Assignment Simple KV Store Report

By: pnalawa@iu.edu

Design:

- We have two parts: **client** and **server**.
- The server can be connected with a **port by giving it via arguments**.
- The server can accept multiple client connections concurrently via **multi-threading**.
- The **max concurrent connections** set are **1024** as per the **memcache protoco**l.
- We are using **TCP** as the underlying communication between client and server with low level **socket programming**.
- Server spawns a new thread for each client connected.

Operations:

- We have implemented 3 operations:
    1. SET
    2. GET
    3. END
- **SET**
    - **Command:**
        - set <key> <size>\r\n <value>\r\n
    - If length is less not equal to 4 it will return ERROR\r\n
    - If key length is greater than 250, it will return ERROR\r\n
    - If the key is already present in the kv store, it will just update it.
    - If <size> is not equal to the <value> it will return ERROR\r\n
- **GET**
    - **Command**
        - get <key>\r\n
    - If length is less not equal to 2 it will return ERROR\r\n
    - If key does not exist it returns NOT-FOUND\r\n
    - If key exists it returns
        - VALUE <key> <size>\r\n <value>\r\n END\r\n
- **END**
    - **Command**
        - end\r\n
    - Closes client thread in server.

Limitations:

- Currently, the race condition where two writes or set operations are performed at same time is not handled. This can be taken as **future improvement** by **adding timestamp** in persistent storage.

- We are limiting the number of concurrent connections to 1024 as per memcache protocol.

Steps to reproduce:

- chmod 755 start_client.sh
- chmod 755 start_server.sh
- Start Server
    - **./start_server.sh <server_port>**
- You can run **multiple clients** by this command
    - **./start_client.sh <server_port>**