Understanding Distributed Aspects of HBase

Jay Patel
Concordia University
jaybharatpatel1999@gmail.com

Mahek Master
Concordia University
mahekmaster@gmail.com

Poornaa Bhattacharya
Concordia University
poornaa.bh25@gmail.com

Yash Trivedi
Concordia University
trivediyash022@gmail.com

ABSTRACT

In the past decade we have seen the boom of information and it has been continually tiring for us to store and recover the information. Until the 1970s we were utilizing RDBMS yet that was not adequate to deal with a lot of information. Enormous information has substantiated itself as an immense fascination point for some specialists and scholastics across the world. Because of tremendous utilization of web-based media applications, information is developing quickly these days. This information is frequently indistinct, complicated and capricious. Accumulating and examining this information is certifiably not a simple job. Nevertheless, NoSQL databases can deal with and extricate this information effortlessly. The ascent of developing information gave us the NoSQL datasets and HBase is one of those based on top of Hadoop. This report delineates the HBase (utilized generally in many enormous organizations) dataset, its construction and aspects. [1]

1 INTRODUCTION

HBase is an open-source NoSQL information base which is a java execution of Google's Big Table and is an acronym for the Hadoop Database that runs on top of the HDFS as a scalable big data store. Moreover, it enjoys the benefits (faster and fault-tolerant with random access) of Hadoop's distributed file system and MapReduce model.Hbase gives a twin technique to data access. While it's row key primarily based table scans offer regular and actual time reads/writes, it additionally leverages hadoop mapreduce for batch jobs. This makes it bestfor each batch analytics as well as real-time querying Apart from these, HBase has capacities of permitting admittance to scattered information - significant information inside the tremendous volume of unstructured information utilized for Big Data analytics. It has the capacity to report blunders involuntarily, information duplication all through agglomeration and logical read and write. Hence, it is appropriate for the applications which require a continuous read/compose admittance to colossal datasets.

2 ABOUT THE SYSTEM

HBase runs on top of Hadoop therefore Hadoop version 3.3.1 configuration is used with HBase version 2.0. This project explores the aspects of distributed systems using the Cloudera cloud platform running through Oracle VM VirtualBox Linux operating system. Cloudera is a platform that supports multiple Hadoop ecosystem components including Hbase and offers a platform for managing HBase and Map Reduce easily. Cloudera requires a minimum allocation of 10 GB of RAM.Cloudera has multiple versions like Cloudera Express Version and Cloudera Enterprise version. However, there is no support available for the Cloudera Manager for these versions for free. Hence for this project We have implemented the distributed design functionalities in the standard version of Cloudera that is open source.

3 DATASET

Dataset is a collection of related sets of information that is composed of separate elements but can be manipulated as a unit by a computer.There are tons of dataset readily available on an open source platform - Kaggle. However, this distributed system requires a dataset that has got no null values or pictorial data in it. Due to this, we selected two datasets namely bbc sitemap and nyc taxi data 2014 after cumbersome research.

Since this project involves data wrangling tasks, trial and error was performed on the bbc sitemap data which is 600 MBs approximately with 3 columns in it. The other dataset which was about New York Taxi 2014 data was 2.5 GB with 18 columns in it. As a part of this project, we have selected the bbc dataset since the data was clean and we found better scope of analytics using the dataset.

The bbc dataset has three columns 'loc', 'lastmod' and 'sitemap'.The dataset is about all the articles of different locations that were published on the bbc website along with their last modified date and the sitemap in XML format.An XML sitemap will list the website's important pages making sure that Google can crawl all the results. We have 104857 total rows in the dataset.

4 HBASE ARCHITECTURE

As HBase has Java client API, the tables in HBase can be utilized as an input and output focus for MapReduce work. HBase uses key-value format to store data in tables. Data in HBase can be stored in petabytes and keys allow fast random reads and writes. HBase utilizes the Zookeeper which is an open-source undertaking of Apache particularly utilized for the administration of fragmented malfunctions in information bases. Furthermore, it additionally gives the support of grouping data and distributed synchronization. Zookeeper has short lived hubs which address the region servers which are utilized to follow the collapses and organization allotments.[1] HBase architecture mainly has three main parts, and they are as follows: [2]

