ACADGILD

Advance HBase

Assignment 11

Task1

Explain the below concepts with an example in brief.

Nosql Databases

Types of Nosql Databases

CAP Theorem

HBase Architecture

HBase vs RDBMS

Nosql Databases

NoSQL stands for Not only SQL. NoSQL are non-relational, open source, schema less and distributed database.

They are highly popular today because of their ability to scale out or scale horizontally and flexibility to deal with a variety of data : structured, semi-structered and unstructured data.

Types of Nosql Databases

There are 4 types of NoSQL :

Key-Value pair /Key-Value store:

It maintains data in key-value pairs. It uses a hash table in which there exists a unique key and a pointer to a particular item of data.

e.g.

Key	**Value**
First Name	Sachin
Last Name	Gorade
Address	Mumbai, India

Amazon DynamoDB, LinkedIn uses key-value store NoSQL database.

Document store:

It maintains data in collections constituted of documents.

It pair each key with a complex data structure known as a document.

Documents can contain many different key-value pairs, or key-array pairs, or even nested documents.

e.g.

```
{
  _id: ObjectId(fdf789)
  title: 'MongoDB Overview',
  description: 'MongoDB is no sql database',
  tags: ['mongodb', 'database', 'NoSQL'],
}
```

Assignment 11

Walmart uses Graph NoSQL.

CAP Theorem

The CAP theorem applies to distributed systems that store state. It states that networked shared-data systems can only guarantee or strongly support two of the following three properties:

Consistency -

A guarantee that every node in a distributed cluster returns the same, most recent, successful write. Consistency refers to every client having the same view of the data.

Availability –

Every non-failing node returns a response for all read and write requests in a reasonable amount of time. To be available, every node on (either side of a network partition) must be able to respond in a reasonable amount of time.

Partition Tolerant –

The system continues to function and upholds its consistency guarantees in spite of network partitions. Network partitions are a fact of life. Distributed systems guaranteeing partition tolerance can gracefully recover from partitions once the partition heals.

The CAP theorem categorizes systems into three categories. We can have system with one of these three categories:

CP (Consistent and Partition Tolerant) –

CP is referring to a category of systems where availability is sacrificed only in the case of a network partition.

CA (Consistent and Available) –

CA systems are consistent and available systems in the absence of any network partition.

Assignment 11

AP (Available and Partition Tolerant) –

These are systems that are available and partition tolerant but cannot guarantee consistency.

In case of a network partition (a rare occurrence) one needs to choose between Availability and partition tolerance. In any networked shared-data systems partition tolerance is a must. Network partitions and dropped messages are a fact of life and must be handled appropriately. Consequently, system designers must choose between consistency and availability. A network partition forces designers to either choose perfect consistency or perfect availability. Picking consistency means not being able to answer a client's query as the system cannot guarantee to return the most recent write. This sacrifices availability. Network partition forces nonfailing nodes to reject clients' requests as these nodes cannot guarantee consistent data. At the opposite end of the spectrum, being available means being able to respond to a client's request but the system cannot guarantee consistency, i.e., the most recent value written. Available systems provide the best possible answer under the given circumstance.

HBase Architecture

As we know, HBase is a column-oriented NoSQL database.

In a column-oriented databases, all the column values are stored together like first column values will be stored together, then the second column values will be stored together and data in other columns are stored in a similar manner.

When the amount of data is very huge, like in terms of petabytes or exabytes, we use column-oriented approach, because the data of a single column is stored together and can be accessed faster.
While row-oriented approach comparatively handles less number of rows and columns efficiently, as row-oriented database stores data is a structured format.
When we need to process and analyse a large set of semi-structured or unstructured data, we use column oriented approach. Such as applications dealing with **Online Analytical Processing** like data mining, data warehousing, applications including analytics, etc.
Whereas, **Online Transactional Processing** such as banking and finance domains which handle structured data and require transactional properties (ACID properties) use row-oriented approach.

HBase Architecture: Components of HBase Architecture
HBase has three major components i.e., **HMaster Server**, **HBase Region Server, Regions** and **Zookeeper**.

