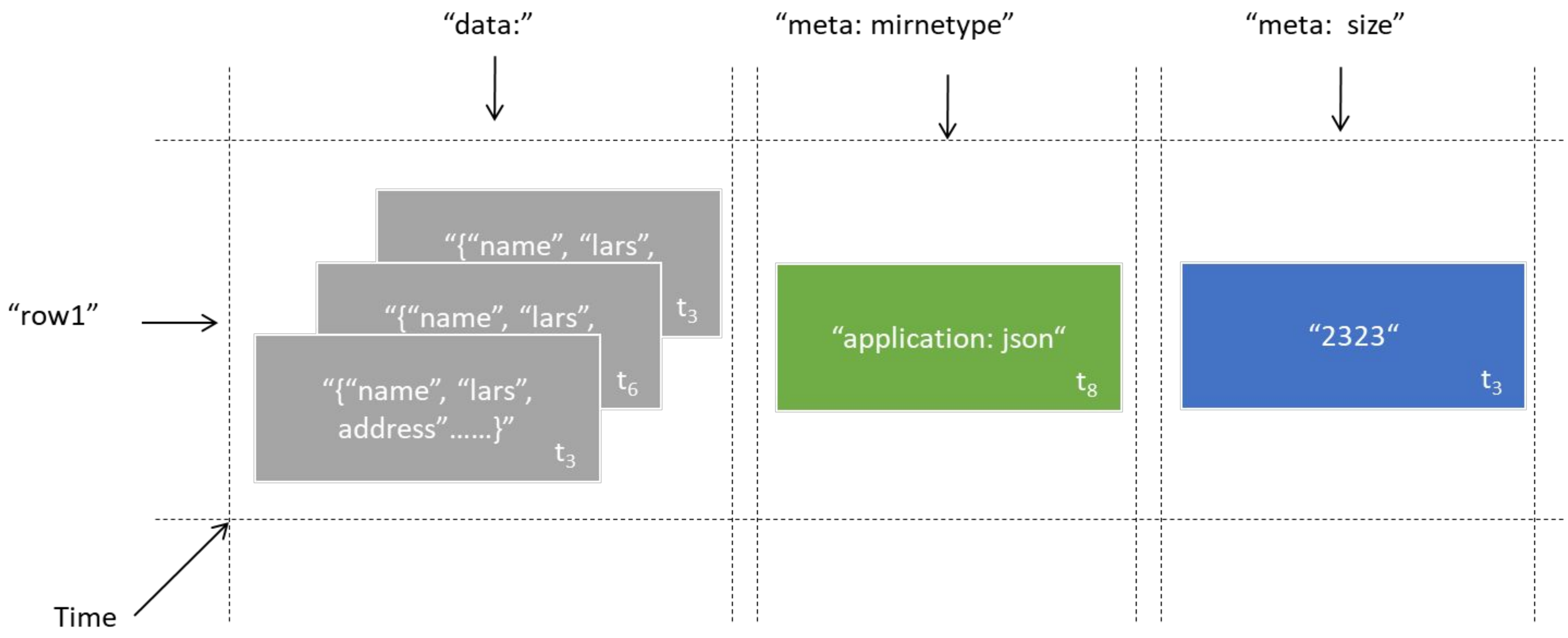
- contains the details about an HBase table

