CSC409 Assignment 1: URL Shortner

Members: Yakob Asihk, Jasmeet Brar, Oluwaseun Cardoso

Architecture:

For our architecture we used 4 servers, one as a proxy and 3 as URL Shortners. The proxy receives PUT/GET requests, hashes them using MD5 hasher and distributes the requests to the 3 servers. The servers obtain the request and either do a PUT into their respective database.txt or they do a GET and return some data to the proxy which redirects the client to the website that they want.

We decided to back up every server using the next server. We have a cache on ram made up of one Concurrent Hashmap datatype. It stores all of the PUT calls and every hour it backs up the data to the next server. So for example, after one hour Server 1 will take everything inside of its cache and send requests to Server 2 so that a back up text file can remain current with Server 1’s main database file.

We are going to assume that at most one Server will go down and we are assuming that it cannot be the proxy Server; if our proxy goes down then the entire website is unavailable. Since there is supposed to be one point of origin we will make the proxy that point. Now if a Server goes down we will see that on our monitoring tool. We then have the option to Revive it instantly or redirect the traffic. We allow this behavior so that you may see how we redirect the traffic in case of a server failure; ideally we would try to revive automatically or revive then connect but we have done this so that you may see our work. If we redirect traffic, we tell the proxy Server to move the hash that goes to the broken server to its backup Server. Later we can heal that broken server and the Server that was handling its traffic will send it requests to update it on all the requests it was missing.

Concerns:

Our Server tries to maintain Availability and Partition Tolerance and it gives up on Consistency, knowing that it will eventually become consistent when the hour strikes and the servers update each other on the calls that were made. We give up consistency because we want our website to be available no matter which server goes down. We also optimize a ton of the processes since we are dividing the requests into 3 servers, which are all multithreaded.

To make everything run quicker we did three things: multithread both the proxy and the servers, optimize saving and finding by using a hashing system to write to and read from the exact bytes we want on the text file, and distributing the requests among three servers so that we have more CPU power. We were running a length-based hashing algorithm to distribute the traffic, but we realized that this would allow for many shorts hashing to the same server, putting too much traffic on it. Hence, we resorted to hashing the shorts using MD5, converting it to an integer, and perform mod 3 on it, so that we can have the shorts be more evenly distributed on all three servers, since the hashes we get from MD5 is less predictable than if we try to use string length.

Instructions to run our code:

1. Go to DH2026 on Pc 12 and run python Orchastrate.py in the folder. This will start our system.
2. Open another Terminal window and run monitor. This will show you a monitor of all the online servers.
3. To see the web page, go to dh2026pc12.utm.utoronto.ca:1235, not localhost. You can submit by clicking the button after entering a short and long, which are respectively capped at 24 and 120 characters.
4. To kill a single service, in another terminal run python KillService.py and give it an argument that corresponds to one of the servers, either 1,2 or 3. You will see on the monitor that the server is unresponsive, and it will prompt you for a Divert or Revive, use D and R respectively.
5. If you Divert the proxy will send the traffic that the unresponsive server was supposed to receive to the next server.
6. If you Revive it will automatically go back on.
7. If you Divert traffic you can later run Heal.py with the argument for the broken server (the one you killed) to heal it and it will back it up with all the diverted traffic that were meant for it.

