Report

This code is an implementation of a distributed hash table (DHT) using a consistent hash ring. The DHT is distributed across multiple servers, and the hash ring is used to map the keys to the servers in the DHT. The hash ring is consistent, meaning that when nodes are added or removed from the ring, as few keys as possible are moved to new servers.

The code starts by importing the necessary packages, such as random, faker, and pymemcache. The insert_sorted function is defined to find the position in a sorted array where an element should be inserted. It is used to determine in which server a key hashed on the ring will be added. Then, the Node class is defined to represent a memcached node. The class has two attributes, the IP and port of the node and a connection to memcached.

Next, the ConsistentHashRing class is defined, which represents the hash ring. The class has a replication factor attribute, which determines the number of virtual nodes per server and the number of servers on which the data is replicated. The ring is implemented as a list of hashed keys, and each key maps to a node in the DHT. The nodes attribute is a dictionary that maps the hashed keys to the nodes. The add_node method is used to add a new node to the ring with virtual nodes, and the remove_node method is used to remove a node from the ring with its virtual nodes. The get_hashed_key method generates a hashed key using the MurmurHash algorithm, and the get_node method returns the node responsible for a given key.

Finally, the DistributedHashTable class is defined, which implements the DHT using the ConsistentHashRing class. The class has three attributes, the hash ring, a dictionary that maps the string representation of a node to a Node object, and a dictionary that maps the string representation of a node to a list of keys. The find_next_two method is used to return the next two unique nodes in the hash ring given a current node. The replicate method is used to replicate the keys across the hash ring to ensure fault tolerance. The add_node method is used to add a new node to the DHT, and it is responsible for replicating the keys to the new node and reassigning the keys to the new node. The remove_node method is used to remove the given server and reassign the keys stored on that server to new servers.

An interactive shell has also been implemented at the end to allow the users to issue put/get keys, and add/remove nodes.

Instructions to run: python3 dht_starter.py