1. Column
2. Compaction
3. Durability
4. File size
5. Memstore size

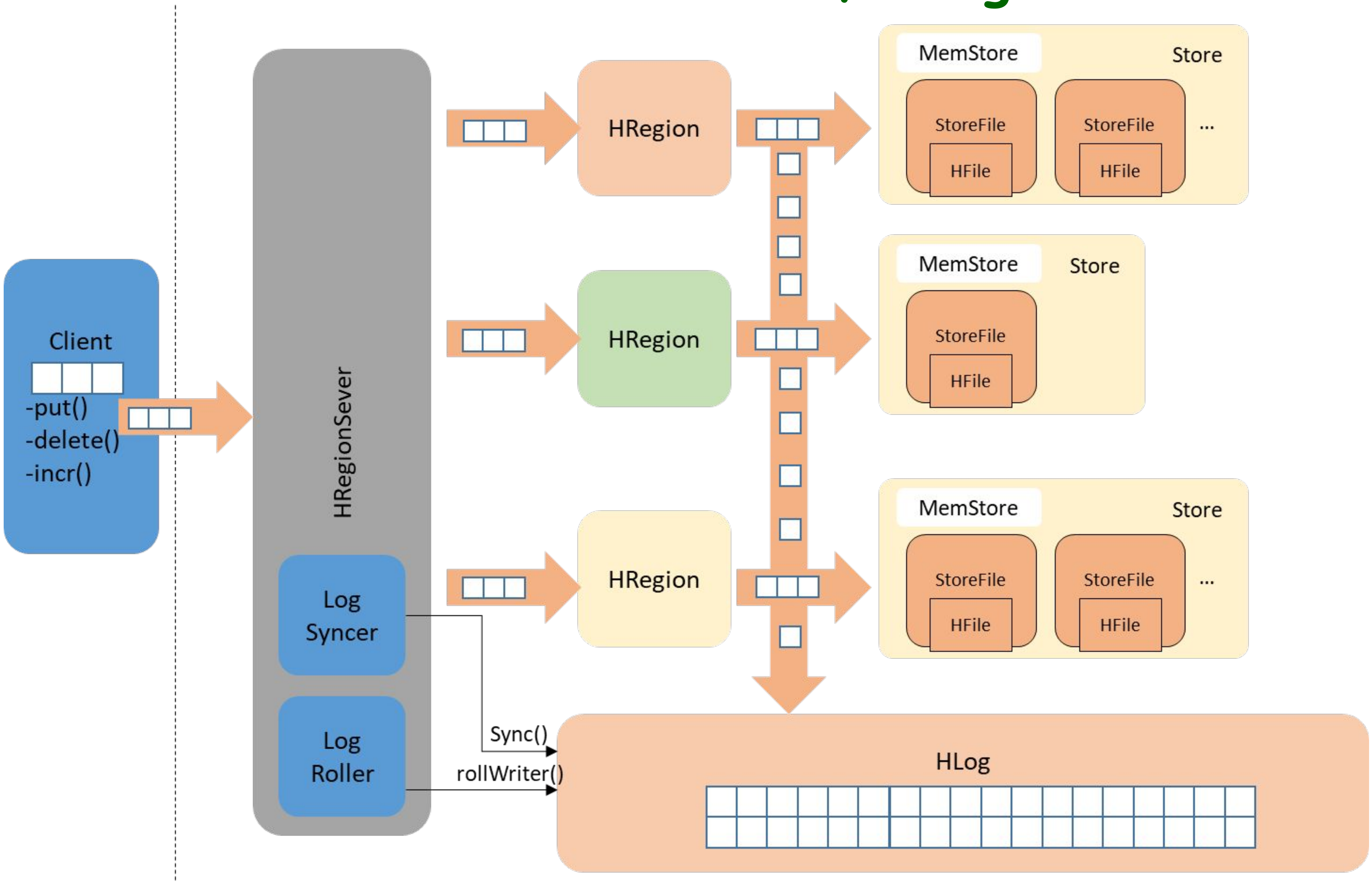
API Overview (5/5): Admin Interface

Method Name	Description
`void modifyColumn(final TableName tableName, final HColumnDescriptor descriptor) throws IOException;`	Modify a column
`void modifyTable(final TableName tableName, final HTableDescriptor htd) throws IOException;`	Modify a table
`void deleteColumn(final TableName tableName, final byte[] columnName) throws IOException;`	Deletes a column
`void deleteTable(final TableName tableName) throws IOException;`	Deletes a table
`void flush(final TableName tableName) throws IOException;`	Flush a table
`ClusterStatus getClusterStatus() throws IOException;`	Return the cluster status
`void enableTable(final TableName tableName) throws IOException;`	Enable table and wait on completion
`void disableTable(final TableName tableName) throws IOException;`	Disable table and wait on completion
`HTableDescriptor[] listTables() throws IOException;`	List all the userspace tables.
`void majorCompact(TableName tableName) throws IOException;`	Major compact a table
`void mergeRegions(final byte[] encodedNameOfRegionA, final byte[] encodedNameOfRegionB, final boolean forcible) throws IOException;`	Merge two regions

Timestamp


“data:”
“meta: mirnetype”
“meta: size”
“row1”
“{“name”, “lars”,
t3
“{“name”, “lars”,
t6
“{“name”, “lars”,
address”……}”
t3
“application: json“
t8
“2323“
t3
Time

Architecture: MemStore, HLog and HFile



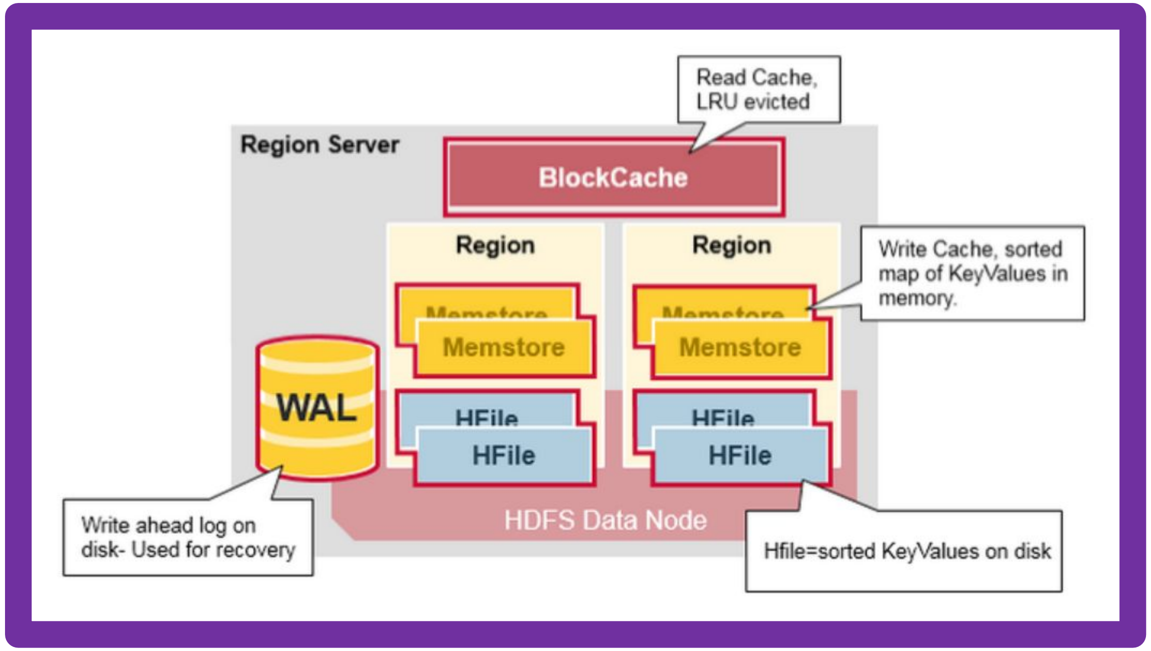
Write-Ahead Log (HLog)

