

ASSIGNMENT

Task1

a) What is NoSQL databases:

Sol: NoSQL is a modified approach to handle the growing data in the web world which is not handled by relational database management systems (RDBMS). NoSQL databases do not rely on table, column etc only and use more flexible data models. NoSQL can mean “not SQL” or “not only SQL.” As RDBMS have increasingly failed to meet the performance, scalability, and flexibility needs that next-generation, data-intensive applications require, NoSQL databases have been adopted by mainstream enterprises. NoSQL is particularly useful for storing semi-structured and unstructured data, which is growing far more rapidly than structured data and does not fit the relational schemas of RDBMS. Common types of unstructured data include: user and session data; chat, messaging, and log data; time series data such as IoT and device data; and large objects such as video and images.

there are mainly 4 types of nosql databases available for use in enterprises:

- i) wide column based e.g cassandra
- ii) key-value based e.g hbase
- iii) graph based e.g apache giraffe
- iv) document based e.g MongoDB

b) Types of NoSQL databases?

Sol: i) Document based NOSQL system: These systems stores data in the form of documents in known format such as json (javaScript Object Notation). Document can be accessed via their document id and can also be accessed by other indexes. E.g. MongoDB

ii) NOSQL key-value stores: These systems are simple data model based on fast access based on key to their corresponding values. The value can be a record, an object, a document or can be more complex data values. E.g. Hbase.

iii) Column based or wide column NOSQL systems: These system partition the table into column under column families where each column family is stored in its on file. They also allow versioning of data values. E.g. Cassandra.

iv) Graph based NOSQL system: Data is represented in the form of graphs, related nodes can be found by traversing the edges using path expressions. E.g. apache giraffe

c) CAP theorem:

Sol: The CAP theorem states that it is not possible to guarantee 3 required properties i.e. **consistencies, availability and partition tolerance** at the same time in a distributed system with data replication. Thus, the designer guarantees two out of three properties. In NOSQL databases weak consistency is often acceptable unlike RDMS instead of guaranteeing serializability. The other two properties i.e. availability and partition tolerance is important. NOSQL database uses eventual consistency.

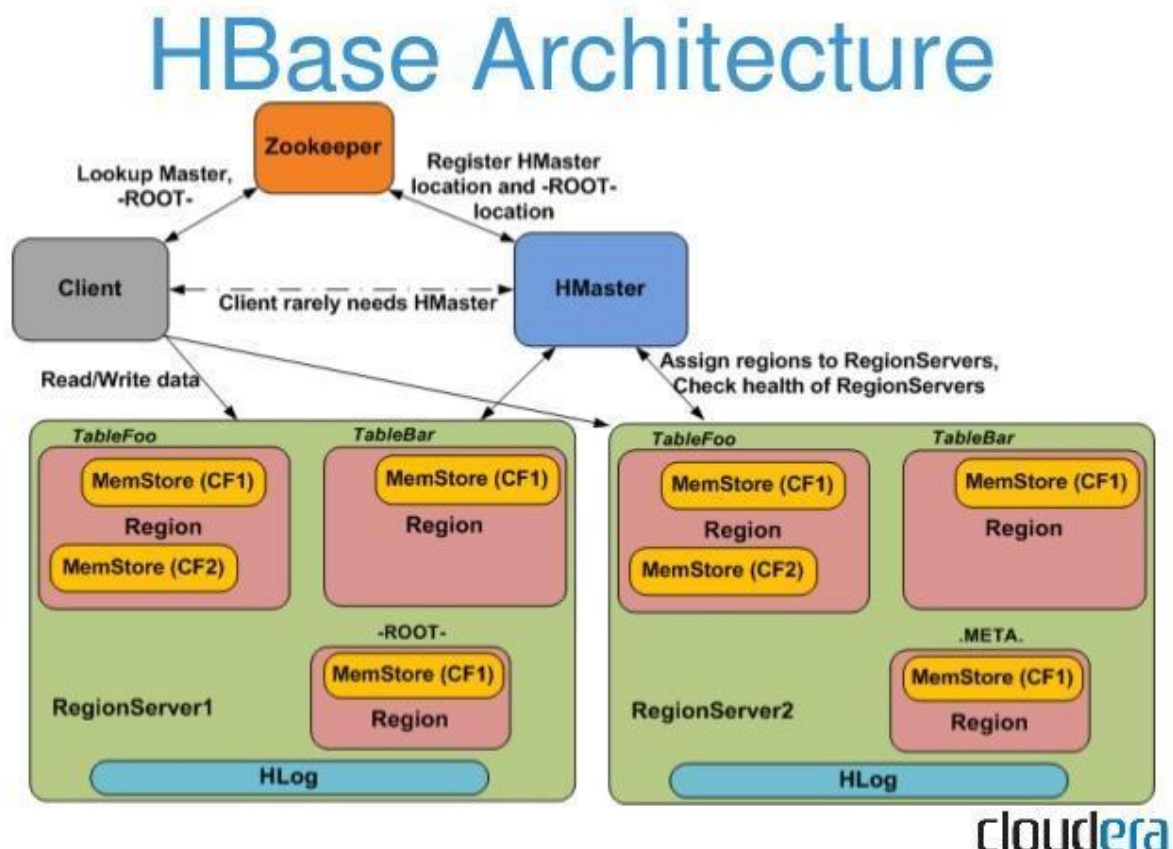
d) Hbase architecture ?