- Master Server or HMaster: The Master server is answerable for observing all RegionServer occurrences in the group and is the interface for all metadata changes.
- Region Server: It is liable for serving and overseeing regions. In a disseminated cluster, a RegionServer runs on a DataNode. It can run several background threads to maintain the servers.
- Zookeeper: It acts as a coordinator to manage distributed system services like configuration information maintenance, naming and synchronization. Zookeeper is also the mediator for communication between the client and the region servers. The coordination service can be implemented from the start. It uses Java to run and supports bindings in C and Java both. Zookeeper adheres to certain design goals:[9]
  - Simple: It is structured in a simple manner where it is ordered like a file system having hierarchical namespace for processes to communicate with each other.
  - Replication: The servers have information on each other. For smooth coordination the Zookeeper promotes the replication over an ensemble. A TCP server connects a client to the server, if the connection with the single zookeeper is disconnected then the client is routed to a different server.
  - Ordered: For synchronization purposes the events and transaction are maintained serially in the order of happening using a stamp to identify it.
  - Fast: It performs extremely well when there are higher number of writes and retrieves consistent data very quickly.

5 DISTRIBUTED SYSTEM ASPECTS OF HBASE

A distributed system is a framework where parts are situated on various PCs on different networks, which can impart and organize their activities by passing messages to each other. The parts connect with each other to accomplish a shared objective. There are different significant objectives that should be met to assemble this framework worth the work. A conveyed framework ought to handily interface clients to assets, it should conceal the way that assets are dispersed across an organization, should be open, and should be adaptable.

5.1 DATA CONSISTENCY

In distributed frameworks, a consistency model is an agreement between the framework and the designer who utilizes it. A system is said to help a specific consistency model on the off chance that it carries on with the principles defined by the model. Database consistency is important because it regulates the data that is coming in and rejects the data that doesn't fit into the rules. In order to perform the concurrent reads and writes and understand them on the shared data, consistency is needed [3].

There are various types of consistencies, namely: strong, weak and eventual.

At the point when a system uncovers an unbridled number of anomalies and acts unstable, generally it is said to be conflicting/inconsistent.

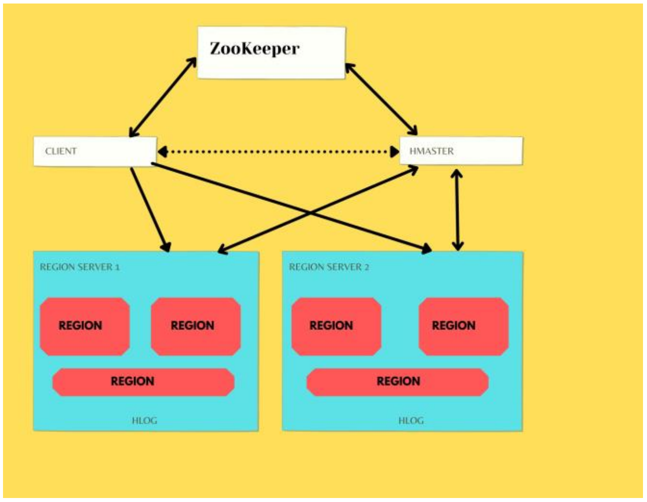


Figure 1: Architecture of HBase

At the point when a duplicated storage framework acts indistinctly from a storage system running on a solitary PC, we say it is strongly consistent.

On the other hand, weakly consistent distributed frameworks don't act indistinctly from storage frameworks running on a solitary PC. While they do uncover different irregular practices, moreover, they don't totally toss consistency out the window. Instead, fall somewhere close to being unstable and acting with solid consistency. They give some number of fundamental guarantees that are ideally adequate to satisfy customers [2].

An eventually consistent system guarantees that if all clients stop issuing requests for a while, then all the system's replicas will converge to the same state.

5.2 SCALABILITY

The ability of a system to manage an increasing quantity of work by adding resources to the system is defined as scalability.[4]

An Hbase system has the capability to scale horizontally into regions. Regions can be described as the division of the table's data stored in a continuous and organized set of rows and multiple regions are served by a Region Server. Hbase acts on the top of HDFS providing optimized random reads and writes with minimum delay in processing with the ability to operate on a huge volume of data. Moreover, Hbase supports auto-sharding which allows the tables to be automatically split as the size of the tables increase. [5]

The regions and their region server share a many to one relationship i.e more than one region can fall under a region server and the regions uniquely correspond to one region server. The mapping of this is stored in a system table which is labelled as META.

In HBase architecture, HMaster acts as the master service which oversees dealing with clusters and coordinating the implementation of administrative tasks while HRegionServer acts as the slave which oversees the data stored in the regions.

