Real Time Micro-Blog Summarization based on Hadoop/HBase

-Sanghoon Lee, Sunny Shakya

Outline

❑ Introduction

- Hadoop
- HDFS
- MapReduce
- HBase
- The Big Picture

❑ HBase Operation

❑ Application

- Twitter
- Application Architecture
- Demo

What is Apache Hadoop?

- ❑ Open source framework that supports data intensive distributed applications
- ❑ Created by Doug Cutting, the creator of Apache Lucene.
- ❑ Derived from Google's MapReduce and Google File System (GFS) papers.
- ❑ Solution for Big Data
  - ▪ Deals with complexities of high volume, velocity and variety of data
- ❑ Transforms commodity hardware into services that
  - ▪ Store petabytes of data reliably
  - ▪ Allows huge distributed computations

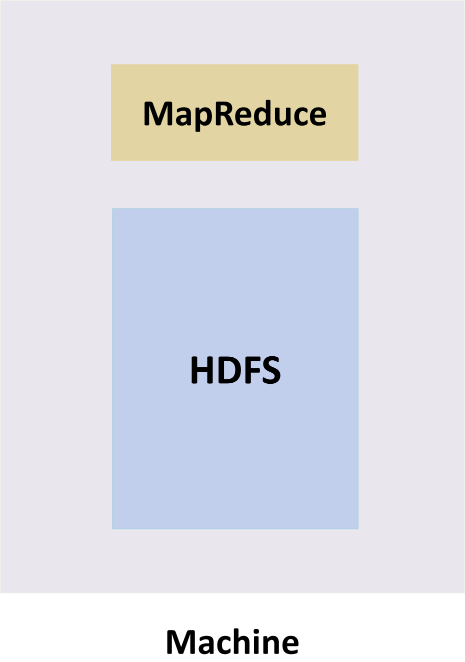
What is Apache Hadoop?

❑ Key Attributes

- Redundant and reliable (no data loss)
- Extremely powerful
- Batch processing centric
- Easy to program distributed applications
- Run on commodity hardware.
- Easily Scalable

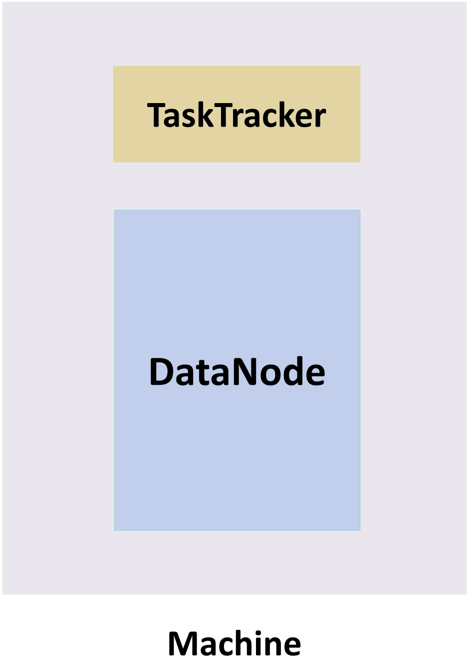
What is Apache Hadoop?

- ❑ MapReduce is the processing part of Hadoop
- ❑ HDFS is the data part of Hadoop



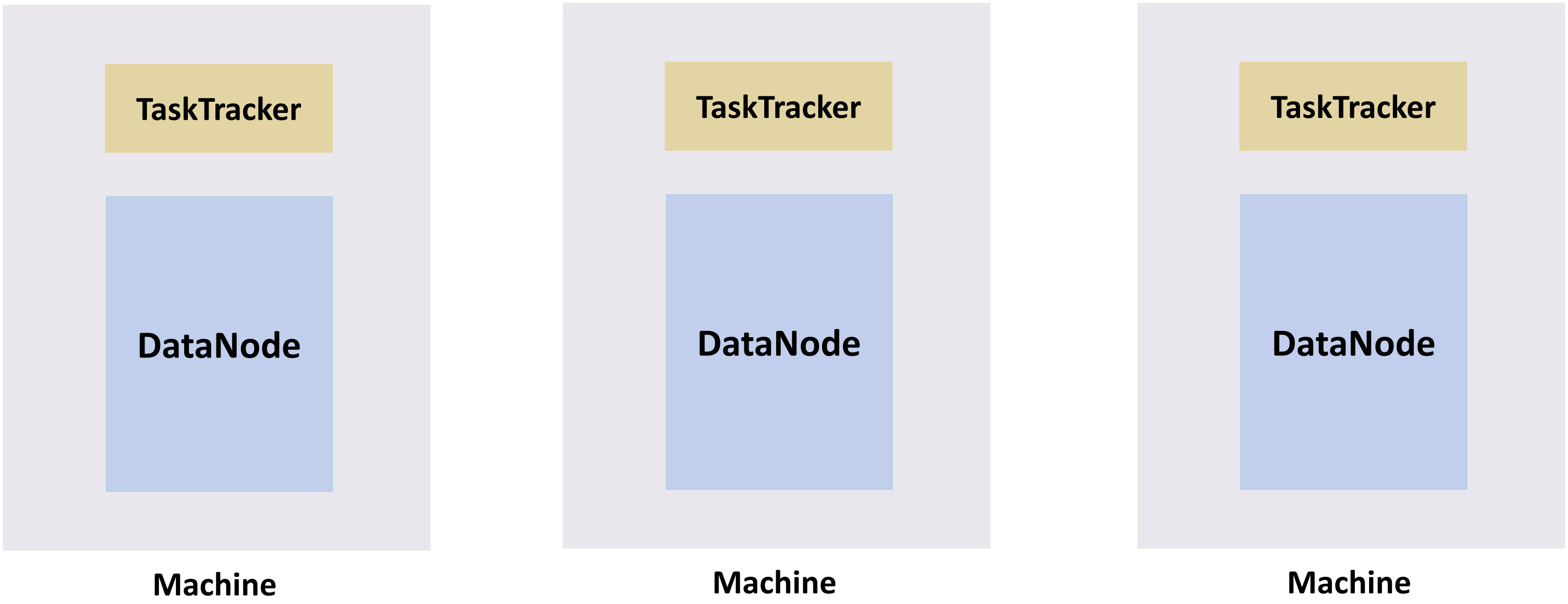
What is Apache Hadoop?

- ❑ The MapReduce server on a typical machine is called a TaskTracker
- ❑ The HDFS server on a typical machine is called a DataNode



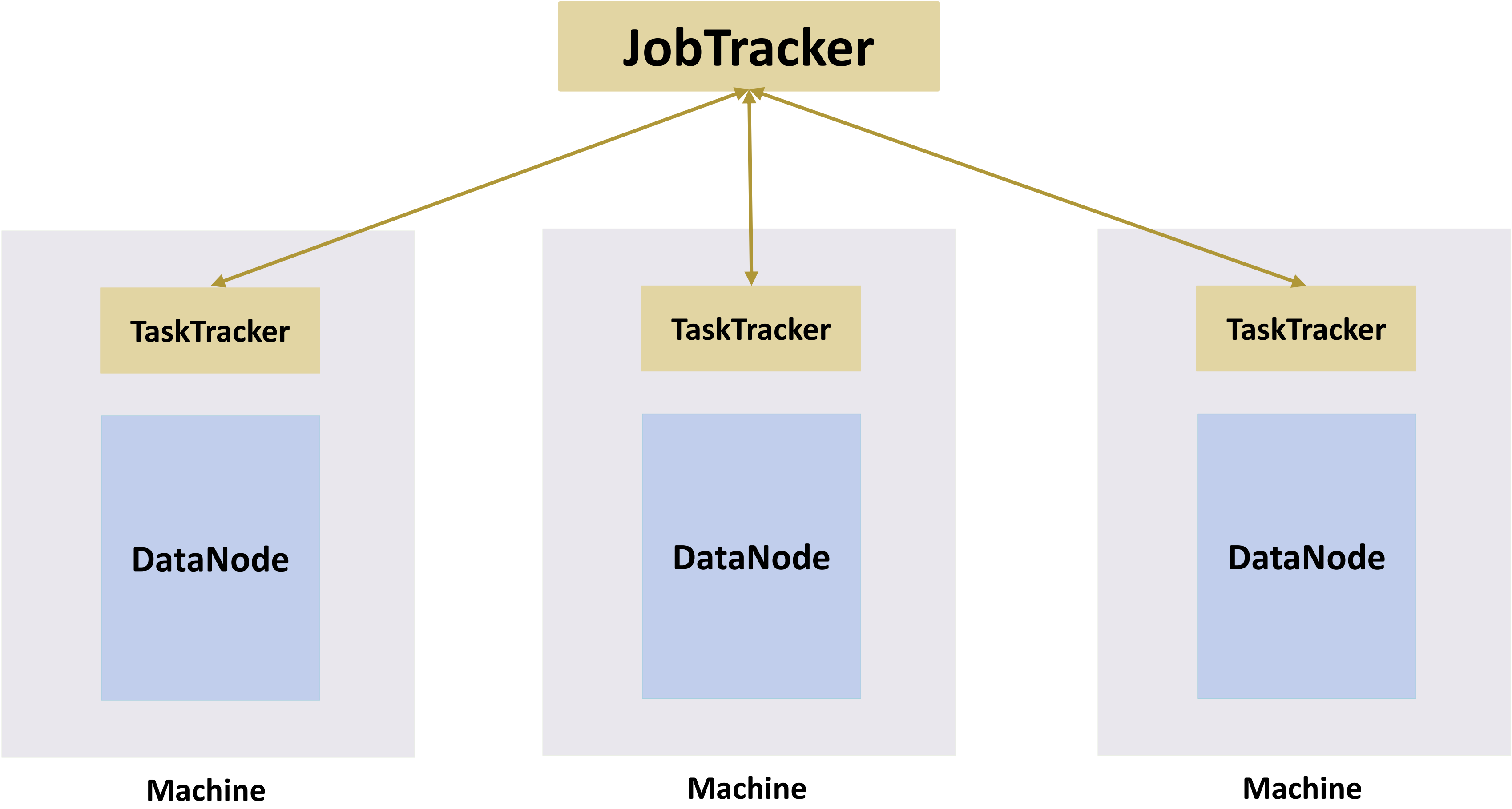
What is Apache Hadoop?

