Bigdata Assignment 4

1. Introduction

This assignment will help you to consolidate the concepts learnt in the session.

2. Problem Statement

HBase Basics:

Task 1:

Answer in your own words with example.

1. What is NoSQL database?

Ans: NoSQL is an approach to database design that can accommodate a wide variety of data models, including key-value, document, columnar and graph formats. NoSQL, which stands for "not only SQL." is an alternative to traditional relational databases in which data is placed in tables and data schema is carefully designed before the database is built. NoSQL database is built. NoSQL databases are especially useful for working with large sets of distributed data.

NoSQL databases are purpose built for specific data models and have flexible schemas for building modern applications. They are widely recognized for their ease of development, functionality, and performance at scale. They use a variety of data models, including document, graph, key-value, in-memory, and search.

2. How does data get stored in NoSQL database?

Ans: There are various NoSQL Databases. Each one uses a different method to store data. Some might use column store, some document, some graph, etc., Each database has its own unique characteristics.

Key-value store: Key-value (KV) stores use the associative array (also known as a map or dictionary) as their fundamental data model. In this model, data is represented as a collection of key-value pairs, such that each possible key appears at most once in the collection.
The key-value model is one of the simplest non-trivial data models, and richer data models are often implemented as an extension of it. This extension is computationally powerful, in that it can efficiently retrieve selective key ranges. Key-value stores can use consistency models ranging from eventual consistency to serializability. Some databases support ordering of keys. Examples of key-value storage are Berkeley DB, Dynamo and so on.

Document store: Each document-oriented database implementation differs on the details of this definition, in general, they all assume that documents encapsulate and encode in some standard formats or encodings. Encodings in use include XML, YAML, and JSON as well as

binary forms like BSON. Documents are addressed in the database via a unique key that represents that document. One of the other defining characteristics of a document-oriented database is that in addition to the key lookup performed by a key-value store, the database also offers an API or query language that retrieves documents based on their contents. Examples are Mongo DB.

Graph: This kind of database is designed for data whose relations are well represented as a graph consisting of elements interconnected with a finite number of relations between them. The type of data could be social relations, public transport links, road maps, network topologies, etc. Example Apache Giraph

Wide-column stores: Wide-column stores organize data tables as columns instead of as rows. Wide-column stores can be found both in SQL and NoSQL databases. Wide-column stores can query large data volumes faster than conventional relational databases. A wide-column data store can be used for recommendation engines, catalogs, fraud detection and other types of data processing. Google BigTable, Cassandra and HBase are examples of wide-column stores.

3.What is a column family in HBase?

Ans: Columns in Apache HBase are grouped into column families. A column family defines shared features to all columns that are created within them (think of it almost as a sub-table within your larger table). You will notice that HBase columns are composed of a combination of the column family and column qualifier (or column key): 'family: qualifier'.

All column members of a column family have the same prefix. The colon character (:) delimits the column family from the column. The column family prefix must be composed of printable characters. The qualifying tail, the column family qualifier, can be made of any arbitrary bytes. Column families must be declared up front at schema definition time whereas columns do not need to be defined at schema time but can be conjured on the fly while the table is up and running. For example, the columns *courses: history* and *courses: math* are both members of the courses column family

4.How many maximum number of columns can be added to HBase table?

Ans: There is no limit on number of columns in HBase. Some shortcomings in the current HBase implementation do not properly support large number of column families in a single table. That number should be in low tens. Most of the time up to three column families should work fine without any significant performance drawback. Ideally you should go with a single column family.

Currently, flushing and compactions are done on a per Region basis so if one column family is carrying the bulk of the data bringing on flushes, the adjacent families will also be flushed even though the amount of data they carry is small. When many column families exist the flushing and compaction interaction can make for a bunch of needless i/o.