In HBase, the user doesn't have to go through the master to carry out every operation. The Master is needed for table creation, modification and deletion. During a read/write operation, the META can be accessed to recognize the region from which the data needs to be retrieved and therefore in this case the master doesn't need to be consulted at all. This eases the load on the master when there are enormous requests from the client side. Since every request is not forwarded to the HMaster, the reads and writes are fetched fast with low latency making the scalable system feasible and efficient.

5.3 DATA REPLICATION

HBase uses data replication to duplicate data from one source HBase cluster (referred as active) to another destination HBase cluster (referred as passive). When the data is copied from an active cluster, the metadata of the replication is tracked with the help of a cluster ID. [5] HBase replication is useful in a scenario where the system crashes and needs to recover from failure. The slave cluster can take the responsibilities of the master cluster while it is down and handle its traffic in the meantime. However, this doesn't happen automatically and must be initiated by the client. HBase supports three types of replications: [7]

- Master-Slave: The data is replicated from one cluster to another unidirectionally.
- Master-Master: The data is transferred bilaterally in the clusters
- Cyclic: More than one cluster is part of the replication process.

5.4 FAULT TOLERANCE

To achieve fault tolerance HBase gives the facility to implement region replication.This feature makes it possible for the system to have high availability reads. The data accessed through all the reads and writes are consistent because they are directed through the RegionServer so that the writes are in the consecutive order and the read retrieves the latest commit.[8] The three steps of fault tolerance are the Detection phase, Assignment phase, Recovery phase. The WAL (Write Ahead Log) is responsible for recording all the data and its changes in HBase. In Case the RegionServer crashes, the WAL reiterates the data at the time of recovery and the region is allocated to a different server.

As seen in figure 2, there always exists a backup master called Inactive Hmaster and multiple region servers. Each region server and master server send a heartbeat to zookeeper. Zookeeper recieves the heartbeat message to ensure that the region and master servers are active. In case of any abnormality, zookeeper detects the failure and transforms the control to the backup master and region servers in order to handle the failures. The entire process is extremely smooth and Hbase deals with failure efficiently and quickly.

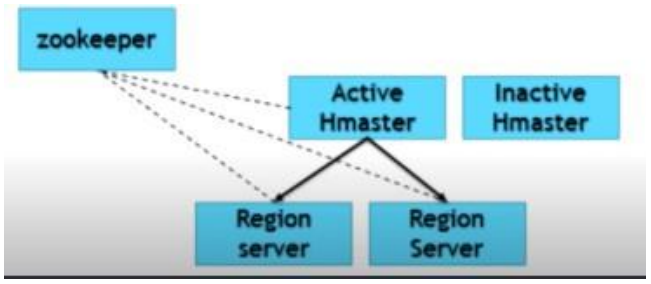


Figure 2: Handling failures

6 IMPLEMENTATION IN PROJECT

The major focus in our project has been understanding the aspects of Distributed System Design. To demonstrate the data consistency of Hbase We uploaded our 'bbc' datset on the Hbase server and performed various CRUD (Create, Read, Update and Delete) operations. We created the table, updated its rows, read the values from table and deleted the table entries. On performing those operations, we can infer that they showcase consistency properties even when they are implemented on concurrent threads. The consistency mechanism is explained in section 5.1.

In order to showcase the scalability aspect, we have demonstrated the load balancing in Hbase. Under this, we generated load using load test tool feature wherein we analyzed the performance of writing 1 million rows of 512 bytes each splitting over 5 regions per region server. By running the load generation, we observed that it took 3 minutes and 25 seconds to write the data on Hbase. The key observation was- with increasing time, the latency will reduce and the number of requests per second will increase. Similarly, We also performed we analyzed the performance of writing 1 million rows using 20 concurrent threads. It took 1 minute and 50 seconds to read a million rows which is faster than the read operation. The number of requests per seconds increases as the time elapses and hence We can say that our Hbase cluster is efficiently able to handle the load among various regions and hence We can infer that on scaling horizontally, Hbase can efficiently handle the load.For faster reads and write Hbase uses Map Reduce for faster performance. We also analyzed the performance of the Map Reduce using hbase -pe command and the results were extremely consistent.

We also demonstrated the data replication and fault tolerance aspect of the Distributed System by creating 3 Replicas of our table. In case a perticular region fails to handle data requests, Hbase will automatically retrieve the data from other replicas. In the demo, we can clearly see the three replicas of 'bbc' table with replica ID 0,1 and 2.