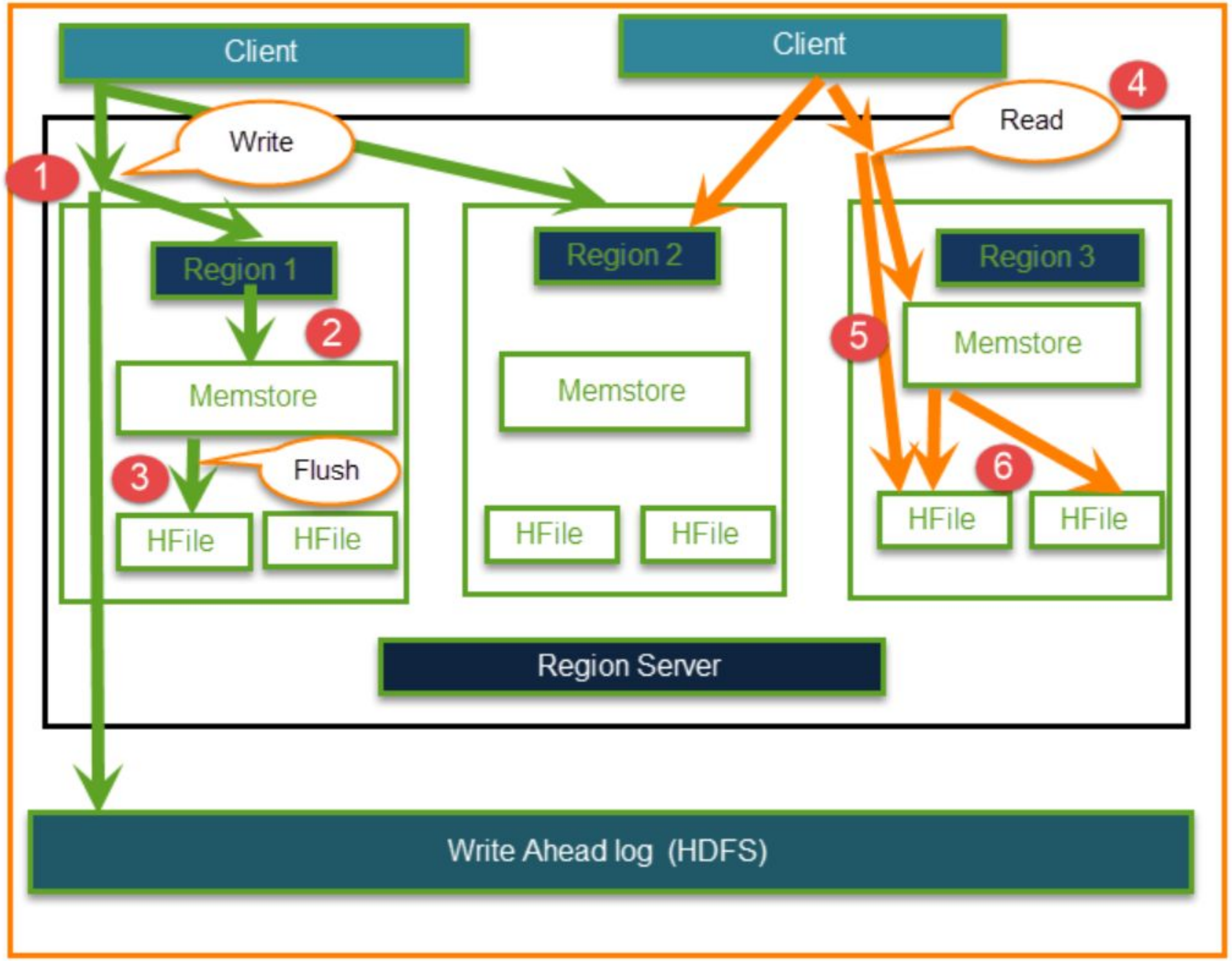
- Region servers keep data in-memory until enough is collected to warrant a flush to disk, avoiding the creation of too many very small files

- By default, each in-memory update is written to a log

- Available options:
  - **ASYNC_WAL**: Write the Mutation to the WAL asynchronously
  - **FSYNC_WAL**: Write the Mutation to the WAL synchronously and force the entries to disk
  - **SKIP_WAL**: Do not write the Mutation to the WAL
  - **SYNC_WAL** (default): Write the Mutation to the WAL synchronously

Get: Data Flow





Splitting a Region

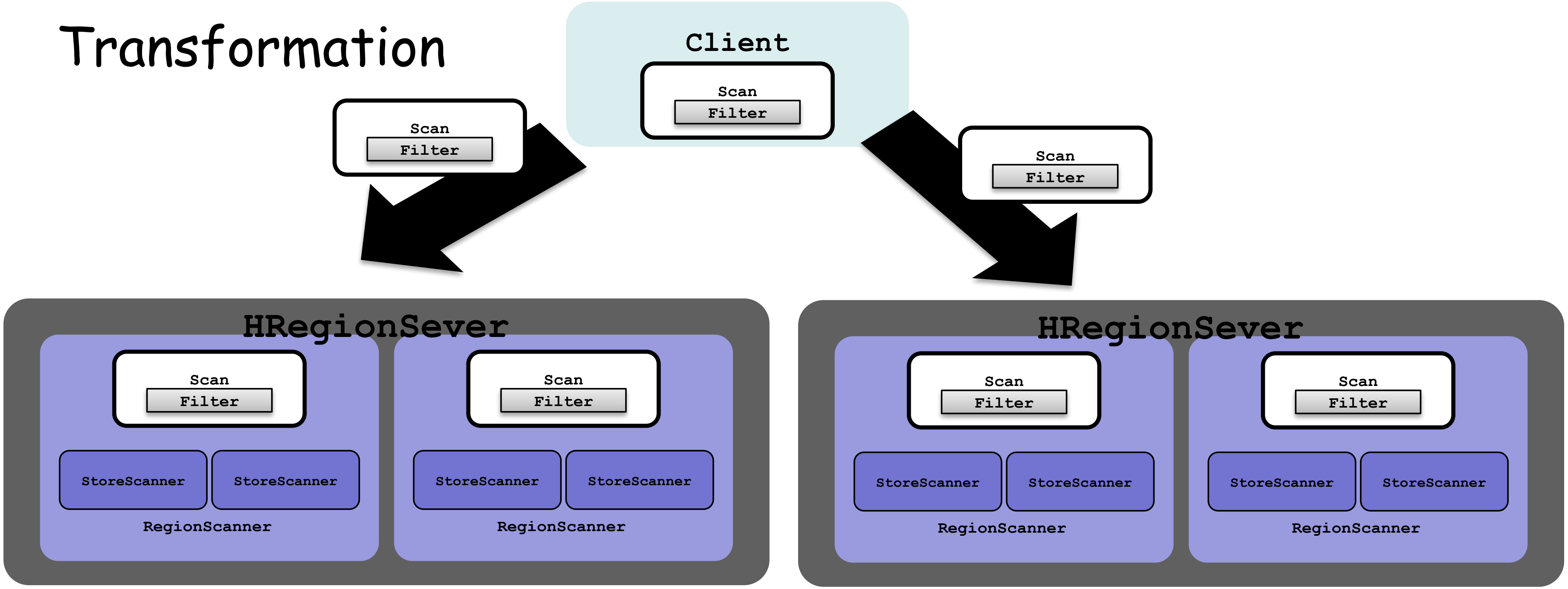
- **A split policy determines how a region should be split**
  - The fundamental unit of split is “row”

- **Various options:**
  - configured in `hbase.regionserver.region.split.policy`
  - `ConstantSizeRegionSplitPolicy` class
  - `IncreasingToUpperBoundRegionSplitPolicy` class (default)
  - `DelimitedKeyPrefixRegionSplitPolicy` class
  - `KeyPrefixRegionSplitPolicy` class
  - `DisabledRegionSplitPolicy` class

Scanner and Filter

- Used in GET() and SCAN() *API* calls

- Fine-grained control over what is returned to the client
  - Less data
  - Transformation



