Mohammed Arham | maj596 | Assignment 3 report

Before running the code run 5 instances of Memcached by writing ***memcached -p 11215*** for example in terminal.

The code defines a list of Memcached instances that will be used as nodes in the DHT. Each instance is represented by a dictionary containing the name, IP address, and port number of the instance.

The script then creates a Memcached client for each instance using a helper function called create_client(). The function takes an instance dictionary as an argument and attempts to create a client for that instance. If the connection fails, the function logs an error and returns None. The list of clients is then stored in a variable called MEMCACHED_CLIENTS.

The code also defines a helper function called get_primary_and_secondary() that takes a key as an argument and returns the primary and secondary nodes responsible for that key. The function uses the murmur3_32 hash function to calculate a hash value for the key and then finds the two Memcached clients whose instance hashes are closest to the key hash.

The code defines two functions for adding and removing Memcached instances from the DHT. The add_node() function takes the name, IP address, and port number of a new instance and adds it to the list of instances, hashes, and clients. The remove_node() function takes the name of an instance and removes it from the list of instances, hashes, and clients.

The code defines two functions for reading and writing key-value pairs to the DHT. The dht_get() function takes a key as an argument, calls get_primary_and_secondary() to find the primary and secondary nodes responsible for that key, and then attempts to read the value for that key from the primary node. If the read fails, the function attempts to read the value from the secondary node. If the value is still not found, the function returns None.

The dht_set() function takes a key-value pair as arguments, calls get_primary_and_secondary() to find the primary and secondary nodes responsible for that key, and then writes the key-value pair to both nodes.

Get_primary_and_secondary : this function should ensure consistent hashing as it follows the consistent hashing approach where it maps the keys and instances to a hash ring, and each key is mapped to a point on the ring based on its hash value. Then, the instance responsible for serving the key is the first instance found in a clockwise direction starting from the point where the key is mapped. In this case, the function iterates through the instance hash values in increasing order until it finds the first instance hash value that is greater than or equal to the key's hash value. This instance becomes the primary instance for the key, and the next instance in a clockwise direction becomes the secondary instance.

The code implements a replication factor of 2 by maintaining two Memcached client instances for each key in the DHT, one primary and one secondary. If the value for a key is not found in the primary, the secondary is checked next, and if the value is not found in the secondary as well, it is assumed to be not present in the DHT. This ensures that the data is replicated across two nodes, providing fault-tolerance and redundancy.