7 CONCLUSION

HBase proves to be a reliable solution for a vast amount of data. With an enormous load of data, it becomes necessary for the distributed systems to be consistent across different machines and HBase fulfills that aspect successfully. The current reads and writes are reflected in the database. The regions and region servers provide the mechanism for the system to scale and replicate in anticipation for failover support and recovery. While HBase can handle large datasets, process datasets faster, can be scaled easily, support fault tolerance, load balancing and provide consistent reads and writes there are a few limitations as well. One of the drawbacks is that HBase can be a victim of a single point of failure when only one HMaster is working. Migration of data from traditional RDBMS to HBase can also be tough. Moreover, HBase requires high performance machines because of which it is expensive and has a high cost for maintenance. In conclusion, there are advantages and disadvantages of using HBase but it is tremendously dependable for management of a distributed system.

8 CITATIONS

A reference to Extracting, Transforming and Loading data into HFiles [10]

REFERENCES

[1] Research paper by Hiren Patel. HBase: A NoSQl Database
DOI: http://dx.doi.org/10.13140/RG.2.2.22974.28480

[2] Official Documentation of HBase
https://hbase.apache.org/book.html#arch.overview

[3] Official Github Developer page from M. Whittaker on Data Consistency
https://mwhittaker.github.io/consistency_in_distributed_systems/1_baseball.html#:~:text=Strong%20and%20Weak%20Consistency%20Replication%20allows%20a%20distributed,store%20replicated%20across%20two%20servers%20%28s1%20and%20s2%29.

[4] Scalability
https://en.wikipedia.org/wiki/Scalability#cite_note-1

[5] Official Documentation of "Working of scalability in Apache HBase"
https://blog.cloudera.com/how-scaling-really-works-in-apache-hbase/

[6] Official Documentation of Cloudera on "HBase Replication"
https://blog.cloudera.com/apache-hbase-replication-overview/

[7] Official Documentation of Cloudera on "HBase Replication"
https://docs.cloudera.com/documentation/enterprise/6/6.3/topics/cdh_bdr_hbase_replication.html

[8] Official Documentation of Cloudera on "Fault Tolerance"
https://docs.cloudera.com/HDPDocuments/HDP3/HDP-3.1.5/fault-tolerance/content/introduction_to_hbase_high_availability.html

[9] Official Documentation of Apache Zookeeper
https://zookeeper.apache.org/doc/r3.7.0/zookeeperOver.html

[10] Official Blog of Cloudera on Bulk Loading
https://blog.cloudera.com/how-to-use-hbase-bulk-loading-and-why/

APACHE HBASE

Team PJMY

Group #19

TABLE OF CONTENTS

About the Project

- Why Hbase
  - Open source NoSQL Database
  - Faster and Fault-tolerant with random access
  - Handles disorganized data with ease
  - Supports parallel processing and MapReduce
- Cloudera
  - A platform that includes all leading Hadoop ecosystem components including Hbase
  - supports parallel processing and MapReduce
- About Dataset
  - Bbc sitemap
  - Consists of ‘loc’, ‘last modified’ and ‘sitemaps’
  - Dataset Link: https://www.kaggle.com/eliasdabbas/news-sitemaps?select=bbc_sitemaps.csv

Designing a Distributed System

- Distributed system - PCs - impart and organize their activities by passing messages
- There are different significant objectives that should be met to assemble this framework worth the work. They are as follows:
    - Consistency
    - Scalability
    - Fault tolerance
    - Data replication

Image Ref: distributed system gif - Bing images

CONSISTENCY

- Consistent read and writes

  ACID Properties
- Bridge between system and designer
- Regulates the data coming in
- Inconsistent - acts unstable
- Strongly consistent - Replicated storage system acts indistinctly
- Weakly consistent - Replicated storage system doesn’t act indistinctly

SCALABILITY

- Manage an increasing quantity of work efficiently
- Scaling horizontally into regions
- Region - RegionServer : Many - One
- Master-Slave architecture - Low latency - Efficient system

FAULT TOLERANCE

- High availability reads
- Data consistent - retrieval of latest commit
- Role of WAL (Write Ahead Log)
- Three phases: Detection, Assignment, Recovery