A column family can have an arbitrary number of columns denoted by a column qualifier which is like a column's label. For example:

Bigdata Assignment 4

```
{
 "row1": {"1": {"color": "green",
              "size": 25},
         "2": {"weight": 52,
              "size": 18}
        },
 "row2": {"1": {"color": "blue"},
         "2": {"height": 192,
              "size": 43}
        }
}
```

As you can see in the example above, the same column family (e.g., "1") in two rows can have different columns. In row "row1", it has columns "color" and "size", while in row "row2", it has only "color" column. It can also have a column that is none of the above. Since rows can have different columns in column families there is no a single way to query for a list of all columns in all column families. This means that you have to do a full table scan.

There is no specific limit on the number of columns in a column family. Actually, you can have millions of columns in the single column family.

5.Why columns are not defined at the time of table creation in HBase?

Ans: Columns are configured based on the requirement and data ingestion but the column families should be in the same generalized access pattern.

6.How does data get managed in HBase?

Ans: HBase is built upon distributed filesystems with file storage distributed across commodity machines. The distributed file systems HBase works with include

- Hadoop's Distributed File System (HDFS) and
- Amazon's Simple Storage Service (SS3).

HDFS provides a scalable and replicated storage layer for HBase. It guarantees that data

is never lost by writing the changes across a configurable number of physical servers.

The data is stored in HFiles, which are ordered immutable key/value maps. Internally, the HFiles are sequences of blocks with a block index stored at the end. The block index is loaded when the HFile is opened and kept in memory. The default block size is 64 KB but it can be changed since it is configurable. HBase API can be used to access specific values and also scan ranges of values given a start and end key.

Since every HFile has a block index, lookups can be performed with a single disk seek. First, HBase does a binary search in the in-memory block index to find a block containing the given key and then the block is read from disk.

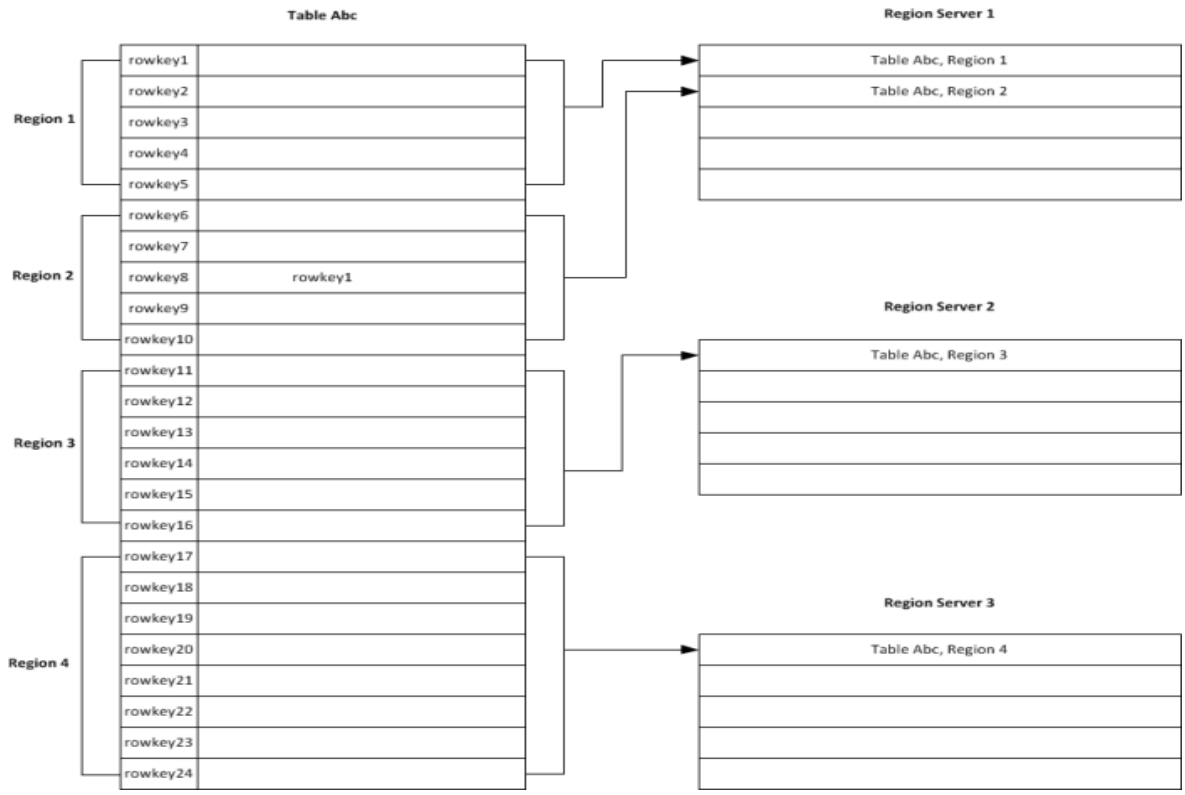
When data is updated it is first written to a commit log, called a write-ahead log (WAL) and then it is stored in the in-memory memstore. When the data in memory exceeds a given maximum value, it is flushed as an HFile to disk and after that the commit logs are discarded up to the last unflushed modification. The system can continue to serve readers and writers without blocking them while it is flushing the memstore to disk. This is done by rolling the memstore in memory where the new empty one is taking the updates and the old full one is transferred into an HFile. At the same time, no sorting or other special processing has to be performed since the data in the memstores is already sorted by keys matching what HFiles represent on disk.

