MapReduce on Google Cloud

Instructions on executing google cloud MapReduce architecture:

- Create key-value store instance using the 'instance_handler.py' script by selecting the appropriate options.
- Then create the master instance using the same script by selecting the appropriate option from the choices.
- Then get the natIP of the master instance using 'get' option in the 'instance_handler.py' script.
- Wait for some time, say 30-45 seconds so that master finishes installing with all of its dependencies.
- Next run the bash script 'initiator.sh' and enter the natIP that has been taken from the instance_handler script.
- Then the script will ask for the options to run MapReduce examples of either wordcount or inverted index. (wordConfig.json and invertedConfig.json must be present in the directory)
- Select the required option and wait to get the response back from the master.

Overall Design and Implementation:

Initial setup:

- MapReduce architecture on google cloud is based on the basic assumption that, the key-value store and the master instances are up and running and the master is an HTTP server which waits for the requests from the clients.
- All the necessary files and scripts are placed in the cloud storage bucket associated with the project.
- Initial setup that we have to do is to start key-value store and the master instances. This is achieved by a python script which interactively handles instance creation, starting, stopping and deleting on google cloud.
- The instance handler script will utilize google cloud APIs to create, start, stop, destroy instances within a given project and a specified storage bucket etc.

```
adarshs_storage_bucket_1

1.Datastore
2.Master
3.Exit
Please enter your choice:
```

```
Please enter your choice:1
Datastore:

1.Create
2.Start
3.Stop
4.Delete
4.Exit
Please enter your choice:
```

```
Please enter your choice:1
Creating instance.
Waiting for operation to finish...
done.
Instances in project adarsh-hegde and zone us-central1-f:
 - datastore-instance
```

- Script will wait till the instance is successfully created and lists out the instances in the bucket once done.

Submitted by: Adarsh Narayan Hegde

- Once the master instance is created, HTTP server waits for the incoming requests from the clients after installing all the dependencies.
- Google cloud caters a way to run initialization scripts (i.e. the code which needs to be executed upon instance creation) as start-up scripts which needs to be specified in metadata.

```
'metadata': {
    'items': [{
        # Startup script is automatically executed by the
        # instance upon startup.
        'key': 'startup-script-url',
        'value': startup_script
    },
```

- Startup script will create firewall rules and installs requires dependencies in our instances.
- We can provide our 'cloud_master.py' script as a startup script upon creation of master-instance which coordinates all MapReduce functionalities. Similarly, we can run our key-value store script upon creation of the datastore-instance.
- Once the datastore-instance and master-instance are up and running, client can send an HTTP request to the master containing MapReduce input configurations, i.e. path to an input file, map-function, reduce-function and output-location.

Cloud operations:

- Once the master receives an HTTP request from the client containing the input configurations, master will validate the input configuration to check whether path to the input file is present, whether the map and reduce functions is supported by the current architecture etc. (we are assuming that the current MapReduce architecture is supported for wordcount and inverted index test cases for simplicity).
- Suppose the mapper or reducer functions are not supported, the master will respond to the user that the map/reduce function is currently not supported.

Ex: `Error: Map Function currently not supported!`

- If the validation is successful, master will create required number of mapper instances (3 in this architecture) dynamically using Google cloud APIs.

Name	Zone	Recommendation	In use by	Internal IP	External IP	Connect
datastore-instance	us-central1-f			10.128.15.216 (nic0)	34.70.121.161	SSH
mapper-1	us-central1-f			10.128.15.218 (nic0)	34.69.77.42	SSH
mapper-2	us-central1-f			10.128.15.221 (nic0)	35.222.199.231	SSH
mapper-3	us-central1-f			10.128.15.222 (nic0)	35.232.99.172	SSH
master-instance	us-central1-f			10.128.15.217 (nic0)	35.225.23.23	SSH

- After creation of all mapper instances, master will wait for 60 seconds to that all the mappers are initialized properly and then makes an RPC (XMLRpc) request to start the map tasks.