- ❑ Having multiple machines with Hadoop creates a cluster



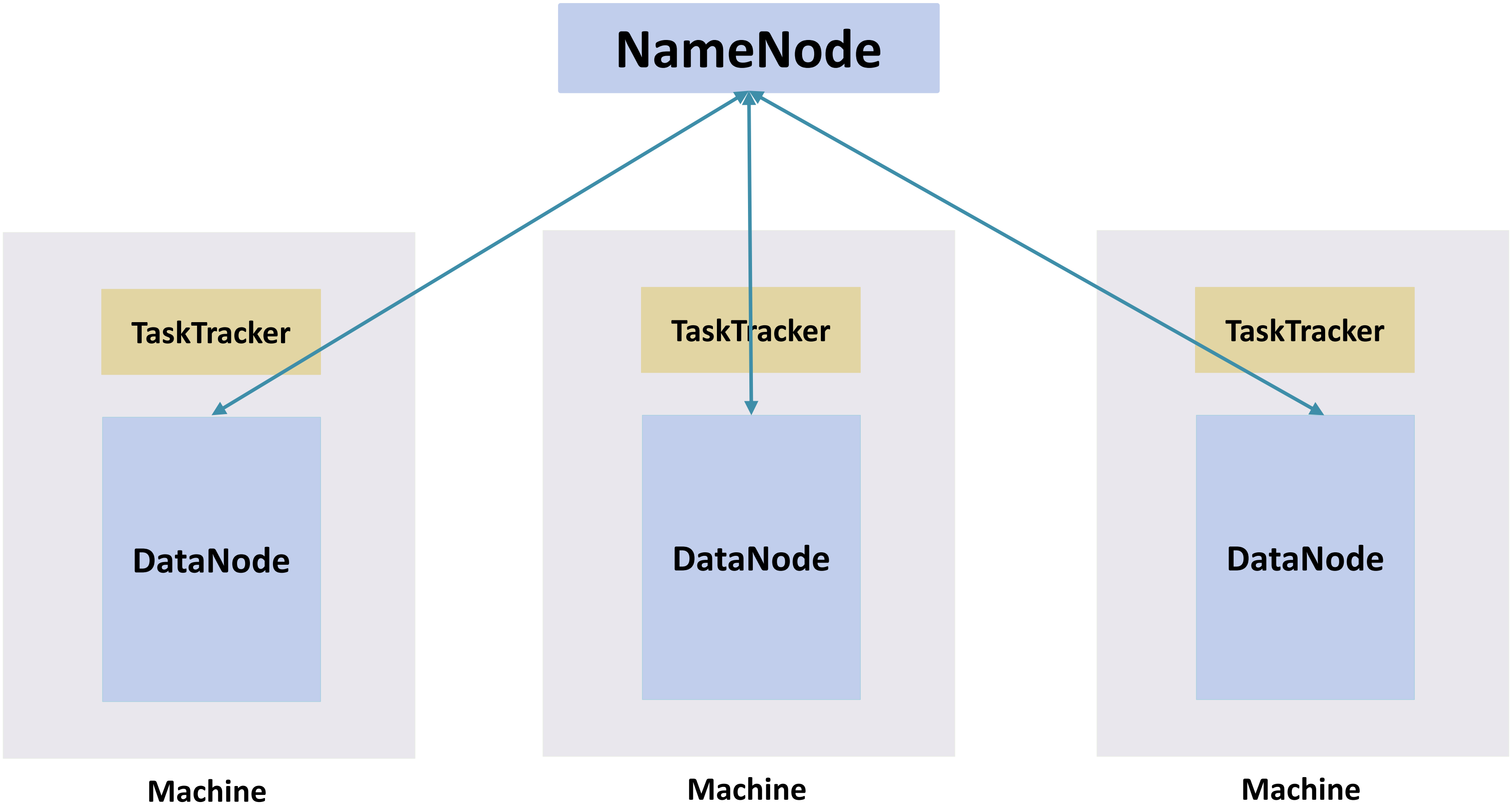
What is Apache Hadoop?

- ❑ JobTracker keeps track of jobs being run



What is Apache Hadoop?

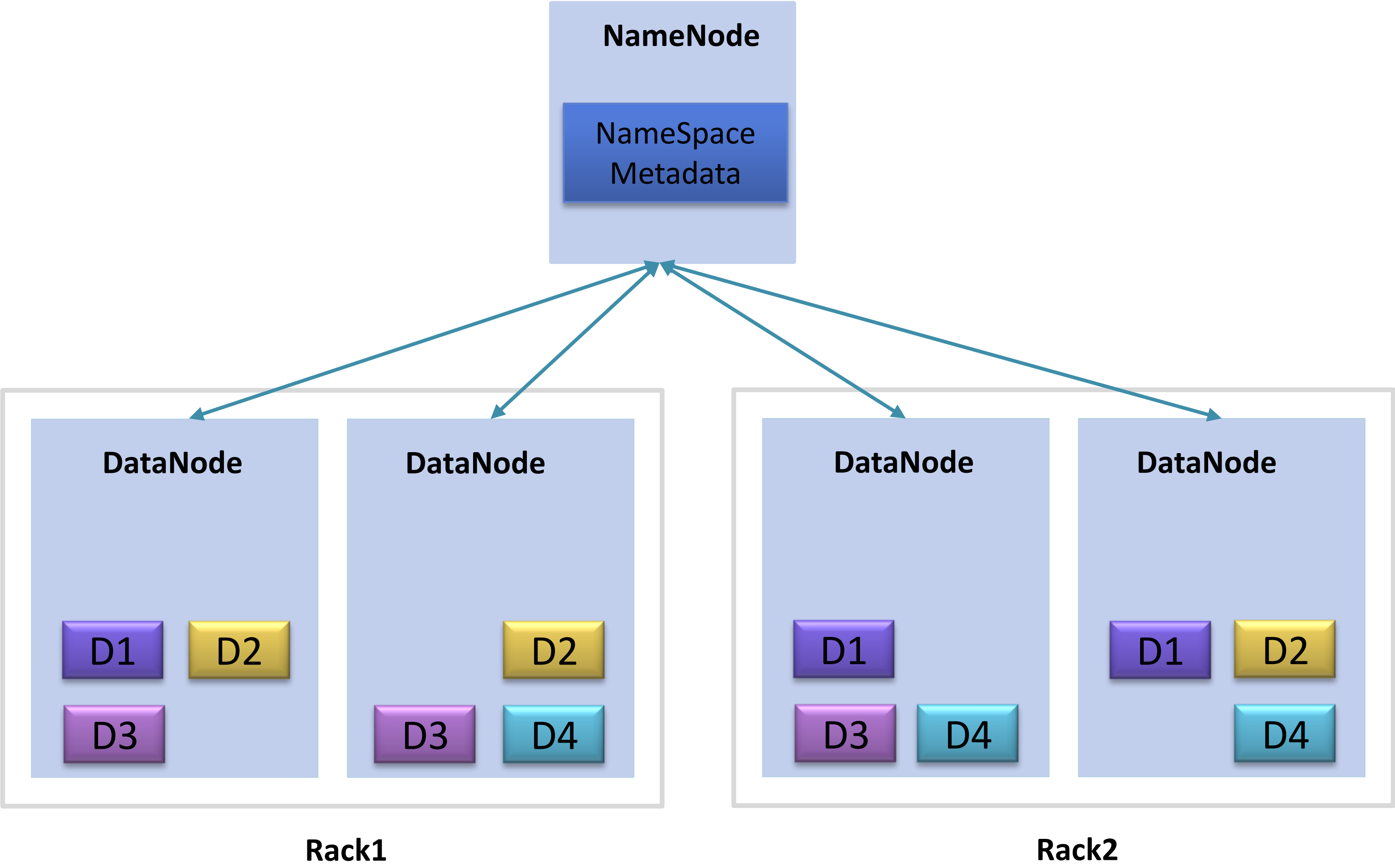
- NameNode keep information about data location



HDFS

- Scalable, Reliable and Manageable
- Highly scalable file system
  - Adds commodity servers and disks to scale storage and IO bandwidth
  - Supports parallel reading and processing of the data
    - Read, Write, Rename and Append
    - Optimized for streaming reads/writes of large files
    - Bandwidth scales linearly with the number of nodes and disks
  - Fault Tolerant and Easy manageable
    - Built-in redundancy
    - Tolerates nodes and disk failures
    - Automatically manages addition/removal of nodes