The write-ahead log (WAL) is used for recovery purposes only. Since flushing memstores to disk causes creation of HFiles, HBase has a housekeeping job that merges the HFiles into larger ones using compaction. Various compaction algorithms are supported.

Other HBase architectural components include the client library (API), at least one master server, and many region servers. The region servers can be added or removed while the system is up and running to accommodate increased workloads. The master is responsible for assigning regions to region servers. It uses Apache ZooKeeper, a distributed coordination service, to facilitate that task.

Data is partitioned and replicated across a number of regions located on region servers.

Bigdata Assignment 4



As mentioned above, assignment and distribution of regions to region servers is automatic. However manual management of regions is also possible. When a region's size reaches a pre-defined threshold, the region will automatically split into two child regions. The split happens along a row key boundary. A single region always manages an entire row. It means that a row is never divided.

7. What happens internally when new data gets inserted into HBase table?

Ans: HBase stores data in a form of a distributed sorted multidimensional persistence maps called Tables. HBase does not overwrite row values. It stores different values per row by time and column qualifier. A row key, column family and column qualifier form a cell that has a value and timestamp that represents the value's version. A timestamp is recorded for each value and it is the time on the region server when the value was written.

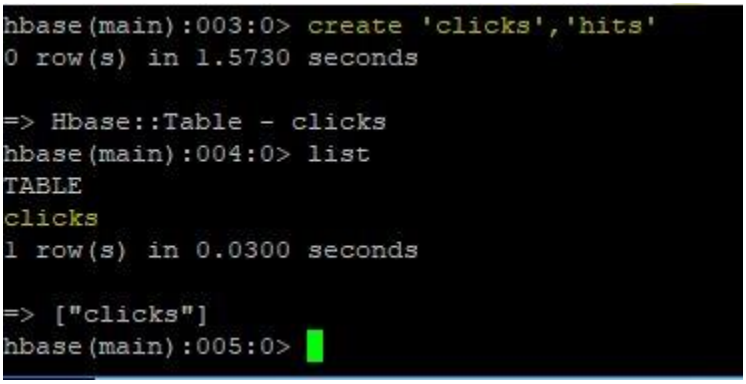
All cell's values are stored in a descending order by its timestamp. When values are retrieved and if the timestamp is not provided then HBase will return the cell value with the latest timestamp. If a timestamp is not specified during the write, the current timestamp is used.

