

Assignment 2: Zookeeper

Project Description: This project is based on the project code provide in class. The `zk_election_another.py` program and `docker-compose.yml` file were extended to meet the assignment requirements. A replicated key-value data store maintained by N servers was implemented and each server maintains a copy of the data store. Zookeeper was used in order to implement leader election. All requests are routed through the leader and propagated to the replicas.

Project Implementation:

(Left) Zookeeper.py with read and add_update methods implemented and (Right) the main method for server.py that handles all test cases.

```
# Return the value for the key in the dictionary, otherwise return empty string
def read(self, key):
    return self.data_store.get(key, "")

# Add or update a key-value pair
def add_update(self, key, value):
    try:
        # Check if path exists before gettings children and detecting leader
        if self.zk.exists(self.leadernode):
            childrens = self.zk.get_children(self.leadernode)

            # If leader, update key-value pair and propagate to replicas
            if self.detectLeader(childrens):
                self.data_store[key] = value
                self.propagate_update(self.data_store)
            else:
                print(f"\033[32mOnly leader can add/update key-value pairs. Sending request to leader.\033[0m")
                self.host_seq_list = [i.split(".") for i in childrens]
                sorted_host_seqvalue = sorted(self.host_seq_list, key=operator.itemgetter(1))
                leader = sorted_host_seqvalue[0][0]
                print(f"LEADER IS: {leader}")
                url = f"http://{leader}/update"
                response = requests.post(url, json={"key": key, "value": value})
            else:
                print(f"\033[31mPath {self.leadernode} does not exist.\033[0m")

    except Exception as e:
        print(f"\033[31mError in add_update: {e}\033[0m")

def main():
    zookeeper_ip = "127.0.0.1"
    zookeeper_port = "21811"
    host = "127.0.0.1"
    ports = ["5000", "5001", "5002"]

    try:
        start_docker()

        servers = [start_server(host, port, zookeeper_ip, zookeeper_port) for port in ports]

        print("\033[32mTesting Add and Read...\033[0m")
        # All updates routed through server on the elected leader. Sending through port 500 as default.
        add_update(host, ports[0], f"key0", f"value0")

        for i in range(3):
            for j in range(3):
                print(f"\033[36mFor Port: {ports[i]}\033[0m")
                read_key(host, ports[i], f"key{j}") # Check existing keys on all ports

        print("\033[32mTesting Leader Election...\033[0m")
        for port, server in servers:
            response = call(host, port)
            is_leader = response.get("is_leader", False)
            if is_leader:
                print(f"\033[32mElected the leader, electing a new one.\033[0m")
                stop_server(server)
                time.sleep(30) # Wait for new leader election
                start_server(host, port, zookeeper_ip, zookeeper_port)

        print("\033[32mTesting Stale Read...\033[0m")
        for i in range(3):
            for j in range(3):
                print(f"\033[36mFor Port: {ports[i]}\033[0m")
                read_key(host, ports[i], f"key{j}") # Check existing keys on all ports, old leader should be empty
            time.sleep(10)

        for i in range(3):
            add_update(host, ports[i], f"key{i}", f"value{i}") # All updates routed through server on some elected leader, update all
        for i in range(3):
            for j in range(3):
                print(f"\033[36mFor Port: {ports[i]}\033[0m")
                read_key(host, ports[i], f"key{j}") # Check existing keys on all ports

        # Handle an exception, so many exceptions... :/
        except Exception as e:
            print(f"Exception: {e}")

    finally:
        for _server in servers:
            stop_server(_server)
        stop_docker()
```

Project Testing:

Test 1:

(a.) *Three servers started in docker containers.*

```
Composing Docker Environment...
[+] Running 5/5
✓Network zookeeper_default Created
✓Container zk3 Started
✓Container zoonavigator Started
✓Container zk1 Started
✓Container zk2 Started
```