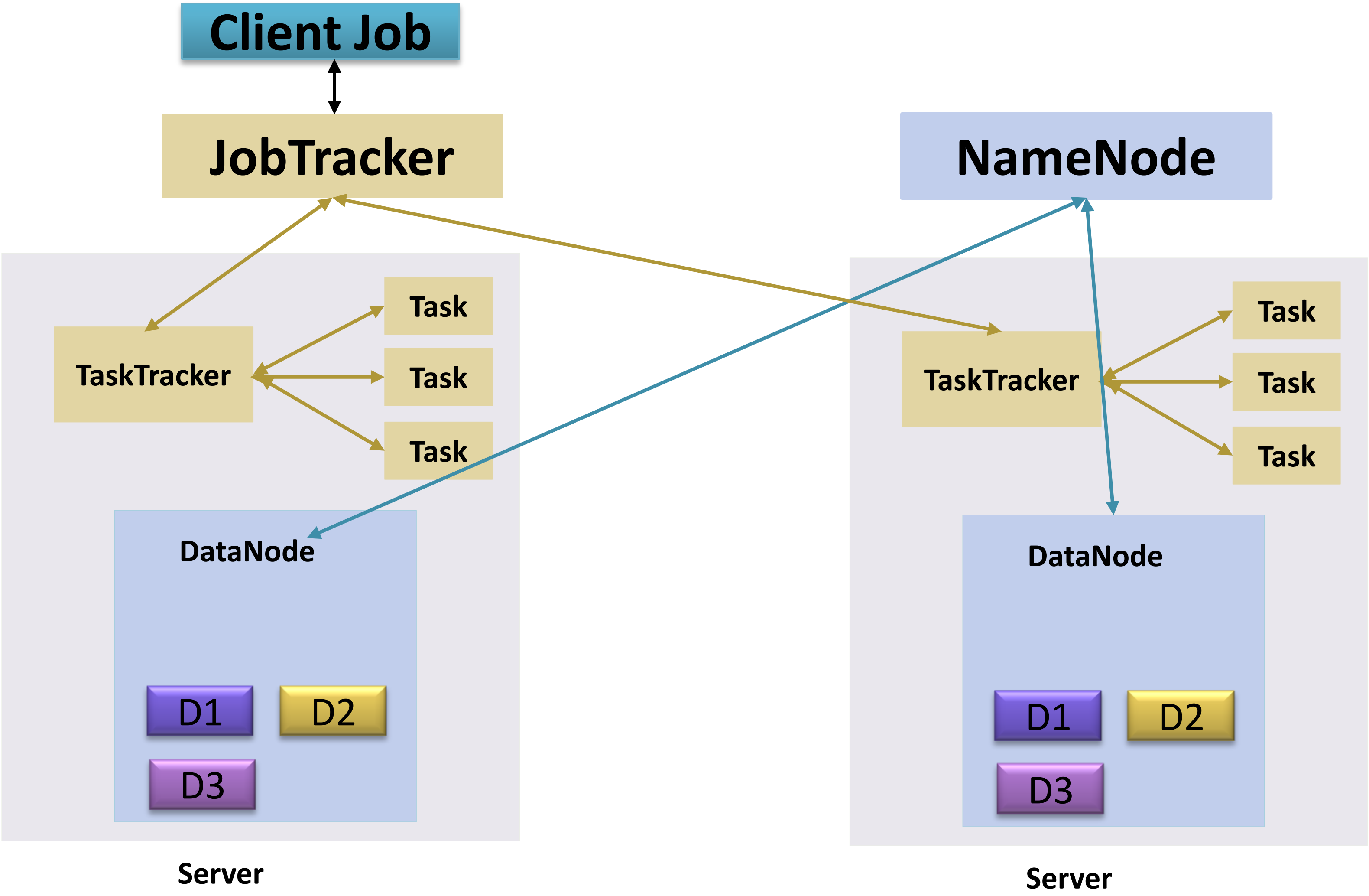
HDFS


NameNode
NameSpace Metadata
DataNode
D1
D2
D3
DataNode
D2
D3
D4
DataNode
D1
D3
D4
DataNode
D1
D2
D4
Rack1
Rack2

HDFS and its Uses

- HDFS provides a reliable, scalable and manageable solution for working with huge amount of data
- HDFS has been successfully deployed in clusters of 10 – 4500 nodes and can store up to 25 petabytes of data

MapReduce


Client Job
JobTracker
NameNode
TaskTracker
Task
Task
Task
DataNode
D1
D2
D3
Server
TaskTracker
Task
Task
Task
DataNode
D1
D2
D3
Server

MapReduce

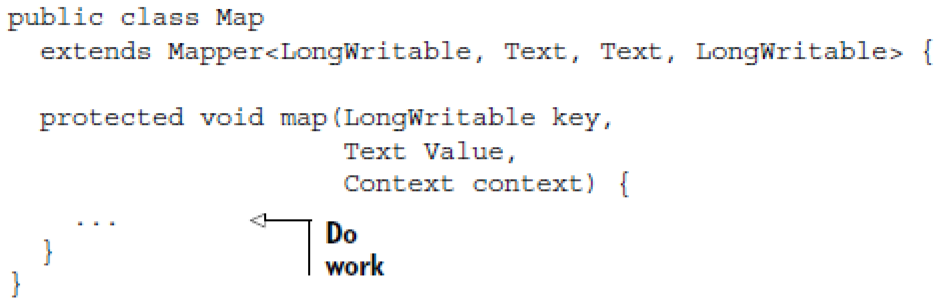
❑ Map Step

- The master node takes the input, divides it into smaller sub-problems, and distributes them to worker nodes.
- The worker node processes the smaller problem, and passes the answer back to its master node.

❑ Reduce Step

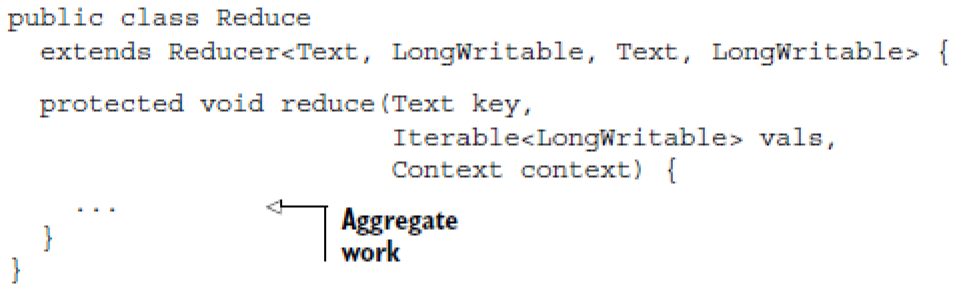
- The master node then collects the answers to all the sub-problems and combines them in some way to form the output – the answer to the problem it was originally trying to solve.

MapReduce



```
public class Map
  extends Mapper<LongWritable, Text, Text, LongWritable> {

  protected void map(LongWritable key,
                     Text Value,
                     Context context) {
    ...
  }
}
```

Do work



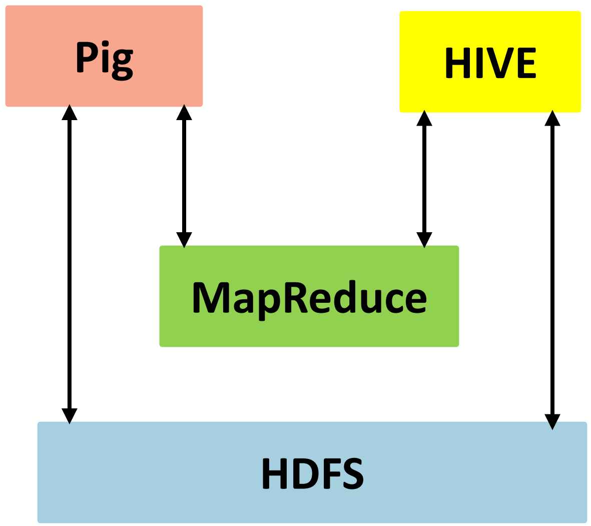
```
public class Reduce
  extends Reducer<Text, LongWritable, Text, LongWritable> {

  protected void reduce(Text key,
                        Iterable<LongWritable> vals,
                        Context context) {
    ...
  }
}
```

Aggregate work

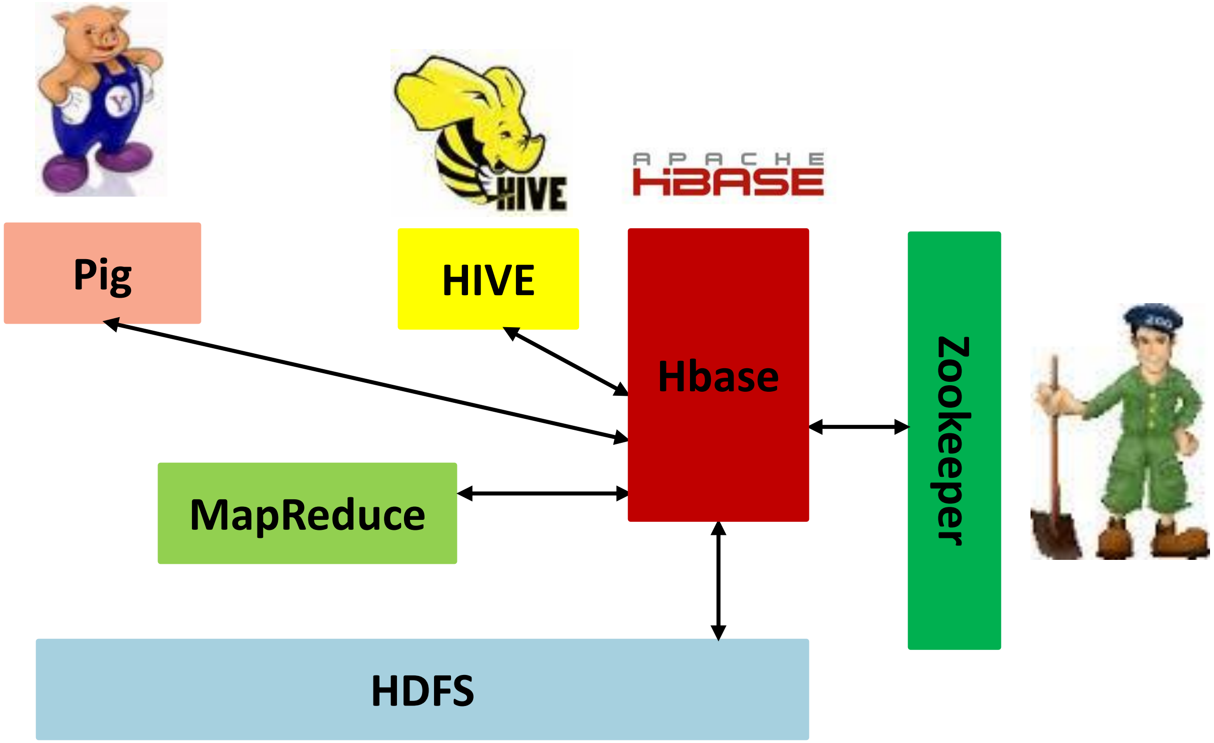
What is Apache Hadoop?