The maximum number of versions for a given column to store is part of the column schema. It is specified at table creation. It can be specified via alter table command as well. The default value is 1. The minimum number of versions can be also set up per column family. You can also globally set up a maximum number of versions per column.

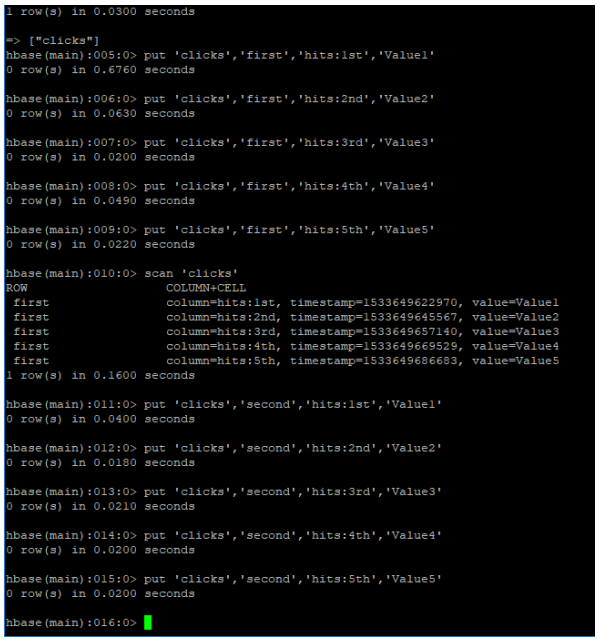
Bigdata Assignment 4

Task 2:

1. Create an HBase table named 'clicks' with a column family 'hits' such that it should be able to store last 5 values of qualifiers inside 'hits' column family.



2. Add few records in the table and update some of them. Use IP Address as row-key. Scan the table to view if all the previous versions are getting displayed.



Bigdata Assignment 4

```
hbase(main):017:0> scan 'clicks'
ROW                   COLUMN+CELL
 first                column=hits:1st, timestamp=1533649622970, value=Value1
 first                column=hits:2nd, timestamp=1533649645567, value=Value2
 first                column=hits:3rd, timestamp=1533649657140, value=Value3
 first                column=hits:4th, timestamp=1533649669529, value=Value4
 first                column=hits:5th, timestamp=1533649686683, value=Value5
 second               column=hits:1st, timestamp=1533649896494, value=Value1
 second               column=hits:2nd, timestamp=1533649911018, value=Value2
 second               column=hits:3rd, timestamp=1533649921029, value=Value3
 second               column=hits:4th, timestamp=1533649930827, value=Value4
 second               column=hits:5th, timestamp=1533649943961, value=Value5
2 row(s) in 0.1230 seconds

hbase(main):018:0> get 'clicks','second'
COLUMN                CELL
 hits:1st             timestamp=1533649896494, value=Value1
 hits:2nd             timestamp=1533649911018, value=Value2
 hits:3rd             timestamp=1533649921029, value=Value3
 hits:4th             timestamp=1533649930827, value=Value4
 hits:5th             timestamp=1533649943961, value=Value5
5 row(s) in 0.1610 seconds
```

Bigdata Assignment 4

Advanced HBase:

Task 1:

Explain the below concepts with an example in brief.

● Nosql Databases

NoSQL basically means "not only SQL" or it also means non-relational database. It's an alternate for traditional relational database. They are especially useful for working with large sets of distributed data. of distributed data.

NoSQL databases are purpose built for specific data models and have flexible schemas for building modern applications. They are widely recognized for their ease of development, functionality, and performance at scale. They use a variety of data models, including document, graph, key-value, in-memory, and search.

Examples of NoSQL databases are HBase, MongoDB, Cassandra, Berkeley DB, Apache Giraph.