HBase Architecture: Region

A table can be divided into a number of regions. A Region is a sorted range of rows storing data between a start key and an end key.
A Region has a default size of 256MB which can be configured according to the need.
A Group of regions is served to the clients by a Region Server.

Advance HBase

Assignment 11

A Region Server can serve approximately 1000 regions to the client.

HBase Architecture: HMaster

As in the below image, you can see the HMaster handles a collection of Region Server which resides on DataNode.

HBase HMaster performs DDL operations (create and delete tables) and assigns regions to the Region servers as you can see in the above image.

It coordinates and manages the Region Server (similar as NameNode manages DataNode in HDFS).

It assigns regions to the Region Servers on startup and re-assigns regions to Region Servers during recovery and load balancing.

It monitors all the Region Server's instances in the cluster (with the help of Zookeeper) and performs recovery activities whenever any Region Server is down.

It provides an interface for creating, deleting and updating tables.

ZooKeeper – The Coordinator

Zookeeper acts like a coordinator inside HBase distributed environment. It helps in maintaining server state inside the cluster by communicating through sessions.

Every Region Server along with HMaster Server sends continuous heartbeat at regular interval to Zookeeper and it checks which server is alive. It also provides server failure notifications so that, recovery measures can be executed.

The inactive server, which acts as a backup for active server. If the active server fails, it comes for the rescue.

The active HMaster sends heartbeats to the Zookeeper while the inactive HMaster listens for the notification send by active HMaster. If the active HMaster fails to send a heartbeat the session is deleted and the inactive HMaster becomes active.

HBase vs RDBMS

HBase	RDBMS
1. Column-oriented	1. Row-oriented
2. Flexible schema, add columns on the Fly	2. Fixed schema
3. Does not supports SQL	3. It supports SQL
4. Good for semi-structured data as well as structured data.	4. Good for structured data only.
5. Supports Horizontal scaling(scale out). So easy to scale.	5. Supports Vertical scaling(scale up). So difficult to scale.
6. It doesn't supports referential integrity.	6. It supports referential integrity.
7. Good with sparse tables	7. Not optimized for sparse tables.
8.Tight integration with MapReduce (MR).	8. No integration with MapReduce (MR).
9. Joins are not optimized and are using MR.	9. Joins are optimized and are not using MR.
10. It does not supports ACID properties. Hence it does not supports transactions.	10. It supports ACID properties. Hence supports transactions.

Assignment 11

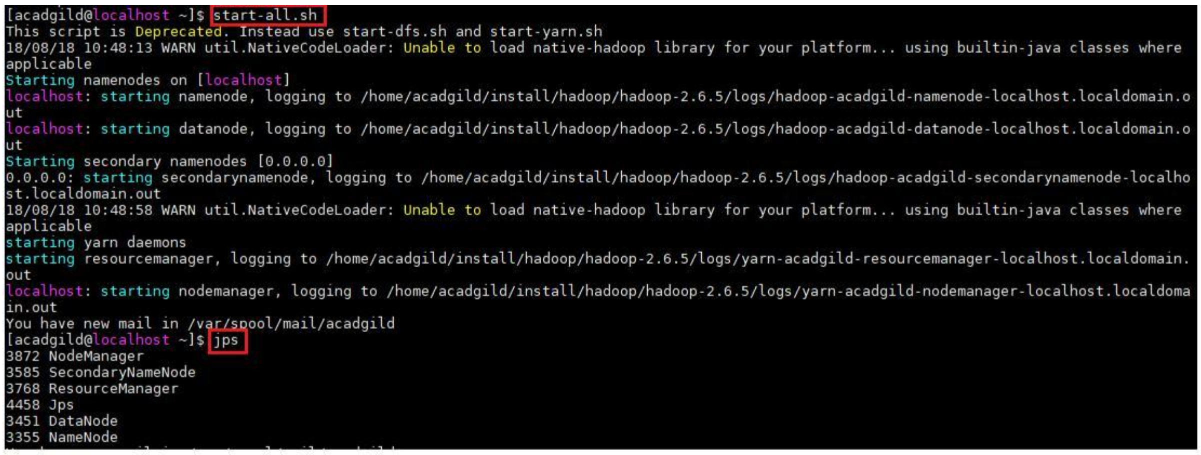
Task2 :

Execute blog present in below link

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

We have started all hadoop daemons by using command : **start-all.sh**

Then we have verified that all hadoop daemons are started by using **jps** command.

