Assignment 3: Distributed Hash Table

Special Topics in Computer Science: Big Data Systems
CS-UH 3260 Spring 2023
MAX 10 points

In this assignment, you will be implementing your own in-memory distributed hash table using Memcached as a backend, with consistent hashing.

1. System installation (not graded):

- Install Memcached on your machine
- To simulate a distributed system, start multiple instances of Memcached on different ports
- You can start five instances on five different terminals (or use -d run as daemon):
    - ./memcached -p 11211
    - ./memcached -p 11212
    - ./memcached -p 11213
    - ./memcached -p 11214
    - ./memcached -p 11215

2. Description

You are provided with a starter code: dist_kv_starter.py that uses Python's Pymemcache library for read/write operations. Use this code base to implement a program that distributes keys/values on a set of Memcached instances. Since Memcached runs in memory, the system should also support replication (use a fixed factor of 2) to ensure fault tolerance. You are free to modify the code as you see fit, as long as the test procedure continues to work correctly.

Requirements:

1. Implement consistent hashing.
2. Use Murmur3 as the hash function.
3. Use a replication factor of 2 for fault tolerance.
4. Ensure that the system can easily add or remove new Memcached instances.
5. Modify the provided test procedure to verify that the program is functioning correctly.

🚨 You cannot use an existing implementation or copy code.

The starter code contains a simple test procedure.

- Create 100 items and adds them to the DHT
- Select 10 random keys and read their values from the DHT.
- Scenario 1: One of the servers drops
  ⇒ Reading the values should yield the same results
- Scenario 2: A new server is added
  ⇒ Reading the values should yield the same results

🚨 You can Modify the provided test procedure to verify that the program is functioning correctly. For example, by increasing the number of elements to insert and to read.

Good luck, and have fun coding a distributed system! 🤩

3 Deliverables and Grading

🎯 **Deliverable:** Python Code + Report with any special instructions

Description	**Score (/10)**
• Correct implementation of consistent hashing without the use of specialized libraries. • Correct implementation of replication (factor 2) • Correct implementation of stub functions to add records to the distributed DHT. • Proper handling of any relevant metadata required for the DHT • Automate the testing function to decide whether the test passes or not.	7
• Proper error handling	1
• The test code passes successfully, indicating that the system is elastic and failure tolerant (to the extent of the simulation)	1
• Documentation: code comments + 1 short report	1
• Bonus point 🙌: With a bit more effort you can create an interactive shell, which should allow users to issue put/get keys, and add/remove nodes.*	1

** Bonus point is used to pad your score up to the maximum score, but the total score of this assignment cannot exceed 10 points.*