Sol: Hbase provides latency based random read and writes on top of HDFS. In Hbase, tables are dynamically distributed by the server whenever the data becomes too large to handle (auto sharding). The simplest unit of horizontal scalability in hbase is region. A continuous sorted set of rows is called region.

Hbase architecture has single set of master nodes known as Hmaster and several region servers as slaves. Each region server (slave) serves a set of

regions, and a region can be served only by one region server. Whenever a client sends a write request, Hmaster receives the request and forwards it to corresponding region server.

Hbase can be run in a multiple master set up, herein there is only single master active at a time. Hbase tables are partitioned into multiple regions with every region storing multiple table's rows.



Components of Hbase architecture

It has 3 main components: Hmaster, Region Server, and Zookeeper.

Hmaster: Hbase Hmaster is lightweight process that assigns the regions to region servers in the hadoop cluster for load balancing.

- a) It manages and monitors the hadoop cluster.
- b) Provide interface for creating, updating and deleting tables.
- c) Controlling the failover.
- d) Whenever the client wants to change the schema and any metadata operations, Hmaster is responsible for that.

Region Servers: These are the worker nodes which execute the read, write, update, and delete requests from clients. Region server process, runs on every node in the Hadoop cluster. It runs on HDFS datanodes and consists:

- a) Block cache : this is the read cache, most frequently read data is stored on read cache and whenever the block cache is full, recently used data is evicted.
- b) Memstore: this is the write cache, it stores the recent data which is not written to the disk. Every column family in a region has a MemStore.
- c) Write Ahead Log (WAL): it is a file that stores the new data that is not persisted to permanent storage.
- d) Hfile is the actual storage file that stores the rows as sorted key values on a disk.

Zookeepers: a) Hbase uses zookeeper as a distributed coordination service for region assignments and to recover any region server crashes by loading them on to another region servers that are functioning.

b) Zookeeper is a centralized monitoring server that maintains configuration information and provides distributed synchronization.

c) Whenever a client wants to communicate with region, it has to contact zookeeper first. Hmaster and region server are registered with zookeeper services, client needs to access zookeeper quorum in order to connect with region server and Hmaster.

d) In case of node failure in Hbase cluster, zookeeper quorum will trigger the error messages and start repairing the failed nodes.

e) Zookeeper keeps a track of all the region servers and which data node is held by which region server. Hmaster contacts the zookeeper to get the details of region servers.

f) Services of zookeeper also involves: establishing client communication with region servers, tracking server failure and network partitions, Maintaining configuration information, provides ephemeral nodes, which represent different region servers.

e) Hbase vs RDBMS?