- ❑ Hadoop is
  - ❑ Reliable
    - Data is held in multiple locations
    - Tasks that fail are redone
  - ❑ Scalable
    - Same program runs on 1, 1000 or 4000 machines
    - Scales linearly
  - ❑ Simple APIs
  - ❑ Very powerful
    - You can process in parallel massive amount of data
      - Petabytes of data
    - Processing in parallel allows for the timely processing of massive amount of data

What is Apache Hadoop?



What is Apache Hadoop?



HBase

- Distributed column-oriented database built on top of HDFS
- Not relational and does not support SQL and is designed to run on a cluster of computers with scalability and ability to deal with any type of data in mind
- HBase is often described as a schema-less database.
- HBase is designed to run on a cluster of computers instead of a single computer.

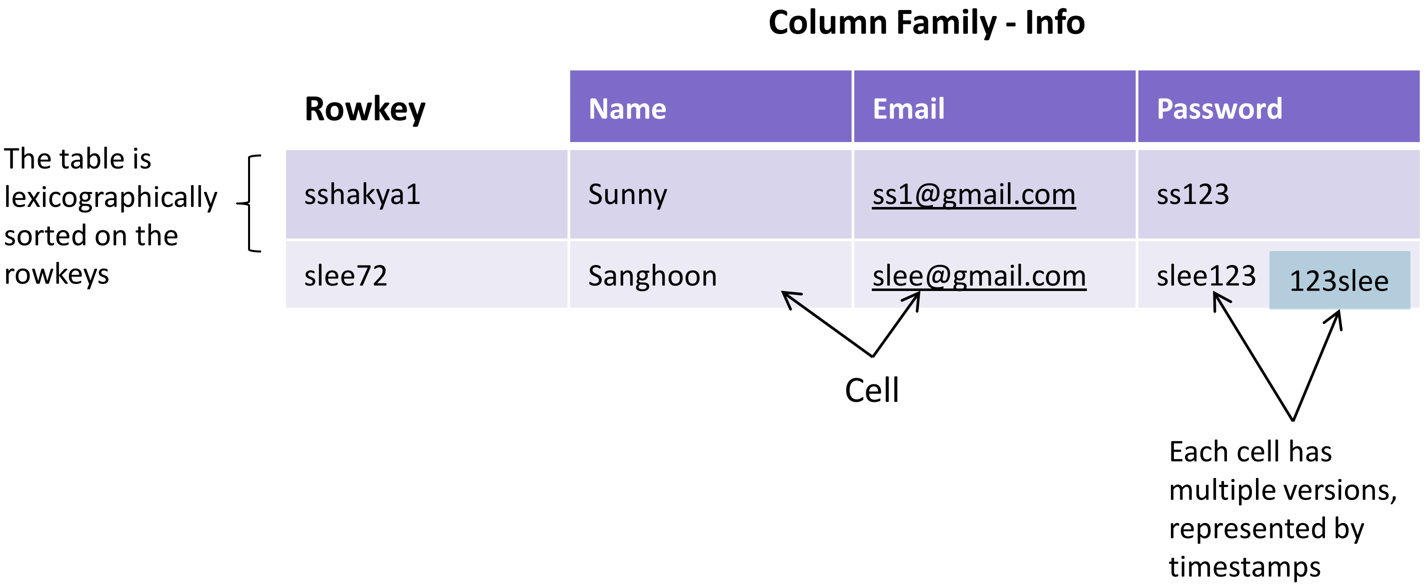
HBase

- ❑ HBase depends on Hadoop primarily for two reasons
  - Hadoop MapReduce provides a distributed computation framework for high throughput data computation.
  - The Hadoop Distributed File System (HDFS) gives HBase a reliable storage layer providing availability and reliability

HBase Table Structure

- Every row in an HBase table has a unique identifier called its rowkey. Rowkey values are distinct across all rows in an HBase table. Every interaction with data in a table begins with the rowkey.
- Table rows are sorted by row key
- A cell which is the intersection of row and column is versioned. By default, their version is a timestamp auto-assigned by HBase at the time of cell insertion.
- A cell’s content is an uninterpreted array of bytes.
- Row columns are grouped into column families. All column family members have a common prefix
- Columns can be added on the fly by the client as long as the column family they belong to preexists

HBase Table Structure



Column Family - Info

Rowkey	Name	Email	Password
sshakya1	Sunny	ss1@gmail.com	ss123
slee72	Sanghoon	slee@gmail.com	slee123 123slee

The table is lexicographically sorted on the rowkeys

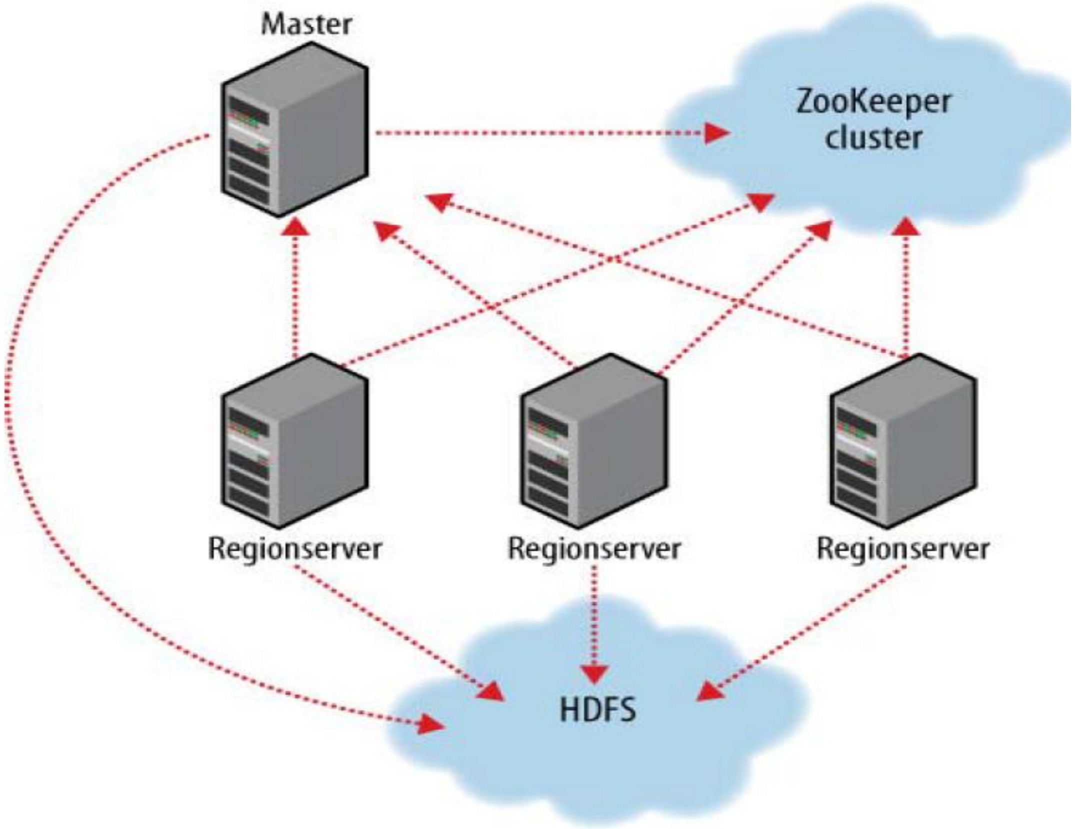
Cell

Each cell has multiple versions, represented by timestamps

HBase Implementation

- Tables are automatically partitioned horizontally by HBase into regions.
- Each region comprises a subset of a table's rows.
- Initially a table comprises a single region but as the size of the region grows, after it crosses a configurable size threshold
- As the table grows, the number of its regions grows. Regions are the units that get distributed over an HBase cluster
- In this way, a table that is too big for any one server can be carried by a cluster of servers with each node hosting a subset of the table's total regions

HBase Implementation



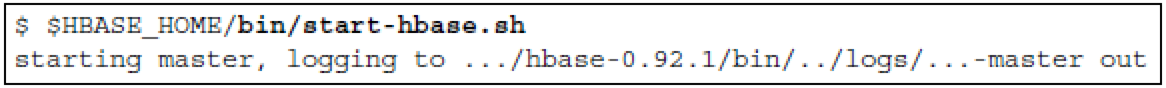
HBase Implementation

- HBase internally keeps special catalog tables named
  - ROOT
  - META

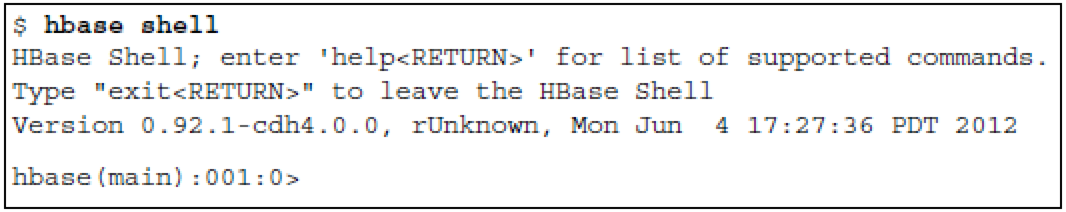
- ROOT table hold the list of META table regions
- META table holds the list of all user-space regions

- Fresh Clients connect to the Zookeeper cluster first to learn the location of ROOT
- Clients then consult ROOT to know the location of the META region.
- The Clients then do a lookup against the found META region to figure the hosting user-space region and its location

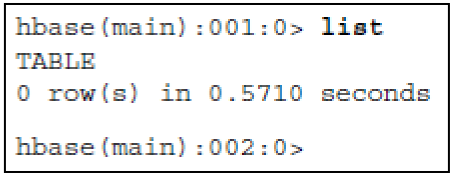
HBase Operations



```
$ $HBASE_HOME/bin/start-hbase.sh
starting master, logging to .../hbase-0.92.1/bin/../logs/...-master out
```



```
$ hbase shell
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 0.92.1-cdh4.0.0, rUnknown, Mon Jun  4 17:27:36 PDT 2012

hbase(main):001:0>
```