DATA REPLICATION

- Duplicate data - one source to another destination cluster

- Recovery from failures

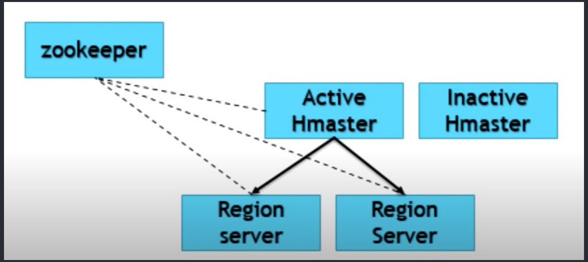


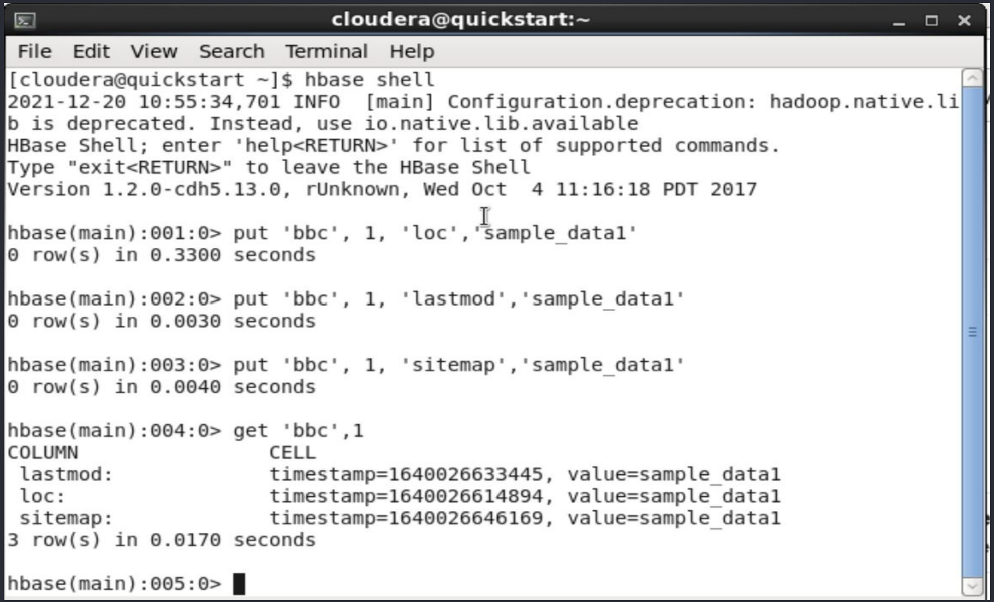
Image Ref: [Hindi] HBase Architecture - YouTube

