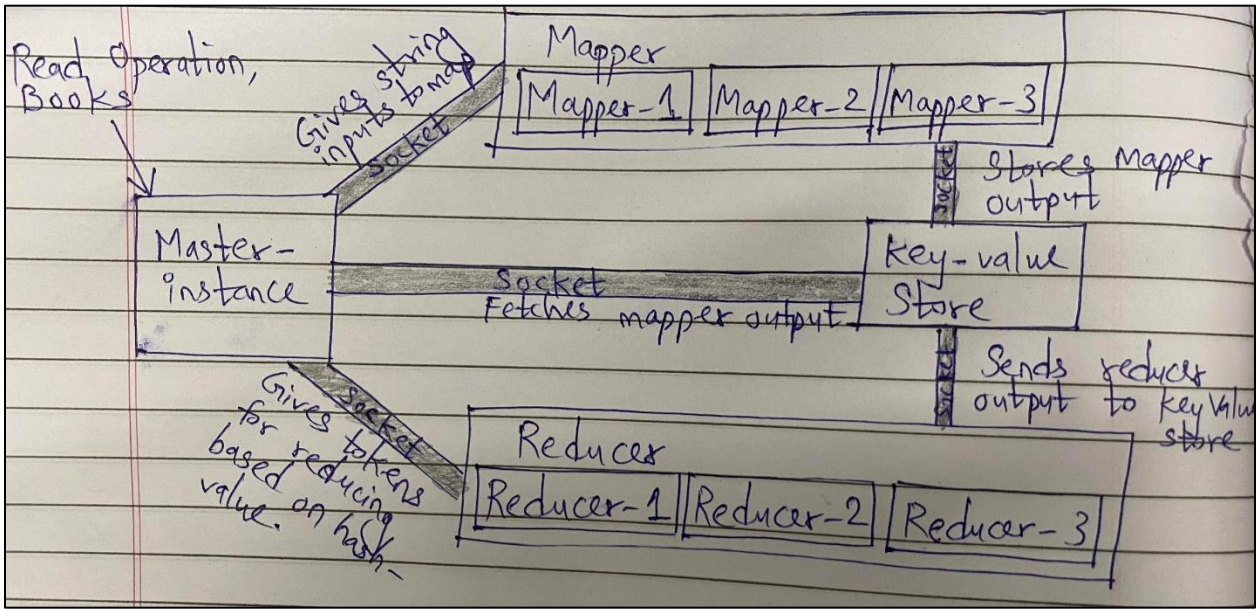
Detailed Design



MapReduce is a programming model designed for processing large volumes of data in parallel by dividing the work into a set of independent tasks. The Master instance accepts the input from user and performs the user-provided operation. Mapper & reducer instances perform mapping & reducing tasks respectively. Key value store instance stores the intermediate & final data. There is a proper synchronization between each instance. Master is the backbone of the system, which co-ordinates with all other entities.

The system uses 4 auxiliary files:

i) output.txt

The output of the program (ie. Word Count/Inverted Index) is stored into this file. This file is created at KeyValueStore.

ii) log.txt

Log of each activity is stored into this file for debugging, error recovery and fault tolerance. This file is created at master instance.

iii) config.csv

Configuration file contains the pair of each environment variable along with its value. This is accessed each time the system starts & initializes the environment variable. This file is user-defined file which has all system configurations such as ports, IPs, number of mappers etc. This file is present at master-instance.

iv) data.csv

The output of the mapper is stored into this Key Value store. This is also called as a intermediate data which is stored in keyvaluestore-instance.

Pseudo code

1. Start the master-instance. Provide proper operation to be performed along with the input file(s) on which the operation is performed.
2. Start the keyvaluestore-instance & Mapper instances (mapper-instance1, mapper-instance2, mapper-instance3). The startup script for executing each java jar files are executed once the instance is started.
3. Connect mapper & key value store instances
4. Read & split the input text data and distribute among different mappers using ***Round-Robin technique***.
5. Mapper scans all tokens & stores the result into keyvalue store.
6. Control is returned back to Master. Master then fetches all mapper output from key value store.
7. Master clears the keyvalue store contents.
8. Hash values of each token is computed, and tokens are forwarded to reducer using the corresponding hash values.
9. Reducer maintains a map data structure to maintain the counts of each token.
10. At the end of processing, reducers set the result into key value store.
11. Master then commands the keyvalue store to export the data into output.txt file on Key value store instance.