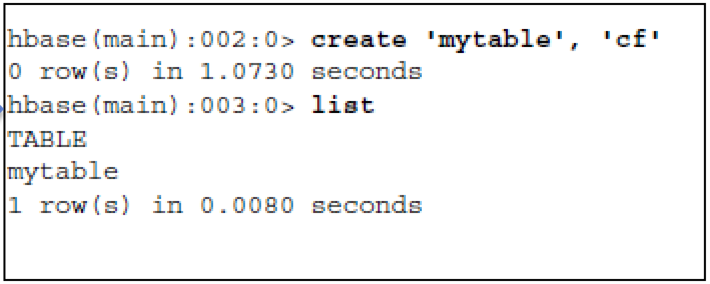


```
hbase(main):001:0> list
TABLE
0 row(s) in 0.5710 seconds

hbase(main):002:0>
```

HBase Operations

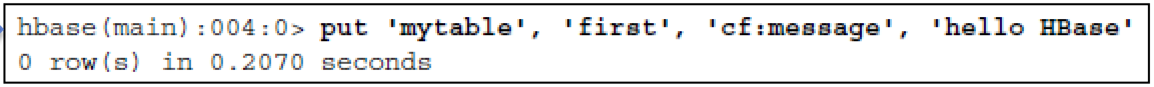
Five primitive commands : Get, Put, Delete, Scan, and Increment.

create a table ‘mytable’ with a single column family ‘cf’



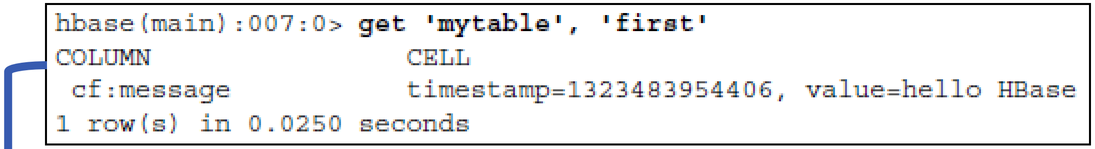
```
hbase(main):002:0> create 'mytable', 'cf'
0 row(s) in 1.0730 seconds
hbase(main):003:0> list
TABLE
mytable
1 row(s) in 0.0080 seconds
```

puts the bytes ‘hello HBase’ to a cell in ‘mytable’ in the ‘first’ row at the ‘cf:message’ columns

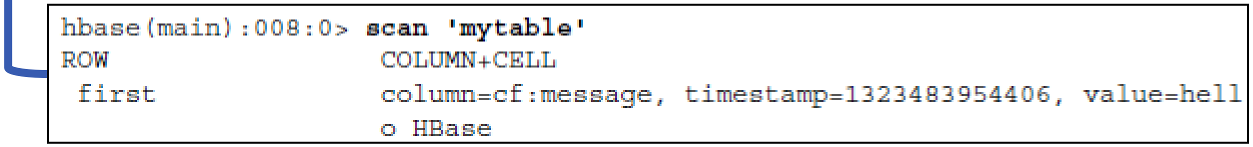


```
hbase(main):004:0> put 'mytable', 'first', 'cf:message', 'hello HBase'
0 row(s) in 0.2070 seconds
```

Two ways to read a table – GET and SCAN



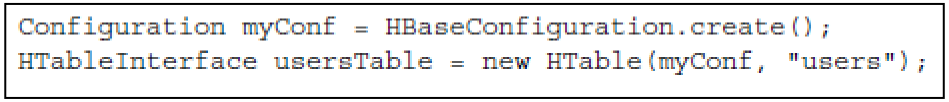
```
hbase(main):007:0> get 'mytable', 'first'
COLUMN                      CELL
 cf:message                 timestamp=1323483954406, value=hello HBase
1 row(s) in 0.0250 seconds
```



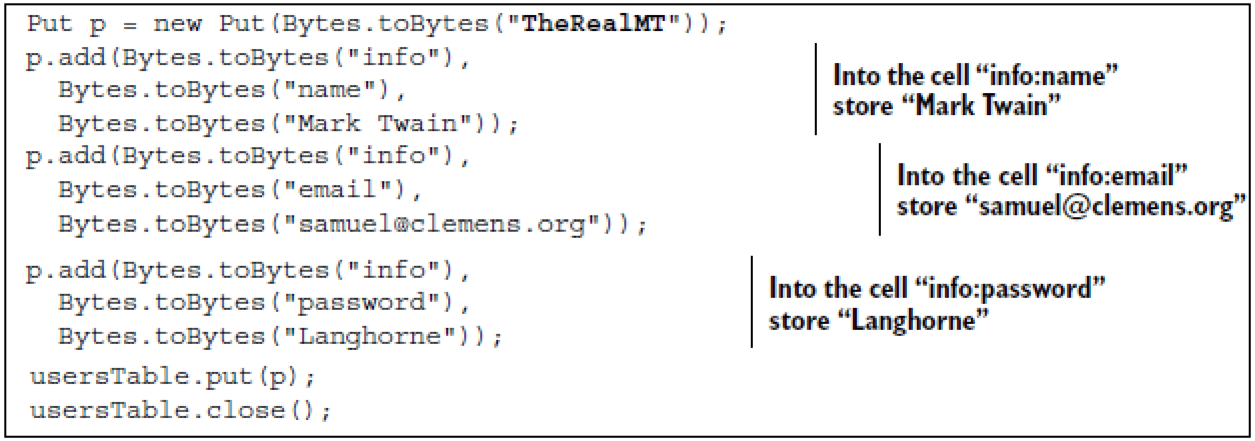
```
hbase(main):008:0> scan 'mytable'
ROW                       COLUMN+CELL
 first                    column=cf:message, timestamp=1323483954406, value=hell
                          o HBase
```

HBase Operations via JAVA Client API

Five primitive commands : Get, Put, Delete, Scan, and Increment.



```
Configuration myConf = HBaseConfiguration.create();
HTableInterface usersTable = new HTable(myConf, "users");
```

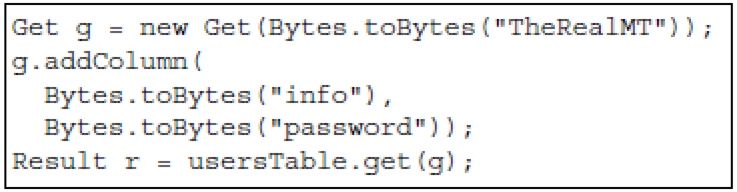


```
Put p = new Put(Bytes.toBytes("TheRealMT"));
p.add(Bytes.toBytes("info"),
  Bytes.toBytes("name"),
  Bytes.toBytes("Mark Twain"));
p.add(Bytes.toBytes("info"),
  Bytes.toBytes("email"),
  Bytes.toBytes("samuel@clemens.org"));
p.add(Bytes.toBytes("info"),
  Bytes.toBytes("password"),
  Bytes.toBytes("Langhorne"));
usersTable.put(p);
usersTable.close();
```

Into the cell “info:name” store “Mark Twain”

Into the cell “info:email” store “samuel@clemens.org”

Into the cell “info:password” store “Langhorne”

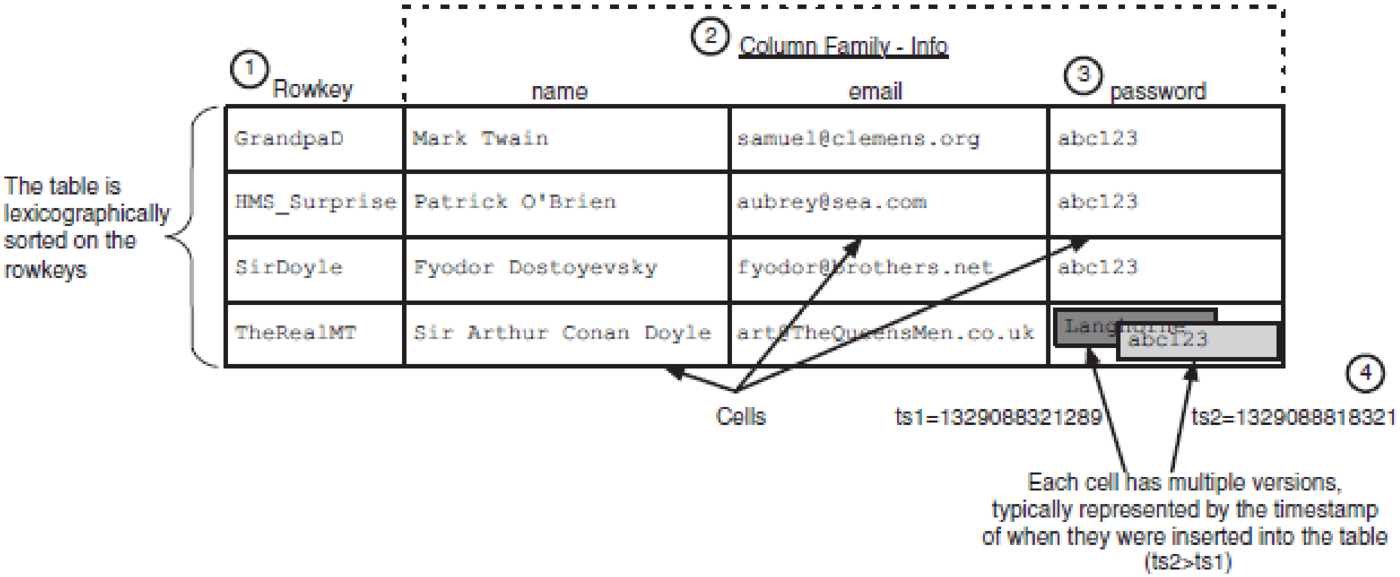


```
Get g = new Get(Bytes.toBytes("TheRealMT"));
g.addColumn(
  Bytes.toBytes("info"),
  Bytes.toBytes("password"));
Result r = usersTable.get(g);
```

Versioned Data

- ❑ In addition to being a schema-less database, HBase is also *versioned*.
- ❑ Every time you perform an operation on a cell, HBase implicitly stores a new version.
- ❑ By default, HBase stores only the last three versions; this is configurable per column family

Data Co-ordinates



	② Column Family - Info		
① Rowkey	name	email	③ password
GrandpaD	Mark Twain	samuel@clemens.org	abc123
HMS_Surprise	Patrick O'Brien	aubrey@sea.com	abc123
SirDoyle	Fyodor Dostoyevsky	fyodor@brothers.net	abc123
TheRealMT	Sir Arthur Conan Doyle	art@TheQueensMen.co.uk	Langhorne abc123

The table is lexicographically sorted on the rowkeys

Cells

ts1=1329088321289

ts2=1329088818321

④

Each cell has multiple versions, typically represented by the timestamp of when they were inserted into the table (ts2>ts1)

Map<RowKey, Map<ColumnFamily, Map<ColumnQualifier, Map<Version, Data>>>>

Modes of Operation

❑ HBase can run in three different modes

- Standalone
  - All of HBase runs in one java process
- Pseudo-distributed
  - A single machine run many java processes
- Full-distributed
  - HBase is fully distributed across a cluster a machines.