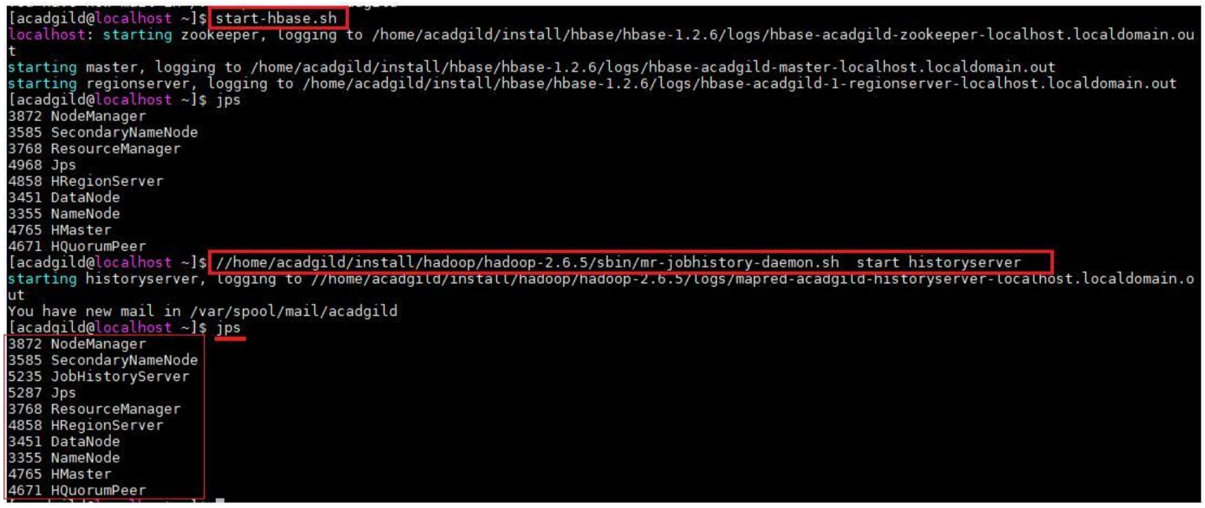


```
[acadgild@localhost ~]$ start-all.sh
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
18/08/18 10:48:13 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where
applicable
Starting namenodes on [localhost]
localhost: starting namenode, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/logs/hadoop-acadgild-namenode-localhost.localdomain.o
ut
localhost: starting datanode, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/logs/hadoop-acadgild-datanode-localhost.localdomain.o
ut
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/logs/hadoop-acadgild-secondarynamenode-localho
st.localdomain.out
18/08/18 10:48:58 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where
applicable
starting yarn daemons
starting resourcemanager, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/logs/yarn-acadgild-resourcemanager-localhost.localdomain.
out
localhost: starting nodemanager, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/logs/yarn-acadgild-nodemanager-localhost.localdoma
in.out
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ jps
3872 NodeManager
3585 SecondaryNameNode
3768 ResourceManager
4458 Jps
3451 DataNode
3355 NameNode
```

After this, we have started hbase and job history daemons by using commands : **start-hbase.sh** and **/home/acadgild/install/hadoop/hadoop-2.6.5/sbin/mr-jobhistory-daemon.sh start historyserver**

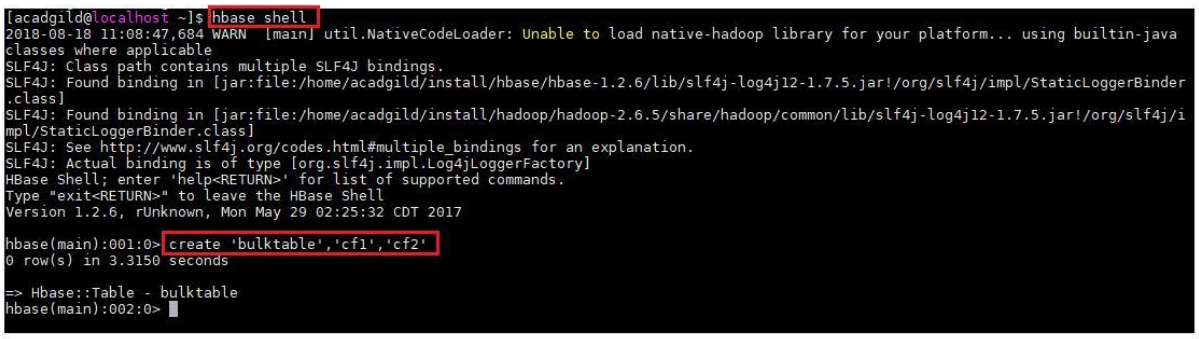
Then we have verified that hbase and job history server daemons are started by using jps command.