The above code also logs each event along with proper timings. At the end of each phase, corresponding instances are terminated (Eg.- after mapping phase, Mappers are terminated).

Program Flow

The program is executed by accessing the master instance. Following are the steps to execute on the google sdk console for accessing (SSH) and executing the jar file (ie. Main.java class file):

- gcloud compute ssh master-instance
- cd cloud_mapreduce
- java -jar cloudMapreduce-1.0-SNAPSHOT-jar-with-dependencies.jar

VMs used

Sr.	Entity	Instance name
1	Master	master-instance
2	Mapper	mapper-instance1, mapper-instance2, mapper-instance3
3	Reducer	reducer-instance1, reducer-instance2, reducer-instance3
4	Key Value Store	keyvaluestore-instance

The above instances are created using Machine type ***- n1-standard-1 (1 vCPU, 3.75 GB memory).***

Following are the instances configurations:

i) **Zone** – us-central1-c
ii) **Firewalls** – Allow HTTP traffic, Allow HTTPS traffic
iii) **OS** - Debian 9
iv) **Size** – 10GB
v) **On host maintenance** – Migrate VM instances
vi) **Cloud API access scopes** – Allow full access to all Cloud APIs
vii) **CPU platform** – Intel Haswell
viii) **Disk Type** – Persistent storage

Startup scripts are also added.
Eg. – For mapper instance,

```
#! /bin/bash
cd /home/sudip
java -jar Mapper-1.0-SNAPSHOT-jar-with-dependencies.jar
```

Gcloud compute logs for start/stop operations

Compute Start Log –


```
{
insertId: "1r9rd5ff4u28sk"
jsonPayload: {
actor: {
user: "sudipadh@iu.edu"
}
event_subtype: "compute.instances.start"
event_timestamp_us: "1575948709906210"
event_type: "GCE_OPERATION_DONE"
operation: {
id: "1170980328165257074"
name: "operation-1575948700905-5995126806ffc-5e12f8cb-3f0f08b9"
type: "operation"
zone: "us-central1-c"
}
resource: {
id: "7275814295438103266"
name: "keyvaluestore-instance"
type: "instance"
zone: "us-central1-c"
}
trace_id: "operation-1575948700905-5995126806ffc-5e12f8cb-3f0f08b9"
version: "1.2"
}
labels: {
compute.googleapis.com/resource_id: "7275814295438103266"
compute.googleapis.com/resource_name: "keyvaluestore-instance"
compute.googleapis.com/resource_type: "instance"
compute.googleapis.com/resource_zone: "us-central1-c"
}
logName: "projects/sudip-padhye/logs/compute.googleapis.com%2Factivity_log"
receiveTimestamp: "2019-12-10T03:31:49.934466215Z"
```

resource: {
labels: {
instance_id: "7275814295438103266"
project_id: "sudip-padhye"
zone: "us-central1-c"
}
type: "gce_instance"
}
severity: "INFO"
timestamp: "2019-12-10T03:31:49.906210Z"
}

