Consistent Hashing Distributed Hash Table (DHT)

Sashank Silwal

Assignment 3

This code implements a distributed hash table using a consistent hashing algorithm. It allows users to store and retrieve key-value pairs across multiple nodes in a cluster.

Class

The `Node` class represents a node in the distributed hash table. It has a name, IP address, and port.

The `ConsistentHashRing` class is the core of the Consistent Hashing algorithm. It takes a list of nodes and a number of replicas as input and creates a hash ring, where each node is assigned a number of virtual nodes based on the number of replicas. It provides methods to add and remove nodes from the ring and to get the node responsible for a given key.

Functions

The `dht_get` and `dht_set` functions are used to get and set values in the distributed hash table.

The `add_node` and `remove_node` functions are used to add and remove nodes from the hash ring, respectively. When a node is added or removed, the data is redistributed across the other nodes in the ring.

Usage

The code creates `Node` objects, which represent individual nodes in the distributed system. Each `Node` has a name, IP address, and port number, and is responsible for storing a subset of the key-value pairs in the hash table.

You can create a new `Node` object as follows:

```
node = Node(name='my_node', ip='127.0.0.1', port=8000)
```

The node is added to the hash ring, by creating a `ConsistentHashRing` object and call its `add` method:

```
ring = ConsistentHashRing(nodes=[node])
```

The `dht_set` function is used to store key-value pairs in the hash table, and the `dht_get` function to retrieve them. The `dht_set` function uses a replication factor of 2, meaning that it stores each key-value pair on two nodes in the hash ring to ensure redundancy.

```
dht_set(key='my_key', value='my_value') result = dht_get(key='my_key')
```

Addition or removal of nodes from the hash ring is done using the `add_node` and `remove_node` functions. When you add a node, the existing data in the hash table is redistributed across the new node and its neighbors in the ring to ensure that the load is balanced. When you remove a node, its data is redistributed to its neighbors before the node is removed from the ring.

```
add_node(name='new_node', ip='127.0.0.1', port=8001) remove_node(name='my_node')
```

Make Sure of the following:

- When adding a new node, make use the IP address and port number is established otherwise will result in an error
- The `add_node` and `remove_node` functions assume that the hash ring is initialized with at least one node. If the hash ring is empty, these functions will give an error

message: No nodes in the ring