Different than Cassandra

Cassandra	HBase
Lacks concept of a Table. It's not common to have multiple keyspaces. Key space in a cluster is shared. Furthermore adding a keyspace requires a cluster restart!	Concept of Table exists. Each table has it's own key space.. You can add and remove table as easily as a RDBMS.
Offers sorting of columns.	Does not have sorting of columns.
Concept of Supercolumn allows you to design very flexible, very complex schemas.	Does not have supercolumns. But you can design a super column like structure as column names and values are binary.
Map Reduce support is new. You will need a Hadoop cluster to run it. Data will be transferred from Cassandra cluster to the Hadoop cluster. No suitable for running large data map reduce jobs.	Map Reduce support is native. HBase is built on Hadoop. Data does not get transferred.
Comparatively simpler to maintain if you don't have to have Hadoop.	Comparatively complicated as you have it has many moving pieces such as Zookeeper, Hadoop and HBase itself.
Does not have a native JAVA API as of now. No java doc. Even though written in Java, you have to use Thrift to communicate with the cluster.	Has a nice native JAVA API. HBase has a thrift interface for other languages too.
No master server, hence no single point of failure.	Although there exists a concept of a master server, HBase itself does not depend on it heavily. HBase cluster can keep serving data even if the master goes down. Hadoop NameNode is a single point of failure.

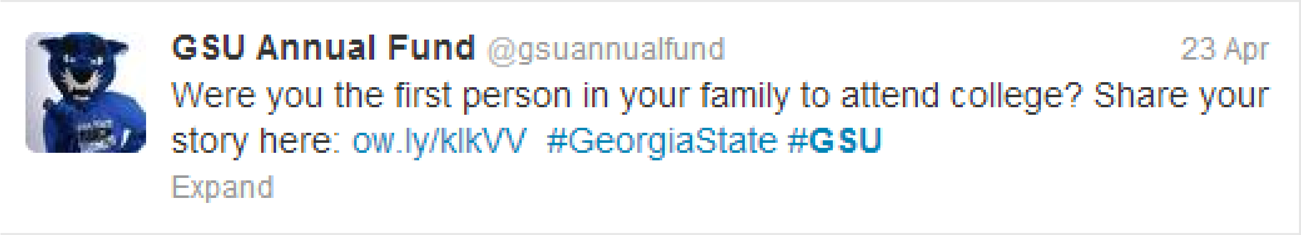
Real time Micro-Blog Summarization

❑ Twitter

- Started in 2006 as the micro blogging sites
- Very popular micro-blogging site where people send short messages of 140 characters called tweets
- By 2013, it has 100 million active user sending 200 million tweets per day.
- A majority of posts are conversational or not meaningful
- 3.6% of the posts concern topic of mainstream news.
- It has become a very popular medium to disperse information.

Real time Micro-Blog Summarization

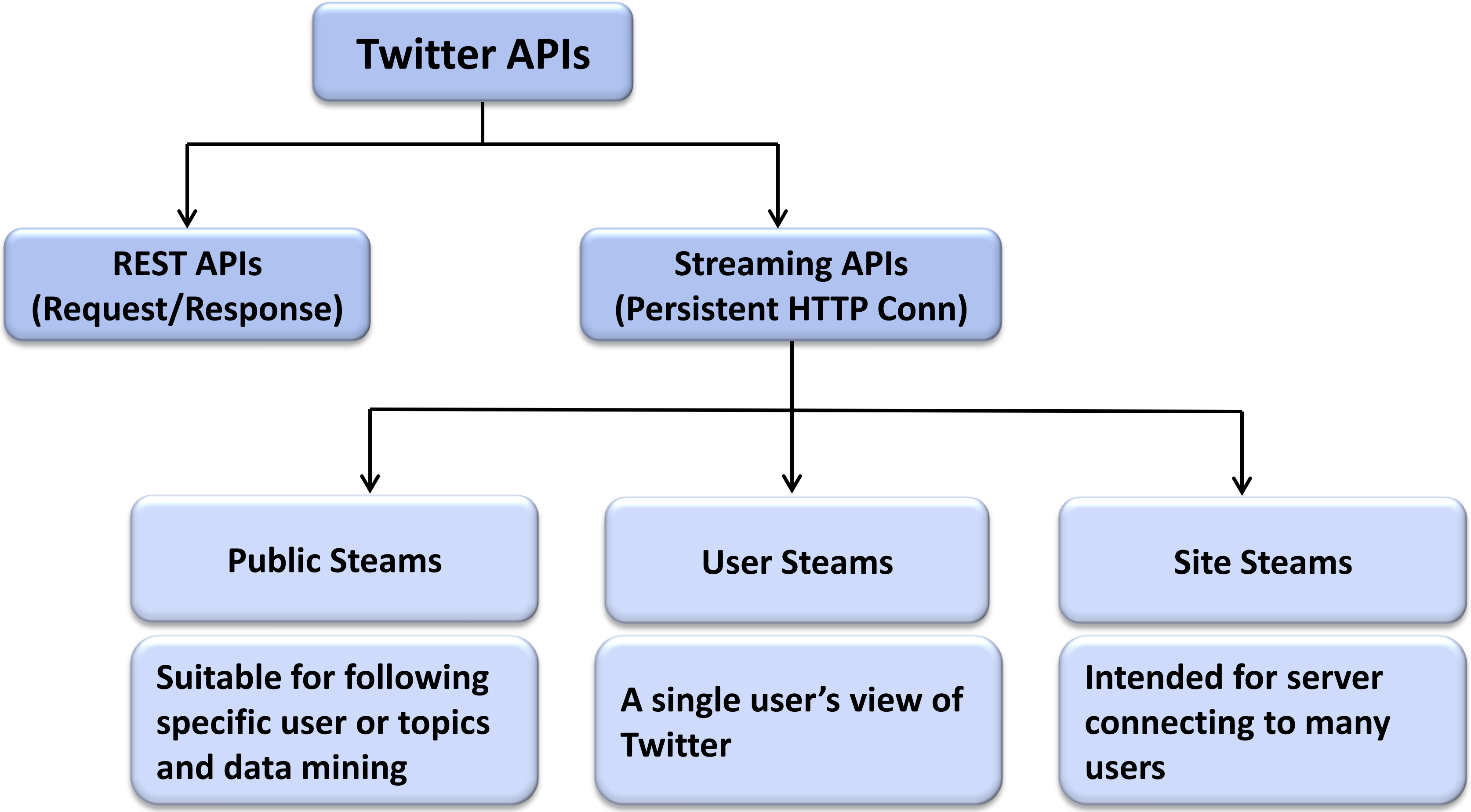
- Trending Topics
  - Twitter provides a list of popular topics.
  - A user retrieve a list of recent posts with the topic phrase.
  - Some trends have pound # sign before the word or phrase.
  - Hashtag is included particularly in Tweets to explain it as relating to a topic.



GSU Annual Fund @gsuannualfund 23 Apr
Were you the first person in your family to attend college? Share your story here: ow.ly/klkVV #GeorgiaState #GSU
Expand