Compute Stop Log -

{
insertId: "5jzxmzg491q3sm"
jsonPayload: {
actor: {
user: "sudipadh@iu.edu"
}
event_subtype: "compute.instances.stop"
event_timestamp_us: "1575948868791146"
event_type: "GCE_OPERATION_DONE"
operation: {
id: "3045520290563140837"
name: "operation-1575948809977-599512d00bf93-1d3743b8-0e1e806c"
type: "operation"
zone: "us-central1-c"
}
resource: {
id: "7275814295438103266"
name: "keyvaluestore-instance"
type: "instance"
zone: "us-central1-c"
}
trace_id: "operation-1575948809977-599512d00bf93-1d3743b8-0e1e806c"
version: "1.2"
}
labels: {
compute.googleapis.com/resource_id: "7275814295438103266"
compute.googleapis.com/resource_name: "keyvaluestore-instance"
compute.googleapis.com/resource_type: "instance"
compute.googleapis.com/resource_zone: "us-central1-c"
}
logName: "projects/sudip-padhye/logs/compute.googleapis.com%2Factivity_log"
receiveTimestamp: "2019-12-10T03:34:28.821377131Z"
resource: {
labels: {
instance_id: "7275814295438103266"
project_id: "sudip-padhye"
zone: "us-central1-c"
}
type: "gce_instance"
}
severity: "INFO"
timestamp: "2019-12-10T03:34:28.791146Z"
}

Output of some test case

In the following test case, the 2 files are processed. Each file is processed using mappers. Later, reducers reduce the mapper output and store the results into the keyvaluestore. Once, output us generated, instances are stopped, and cluster is destroyed.

```
sudip@master-instance:~/cloud_mapreduce$ java -jar cloudMapreduce-1.0-SNAPSHOT-jar-with-dependencies.jar
Enter the Operation Mode[0/1]:
0 - WordCount
1 - InvertedIndex
1
Enter no. of input files: 2
Enter File 1: book.txt
Enter File 2: The_Legend_of_Sleepy_Hollow.txt
Starting KeyValueStore & Mapper instances...Please be patient. This may take some time!!!

Mappers processing...

Processing File: 1
No. of mappers intitiated:  3
Mapper Instance connected at: Socket[addr=/10.128.0.7,port=52408,localport=9200]
Mapper Instance connected at: Socket[addr=/10.128.0.9,port=50210,localport=9200]
Mapper Instance connected at: Socket[addr=/10.128.0.8,port=43674,localport=9200]
Mapper job 1/2 done
Mapper job 2/2 done

Processing File: 2
No. of mappers intitiated:  3
Mapper job 1/3 done
Mapper job 2/3 done
Mapper job 3/3 done
Mapping done...
Terminating Mapper instances...Please be patient. This may take some time!!!
Starting Reducer instances...Please be patient. This may take some time!!!

Reducers processing...
Reducers spawned = 3
Reducer Instance connected at: Socket[addr=/10.128.0.10,port=52222,localport=9200]
Reducer Instance connected at: Socket[addr=/10.128.0.12,port=48790,localport=9200]
Reducer Instance connected at: Socket[addr=/10.128.0.11,port=33846,localport=9200]
Inverted Index operation done successfully
Reducing done
Terminating KeyValueStore & Reducer instances...Please be patient. This may take some time!!!
Cluster 1 destroyed successfully.
```

Performance numbers

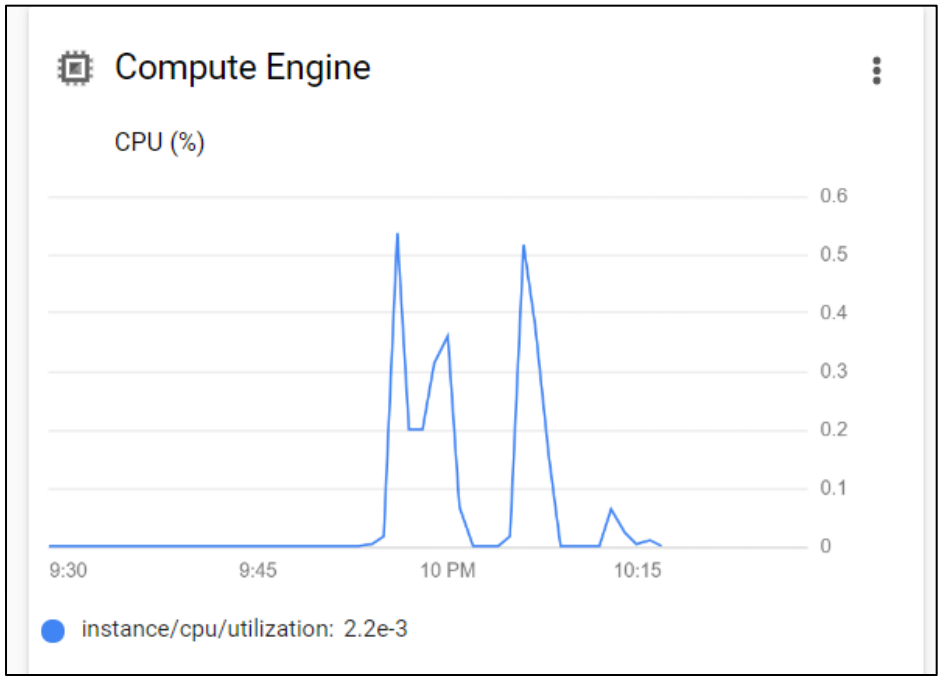
i) **Execution Time -** The system performs the operation within 4 minutes based on the file input size of 10000 words.

