Memcached-lite Deploy Memcached-lite on GCP VMs using APIs

1. Introduction

In the last assignment, we designed and implemented a simple key-value store. A server was implemented, that does the job of storing and retrieving data for multiple clients. The server stores a unique value for each key using the set command and retrieves values using the get command.

This assignment aims to deploy the lite version of the Memcached to GCP VMs. After deploying our version of Memcached, we compare it with google's key value store to gauge the performance.

2. Design Details

2.1. Basic Server

The Memcached-lite server was implemented as a TCP-socket server listening on port 9889 for client connections. The server utilizes file system, in my case a JSON file ('cache.json') for key-value storage. This is stored in the VM's instance's boot storage in this case. It maintains a dictionary/map data structure for efficient key-value management and ensure data persistence even after the server process dies.

The server supports the following commands:

1. set <key> <value> \r\n : Stores a key-value pair on the server.
2. get <key> \r\n : Retrieves the value associated with a given key.

File Structure: I use JSON file structure for this task, since it is a lightweight and straightforward format for key-value data.

2.1.1. Set

The set command consists of two lines:

set <key> <value-size-bytes> \r\n

<value> \r\n

The server responds with either "STORED\r\n" if the data is successfully stored or "NOT-STORED\r\n" if there is an issue.

The reader of data (either server or client) will always know, from a preceding text line, the exact length of the data block being transmitted. (<value-size-bytes >) [1]

Text lines are always terminated by \r\n. Although unstructured data is also terminated by \r\n, but \r, \n or any other 8-bit characters may also appear inside the data. That's the reason why when a client retrieves data from a server, it must use the length of the data block (which it will be provided with (value-size-bytes >) to determine where the data block ends, and not the fact that \r\n follows the end of the data block, even though it does.[1]

The server first takes a command line and expects next data block of the size it got in the command line.

Note: since we were not asked to implement “add” or “replace” commands, “NOT_STORED\r\n” doesn’t mean that the condition for add or replace wasn’t met as in Memcached, but in this scenario, is returned because of an error like wrong command/ invalid (<value-size-bytes >) or absence of <value-size-bytes > in the command.

2.1.2. Get

The get command retrieves data as follows:

VALUE <key> <bytes> \r\n

<data block>\r\n

END\r\n

The server sends "END\r\n" when all items have been transmitted.

After this command, the client expects an item, each of which is received as text line followed by a data block of the size <bytes>. After all the items have been received, the server ends with the string – ‘END\r\n’ to indicate end of response.

In our case since, the server stores unique values for each key, every key will have one unique item.

The server returned ‘INVALID’ message in the case of error.

2.2. Clients and Test Cases

Client programs were implemented accordingly to send commands to the server and receiving and displaying the messages from the server. Client was connected to the server and executed a series of get and set requests for testing purposes.

Following test cases were run:

1. Client: `set Shivani 5`

   `Rahul`

   `Server output: STORED\r\n`

2. Client: `get Shivani`

   Server output:

   `Rahul`

   `END\r\n`

3. Client: `set Rahul 4`

   `Shivani`

Server output:

```
STORED\r\n
```

NOTE: here, the size given was 4 but the client provided with 7 characters. The server will only receive 4 characters, and thus as a result, the rest 3 letter will not be stored.

4. Client: `get Rahul`

   Server output:

```
Shiv

END\r\n
```

5. Client: `set Rahul shivani`

   Server output:

```
NOT_STORED\r\n
```

NOTE: This happened before, the command was invalid, server was expecting a value which was not provided.

Testing concurrency is shown in the next section

2.3 Google's Key-value Store

I used Google Cloud Platform's bucket to store our key-value object and compared the performance with respect to storing key-value object in VM's Boot Storage.