Assignment 11



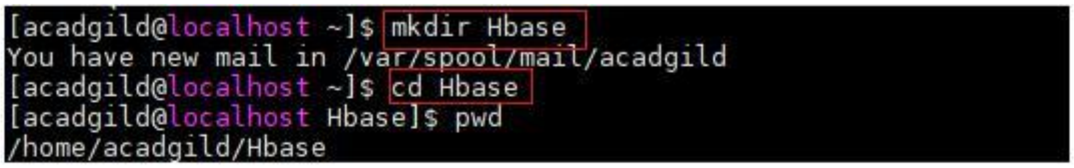
```
[acadgild@localhost ~]$ start-hbase.sh
localhost: starting zookeeper, logging to /home/acadgild/install/hbase/hbase-1.2.6/logs/hbase-acadgild-zookeeper-localhost.localdomain.ou
t
starting master, logging to /home/acadgild/install/hbase/hbase-1.2.6/logs/hbase-acadgild-master-localhost.localdomain.out
starting regionserver, logging to /home/acadgild/install/hbase/hbase-1.2.6/logs/hbase-acadgild-1-regionserver-localhost.localdomain.out
[acadgild@localhost ~]$ jps
3872 NodeManager
3585 SecondaryNameNode
3768 ResourceManager
4968 Jps
4858 HRegionServer
3451 DataNode
3355 NameNode
4765 HMaster
4671 HQuorumPeer
[acadgild@localhost ~]$ //home/acadgild/install/hadoop/hadoop-2.6.5/sbin/mr-jobhistory-daemon.sh  start historyserver
starting historyserver, logging to //home/acadgild/install/hadoop/hadoop-2.6.5/logs/mapred-acadgild-historyserver-localhost.localdomain.o
ut
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ jps
3872 NodeManager
3585 SecondaryNameNode
5235 JobHistoryServer
5287 Jps
3768 ResourceManager
4858 HRegionServer
3451 DataNode
3355 NameNode
4765 HMaster
4671 HQuorumPeer
```

Then we have started **hbase shell** and created table **'bulktable'** with column families **cf1** and **cf2**.



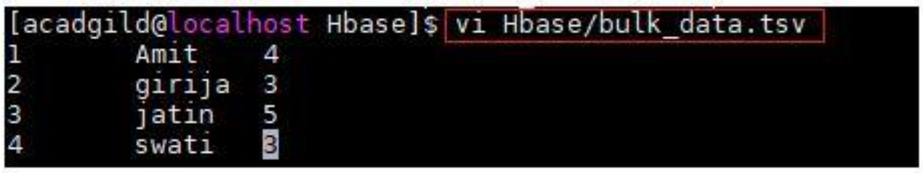
```
[acadgild@localhost ~]$ hbase shell
2018-08-18 11:08:47,684 WARN  [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java
classes where applicable
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hbase/hbase-1.2.6/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder
.class]
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.6.5/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/i
mpl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.6, rUnknown, Mon May 29 02:25:32 CDT 2017

hbase(main):001:0> create 'bulktable','cf1','cf2'
0 row(s) in 3.3150 seconds

=> Hbase::Table - bulktable
hbase(main):002:0>
```

Then we have created Hbase directory in local system.



```
[acadgild@localhost ~]$ mkdir Hbase
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ cd Hbase
[acadgild@localhost Hbase]$ pwd
/home/acadgild/Hbase
```