- Role of Cluster ID

- Three types: Master-Slave, Master-Master, Cyclic

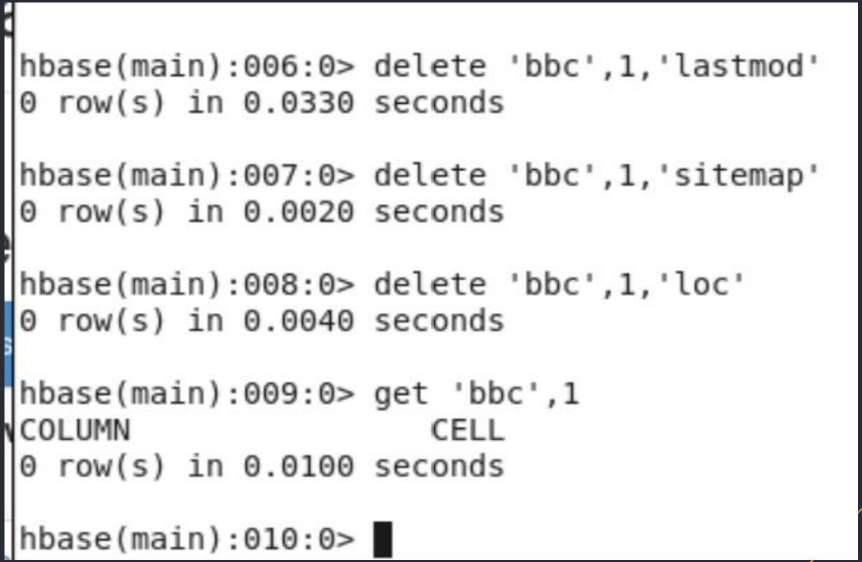
DEMO RESULTS

CRUD Operations on Dataset



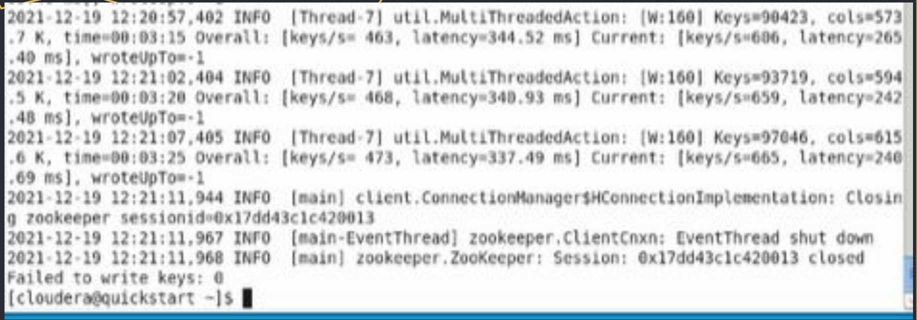
```
[cloudera@quickstart ~]$ hbase shell
2021-12-20 10:55:34,701 INFO  [main] Configuration.deprecation: hadoop.native.li
b is deprecated. Instead, use io.native.lib.available
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.0-cdh5.13.0, rUnknown, Wed Oct  4 11:16:18 PDT 2017

hbase(main):001:0> put 'bbc', 1, 'loc','sample_data1'
0 row(s) in 0.3300 seconds

hbase(main):002:0> put 'bbc', 1, 'lastmod','sample_data1'
0 row(s) in 0.0030 seconds

hbase(main):003:0> put 'bbc', 1, 'sitemap','sample_data1'
0 row(s) in 0.0040 seconds

hbase(main):004:0> get 'bbc',1
COLUMN                     CELL
 lastmod:                  timestamp=1640026633445, value=sample_data1
 loc:                      timestamp=1640026614894, value=sample_data1
 sitemap:                  timestamp=1640026646169, value=sample_data1
3 row(s) in 0.0170 seconds

hbase(main):005:0>
```

Create , Update and Read Operations



```
hbase(main):006:0> delete 'bbc',1,'lastmod'
0 row(s) in 0.0330 seconds

hbase(main):007:0> delete 'bbc',1,'sitemap'
0 row(s) in 0.0020 seconds

hbase(main):008:0> delete 'bbc',1,'loc'
0 row(s) in 0.0040 seconds

hbase(main):009:0> get 'bbc',1
COLUMN                     CELL
0 row(s) in 0.0100 seconds

hbase(main):010:0>
```

Delete Operations



```
2021-12-19 12:20:57,402 INFO  [Thread-7] util.MultiThreadedAction: [W:160] Keys=90423, cols=573
.7 K, time=00:03:15 Overall: [keys/s= 463, latency=344.52 ms] Current: [keys/s=606, latency=265
.40 ms], wroteUpTo=-1
2021-12-19 12:21:02,404 INFO  [Thread-7] util.MultiThreadedAction: [W:160] Keys=93719, cols=594
.5 K, time=00:03:20 Overall: [keys/s= 468, latency=340.93 ms] Current: [keys/s=659, latency=242
.48 ms], wroteUpTo=-1
2021-12-19 12:21:07,405 INFO  [Thread-7] util.MultiThreadedAction: [W:160] Keys=97046, cols=615
.6 K, time=00:03:25 Overall: [keys/s= 473, latency=337.49 ms] Current: [keys/s=665, latency=240
.69 ms], wroteUpTo=-1
2021-12-19 12:21:11,944 INFO  [main] client.ConnectionManager$HConnectionImplementation: Closin
g zookeeper sessionid=0x17dd43c1c420013
2021-12-19 12:21:11,967 INFO  [main-EventThread] zookeeper.ClientCnxn: EventThread shut down
2021-12-19 12:21:11,968 INFO  [main] zookeeper.ZooKeeper: Session: 0x17dd43c1c420013 closed
Failed to write keys: 0
[cloudera@quickstart ~]$ 
```