Sol:

Hbase	RDBMS
Col oriented	Ro oriented mostly
Flexible schema, add col on the fly	Fixed schema
Good with sparse tables	Not optimized for sparsed tables
Joins using MR- not optimized	Optimized for joins
Tight integration with MR	No Map reduce
Horizontal scalability, add hardware	Hard to shard and scale
Good for semi structured data, and unstructured data	Good for structured data

Task2

Executing the blog [“to transfer tsv file from HDFS to Hbase”](#)

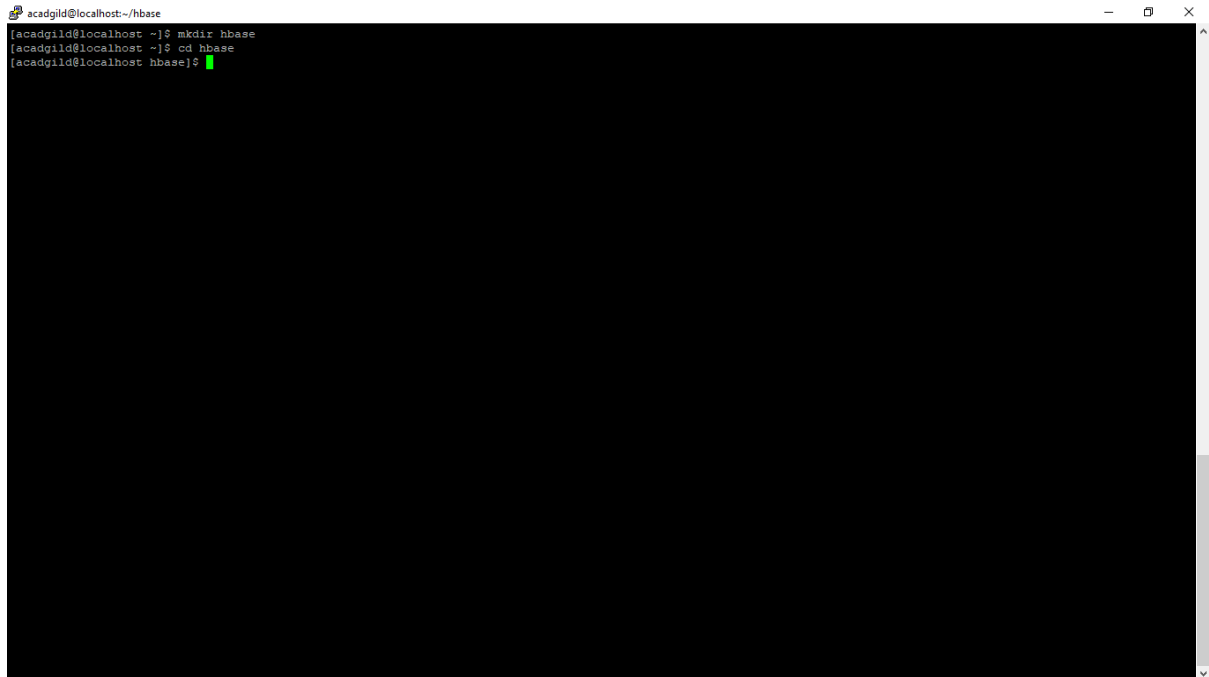
- Start all the nodes -> start-all.sh
- Start hbase -> start-hbase.sh
- Open shell -> hbase shell
- Create the bulktable -> create 'bulktable','cf1','cf2'

```
acdgild@localhost:~$
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acdgild/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/acdgild/install/hadoop/hadoop-2.9.0/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/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 1.6960 seconds

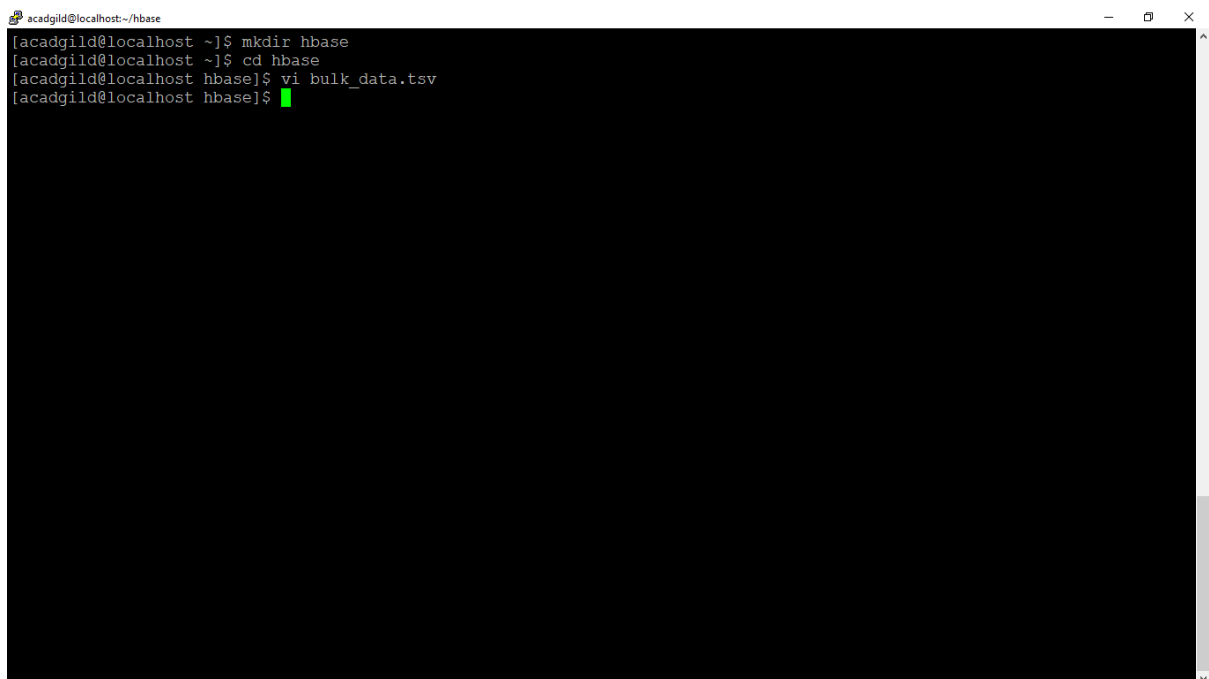
=> Hbase:Table - bulktable
hbase(main):002:0>
hbase(main):003:0*
hbase(main):004:0*
hbase(main):005:0*
hbase(main):006:0*
hbase(main):007:0*
hbase(main):008:0*
hbase(main):009:0*
hbase(main):010:0*
hbase(main):011:0*
hbase(main):012:0*
hbase(main):013:0*
hbase(main):014:0*
hbase(main):015:0*
hbase(main):016:0*
hbase(main):017:0*
hbase(main):018:0*
hbase(main):019:0*
hbase(main):020:0*
hbase(main):021:0*
hbase(main):022:0*
hbase(main):023:0*
hbase(main):024:0*
hbase(main):025:0*
hbase(main):026:0*
hbase(main):027:0*
hbase(main):028:0*
hbase(main):029:0*
hbase(main):030:0*
hbase(main):031:0*
```