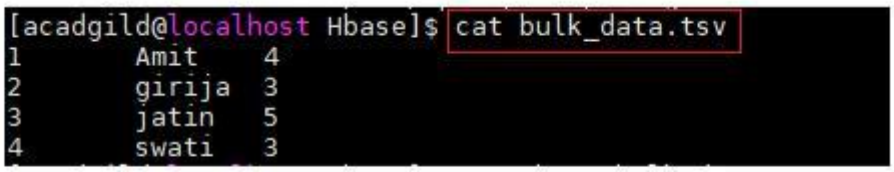
Then we have created and edited bulk_data.tsv file by using **VI editor**.



```
[acadgild@localhost Hbase]$ vi Hbase/bulk_data.tsv
1       Amit    4
2       girija  3
3       jatin   5
4       swati   3
```

Assignment 11

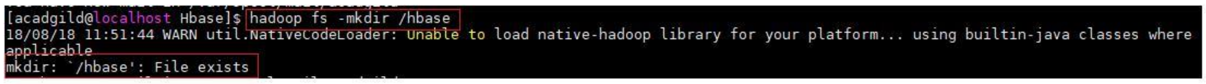
By using below cat command, we could see that data has been inserted successfully in bulk_data.tsv file.



```
[acadgild@localhost Hbase]$ cat bulk_data.tsv
1       Amit    4
2       girija  3
3       jatin   5
4       swati   3
```

As data should be present in HDFS, we have to copy bulk_data.tsv file from local to HDFS.
So we are creating hbase directory in HDFS via command : **hadoop fs -mkdir /hbase**.

But as this hbase directory is already present, we are getting below error as
hbase directory already exists.

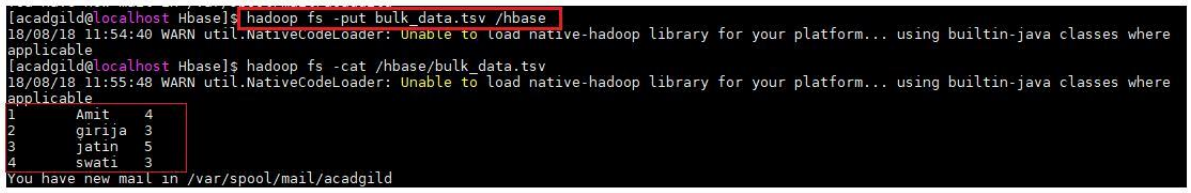


```
[acadgild@localhost Hbase]$ hadoop fs -mkdir /hbase
18/08/18 11:51:44 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where
applicable
mkdir: `/hbase': File exists
```

Then we have copied bulk_data.csv file from local to hbase directory in HDFS.

By using below command, you could see content in bulk_data.tsv file :

hadoop fs -cat /hbase/bulk_data.tsv



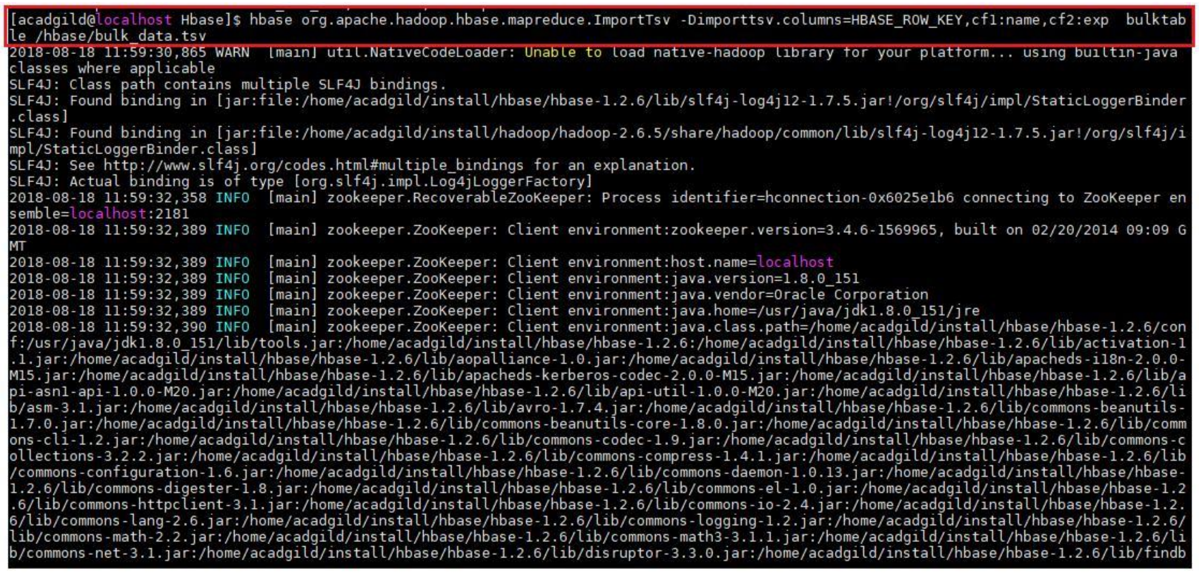
```
[acadgild@localhost Hbase]$ hadoop fs -put bulk_data.tsv /hbase
18/08/18 11:54:40 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where
applicable
[acadgild@localhost Hbase]$ hadoop fs -cat /hbase/bulk_data.tsv
18/08/18 11:55:48 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where
applicable
1       Amit    4
2       girija  3
3       jatin   5
4       swati   3
You have new mail in /var/spool/mail/acadgild
```