Available Filters (cont’d)

Class Name	Description
`ColumnCountGetFilter`	• Return first ***N*** columns on row only
`ColumnPaginationFilter`	• Based on the ColumnCountGetFilter, takes two arguments: limit and offset
`ColumnPrefixFilter`	• Select only those keys with columns that match a particular prefix
`MultipleColumnPrefixFilter`	• Select only those keys with columns that match a set of prefixes
`ColumnRangeFilter`	• Select only those keys with columns that are between **`minColumn`** to **`maxColumn`**
`CompareFilter`	• Can specify an operator (equal, greater, not equal, etc) • To filter by row key, use RowFilter. • To filter by column qualifier, use QualifierFilter. • To filter by value, use SingleColumnValueFilter.
`FirstKeyOnlyFilter`	• Return first KV from each row
`MultiRowRangeFilter`	• Scan multiple row key ranges

Available Filters (cont'd)

Class Name	Description
`FuzzyRowFilter`	• Specify (row key, fuzzy info) to match row keys, where fuzzy info equal to 0: not interested in a particular byte in a key 1: interested in a particular byte in a key
`InclusiveStopFilter`	• Stop once touching a given row
`KeyOnlyFilter`	• Return key of each KV
`PageFilter`	• Limit results in a specific page size
`PrefixFilter`	• Result values that have same row prefix
`RandomRowFilter`	• Rows that are interested with a probability
`SingleColumnValueFilter`	• Filter cells based on value
`SkipFilter`	• Filter an entire row if any of the Cell checks fails

Filter on Your Own

Method Name	Description
`abstract public void reset() throws IOException;`	Reset the state of the filter between rows
`abstract public boolean filterRowKey(byte[] buffer, int offset, int length) throws IOException;`	Filters a row based on row key
`abstract public boolean filterAllRemaining() throws IOException;`	If true, the scan will terminate
`abstract public ReturnCode filterKeyValue(final Cell v) throws IOException;`	Way to filter based on the column family, column qualifier and/or the column value
`abstract public Cell transformCell(final Cell v) throws IOException;`	Give the filter a chance to transform the passed Key-Value
`abstract public void filterRowCells(List<Cell> kvs) throws IOException;`	Alter the contains of specified Cells
`abstract public boolean hasFilterRow();`	Primarily used to check for conflicts with scans such as scans that do not read a full row at a time
`abstract public boolean filterRow() throws IOException;`	Last chance to filter row based on previous **`filterKeyValue(Cell)`** calls
`abstract public Cell getNextCellHint(final Cell currentCell) throws IOException;`	If the filter returns **`SEEK_NEXT_USING_HINT`**, then it should also tell which is the next key it must seek to
`abstract public boolean isFamilyEssential(byte[] name) throws IOException;`	Check a given column family whether is essential to filter
`abstract public byte[] toByteArray() throws IOException;`	Serialize filter to byte array
`public static Filter parseFrom(final byte [] pbBytes) throws DeserializationException`	Failure signal

Coprocessor

- **An alike MapReduce framework that distributes work across the entire cluster (such as filters)**
  - Enable to run arbitrary code directly on each region server

- **Types of coprocessor:**

Observer: comparable to triggers (or callback functions) that are executed when certain events occur

- `RegionObserver` class
- `MasterObserver` class
- `WALObserver` class

Endpoint: user code can be deployed to the servers hosting the data to perform server-local computations

More Features

Feature	Description
Replication	• copy data between HBase deployments
Bloom filter	• predict whether a given element is a member in a set of data
Metrics	• expose a large number of metrics that detail present status
Compression	• compression algorithms for an Hfile
Compaction	• combine HFiles to a few, larger Hfiles
MapReduce Integration	TableInputFormat • Convert HBase tabular data into a format that can be understood by Map/Reduce TableOutputFormat • Convert Map/Reduce output to an HBase table

Compactions (Per Region Based)

- **Minor compaction**
  - 將幾個小的Hfiles整理成一個大的Hfile (to minimize the seek time)

- **Major compaction**
  - HBase特別適合大量資料的寫入，持續產生Hfiles，致使反覆被update的資料衍生數個版本分佈在不同的HFiles裡；同時也移除delete marks!
  - 浪費儲存空間，也增加搜尋資料的時間
  - 希望 "所有HFiles" 裡每筆相異資料的版本個數為使用者所指定，同時也希望所有的Hfiles裡的已經按照keys的global order整理