3. Deployment on GCP

Step1: Create a project on GCP: Shivani-pal-Fall2023

Step2: Downloaded and installed Google cloud SDK

Step3: Run the following commands:

```
gcloud init

gcloud components update

gcloud projects list   #This lists all the projects

gcloud compute networks create psnetwork --subnet-mode=auto
gcloud compute firewall-rules create psnetwork-allow-internal --network
psnetwork --allow tcp,udp,icmp --source-ranges 10.128.0.0/9
gcloud compute firewall-rules create psnetwork-allow-external --network
psnetwork --allow tcp:22,tcp:3389,icmp


gcloud compute instances create instance1

gcloud compute instances list     #list the created instances
```

```
gcloud compute instances start instance1

gcloud compute ssh instance1 # SSH keys will be generated automatically when you try to ssh to a remote instance and the key will be automatically saved in the metadata when you use the cloud SDK.
      mkdir code

      exit

gcloud compute scp --recurse /Users/shivanipal/Documents/IUB/Fall23/ECC/Assignments/assignment1/Assignment1_ShivaniPal/server.py shivanipal@instance1:code

gcloud compute ssh instance1

      cd code

      python3 server.py

gcloud compute instances stop instance1

gcloud compute instances delete instance1
```

Similarly, we run client code in another VM instance(instance2), and then communicate with server in instance1.

In order to test with Google's key-value store, we store the key value pairs ie, cachegs.json in this case in the Google Cloud Platform's Buckets. We store this as an object in the bucket created : bucket-shivanipal

Below are the commands to create the bucket:

```
gsutil mb -c standard gs://bucket-shivanipal/
```

In order to create/read/write objects inside the bucket, we need to change then role of the principal to "Storage Legacy Bucket Owner ".

For our python code to be able to access the bucket and it's object. We make a service account and generate keys and download it to be used by the python script to modify the bucket object(key value pair)

We modify our server-gs.py to get/set key-value pair from the google cloud bucket rather than the default boot space and compare the performance.

Finally,

We delete all the storages and VM instances:

```
Delete google bucket: gsutil rb gs://bucket-shivanipal/
```

Delete instances:

```
gcloud compute instances stop instance1

gcloud compute instances delete instance1
```

```
gcloud compute instances stop instance2
gcloud compute instances delete instance2
```

4. Experimental Evaluation

4.1. Performance

4.1.1. Key-value store in Boot disk

The server was tested for performance using different numbers of concurrent clients in terms of response times. The server showed good performance up to a certain level of concurrency

For 10 concurrent clients, total execution time was: 0.04746222496032715 sec

For 20 concurrent clients, total execution time was: 0.06939125061035156 sec

For 30 concurrent clients, total execution time was: 0.08686971664428711 sec

For 40 concurrent clients, total execution time was: 0.11606574058532715 sec

For 50 concurrent clients, total execution time was: 0.1445622444152832 sec

For 60 concurrent clients, total execution time was: 0.1741471290588379 sec

For 70 concurrent clients, total execution time was: 0.2020564079284668 sec

For 80 concurrent clients, total execution time was: 0.23308134078979492 sec

For 90 concurrent clients, total execution time was: 0.2576456069946289 sec

For 100 concurrent clients, total execution time was: 0.28931260108947754 sec

The response time by the server showed good performance up to a certain level of concurrency, after which response times started to degrade.

3.1.2. Google's Key-value store in GCP Bucket

The server was tested for performance using different numbers of concurrent clients in terms of response times. The server showed good performance up to a certain level of concurrency and was slightly slower than the ket value store in boot-disk