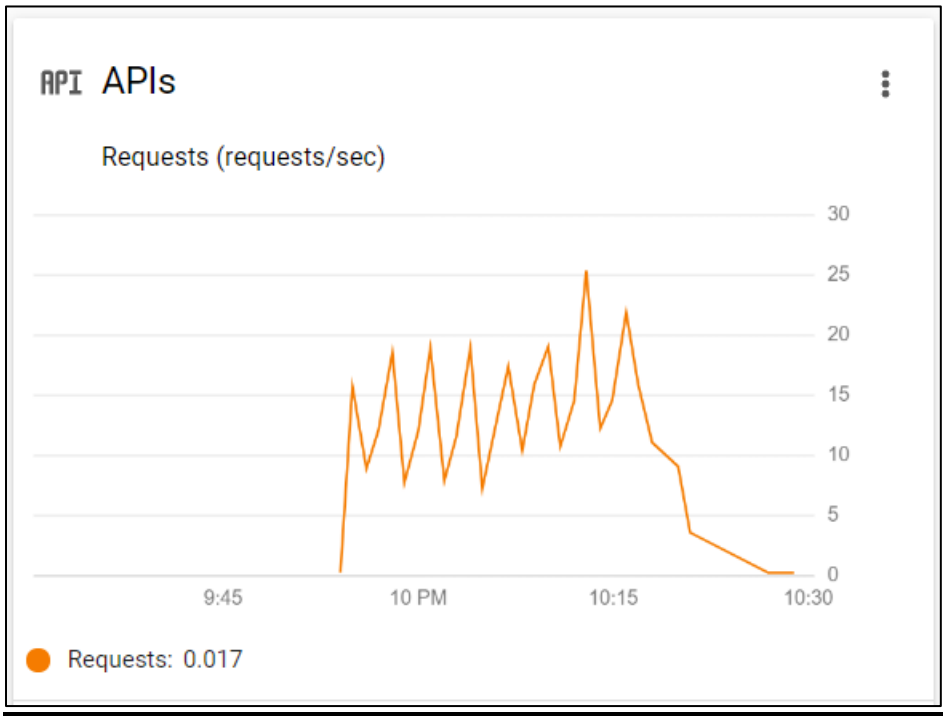
ii) **Memory Usage** – Max. memory available is 10 GB. Memory reserved is 500 Mb for the OS and other system libraries. Out of this, 300 Mb is used for running the experiment and storing the data.

iii) **CPU Usage** – CPU usage goes as high as 51% of the total CPU capacity.

iv) **Instances run at a time** – 8 (1 master, 1 keyvaluestore, 3 mappers, 3 reducers)

Cost of running experiment





The experiment uses max. of 60% of the CPU utilization. It also triggers upto 30 API requests to the gcloud server.

Weaknesses, Design & implementation improvements

The system has following weaknesses which can be improved in the later stage of developments:

1. The system does not support VM creation and destruction.
2. The system is not fault tolerant.
3. The system is run on non-preemptible VMs. Future developments can be added to make it run on transient VMs.