- **Compaction is a background process**

.META Table

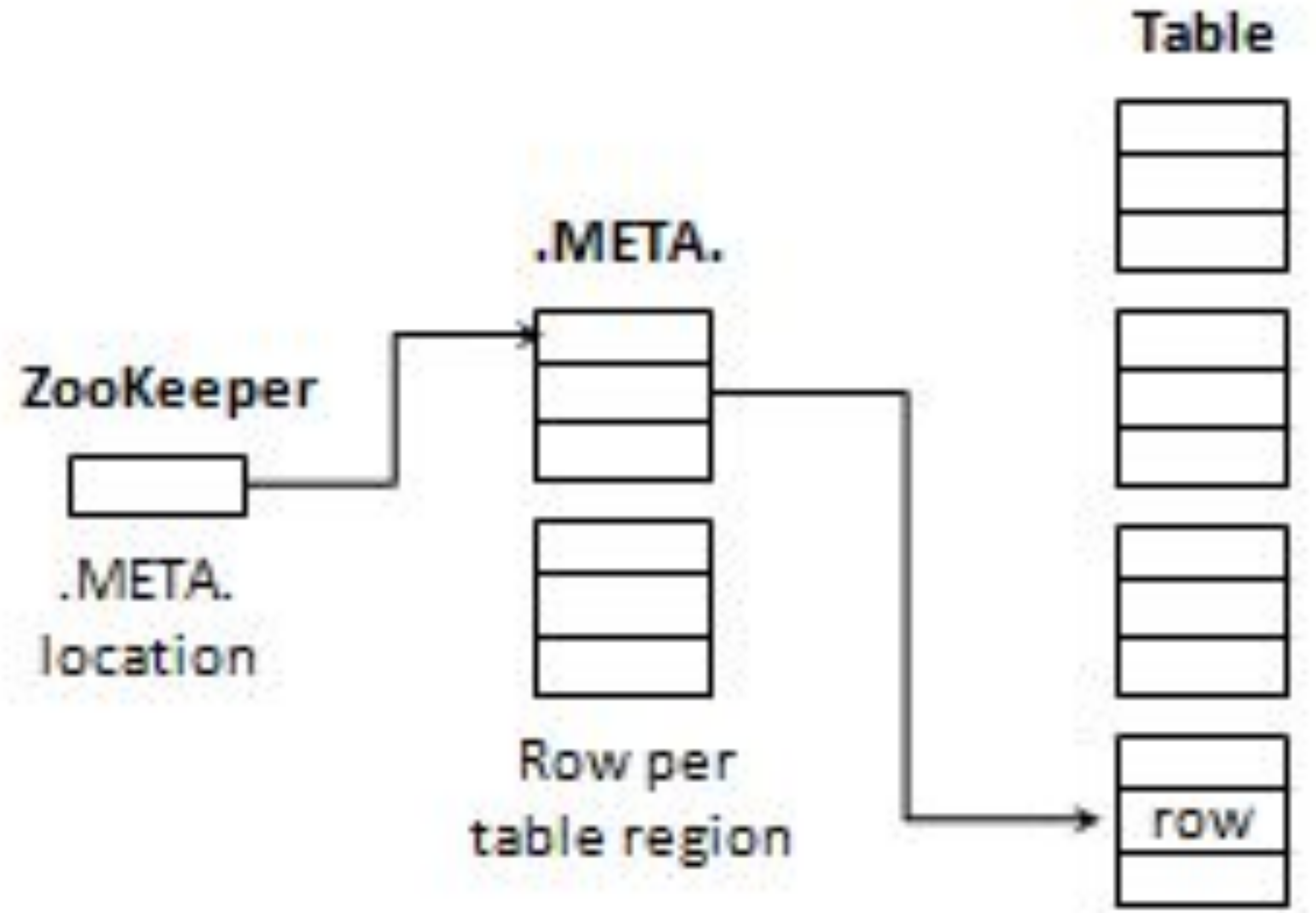
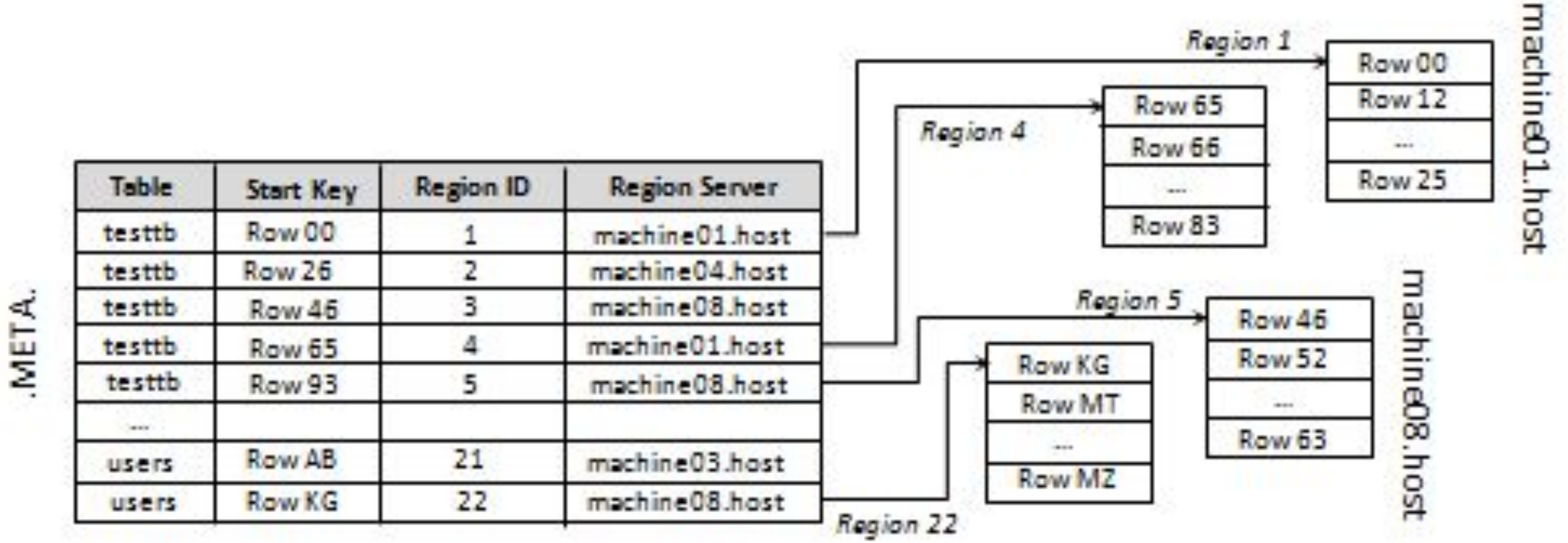