For 10 concurrent clients, total execution time was: 0.051645755767822266 sec

For 20 concurrent clients, total execution time was: 0.07049942016601562 sec

For 30 concurrent clients, total execution time was: 0.0937802791595459 sec

For 40 concurrent clients, total execution time was: 0.12290287017822266 sec

For 50 concurrent clients, total execution time was: 0.16287517547607422 sec

For 60 concurrent clients, total execution time was: 0.18877792358398438 sec

For 70 concurrent clients, total execution time was: 0.21940088272094727 sec

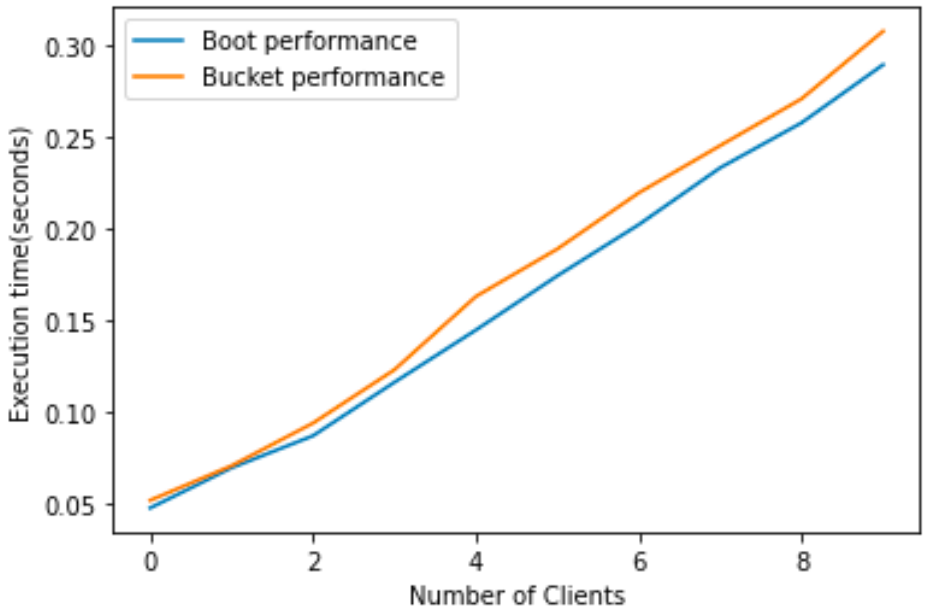
For 80 concurrent clients, total execution time was: 0.2451951503753662 sec

For 90 concurrent clients, total execution time was: 0.27074098587036133 sec

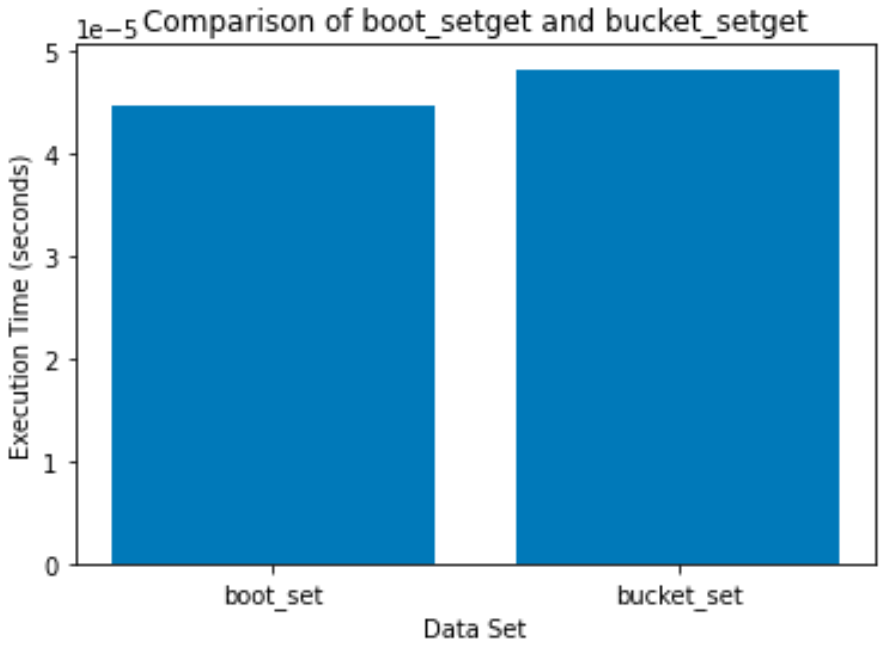
For 100 concurrent clients, total execution time was: 0.3075985908508301 sec

The response time by the server showed good performance up to a certain level of concurrency, after which response times started to degrade. And the key-value store in boot space is slighter faster than key-value stored as a bucket object

Below is the graph comparing the performance:



Below we can see execution time(COST) for a single SET command:



As we can see, initially the performance is same on both VM's boot disk and GCP's Bucket, but the performance degrades as we increase the clients.

4.2. Limitations

Concurrency limits: The server's performance starts to degrade at high levels of concurrency.

