Assignment 2

Description

You will implement a replicated key-value data store maintained by **N** servers. Each server will maintain a copy of the data store and expose two functions

`Read(key):` will read the value associated with the key.
`Add_Update(key, value):` will add/update the value associated with the key.

A client may contact any of the server to read/add/update the data store. All **`Add/Update(key, value)`** requests will be routed to the **leader server**, who will be responsible for any write requests to the key-value. The leader will also be responsible for propagating the updates to other replicas. You will use **ZooKeeper** (a coordination service) to elect the leader (see more information below on how to use ZooKeeper for leader lection).

All **`Read(key)`** request will be served as follows. If the key exists in the local data store, it will send the value even if it is "stale". Else, it will return empty.

Implementation

You will create a server program (e.g., server.py) that spawns the server process and connects to the ZooKeeper service.

```
python server.py –host <hostip> –port <xxxx> -zookeeper <zkip> -zookeeper_port
<zkport>
```

This will spawn a server that will connect to the zookeeper service, create a `/election/` znode and register itself in the `/election/` znode. Servers under the `/election/` znode will participate in the election, and a server will be identified as a leader.
Use Znode type SEQUENTIAL, so when the server registers under /election/ it is assigned a value "server_<id>". e.g., `/election/server_000000001`.

For leader election, query all the children nodes under election and select the znode with the smallest sequence number.

You may use REST protocols or RPCs for implementing the add/update and read functions.

Testing

You will provide a driver-test program that will spawn the servers and connect to the zookeeper service. The test driver program will also be responsible for sending / receiving key value store updates to one of the servers. Use the Docker Zookeeper service script to spawn zookeeper servers and test the following scenarios:

- Add and Read: Start 3 servers. The servers elect a leader and all add requests are routed to the leader. Subsequent Read requests of the key can be fetched from any of the server.

- Leader election: Same as above but kill the leader node. A new leader is elected and all subsequent requests is routed to the new leader.
- Stale Read: The killed leader is back online but is not the leader and may have stale data (if main data store was updated). Since the key value is in memory, it will return empty until the key was updated. Once key value is updated, the server will output the updated value.

Please put appropriate print statements.

Extra Credit (10%)

Start the Zookeeper service on Google's cloud platform and compare the leader election latency with a local zookeeper service. Report the latency over multiple runs.

Submission

Please upload your code and your report on Gradescope. Your submission should contain all the code including the test cases and log files of your execution.

Grading Criteria

Component	**%**
Implementation	60
Testing	30
Code documentation	10
Extra credit	10

Zookeeper Documentation

Leader selection via Zookeeper

ZooKeeper is a distributed, open-source coordination service for distributed applications. It exposes a simple set of primitives that distributed applications can build upon to implement higher level services for synchronization, configuration maintenance, and groups and naming. It is designed to be easy to program to, and uses a data model styled after the familiar directory tree structure of file systems.

Installation:

We are going to use docker to install Zookeeper.

Go get docker on: https://docs.docker.com/desktop/

Once the docker is successfully installed, use your cmd/terminal to check via command:

$ docker --version

```
(base) Yues-MacBook-Pro-2:CS2510 yuedai$ docker --version
Docker version 19.03.12, build 48a66213fe
```

Get Zookeeper from docker on: https://hub.docker.com/_/zookeeper

Then we can use docker to get ZooKeeper Service setup on our machine via command:

$ docker pull zookeeper

```
(base) Yues-MacBook-Pro-2:CS2510 yuedai$ docker pull zookeeper
Using default tag: latest
latest: Pulling from library/zookeeper
bf5952930446: Pull complete
092c9b8e633f: Pull complete
0b793152b850: Pull complete
b612fb485c1a: Pull complete
667cec6570bc: Pull complete
db0e227deeb8: Pull complete
2ecd18ebbb33: Pull complete
5ebd9b1202ae: Pull complete
Digest: sha256:c0c7b8774beaab2d4b06e004fda286526503ad991b6b5677e4edd431f22e135d
Status: Downloaded newer image for zookeeper:latest
docker.io/library/zookeeper:latest
```