(b.) *The servers elect a leader and all requests are routed to the leader.*

```
Testing Add and Read...
Childrens: ['127.0.0.1:5002_0000000002', '127.0.0.1:5001_0000000001', '127.0.0.1:5000_0000000000']
sorted_host_seqvalue: [['127.0.0.1:5000', '0000000000'], ['127.0.0.1:5001', '0000000001'], ['127.0.0.1:5002', '0000000002']]
I am current leader: 127.0.0.1:5000
Replicas: ['127.0.0.1:5001', '127.0.0.1:5002']
Propagating...
127.0.0.1 - - [09/Mar/2025 22:29:07] "POST /propagate HTTP/1.1" 200 -
Successfully propagated update to 127.0.0.1:5001
Propagating...
127.0.0.1 - - [09/Mar/2025 22:29:07] "POST /propagate HTTP/1.1" 200 -
Successfully propagated update to 127.0.0.1:5002
127.0.0.1 - - [09/Mar/2025 22:29:07] "POST /update HTTP/1.1" 200 -
Add/Update response: {'status': 'updated'}
```

(c.) *Subsequent requests of the key can be fetched from any server.*

```
For Port: 5000
127.0.0.1 - - [09/Mar/2025 22:29:07] "GET /read?key=key0 HTTP/1.1" 200 -
Read response: {'key': 'key0', 'value': 'value0'}
For Port: 5000
127.0.0.1 - - [09/Mar/2025 22:29:07] "GET /read?key=key1 HTTP/1.1" 200 -
Read response: {'key': 'key1', 'value': ''}
For Port: 5000
127.0.0.1 - - [09/Mar/2025 22:29:07] "GET /read?key=key2 HTTP/1.1" 200 -
Read response: {'key': 'key2', 'value': ''}
For Port: 5001
127.0.0.1 - - [09/Mar/2025 22:29:07] "GET /read?key=key0 HTTP/1.1" 200 -
Read response: {'key': 'key0', 'value': 'value0'}
For Port: 5001
127.0.0.1 - - [09/Mar/2025 22:29:07] "GET /read?key=key1 HTTP/1.1" 200 -
Read response: {'key': 'key1', 'value': ''}
For Port: 5001
127.0.0.1 - - [09/Mar/2025 22:29:07] "GET /read?key=key2 HTTP/1.1" 200 -
Read response: {'key': 'key2', 'value': ''}
For Port: 5002
127.0.0.1 - - [09/Mar/2025 22:29:07] "GET /read?key=key0 HTTP/1.1" 200 -
Read response: {'key': 'key0', 'value': 'value0'}
For Port: 5002
127.0.0.1 - - [09/Mar/2025 22:29:07] "GET /read?key=key1 HTTP/1.1" 200 -
Read response: {'key': 'key1', 'value': ''}
For Port: 5002
127.0.0.1 - - [09/Mar/2025 22:29:07] "GET /read?key=key2 HTTP/1.1" 200 -
Read response: {'key': 'key2', 'value': ''}
```

Test 2:

(a.) Kill the leader.

```
Testing Leader Election...
I am the current leader to kill: 127.0.0.1:5000
127.0.0.1 - - [09/Mar/2025 22:29:07] "GET /kill HTTP/1.1" 200 -
Kill response: {'is_leader': True}
Killing the leader, electing a new one.
Stopping Server...
```

(b.) A new leader is elected.

```
LEADER IS: 127.0.0.1:5001
Childrens: ['127.0.0.1:5002_0000000002', '127.0.0.1:5000_0000000003', '127.0.0.1:5001_0000000001']
sorted_host_seqvalue: [['127.0.0.1:5001', '0000000001'], ['127.0.0.1:5002', '0000000002'], ['127.0.0.1:5000', '0000000003']]
I am current leader: 127.0.0.1:5001
Replicas: ['127.0.0.1:5002', '127.0.0.1:5000']
```

(c.) Subsequent requests are routed through the new leader.

```
For Port: 5001
127.0.0.1 - - [09/Mar/2025 22:29:47] "GET /read?key=key0 HTTP/1.1" 200 -
Read response: {'key': 'key0', 'value': 'value0'}
For Port: 5001
127.0.0.1 - - [09/Mar/2025 22:29:47] "GET /read?key=key1 HTTP/1.1" 200 -
Read response: {'key': 'key1', 'value': ''}
For Port: 5001
127.0.0.1 - - [09/Mar/2025 22:29:47] "GET /read?key=key2 HTTP/1.1" 200 -
Read response: {'key': 'key2', 'value': ''}
For Port: 5002
127.0.0.1 - - [09/Mar/2025 22:29:47] "GET /read?key=key0 HTTP/1.1" 200 -
Read response: {'key': 'key0', 'value': 'value0'}
For Port: 5002
127.0.0.1 - - [09/Mar/2025 22:29:47] "GET /read?key=key1 HTTP/1.1" 200 -
Read response: {'key': 'key1', 'value': ''}
For Port: 5002
127.0.0.1 - - [09/Mar/2025 22:29:47] "GET /read?key=key2 HTTP/1.1" 200 -
Read response: {'key': 'key2', 'value': ''}
```