Table
.META.
ZooKeeper
.META.
location
Row per
table region
row

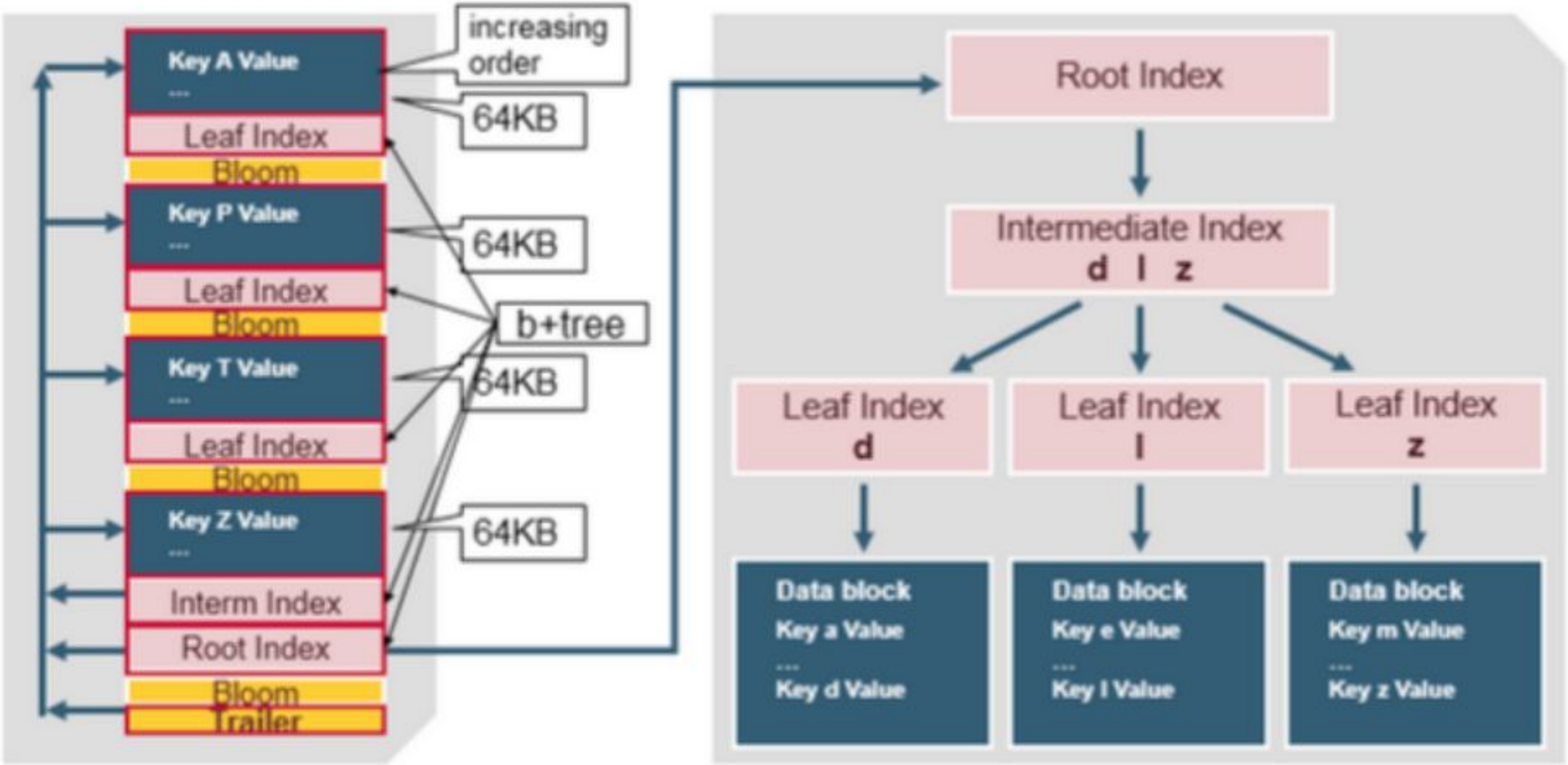
.META Table (cont'd)



.META.

Table	Start Key	Region ID	Region Server
testtb	Row 00	1	machine01.host
testtb	Row 26	2	machine04.host
testtb	Row 46	3	machine08.host
testtb	Row 65	4	machine01.host
testtb	Row 93	5	machine08.host
...			
users	Row AB	21	machine03.host
users	Row KG	22	machine08.host

HFile Structure



Bloom Filters

- A concise representation of a set S with m elements using a n-bit vector, given:
  - Denote by $S = \{x_1, x_2, x_3, \cdots, x_m\}$
  - k hash functions (denoted by H_i, where $1 \leq i \leq k$), each picking a value from $\{1, 2, 3, \cdots, n\}$ uniformly at random
- The n-bit standard Bloom filter ($\mathbb{B}$) is initially set to $0_1 0_2 0_3 \cdots 0_n$
- Insert an element y into $\mathbb{B}$: set bit $H_i(y) = 1$ for all $i = 1, 2, \cdots, k$

Example

$1_1 0_2 0_3 1_4 1_5$ for $n = 5$ and $k = 3$

- An element $y \in \mathbb{B}$ if $H_i(y) = 1$ for all $i = 1, 2, \cdots, k$

Bloom Filters (cont'd)

- False positive: An element $y \notin \mathbb{B}$ is claimed in $\mathbb{B}$
- The probability of a specific bit is zero is

$$p = \left(1 - \frac{1}{n}\right)^{km} \approx e^{-\frac{km}{n}} \tag{1}$$

- The false positive rate f is

$$f = (1 - p)^k \tag{2}$$

Example

$f \approx 0.02$ if $n = 8m$ and $k = 5$ (or $k = 6$)

Bloom Filters (cont'd)

- Observation:
  - Larger k, more chance to find a 0-bit for $y \notin S$
  - Smaller k, more fraction of 0 bit and thus smaller f