- Each mapper instances are initialized with the mapper startup scripts which essentially support mapper functionalities supported by the architecture (wordcount and inverted index) as a function.
- Each mapper will fetch the input from key-value store running on datastore-instance by opening a socket connection and then do the necessary processing on the input data and marshals its output using 'struct' library and sends back its output to the key-value store by opening a socket connection.
- Once the mappers are done, they will acknowledge their completion via RPC call.
- After all the mappers acknowledges their completion, master will terminate all the mapper instances as their jobs are completed and to reduce the cloud costs.
- After all mapper instances are deleted, master will request the key-value store to run the shuffle task by sending a socket request.
- Once the shuffle task is completed, master will start the required number of reducer instances (3 in this architecture) and waits for 60 seconds for their initializations.

Name	Zone	Recommendation	In use by	Internal IP	External IP	Connect
datastore-instance	us-central1-f			10.128.15.216 (nic0)	34.70.121.161	SSH
master-instance	us-central1-f			10.128.15.217 (nic0)	35.225.23.23	SSH
reducer-1	us-central1-f			10.128.15.229 (nic0)	34.67.2.143	SSH
reducer-2	us-central1-f			10.128.15.230 (nic0)	35.226.240.62	SSH
reducer-3	us-central1-f			10.128.15.231 (nic0)	35.223.135.243	SSH

- All the reducer instances are initialized with a startup script which essentially includes the implementation methods for wordcount and inverted index examples which we are supporting.
- Master then makes RPC calls to the reducer instances to start reduce jobs.
- Each reducer instances will run their reduce tasks upon receiving request from the master and store their output to the key-value server by marshaling the outputs using pack option of struct library.
- Reducer instances will acknowledge their completion to the master. Then the master will terminate all the reducer instances dynamically to reduce the cost
- Finally, the master will read the reducer output which is either wordcounts or inverted indices from the key value server and writes it on to the output location specified by the client.
- For interactivity, the contents in the output location are sent back as response and stored in local file system where the client is running.
- After we done with all of our experiments datastore-instance and master-instance are destroyed explicitly from instance-handler script.
- In case of any failures if any mappers or reducers instances are not destroyed, they are destroyed automatically by a handler method at the end.

Sample Response from the master instance:

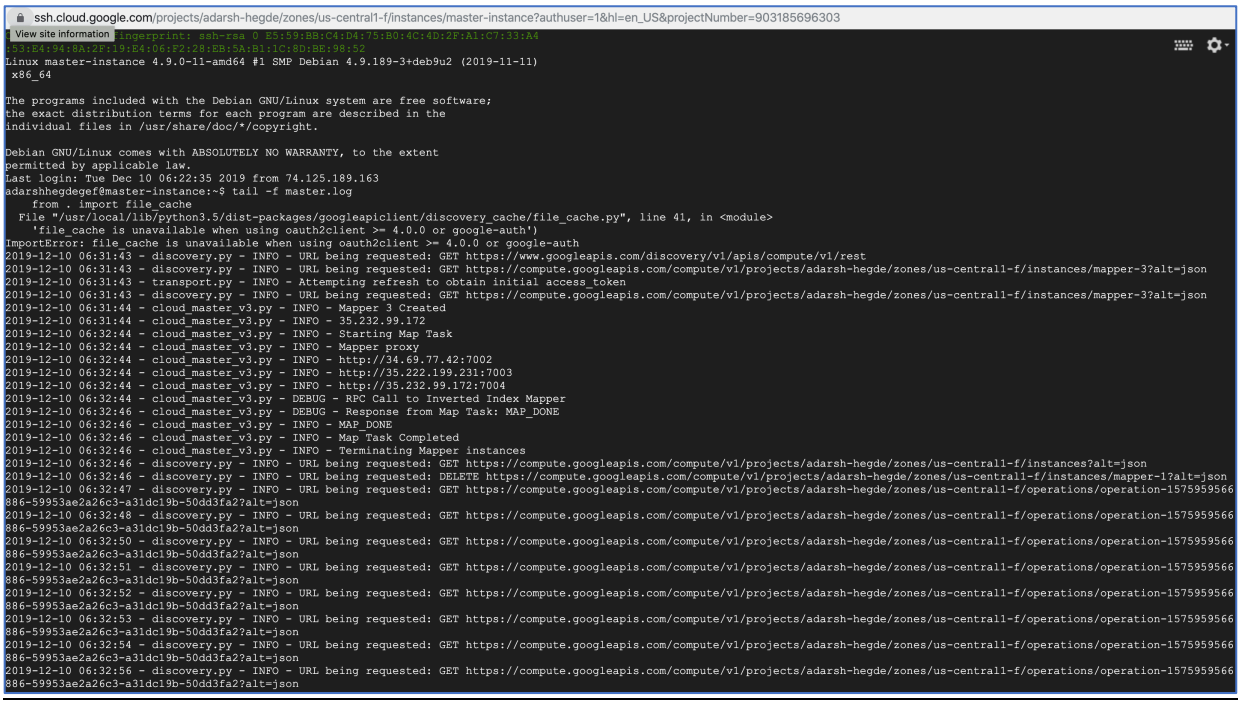
```
MapReduce Output:
WORD_COUNT:
10:1
1550:1
16:1
17:1
1888:2
A:59
ABOUT:1
AFRICA:1
APES:3
ARBRE:1
ARCHER:1
AWAY:1
About:7
Across:2
Adjusting:1
```

```
MapReduce Output:
INVERTED_INDEX:
ago: document_mapper_0:1 | document_mapper_1:11 | document_mapper_2:1 |
own: document_mapper_0:5 | document_mapper_1:1 |
rain: document_mapper_1:2 |
With: document_mapper_0:2 | document_mapper_1:2 |
without: document_mapper_1:5 | document_mapper_2:1 |
trifling: document_mapper_1:1 |
slowly: document_mapper_0:2 | document_mapper_2:1 |
chintz: document_mapper_0:1 |
level: document_mapper_1:1 |
preserve: document_mapper_0:1 |
shutting: document_mapper_0:1 |
it: document_mapper_0:28 | document_mapper_1:41 | document_mapper_2:21 |
wiser: document_mapper_0:1 |
Hence: document_mapper_1:1 |
subscribers: document_mapper_1:4 |
missed: document_mapper_0:1 |
oflife: document_mapper_2:1 |
```

Some performance numbers:

- 0:07:34 mins for completing wordcount test case on a file with 120 seconds wait time inside master.
- 0:08:12 mins for completing inverted index test case with 3 files to process with 120 seconds wait time inside master.

A Sample log at master instance:



Some VM details:

Instance Id

7422695549826143637

Machine type

n1-standard-1 (1 vCPU, 3.75 GB memory)

Reservation

Automatically choose (default)

CPU platform

Intel Haswell

Boot disk

Name	Image	Size (GB)	Device name	Type	Encryption	Mode
datastore-instance	debian-9-stretch-v20191121	10	persistent-disk-0	Standard persistent disk	Google managed	Boot, read/write

Availability policies

Preemptibility	Off (recommended)
On host maintenance	Migrate VM instance (recommended)
Automatic restart	On (recommended)

Custom metadata

startup-script-url	gs://adarshs_storage_bucket_1/datastore-startup.sh
url	http://storage.googleapis.com/gce-demo-input/photo.jpg

Submitted by: Adarsh Narayan Hegde

Few Test Cases:

- No input file passed:

  Response from master:
  ```
  Error: No input received!
  ```

- Unsupported map function passed:

  Response from master:
  ```
  Error: Map Function currently not supported!
  ```

- Unsupported reduce function passed:

  Response from master:
  ```
  Error: Reduce Function currently not supported!
  ```

- No output location passed:

  Response from master:
  ```
  Error: Output location not provided!
  ```

- Correct map and Reduce function passed:

  Cloud MapReduce architecture returns desired results.