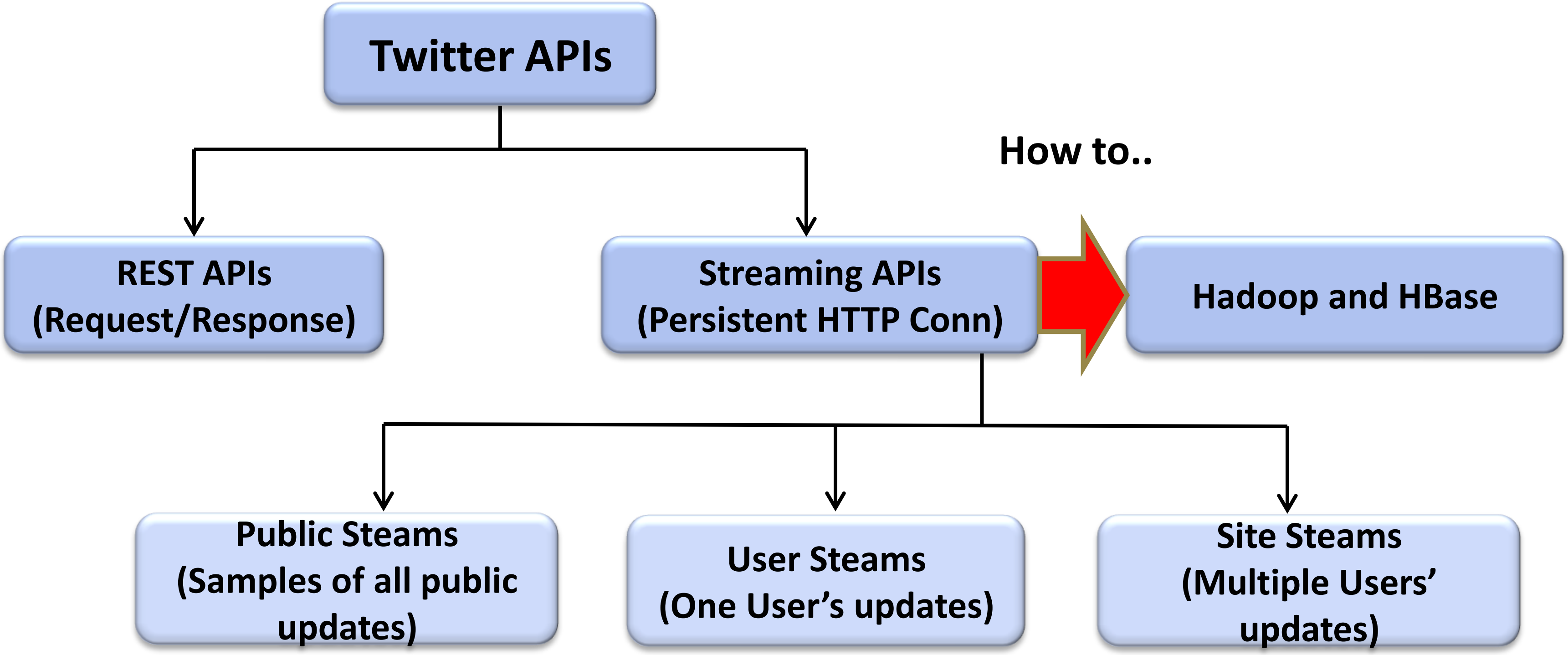
  - Problem: the user have to read manually through the posts for understanding a specific topic because the posts are sorted by recency, not relevancy.

Real time Micro-Blog Summarization



Twitter APIs
REST APIs (Request/Response)
Streaming APIs (Persistent HTTP Conn)
Public Steams
User Steams
Site Steams
Suitable for following specific user or topics and data mining
A single user's view of Twitter
Intended for server connecting to many users

Real time Micro-Blog Summarization



Twitter APIs

REST APIs (Request/Response)

Streaming APIs (Persistent HTTP Conn)

How to..

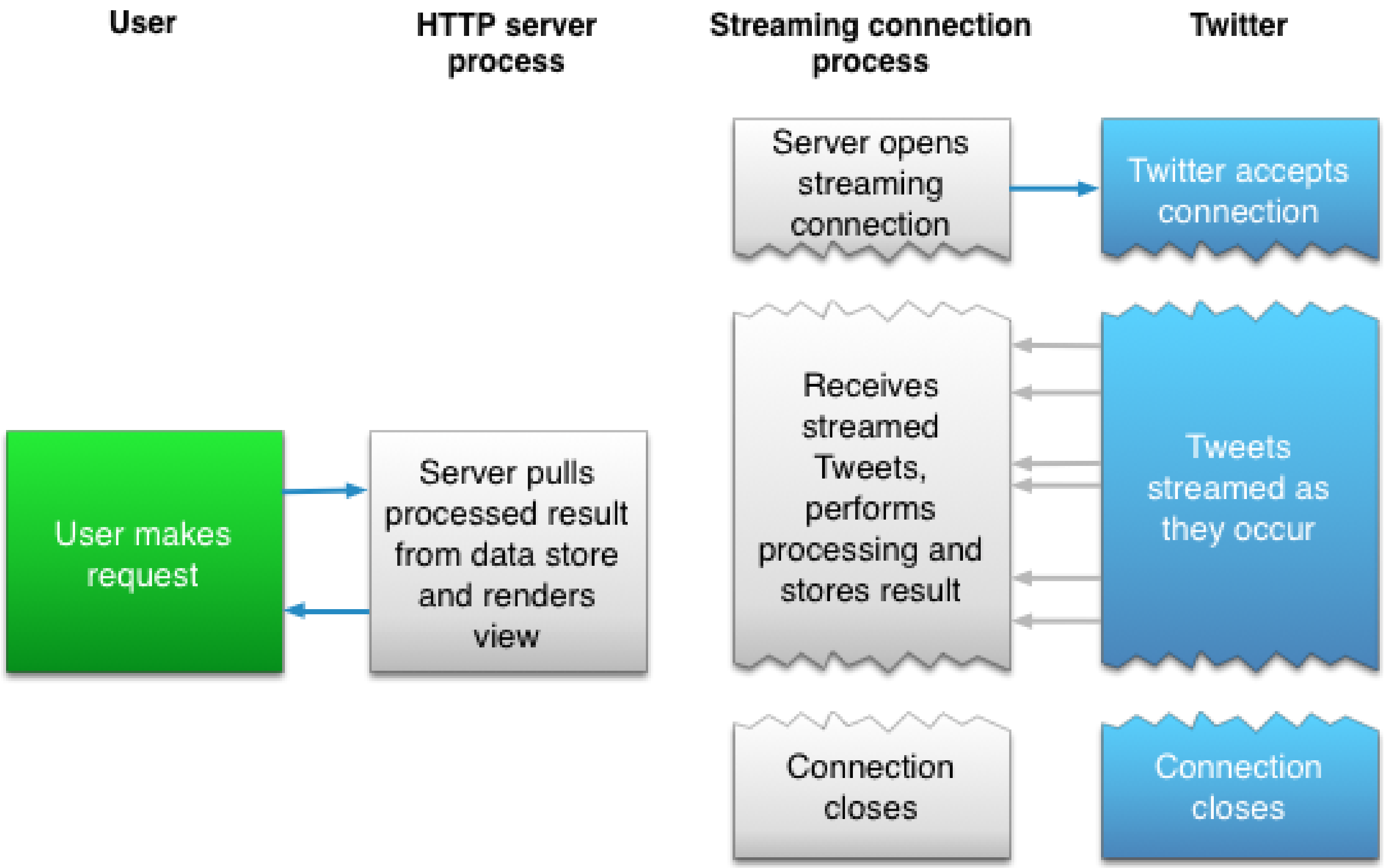
Hadoop and HBase

Public Steams (Samples of all public updates)

User Steams (One User's updates)

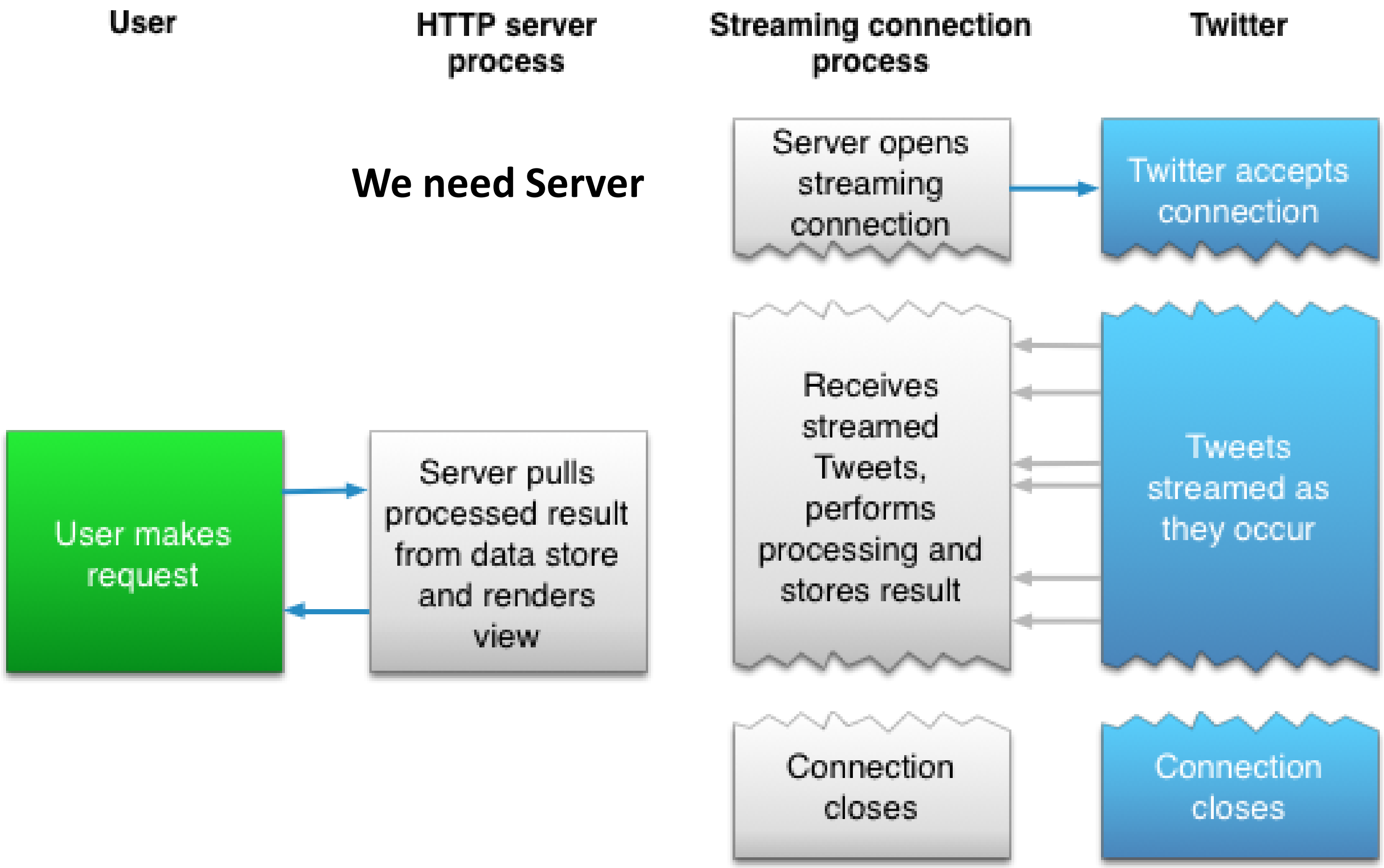
Site Steams (Multiple Users' updates)