Theorem

Given n and m, k minimizes f if

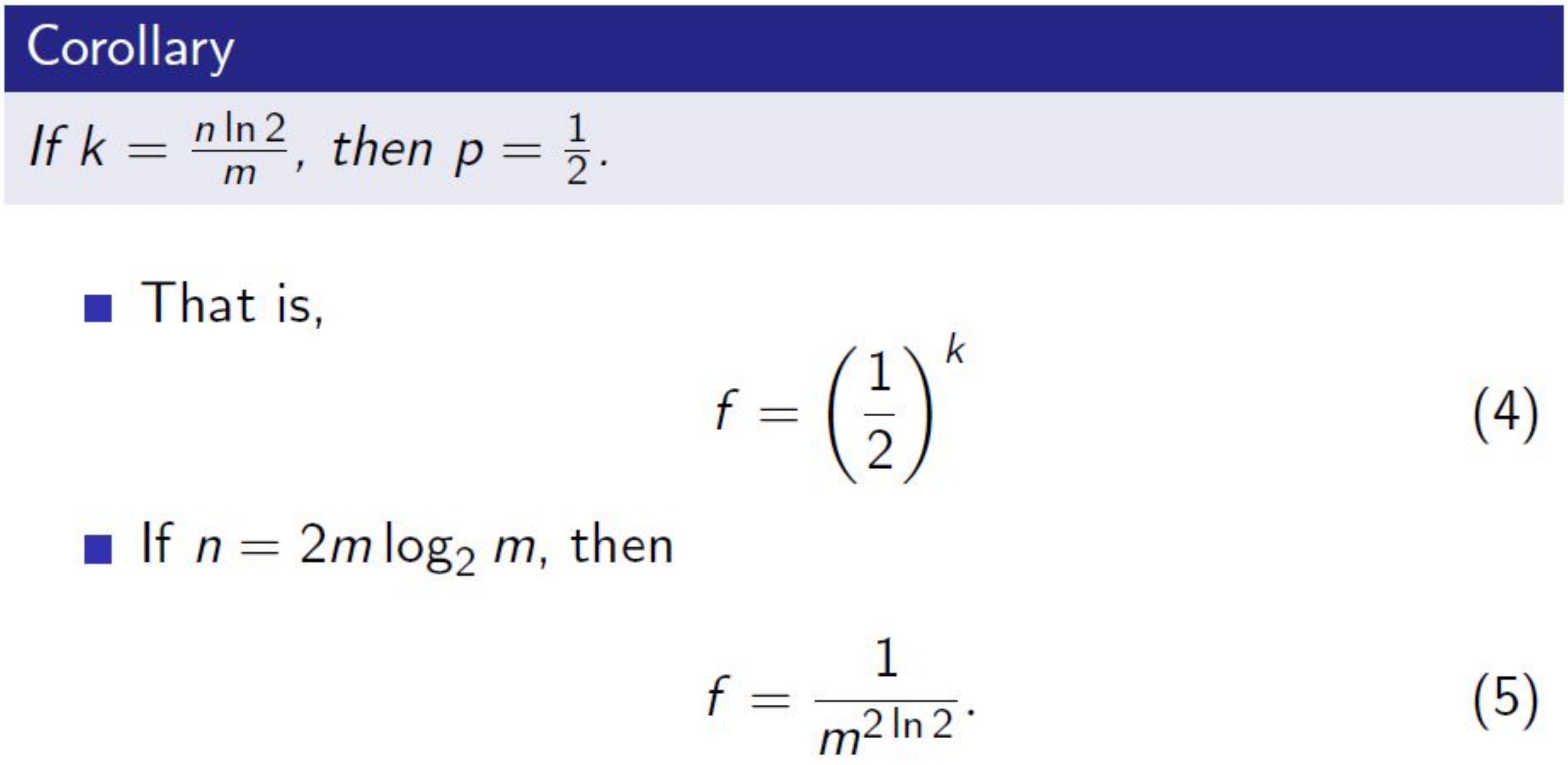
$$k = \frac{n \ln 2}{m}. \tag{3}$$

Proof.

1. $f = (1-p)^k = e^g$, where $g = k \ln(1-p)$.
2. Minimizing f is identical to minimize g, and let $\frac{dg}{dk} = 0$.

□

Bloom Filters (cont'd)



Corollary

If $k = \frac{n \ln 2}{m}$, then $p = \frac{1}{2}$.