● Types of Nosql Databases

There are 4 basic types of NoSQL databases:

1. **Key-Value Store** – It has a Big Hash Table of keys & values Example- Riak, Amazon S3 (Dynamo)
2. **Document-based Store-** It stores documents made up of tagged elements. Example- CouchDB
3. **Column-based Store-** Each storage block contains data from only one column. Example- HBase, Cassandra

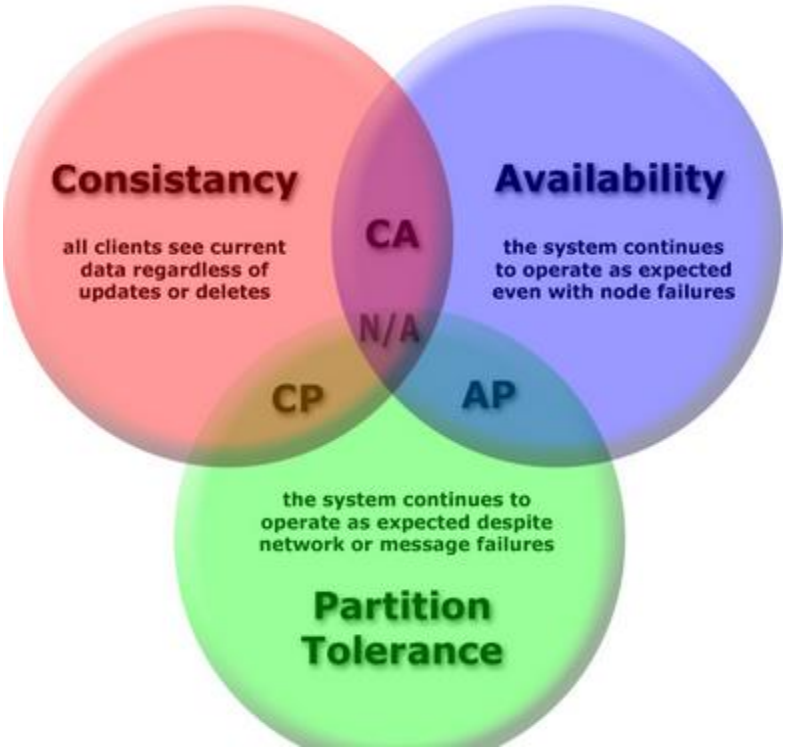
Graph-based-A network database that uses edges and nodes to represent and store data. Example- Neo4J

● CAP Theorem

For any distributed system, CAP Theorem reiterates the need to find balance between Consistency, Availability and Partition tolerance.

- Consistency - This means that the data in the database remains consistent after the execution of an operation. For example, after an update operation, all clients see the same data.
- Availability - This means that the system is always on (service guarantee availability), no downtime.

- Partition Tolerance - This means that the system continues to function even if the communication among the servers is unreliable, i.e. the servers may be partitioned into multiple groups that cannot communicate with one another.



● HBase Architecture

Hbase architecture consists of mainly HMaster, HRegionserver, HRegions and Zookeeper. Zookeeper is a centralized monitoring server which maintains configuration information and provides distributed synchronization. If the client wants to communicate with regions servers, client has to approach Zookeeper.

HMaster is the master server of Hbase and it coordinates the HBase cluster. It is responsible for the administrative operations of the cluster. A region server serves a region at the start of the application. During failure of region server, HMaster assign the region to another Region server. HMaster can also assign a region to another region server as part of load balancing.

HRegions Servers: It will perform the following functions in communication with HMaster and Zookeeper.

