Assignment 2: Clocks, Vector Clocks and Everything

Spring 2022

Problem

You will implement a replicated key-value data store maintained by **N** servers. Each server will maintain a copy of the data store, a vector clock for each entry in the data store and expose two functions

`Read(key):` will read the value associated with the key and the vector clock value. If there is a conflict, it will return all conflicted values and their corresponding value.
`Add_Update(key, value):` will add/update the value associated with the key and return the vector clock value to the client.

A client may contact any servers to read/add/update the data store. When the server receives a new value, it updates the local clock for that entry and propagates it to other servers. In addition, the servers will check for conflict by comparing the local vector clock with the received vector.

Example: N=3

Local clock	Received clock	
[3, 1, 3]	[1, 0, 3]	No conflict
[1, 0, 3]	[1, 1, 1]	Conflict

If there is no conflict, the value in the data store and the local vector clock are updated based on the highest vector clock. If there is a conflict, the client receives all the conflicting values.

Implementation

You will need a server program (e.g., server.py) that spawns the server process.
For example,

```
python server.py -host <hostip> -port <xxxx> -N <numberofreplicas> -hosts
[hostsipaddres] -ports [ports array]

python server.py -host 127.0.0.0 -port 7000 -N 3 -hosts 130.2.xx.xx,192.xx.xx.xx
-ports 7001,7002
```

The server will use the *hosts* and *ports* parameter to communicate with other replicas.

You will also implement a client program (e.g., client.py) that can contact any server replica to read/write/update the data store.

You may use any programming language and any communication abstraction (e.g., sockets, RPC, REST) that might be needed to establish communication.

Testing

You will provide a driver test program (e.g., driver.py) that will spawn the servers and client to test the following scenarios.

Deploy at least 5 servers. A server can become offline anytime. If a server is offline, the server will have outdated values when a client makes an add/update request to other servers. Simulate the scenario when (a) there is no conflict and (b) there is conflict.

When there is no conflict, the sever value should merge the key values and local clock based on the clock values (local versus remote). This can be done by the client sending an update request twice --- when the server is down, and the failed server is back up.

When there is a conflict, the server should store conflicted values and return these to the client. To simulate conflicts, you can make clients perform simultaneous updates to different replicas.

Please put appropriate print statements.

Extra Credit

You may run the service on Google's cloud platform and run these servers in different parts of the world. Report the latency for client operations and any insights you may have from running servers in different parts of the world.

Submission

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

Grading Criteria

Component	**%**
Implementation	65
Testing	30
Code documentation	5
Extra credit	10