Test 3:

(a.) *The killed leader is back online and may have stale data.*

```
Testing Stale Read...
For Port: 5000
127.0.0.1 - - [09/Mar/2025 22:29:47] "GET /read?key=key0 HTTP/1.1" 200 -
Read response: {'key': 'key0', 'value': ''}
For Port: 5000
127.0.0.1 - - [09/Mar/2025 22:29:47] "GET /read?key=key1 HTTP/1.1" 200 -
Read response: {'key': 'key1', 'value': ''}
For Port: 5000
127.0.0.1 - - [09/Mar/2025 22:29:47] "GET /read?key=key2 HTTP/1.1" 200 -
Read response: {'key': 'key2', 'value': ''}
For Port: 5001
127.0.0.1 - - [09/Mar/2025 22:29:47] "GET /read?key=key0 HTTP/1.1" 200 -
Read response: {'key': 'key0', 'value': 'value0'}
For Port: 5001
127.0.0.1 - - [09/Mar/2025 22:29:47] "GET /read?key=key1 HTTP/1.1" 200 -
Read response: {'key': 'key1', 'value': ''}
For Port: 5001
127.0.0.1 - - [09/Mar/2025 22:29:47] "GET /read?key=key2 HTTP/1.1" 200 -
Read response: {'key': 'key2', 'value': ''}
For Port: 5002
127.0.0.1 - - [09/Mar/2025 22:29:47] "GET /read?key=key0 HTTP/1.1" 200 -
Read response: {'key': 'key0', 'value': 'value0'}
For Port: 5002
127.0.0.1 - - [09/Mar/2025 22:29:47] "GET /read?key=key1 HTTP/1.1" 200 -
Read response: {'key': 'key1', 'value': ''}
For Port: 5002
127.0.0.1 - - [09/Mar/2025 22:29:47] "GET /read?key=key2 HTTP/1.1" 200 -
Read response: {'key': 'key2', 'value': ''}
```

(b.) *Once key is updated, output. In this example, I update all key/value pairs.*

```
Add/Update response: {'status': 'updated'}
For Port: 5000
127.0.0.1 - - [09/Mar/2025 22:29:57] "GET /read?key=key0 HTTP/1.1" 200 -
Read response: {'key': 'key0', 'value': 'value0'}
For Port: 5000
127.0.0.1 - - [09/Mar/2025 22:29:57] "GET /read?key=key1 HTTP/1.1" 200 -
Read response: {'key': 'key1', 'value': 'value1'}
For Port: 5000
127.0.0.1 - - [09/Mar/2025 22:29:57] "GET /read?key=key2 HTTP/1.1" 200 -
Read response: {'key': 'key2', 'value': 'value2'}
For Port: 5001
127.0.0.1 - - [09/Mar/2025 22:29:57] "GET /read?key=key0 HTTP/1.1" 200 -
Read response: {'key': 'key0', 'value': 'value0'}
For Port: 5001
127.0.0.1 - - [09/Mar/2025 22:29:57] "GET /read?key=key1 HTTP/1.1" 200 -
Read response: {'key': 'key1', 'value': 'value1'}
For Port: 5001
127.0.0.1 - - [09/Mar/2025 22:29:57] "GET /read?key=key2 HTTP/1.1" 200 -
Read response: {'key': 'key2', 'value': 'value2'}
For Port: 5002
127.0.0.1 - - [09/Mar/2025 22:29:57] "GET /read?key=key0 HTTP/1.1" 200 -
Read response: {'key': 'key0', 'value': 'value0'}
For Port: 5002
127.0.0.1 - - [09/Mar/2025 22:29:57] "GET /read?key=key1 HTTP/1.1" 200 -
Read response: {'key': 'key1', 'value': 'value1'}
For Port: 5002
127.0.0.1 - - [09/Mar/2025 22:29:57] "GET /read?key=key2 HTTP/1.1" 200 -
Read response: {'key': 'key2', 'value': 'value2'}
Stopping Server...
Stopping Server...
Stopping Server...
Stopping Docker Environment...
```