Now to import data from HDFS to HBase, we have used following command along with arguments<tablename> and <path of data in HDFS>.

hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -Dimporttsv.columns=HBASE_ROW_KEY,cf1:name,cf2:exp bulktable /hbase/bulk_data.tsv

Here, tablename is **bulktable** & path of data file is **/hbase/bulk_data.tsv.**

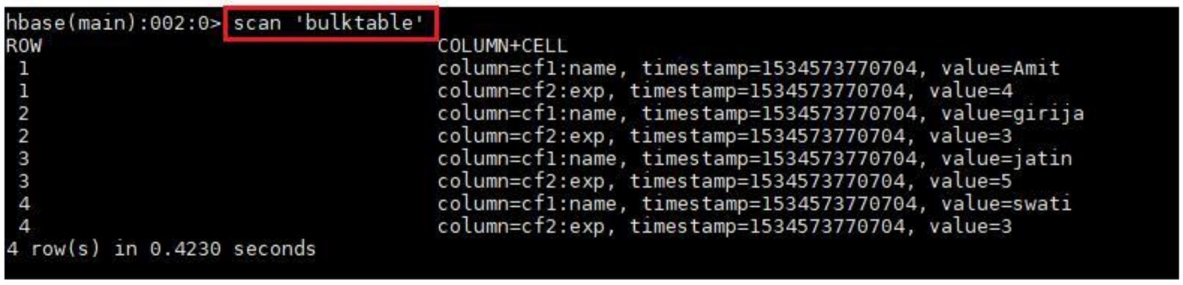
Assignment 11



```
[acadgild@localhost Hbase]$ hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -Dimporttsv.columns=HBASE_ROW_KEY,cf1:name,cf2:exp  bulktab
le /hbase/bulk_data.tsv
2018-08-18 11:59:30,865 WARN  [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java
classes where applicable
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hbase/hbase-1.2.6/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder
.class]
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.6.5/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/i
mpl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
2018-08-18 11:59:32,358 INFO  [main] zookeeper.RecoverableZooKeeper: Process identifier=hconnection-0x6025e1b6 connecting to ZooKeeper en
semble=localhost:2181
2018-08-18 11:59:32,389 INFO  [main] zookeeper.ZooKeeper: Client environment:zookeeper.version=3.4.6-1569965, built on 02/20/2014 09:09 G
MT
2018-08-18 11:59:32,389 INFO  [main] zookeeper.ZooKeeper: Client environment:host.name=localhost
2018-08-18 11:59:32,389 INFO  [main] zookeeper.ZooKeeper: Client environment:java.version=1.8.0_151
2018-08-18 11:59:32,389 INFO  [main] zookeeper.ZooKeeper: Client environment:java.vendor=Oracle Corporation
2018-08-18 11:59:32,389 INFO  [main] zookeeper.ZooKeeper: Client environment:java.home=/usr/java/jdk1.8.0_151/jre
2018-08-18 11:59:32,390 INFO  [main] zookeeper.ZooKeeper: Client environment:java.class.path=/home/acadgild/install/hbase/hbase-1.2.6/con
f:/usr/java/jdk1.8.0_151/lib/tools.jar:/home/acadgild/install/hbase/hbase-1.2.6:/home/acadgild/install/hbase/hbase-1.2.6/lib/activation-1
.1.jar:/home/acadgild/install/hbase/hbase-1.2.6/lib/aopalliance-1.0.jar:/home/acadgild/install/hbase/hbase-1.2.6/lib/apacheds-i18n-2.0.0-
M15.jar:/home/acadgild/install/hbase/hbase-1.2.6/lib/apacheds-kerberos-codec-2.0.0-M15.jar:/home/acadgild/install/hbase/hbase-1.2.6/lib/a
pi-asn1-api-1.0.0-M20.jar:/home/acadgild/install/hbase/hbase-1.2.6/lib/api-util-1.0.0-M20.jar:/home/acadgild/install/hbase/hbase-1.2.6/li