Real time Micro-Blog Summarization

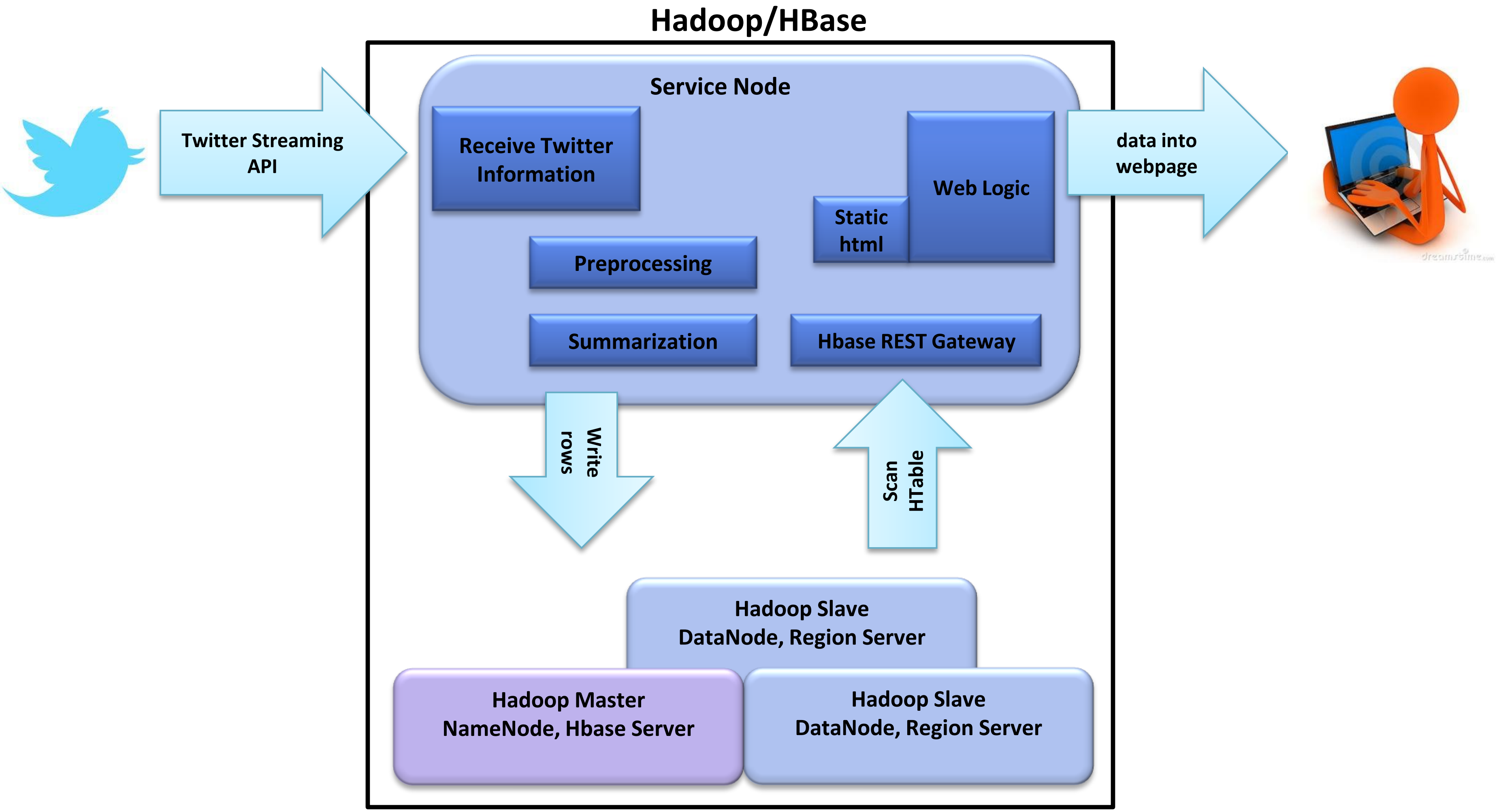

User
HTTP server process
Streaming connection process
Twitter
Server opens streaming connection
Twitter accepts connection
Receives streamed Tweets, performs processing and stores result
Tweets streamed as they occur
User makes request
Server pulls processed result from data store and renders view
Connection closes
Connection closes

Real time Micro-Blog Summarization

We need Server


User
HTTP server process
Streaming connection process
Twitter
Server opens streaming connection
Twitter accepts connection
Receives streamed Tweets, performs processing and stores result
Tweets streamed as they occur
User makes request
Server pulls processed result from data store and renders view
Connection closes
Connection closes

Application Architecture



Hadoop/HBase

Twitter Streaming API

Service Node

Receive Twitter Information

Preprocessing

Summarization

Static html

Web Logic

Hbase REST Gateway

data into webpage

Write rows

Scan HTable

Hadoop Slave
DataNode, Region Server

Hadoop Master
NameNode, Hbase Server

Hadoop Slave
DataNode, Region Server

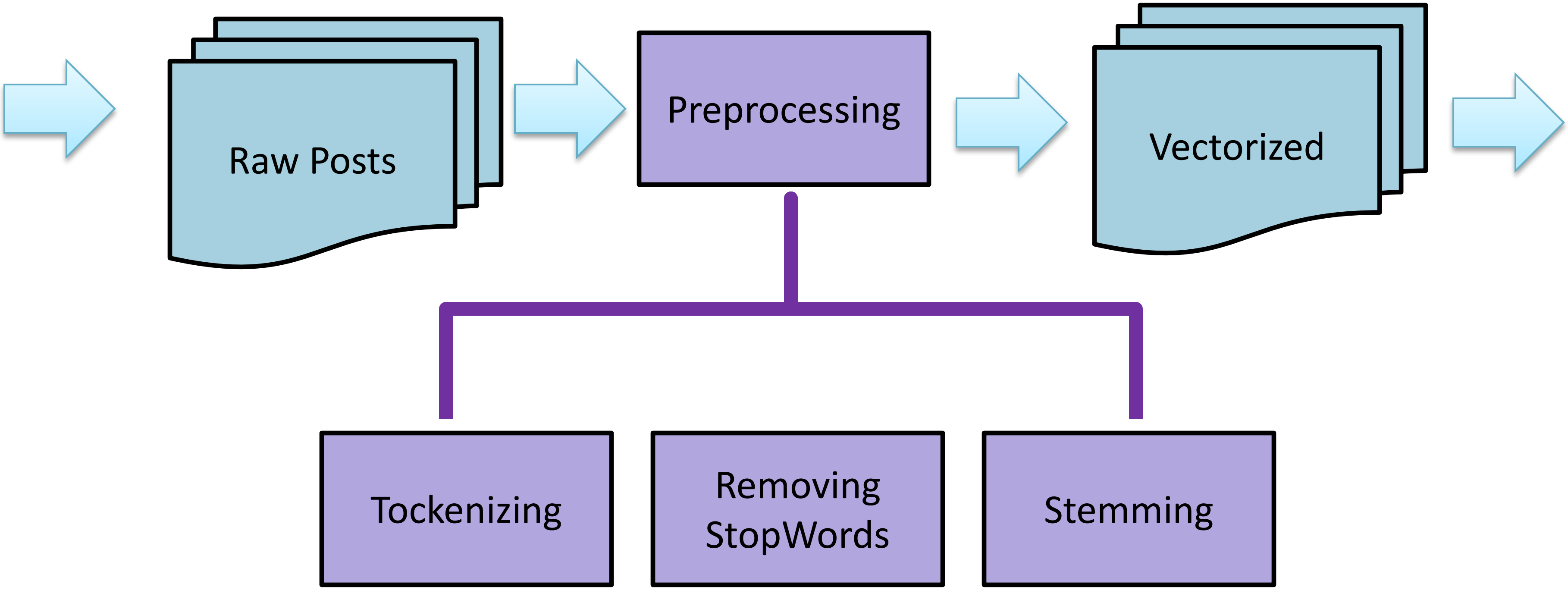
Summary Procedure

Rowkey	ColumnFamily	Column name	Timestamp	Value
Username/Time	UserInfo	Username	13452684	CSc8711
		UserID	13452684	Xke1kdfk
		Location	13452684	CL400
		Post	13452684	This is column #database
		HashTag	13452684	database

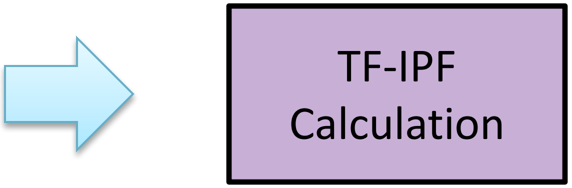
Extract Post

Rowkey	ColumnFamily	Column name	Timestamp	Value
database	HashTag	Number	13452684	1

Summary Procedure



Summary Procedure



TF-IPF Calculation

$$TF \cdot IPF = TF(t,p) \times IPF(t)$$

$$IPF = 1 + log \frac{totalPost}{numPost + 1}$$

$$Score = \frac{\sum_{i=1}^{k} TF_i \times IPF_i}{Max(\sum_{i=1}^{k} TF_i^n \times IPF_i^n)}$$

•*TF(t, p)* is the number of term t in the post

•*IPF (t)* is the inverse post frequency of the term t.

•*totalPost* is the total number of posts.

•*numPost* is the number of posts that the term t occurs.

Rowkey	ColumnFamily	Column name	Timestamp	Value
database/Time	Top10Summary	Summary	13452684	This is column #database

Application Demo

Future Work

- ❑ Evaluation of Summaries
  - Summaries are generated but not evaluated or compared with other types of summaries such as human summary

- ❑ Utilize Hadoop/HBase in full distributed mode.

Thank You

Any Questions?