e) Make hbase directory -> mkdir hbase -> cd hbase

A terminal window with a black background and white text. The title bar at the top reads 'acadgild@localhost:~/hbase'. The terminal shows the following commands and their outputs: '[acadgild@localhost ~]\$ mkdir hbase', '[acadgild@localhost ~]\$ cd hbase', and '[acadgild@localhost hbase]\$' with a green cursor. The rest of the terminal area is empty.

```
acadgild@localhost:~/hbase
[acadgild@localhost ~]$ mkdir hbase
[acadgild@localhost ~]$ cd hbase
[acadgild@localhost hbase]$
```

f) Create a file bulkData.tsv and push the data into it -> vi bulkData.tsv

A terminal window with a black background and white text. The title bar at the top reads 'acadgild@localhost:~/hbase'. The terminal shows the following commands and their outputs: '[acadgild@localhost ~]\$ mkdir hbase', '[acadgild@localhost ~]\$ cd hbase', '[acadgild@localhost hbase]\$ vi bulk_data.tsv', and '[acadgild@localhost hbase]\$' with a green cursor. The rest of the terminal area is empty.

```
acadgild@localhost:~/hbase
[acadgild@localhost ~]$ mkdir hbase
[acadgild@localhost ~]$ cd hbase
[acadgild@localhost hbase]$ vi bulk_data.tsv
[acadgild@localhost hbase]$
```