In the last assignment, I mentioned that Data consistency in a multi-threaded server like this is a limitation in this scenario since multiple threads are simultaneously reading from and writing to shared data file – cache.json. There might be situations when multiple threads may attempt to read and write to the JSON file simultaneously, leading to data corruption or unexpected behavior. I solved the issue in this assignment by implementing Threading.Lock(), way to synchronize access to shared resources in a multithreaded environment. It is used to prevent multiple threads from accessing a shared resource concurrently, ensuring that only one thread can access the resource at a time.

But there still exists the limitation that since threads may access the shared resource in any order. While using a threading.Lock helps prevent race conditions and ensures that only one thread can access the shared resource at a time, it does not inherently guarantee consistency in the order of execution. Threads can still access the shared resource in a non-deterministic order, leading to potential inconsistencies in the program's behavior.

4.3 Potential improvements

To achieve greater consistency and control over the order of execution of threads, we can consider using additional synchronization mechanisms like semaphores and condition variables to coordinate and control the flow of execution among threads more precisely.

5. References :

1. https://github.com/memcached/memcached/blob/master/doc/protocol.txt
2. https://github.com/memcached/memcached/wiki/Commands#set
3. https://pymemcache.readthedocs.io/en/latest/getting_started.html
4. https://docs.python.org/3/library/socket.html
5. https://pymotw.com/2/socket/tcp.html
6. https://docs.python.org/3/library/subprocess.html

6. Files:

- Server.py : code for server implementation (key-value stored on VMs instance)
- Server-gs.py : code for server implementation using google’s key-value store (buckets)
- Client.py: code for client implementation
- Multiple_client_test.py : code for starting multiple clients for concurrent requests to server.
- Cache.json : sample file system for storage and retrieval of key-value(will be created by server if already doesn’t exist)
- shivani-pal-fall2023.json. :Service account generated private key json
- ecc.sh :bash code for initializing gcloud, creating starting stopping and deleting VM Instances.
- Plot.ipynb : code to plot performance graphs