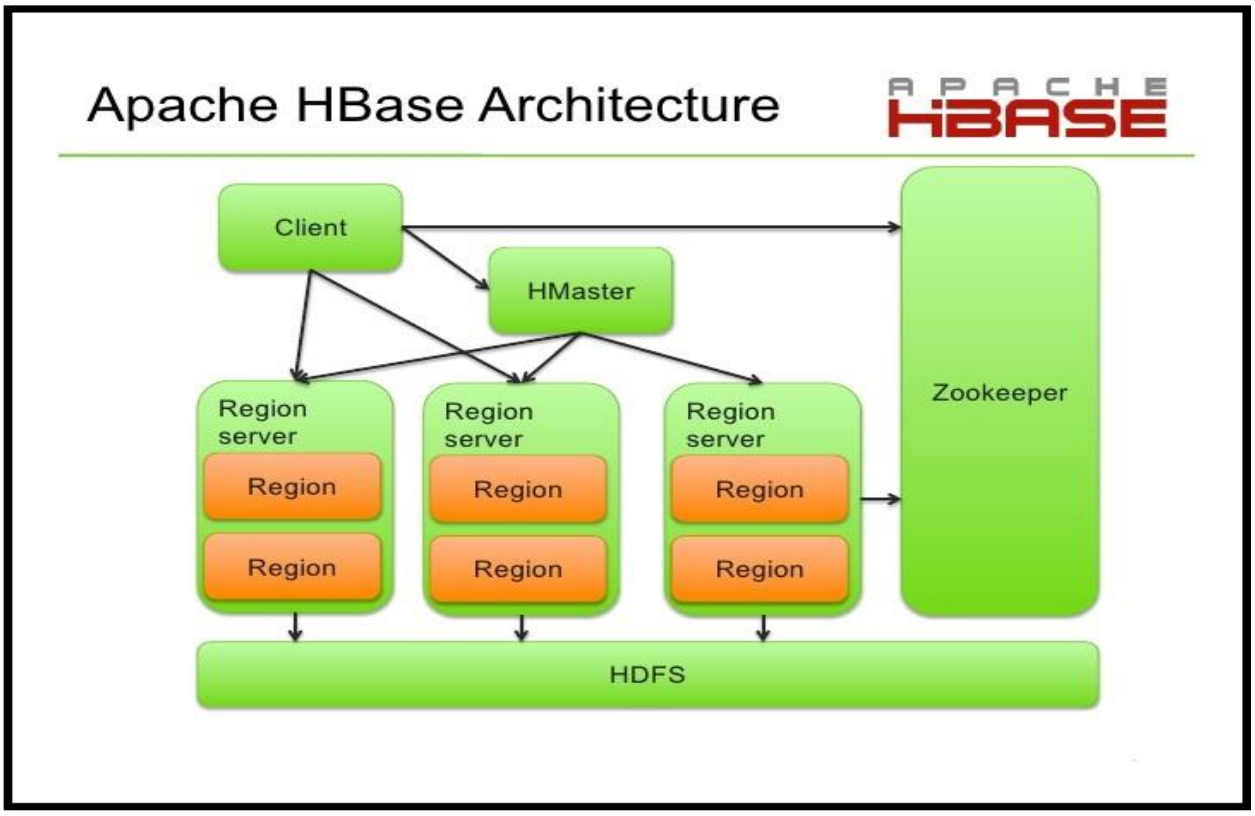
- Hosting and managing regions.
- Splitting regions automatically.
- Handling read and writes requests.
- Communicating with clients directly.

Bigdata Assignment 4

HRegions: For each column family, HRegions maintain a store. Main components of HRegions are

- Memstore - Holds in-memory modifications to the store
- Hfile

BASIC ARCHITECTURE OF HBASE



● HBase vs RDBMS

HBASE	RDBMS
Schema-less in database.	Having fixed schema in database.
Column oriented database.	Row oriented data store.
Designed to store De-normalized data.	Designed to store Normalized data.