LOAD HANDLING ANALYSIS

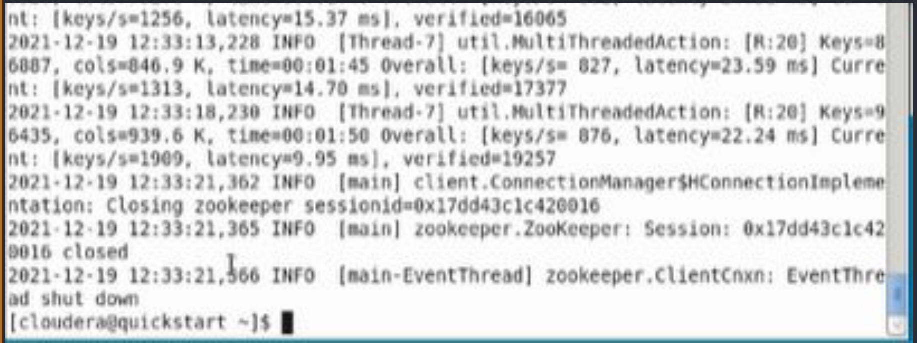
Handling write requests for 1 Million Rows took 3 minutes 25 seconds

Demo Link:
https://www.loom.com/share/b6572eae19b64a5da0a0865d97476d15

LOAD HANDLING ANALYSIS

Handling read requests for 1 Million Rows took 1 minute and 50 seconds

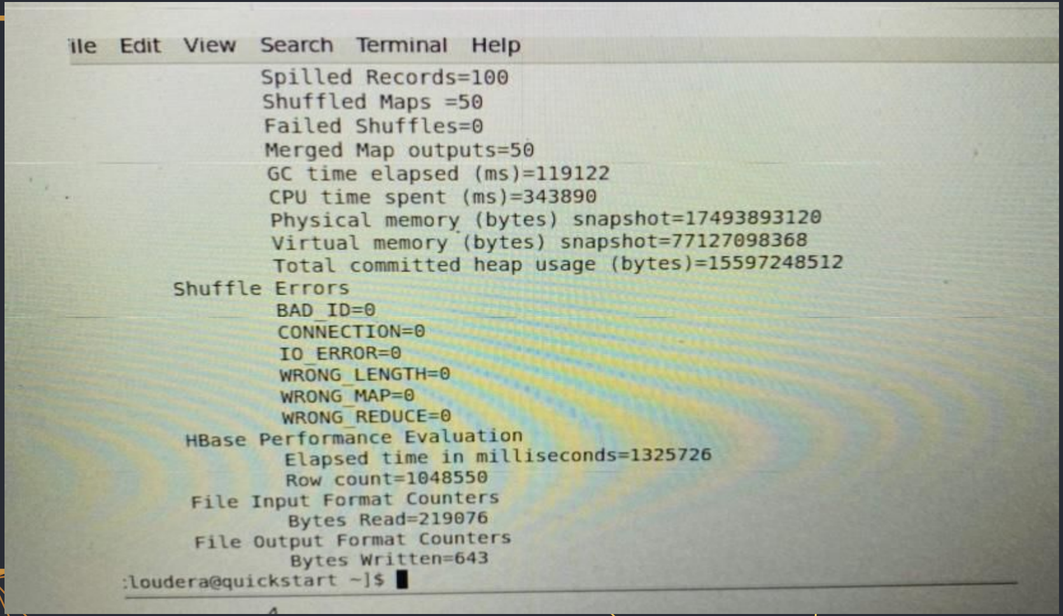
Demo Link:
https://www.loom.com/share/8f19ae6b058b4cacb5399d56c1872ade



```
nt: [keys/s=1256, latency=15.37 ms], verified=16065
2021-12-19 12:33:13,228 INFO  [Thread-7] util.MultiThreadedAction: [R:20] Keys=8
6887, cols=846.9 K, time=00:01:45 Overall: [keys/s= 827, latency=23.59 ms] Curre
nt: [keys/s=1313, latency=14.70 ms], verified=17377
2021-12-19 12:33:18,230 INFO  [Thread-7] util.MultiThreadedAction: [R:20] Keys=9
6435, cols=939.6 K, time=00:01:50 Overall: [keys/s= 876, latency=22.24 ms] Curre
nt: [keys/s=1909, latency=9.95 ms], verified=19257
2021-12-19 12:33:21,362 INFO  [main] client.ConnectionManager$HConnectionImpleme
ntation: Closing zookeeper sessionid=0x17dd43c1c420016
2021-12-19 12:33:21,365 INFO  [main] zookeeper.ZooKeeper: Session: 0x17dd43c1c42
0016 closed
2021-12-19 12:33:21,366 INFO  [main-EventThread] zookeeper.ClientCnxn: EventThre
ad shut down
[cloudera@quickstart ~]$ 
```