Screenshots:

```
(base) shivanipal@Shivanis-MacBook-Air assignment2 % gcloud init
Welcome! This command will take you through the configuration of gcloud.

Settings from your current configuration [default] are:
core:
  account: shipal@iu.edu
  disable_usage_reporting: 'False'
  project: shivani-pal-fall2023

Pick configuration to use:
 [1] Re-initialize this configuration [default] with new settings
 [2] Create a new configuration
Please enter your numeric choice:  1

Your current configuration has been set to: [default]

You can skip diagnostics next time by using the following flag:
  gcloud init --skip-diagnostics

Network diagnostic detects and fixes local network connection issues.
Checking network connection...done.
Reachability Check passed.
Network diagnostic passed (1/1 checks passed).

Choose the account you would like to use to perform operations for this
configuration:
 [1] shipal@iu.edu
 [2] Log in with a new account
Please enter your numeric choice:  1

You are logged in as: [shipal@iu.edu].

Pick cloud project to use:
 [1] abrehman516-401023
 [2] amit-singh-fall2023-400922
 [3] aniruddho-chatterjee-fall2023
 [4] dhruvil-dholariya-fall23
 [5] fa23-engr-e516-prsuman
 [6] jayaraman-sridharan-fall2023
 [7] nikhil-srirangam-fall2023
 [8] nikhil-vemula-fall2023
 [9] owen-gordon-fall2023
 [10] param-patil-fall2023
 [11] piyush-chaudhari-fall2023
 [12] pmahaja-keyvalue
 [13] prashul-kumar-fall2023
 [14] shivani-pal-fall2023
 [15] Enter a project ID
 [16] Create a new project
Please enter numeric choice or text value (must exactly match list item):  14

Your current project has been set to: [shivani-pal-fall2023].

Do you want to configure a default Compute Region and Zone? (Y/n)?  y

Which Google Compute Engine zone would you like to use as project default?
If you do not specify a zone via a command line flag while working with Compute
Engine resources, the default is assumed.
 [1] us-east1-b
 [2] us-east1-c
 [3] us-east1-d
 [4] us-east4-c
 [5] us-east4-b
```

```
(base) shivanipal@Shivanis-MacBook-Air assignment2 % gcloud projects list
PROJECT_ID                     NAME                           PROJECT_NUMBER
abrehman516-401023             abrehman516                    1060789881740
amit-singh-fall2023-400922     Amit-Singh-Fall2023            715324415902
aniruddho-chatterjee-fall2023  Aniruddho-Chatterjee-Fall2023  479686037232
dhruvil-dholariya-fall23       Dhruvil-Dholariya-Fall23       753436724012
fa23-engr-e516-prsuman         FA23-ENGR-E516-prsuman         514484346661
jayaraman-sridharan-fall2023   Jayaraman-Sridharan-fall2023   1094872199980
nikhil-srirangam-fall2023      Nikhil-Srirangam-Fall2023      962507086823
nikhil-vemula-fall2023         Nikhil-Vemula-Fall2023         307587786697
owen-gordon-fall2023           Owen-Gordon-Fall2023           657024920085
param-patil-fall2023           Param-Patil-Fall2023           1071530240737
piyush-chaudhari-fall2023      Piyush-Chaudhari-Fall2023      829646497935
pmahaja-keyvalue               Key Value Store PM             220513624170
prashul-kumar-fall2023         Prashul-Kumar-Fall2023         244557340171
shivani-pal-fall2023           Shivani-Pal-Fall2023           782794120773
(base) shivanipal@Shivanis-MacBook-Air assignment2 % gcloud components update
Beginning update. This process may take several minutes.


Your current Google Cloud CLI version is: 446.0.1
You will be upgraded to version: 449.0.0

┌─────────────────────────────────────────────────────────────────────────────┐
│                      These components will be updated.                      │
├─────────────────────────────────────────────────────┬────────────┬──────────┤
│                         Name                        │  Version   │   Size   │
├─────────────────────────────────────────────────────┼────────────┼──────────┤
│ BigQuery Command Line Tool (Platform Specific)      │     2.0.98 │  < 1 MiB │
│ Cloud Storage Command Line Tool                     │       5.26 │ 11.3 MiB │
│ Cloud Storage Command Line Tool (Platform Specific) │       5.25 │  < 1 MiB │
│ Google Cloud CLI Core Libraries                     │ 2023.10.02 │ 21.8 MiB │
│ Google Cloud CLI Core Libraries (Platform Specific) │ 2023.09.15 │  < 1 MiB │
│ Google Cloud CRC32C Hash Tool                       │      1.0.0 │  1.2 MiB │
│ anthoscli                                           │     0.2.40 │ 66.8 MiB │
└─────────────────────────────────────────────────────┴────────────┴──────────┘

The following release notes are new in this upgrade.
Please read carefully for information about new features, breaking changes,
and bugs fixed.  The latest full release notes can be viewed at:
  https://cloud.google.com/sdk/release_notes

449.0.0 (2023-10-03)
  Breaking Changes
      ▪ **(Compute Engine)** Modified gcloud beta compute future-reservations
        update to replace the paths query parameter with update_mask.

  App Engine
      ▪ Enable devappserver support for Go 1.21 runtime.

  Cloud Dataproc
      ▪ Fixed a bug that caused Instance Flexibility Policy to be disabled if
        a user only specified --secondary-worker-machine-types in dataproc
        clusters create command for the secondary workers flags.

  Cloud Functions
      ▪ Updated gcloud functions add-invoker-policy-binding and gcloud
        functions remove-invoker-policy-binding to support Cloud Functions (1st
        gen).

  Cloud Run
      ▪ Show mounted volumes in gcloud run services describe and gcloud run
        revisions describe.
```