Bigdata Assignment 4

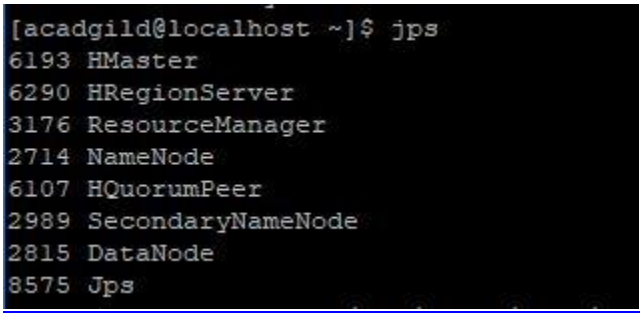
Wide and sparsely populated tables present in Hbase.	Contains thin tables in database.
Supports automatic partitioning.	Has no built in support for partitioning.
Well suited for OLAP systems.	Well suited for OLTP systems.
Read only relevant data from database.	To retrieve one row at a time and hence could read unnecessary data if only some of the data in a row is required.
Structured and semi structure data can be stored and processed using Hbase.	Structured data can be stored and processed using an RDBMS.
Enables aggregation over many rows and columns.	Aggregation is an expensive operation.

Task 2:

Execute blog present in below link.

https://acadgild.com/blog/importtsv-data-from-hdfs-into-hbase/

1. HMaster Running



Bigdata Assignment 4

2. Create table

```
hbase(main):002:0> create 'bulktable','cf1','cf2'
0 row(s) in 1.3980 seconds

=> Hbase::Table - bulktable
hbase(main):003:0> list
TABLE
bulktable
clicks
2 row(s) in 0.0210 seconds

=> ["bulktable", "clicks"]
hbase(main):004:0>
```

3. Creating a dataset

```
=> ["bulktable", "clicks"]
hbase(main):004:0> You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ mkdir hbase
[acadgild@localhost ~]$ cd hbase
[acadgild@localhost hbase]$ vi bulk_data.tsv
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost hbase]$ cat bulk_data.tsv
1       Amith   4
2       Girija  3
3       Jathin  5
4       Swathi  3

[acadgild@localhost hbase]$
```

4. Creating a hdfs directory and moving the dataset to it.

```
[acadgild@localhost hbase]$ hadoop fs -mkdir /hbase
18/08/07 19:43:36 WARN util.NativeCodeLoader: Unable to load native-hadoop library
 for your platform... using builtin-java classes where applicable
mkdir: `/hbase': File exists
[acadgild@localhost hbase]$ hadoop fs -put bulk_data.tsv /hbase/
18/08/07 19:46:11 WARN util.NativeCodeLoader: Unable to load native-hadoop library
 for your platform... using builtin-java classes where applicable
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost hbase]$ hadoop fs -cat /hbase/bulk_data.tsv
18/08/07 19:47:51 WARN util.NativeCodeLoader: Unable to load native-hadoop library
 for your platform... using builtin-java classes where applicable
1       Amith   4
2       Girija  3
3       Jathin  5
4       Swathi  3

You have new mail in /var/spool/mail/acadgild
[acadgild@localhost hbase]$
```

Bigdata Assignment 4

5. Table from Hadoop to Hbase

```
hbase(main):050:0> scan 'bulktable'
ROW                                             COLUMN+CELL
 1                                              column=cf1:name, timestamp=1532639202412, value=Amit
 1                                              column=cf2:exp=>4, timestamp=1532639135965, value=4
 2                                              column=cf1:name, timestamp=1532639313109, value=girja
 2                                              column=cf2:exp, timestamp=1532639343231, value=3
 3                                              column=cf1:name, timestamp=1532639370325, value=jatin
 3                                              column=cf2:exp, timestamp=1532639386685, value=5
 4                                              column=cf1:name, timestamp=1532639416833, value=swati
 4                                              column=cf2:exp, timestamp=1532639402908, value=3
4 row(s) in 0.0670 seconds
```

Oozie & Flume:

Task 1:

Create a flume agent that streams data from Twitter and stores in the HDFS.

<Was not taught in class and the tutorial in the blog is not clear>

3. Expected Output

Solution should contain the task and the question number, the problem statement, then the commands / code used to solve the task with comments and screenshot of the output.

Report shall be in PDF format. Submitted in GitHub.