b/asm-3.1.jar:/home/acadgild/install/hbase/hbase-1.2.6/lib/avro-1.7.4.jar:/home/acadgild/install/hbase/hbase-1.2.6/lib/commons-beanutils-
1.7.0.jar:/home/acadgild/install/hbase/hbase-1.2.6/lib/commons-beanutils-core-1.8.0.jar:/home/acadgild/install/hbase/hbase-1.2.6/lib/comm
ons-cli-1.2.jar:/home/acadgild/install/hbase/hbase-1.2.6/lib/commons-codec-1.9.jar:/home/acadgild/install/hbase/hbase-1.2.6/lib/commons-c
ollections-3.2.2.jar:/home/acadgild/install/hbase/hbase-1.2.6/lib/commons-compress-1.4.1.jar:/home/acadgild/install/hbase/hbase-1.2.6/lib
/commons-configuration-1.6.jar:/home/acadgild/install/hbase/hbase-1.2.6/lib/commons-daemon-1.0.13.jar:/home/acadgild/install/hbase/hbase-
1.2.6/lib/commons-digester-1.8.jar:/home/acadgild/install/hbase/hbase-1.2.6/lib/commons-el-1.0.jar:/home/acadgild/install/hbase/hbase-1.2
.6/lib/commons-httpclient-3.1.jar:/home/acadgild/install/hbase/hbase-1.2.6/lib/commons-io-2.4.jar:/home/acadgild/install/hbase/hbase-1.2.
6/lib/commons-lang-2.6.jar:/home/acadgild/install/hbase/hbase-1.2.6/lib/commons-logging-1.2.jar:/home/acadgild/install/hbase/hbase-1.2.6/
lib/commons-math-2.2.jar:/home/acadgild/install/hbase/hbase-1.2.6/lib/commons-math3-3.1.1.jar:/home/acadgild/install/hbase/hbase-1.2.6/li
b/commons-net-3.1.jar:/home/acadgild/install/hbase/hbase-1.2.6/lib/disruptor-3.3.0.jar:/home/acadgild/install/hbase/hbase-1.2.6/lib/findb
```

Then to check whether data got inserted from HDFS into HBase, we have fetched data from bulktable by using HBase command as **scan 'bulktable' .**

We could see that all data (4 rows) which is present in **bulk_data.tsv** file got inserted into bulktable in HBase successfully.



```
hbase(main):002:0> scan 'bulktable'
ROW                                   COLUMN+CELL
 1                                    column=cf1:name, timestamp=1534573770704, value=Amit
 1                                    column=cf2:exp, timestamp=1534573770704, value=4
 2                                    column=cf1:name, timestamp=1534573770704, value=girija
 2                                    column=cf2:exp, timestamp=1534573770704, value=3
 3                                    column=cf1:name, timestamp=1534573770704, value=jatin
 3                                    column=cf2:exp, timestamp=1534573770704, value=5
 4                                    column=cf1:name, timestamp=1534573770704, value=swati
 4                                    column=cf2:exp, timestamp=1534573770704, value=3
4 row(s) in 0.4230 seconds
```