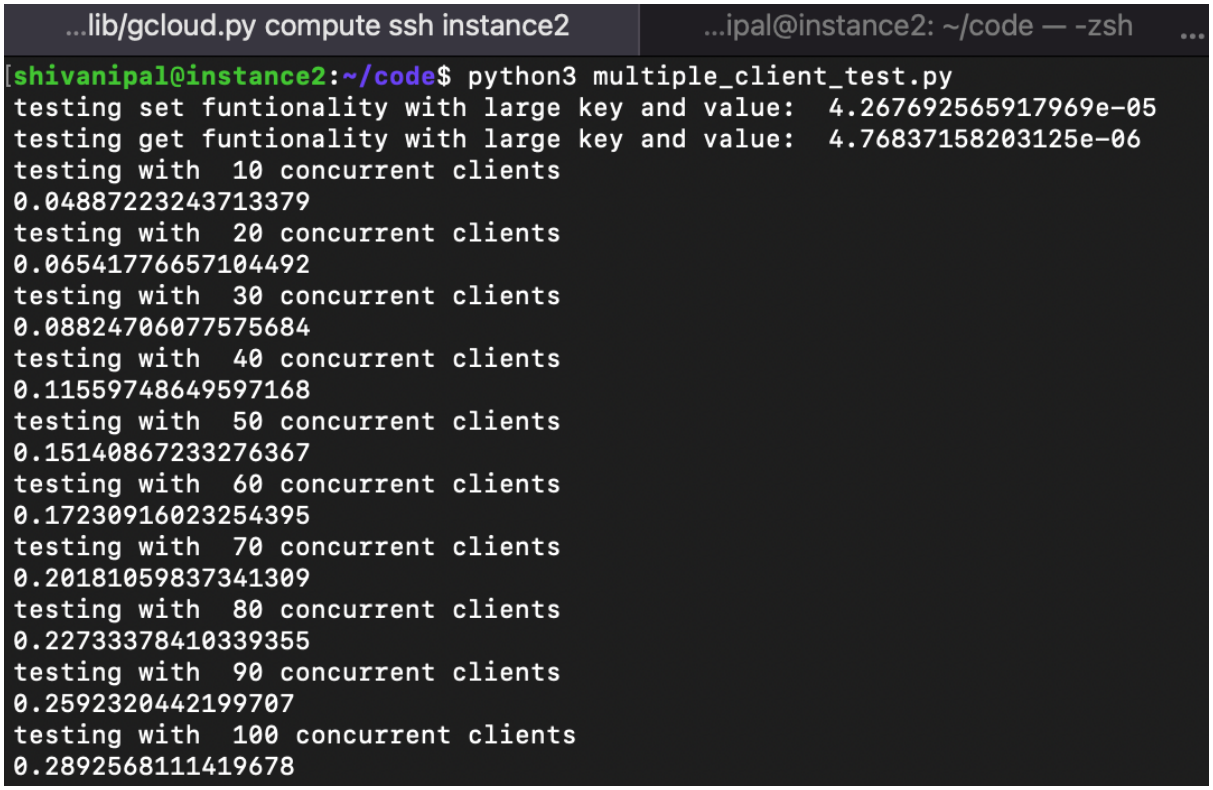
Run Server on instance1:

```
shivanipal@instance1:~/code$ python3 server-gs.py
[STARTING] Server is starting
[LISTENING] Server is listening on 10.128.0.3:9889
[ACTIVE CONNECTIONS] Active connections: 1
[DISCONNECTED] Server Disconnected
```

Run Client on instance2:

```
shivanipal@instance2:~/code$ python3 client.py
Enter: set ssss 4
gfjjd
STORED

Enter: get ssss
gfjj
END

Enter: set rahul 4
shivani
STORED

Enter: get rahul
shiv
END

Enter:
```

Run tests on instance2:


...lib/gcloud.py compute ssh instance2
...ipal@instance2: ~/code — -zsh
shivanipal@instance2:~/code$ python3 multiple_client_test.py
testing set funtionality with large key and value: 4.267692565917969e-05
testing get funtionality with large key and value: 4.76837158203125e-06
testing with 10 concurrent clients
0.04887223243713379
testing with 20 concurrent clients
0.06541776657104492
testing with 30 concurrent clients
0.08824706077575684
testing with 40 concurrent clients
0.11559748649597168
testing with 50 concurrent clients
0.15140867233276367
testing with 60 concurrent clients
0.17230916023254395
testing with 70 concurrent clients
0.20181059837341309
testing with 80 concurrent clients
0.22733378410339355
testing with 90 concurrent clients
0.2592320442199707
testing with 100 concurrent clients
0.2892568111419678