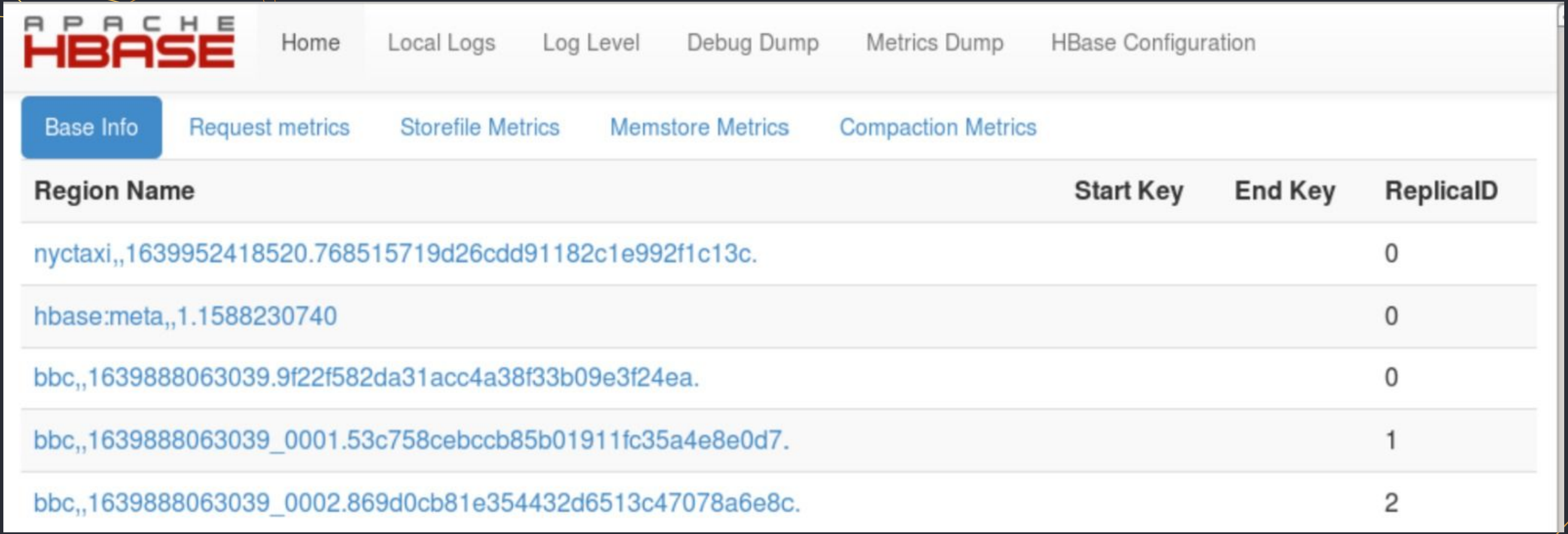
Map Reduce Performance Evaluation

Handling 1048550 rows took around 1000 seconds



```
                    Spilled Records=100
                    Shuffled Maps =50
                    Failed Shuffles=0
                    Merged Map outputs=50
                    GC time elapsed (ms)=119122
                    CPU time spent (ms)=343890
                    Physical memory (bytes) snapshot=17493893120
                    Virtual memory (bytes) snapshot=77127098368
                    Total committed heap usage (bytes)=15597248512
            Shuffle Errors
                    BAD_ID=0
                    CONNECTION=0
                    IO_ERROR=0
                    WRONG_LENGTH=0
                    WRONG_MAP=0
                    WRONG_REDUCE=0
            HBase Performance Evaluation
                    Elapsed time in milliseconds=1325726
                    Row count=1048550
            File Input Format Counters
                    Bytes Read=219076
            File Output Format Counters
                    Bytes Written=643
:loudera@quickstart ~]$
```

Data Replication of ‘bbc’ table



Region Name	Start Key	End Key	ReplicaID
nyctaxi,,1639952418520.768515719d26cdd91182c1e992f1c13c.			0
hbase:meta,,1.1588230740			0
bbc,,1639888063039.9f22f582da31acc4a38f33b09e3f24ea.			0
bbc,,1639888063039_0001.53c758cebccb85b01911fc35a4e8e0d7.			1
bbc,,1639888063039_0002.869d0cb81e354432d6513c47078a6e8c.			2

OUR TEAM

Jay Patel
40197003

Mahek Master
40193721

Poornaa Bhattacharya
40195362

Yash Trivedi
40189803

Thank You!