Check your installation with

$ docker image ls

```
(base) Yues-MacBook-Pro-2:CS2510 yuedai$ docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
docker101tutorial   latest              d9628c37727e        22 minutes ago      27.3MB
<none>              <none>              4665844c438e        22 minutes ago      85.5MB
<none>              <none>              8a9605cdb1e4        22 minutes ago      224MB
<none>              <none>              7b489d100901        22 minutes ago      72MB
nginx               alpine              6f715d38cfe0        3 weeks ago         22.1MB
python              alpine              44fceb565b2a        4 weeks ago         42.7MB
zookeeper           latest              6ad6cb039dfa        4 weeks ago         252MB
node                12-alpine           18f4bc975732        5 weeks ago         89.3MB
(base) Yues-MacBook-Pro-2:CS2510 yuedai$
```

To lunch a Zookeeper server,

```
$ docker run --name my-zookeeper --restart always -d zookeeper
```

Where the –name sets the name of the container you started and -d sets the image the container utilized. You can check running containers with

```
$ docker container ls
```

```
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                              NAMES
fa84c3370b14        zookeeper           "/docker-entrypoint.…"   5 seconds ago       Up 4 seconds        2181/tcp, 2888/tcp, 3888/tcp, 8080/tcp   my-zookeeper
```

By default, your zookeeper will use following ports

Client port::2181
Follower port::2888
Election port::3888
AdminServer port::8080

When a server is on, you can connect via commands,

```
$ docker run -it --rm --link my-zookeeper:zookeeper zookeeper zkCli.sh -server zookeeper
```

Where use your container's name to replace my-zookeeper

Programming in ZooKeeper

Online resources: https://www.tutorialspoint.com/zookeeper/index.htm)

Example Python implementation : KAZOO

If you are using python. Kazoo is a good choice as an API for zookeeper.
Documentation: https://kazoo.readthedocs.io/en/latest/basic_usage.html

Usage:

```
from kazoo.client import KazooClient
zk = KazooClient(hosts='localhost:2181')
zk.start()
```

It supports all UI required for leader election such as *zk.create(), zk.set(), zk.delete()*. Etc.
You can check znode under current znode with: *zk.get_children(dir)*. The returned object will be a list of children's names.

Docker script to run your Zookeeper server on replicate mode

To build a distributed file system we need **at least 3 zookeeper servers**, this requires us to run 3 zookeeper servers as a service in docker. To do that:

1. Initialize your docker to swarm mode to simulate distributed environment: `docker swarm init`
2. Use docker stack deploy to launch 3 servers (in following example we use *zookeeper* as our service name, you can replace that with any name you what when practicing):

   a. Use a config file which specifies server configs in a *.yml file (such as "zookeeper.yml"): an example on Appendix
   b. Then you can deploy stack via: docker stack deploy --compose-file zookeeper.yml *zookeeper*
   c. You can check the running service by: docker stack services *zookeeper*
3. To gracefully exit (in following example we use *zookeeper* as our service name, you can replace that with any name you what when practicing):
   a. Bring stack down via: docker stack rm *zookeeper*
   b. (optional)Bring service down via: docker service rm *zookeeper*
   c. Quit swam mode by: docker swarm leave –force

To run your leader election, first make sure you have a znode created for such purpose (such as "/leader", "/election")

Appendix

```
version: '3.1'

services:
  zoo1:
    image: zookeeper
    restart: always
    hostname: zoo1
    ports:
      - 2181:2181
    environment:
      ZOO_MY_ID: 1
      ZOO_SERVERS: server.1=0.0.0.0:2888:3888;2181
server.2=zoo2:2888:3888;2181 server.3=zoo3:2888:3888;2181

  zoo2:
    image: zookeeper
    restart: always
    hostname: zoo2
    ports:
      - 2182:2181
    environment:
      ZOO_MY_ID: 2
      ZOO_SERVERS: server.1=zoo1:2888:3888;2181
server.2=0.0.0.0:2888:3888;2181 server.3=zoo3:2888:3888;2181

  zoo3:
    image: zookeeper
    restart: always
    hostname: zoo3
    ports:
      - 2183:2181
    environment:
      ZOO_MY_ID: 3
      ZOO_SERVERS: server.1=zoo1:2888:3888;2181 server.2=zoo2:2888:3888;2181
server.3=0.0.0.0:2888:3888;2181
```