- That is,

$$f = \left(\frac{1}{2}\right)^k \tag{4}$$

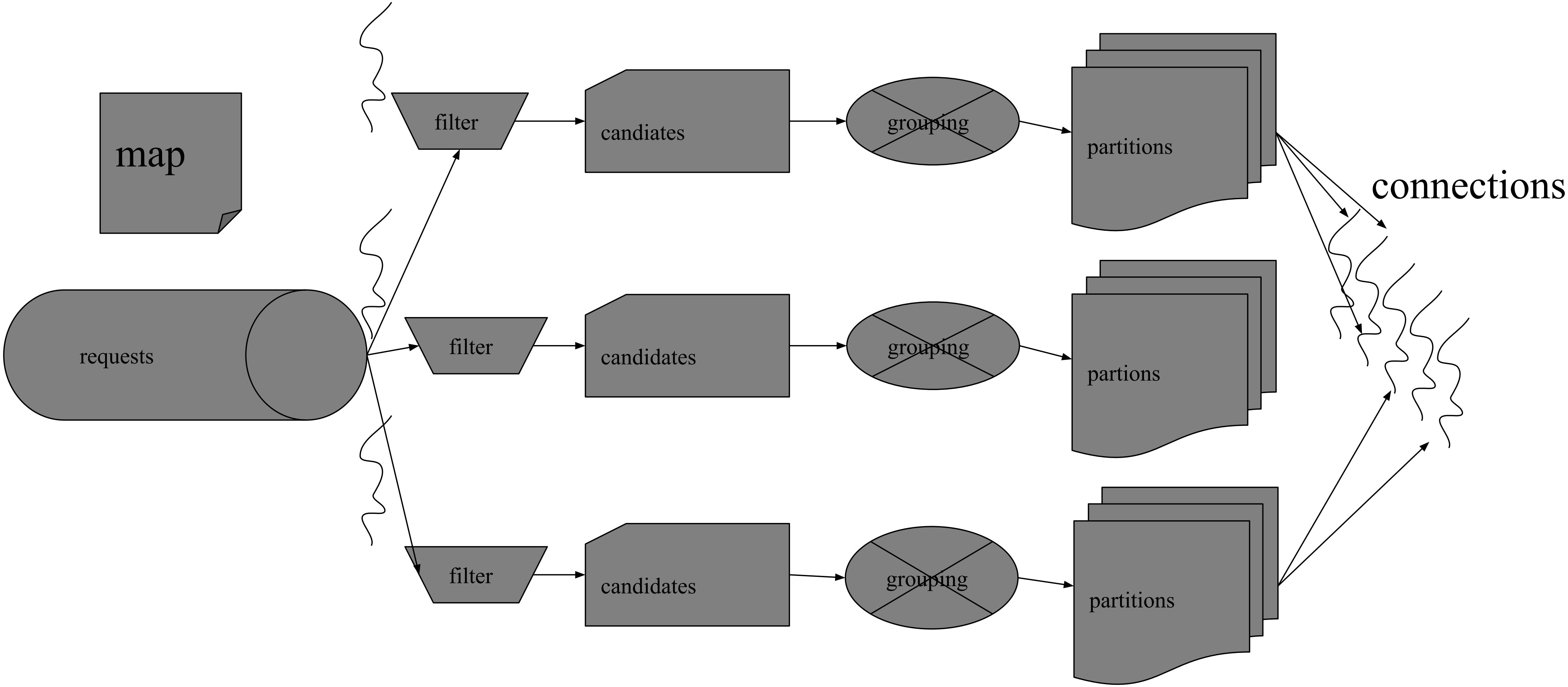
- If $n = 2m \log_2 m$, then

$$f = \frac{1}{m^{2 \ln 2}}. \tag{5}$$

Bloom Filters in HBase

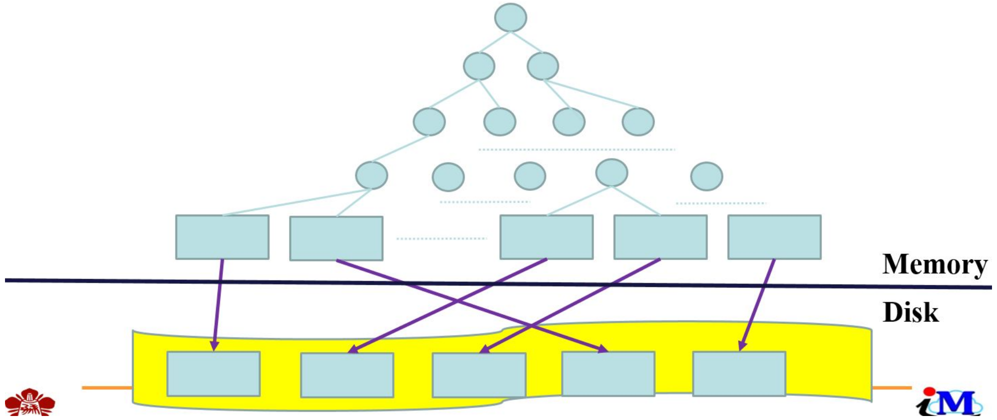
- Bloom filter B:
  - B一個bit vector表示一個集合的元素
  - 元素Y in B: Y可能在B裡，Y不在B裡的機率是f (false positive probability)
  - 元素Y not in B: Y一定不在B裡
- HFile裡存了若干個bloom filter用來表示該file裡所有出現過的keys
- Given a key k and a Hfile h，我們不確定要查找的k是否在h裡，HBase region server會先檢查該h的bloom filter，若檢查後發現h沒有k，則h就不會是要查找k所對應value的對象
- Cached在region servers的記憶體裡

Client-Side Connections and Framework



HBase的適用場合

- 勿將之當作traditional relational database (RDB)來使用, 在big data大海裡撈針
- Against RDB



- 放棄transactions操作, due to CAP theorem and PAXOS
- 適合連續row keys且大量的資料操作應用, 例如AI model training時的data access (適合大量且批次的讀寫)
- 極適合用來存time-series data或structured log (e.g., CSV)

Experiences and Lessons

- HBase is a good solution conquering Moore's Law
- HBase is "extremely" flexible, but complex: dozens of parameters/design options to consider
- To manipulate a distributed database management system in a production state (specifically for big data) needs a team
  - Our lab: HBase storage engine (4 persons), Phoenix SQL (3) and R (2) each yr
- Issues: speed vs space, bug fix, extra functionalities expansion, backup and recovery, learning curve (for administrators and application developers), …
- Cooperation: ITRI, III, Delta, ETC, UMC and ASE

Thank You

40

Advanced (1/5): Load Balancer

- **Makes decisions for placement and movement of regions across RegionServers**
  - Cluster-wide load balancing will occur only when there are no regions in transition and according to a fixed period of a time

- **By default, being executed every five minutes**
  - `hbase.balancer.period`

- **A number of load balancers implemented:**
  - configured in `hbase.master.loadbalancer.class`
  - `SimpleLoadBalancer` class
  - `FavoredNodeLoadBalancer` class
  - `StochasticLoadBalancer` class (default)

Load Balancers (cont'd)

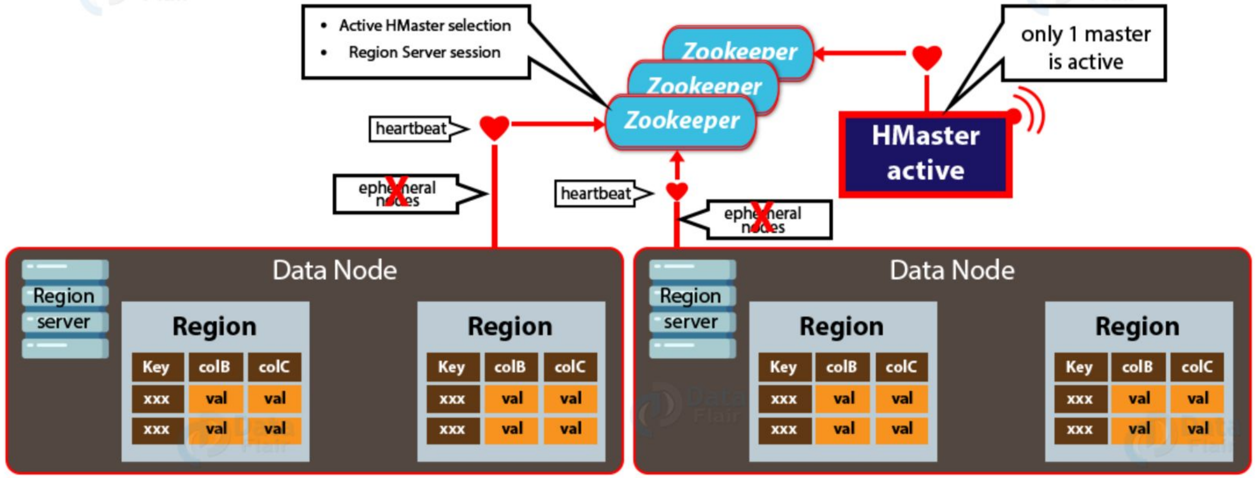
Class Name	Description
`FavoredNodeLoadBalancer`	• Assign favored nodes for each region • Roles: primary RegionServer, secondary and tertiary RegionServers
`SimpleLoadBalancer`	• Invariant: number of regions each server manages shall be <= average +/- 1
`StochasticLoadBalancer`	• Given a cost function F(C) => x, the cluster will randomly try and mutate to Cprime • If F(Cprime) < F(C), then switch the cluster to Cprime • Cost function F() refers to: • region load • table load • data locality • memstore sizes • storefile sizes

Split Policies (cont'd)

Class Name	Description
`ConstantSizeRegionSplitPolicy`	• Split a region as soon as any of its store files exceeds a maximum configurable size
`IncreasingToUpperBoundRegionSplitPolicy`	• Split size is increased proportionally to (number of regions in a column store)^3
`DelimitedKeyPrefixRegionSplitPolicy`	• Group rows by a prefix of the row-key with a delimiter
`KeyPrefixRegionSplitPolicy`	• Group rows by a prefix of the row-key
`DisabledRegionSplitPolicy`	• Disables region splits

HMaster

- Responsible for administrative commands
- Load balancing by migration regions
- Handle failures of region servers and regions recovery
- Note: HMaster checks the aliveness of region servers through Zookeeper
  - Region servers heartbeat w



• Active HMaster selection
• Region Server session
only 1 master is active
Zookeeper
Zookeeper
Zookeeper
HMaster active
heartbeat
heartbeat
ephemeral nodes
ephemeral nodes
Data Node
Data Node
Region server
Region server
Region
Key
colB
colC
xxx
val