g) Push the file into HDFS and check it -> `hadoop fs -cat /hbase1/bulkData.tsv`

```
acadgild@localhost:~/hbase
[acadgild@localhost hbase]$ hadoop fs -cat bulkData.tsv
18/04/20 15:54:46 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
cat: 'bulkData.tsv': No such file or directory
[acadgild@localhost hbase]$ hadoop fs -cat /hbase1/bulkData.tsv
18/04/20 15:54:58 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
[acadgild@localhost hbase]$
```

h) Import the data into hbase

➔ `hbase org.apache.hadoop.hbase.mapredce.ImportTsv -Dimporttsv.columns= HBASE_ROW_KEY,cf1:name,cf2:exp tab4 /hbase1/bulkData.tsv`

```
acadgild@localhost:~/hbase
[acadgild@localhost hbase]$ hadoop fs -cat bulkData.tsv
[acadgild@localhost hbase]$ hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -Dimporttsv.columns=HBASE_ROW_KEY,cf1:name,cf2:exp tab4 /hbase1/bulkData.tsv
2018-04-20 15:29:39,825 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-2.9.0/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/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]
2018-04-20 15:29:40,586 INFO [main] zookeeper.RecoverableZooKeeper: Process identifier=hconnection-0x131ef10 connecting to ZooKeeper ensemble=localhost:2181
2018-04-20 15:29:40,602 INFO [main] zookeeper.ZooKeeper: Client environment:zookeeper.version=3.4.6-1569965, built on 02/20/2014 09:09 GMT
2018-04-20 15:29:40,602 INFO [main] zookeeper.ZooKeeper: Client environment:host.name=localhost
2018-04-20 15:29:40,602 INFO [main] zookeeper.ZooKeeper: Client environment:java.version=1.8.0_151
2018-04-20 15:29:40,602 INFO [main] zookeeper.ZooKeeper: Client environment:java.vendor=Oracle Corporation
2018-04-20 15:29:40,602 INFO [main] zookeeper.ZooKeeper: Client environment:java.home=/usr/java/jdk1.8.0_151/jre
2018-04-20 15:29:40,602 INFO [main] zookeeper.ZooKeeper: Client environment:java.class.path=/home/acadgild/install/hbase/hbase-1.2.6/conf:/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/api-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/lib/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/commons-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-collections-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/lib/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/findbugs-annotations-1.3.9-1.jar:/home/acadgild/install/hbase/hbase-1.2.6/lib/guava-12.0.1.jar:/home/acadgild/install/hbase/hbase-1.2.6/lib/guice-3.0.jar:/home/acadgild/install/hbase/hbase-1.2.6/lib/guice-servlet-3.0.jar:/home/acadgild/install/hbase/hbase-1.2.6/lib/hadoop-annotations-2.5.1.jar:/home/acadgild/install/hbase/hbase-1.2.6/lib/hadoop-auth-2.5.1.jar:/home/acadgild/install/hbase/hbase-1.2.6/lib/hadoop-client-2.5.1.jar:/home/acadgild/install/hbase/hbase-1.2.6/lib/hadoop-common-2.5.1.jar:/home/acadgild/install/hbase/hbase-1.2.6/lib/hadoop-hdfs-2.5.1.jar:/home/acadgild/install/hbase/hbase-1.2.6/lib/hadoop-mapreduce-client-app-2.5.1.jar:/home/acadgild/install/hbase/hbase-1.2.6/lib/hadoop-mapreduce-client-common-2.5.1.jar:/home/acadgild/install/hbase
```

i) scan 'tab4' -> to check the data in the table

```
acagild@localhost:~$ hbase(main):013:0> scan 'tab4'
ROW          COLUMN+CELL
0 row(s) in 0.0180 seconds

hbase(main):014:0> scan 'tab4'
ROW          COLUMN+CELL
1            column=cf1:name, timestamp=1524218379756, value=Amit
1            column=cf2:exp, timestamp=1524218379756, value=4
2            column=cf1:name, timestamp=1524218379756, value=Girija
2            column=cf2:exp, timestamp=1524218379756, value=3
3            column=cf1:name, timestamp=1524218379756, value=Jatin
3            column=cf2:exp, timestamp=1524218379756, value=5
4            column=cf1:name, timestamp=1524218379756, value=Swati
4            column=cf2:exp, timestamp=1524218379756, value=3
4 row(s) in 0.1080 seconds

hbase(main):015:0> █
```