MapReduce Output:	MapReduce Output:
WORD_COUNT:	INVERTED_INDEX:
10:1	ago: document_mapper_0:1 | document_mapper_1:11 | document_mapper_2:1 |
1550:1	own: document_mapper_0:5 | document_mapper_1:1 |
16:1	rain: document_mapper_1:2 |
17:1	With: document_mapper_0:2 | document_mapper_1:2 |
1888:2	without: document_mapper_1:5 | document_mapper_2:1 |
A:59	trifling: document_mapper_1:1 |
ABOUT:1	slowly: document_mapper_0:2 | document_mapper_2:1 |
AFRICA:1	chintz: document_mapper_0:1 |
APES:3	level: document_mapper_1:1 |
ARBRE:1	preserve: document_mapper_0:1 |
ARCHER:1	shutting: document_mapper_0:1 |
AWAY:1	it: document_mapper_0:28 | document_mapper_1:41 | document_mapper_2:21 |
About:7	wiser: document_mapper_0:1 |
Across:2	Hence: document_mapper_1:1 |
Adjusting:1	subscribers: document_mapper_1:4 |
	missed: document_mapper_0:1 |
	oflife: document_mapper_2:1 |

Instance creation/start response:

Start:


```
{'id': '5192050786535929190',
 'insertTime': '2019-12-09T23:38:49.710-08:00',
 'kind': 'compute#operation',
 'name': 'operation-1575963529267-599549a574761-3a728291-a4aedbbd',
 'operationType': 'stop',
 'progress': 0,
 'selfLink': 'https://www.googleapis.com/compute/v1/projects/adarsh-hegde/zones/us-central1-f/operations/operation-1575963529267-599549a574761-3a728291-a4aedbbd',
 'startTime': '2019-12-09T23:38:49.735-08:00',
 'status': 'RUNNING',
 'targetId': '6072382914572776855',
 'targetLink': 'https://www.googleapis.com/compute/v1/projects/adarsh-hegde/zones/us-central1-f/instances/datastore-instance',
 'user': 'adarsh-narayan-hegde@adarsh-hegde.iam.gserviceaccount.com',
 'zone': 'https://www.googleapis.com/compute/v1/projects/adarsh-hegde/zones/us-central1-f'}
```

Stop:


```
{'id': '7497321422637953910',
 'insertTime': '2019-12-09T23:47:38.002-08:00',
 'kind': 'compute#operation',
 'name': 'operation-1575964057627-59954b9d56c8e-9f3601bb-5d0690de',
 'operationType': 'stop',
 'progress': 0,
 'selfLink': 'https://www.googleapis.com/compute/v1/projects/adarsh-hegde/zones/us-central1-f/operations/operation-1575964057627-59954b9d56c8e-9f3601bb-5d0690de',
 'startTime': '2019-12-09T23:47:38.020-08:00',
 'status': 'RUNNING',
 'targetId': '9110733394983187301',
 'targetLink': 'https://www.googleapis.com/compute/v1/projects/adarsh-hegde/zones/us-central1-f/instances/datastore-instance',
 'user': 'adarsh-narayan-hegde@adarsh-hegde.iam.gserviceaccount.com',
 'zone': 'https://www.googleapis.com/compute/v1/projects/adarsh-hegde/zones/us-central1-f'}
Waiting for operation to finish...
done.
```

Cost of running experiments:

- Cost of running these experiments are very low as we are deleting all the VMs as soon as they are done.
- Initial account credit was $50. Credits remaining after thoroughly testing MapReduce architecture is around $49.15

Assumptions and needed improvements:

- Current MapReduce architecture on google cloud supports a maximum of 3 mappers and 3 Reducers. This can be further enhanced to dynamically create and support more mapper and reducer instances when needed.