CS6650 Fall 2018 Assignment 3

Summary

This assignment builds on Assignment 2. The problem and test scenario do not change, so your client should not need any modification besides bug fixes and refactoring for efficiency or new convenience features.

You are also allowed to work in pairs for this assignment. Both people submit the same pdf to blackboard. You'll both get the same grade.

Step 1

Easy - modify your server to run on AWS Lambda. :)

You are free to just refactor your Java code, or if you are feeling chilled and confident, use any language that is supported by Lambda. There are a few fun ones to choose from!

Things to think about:

Lambda services make it difficult to do database connection pooling. This may impact your performance compared to assignment 2. Here's a tutorial that shows how to build a lambda service in Python with a free tier MySql RDS instance. The design pattern for Java should be identical.

Watch your costs! Free tier usage has restrictions described here. If you keep your Lambda service small in terms of memory footprint, which should be fine, it should be pretty cheap.

Test with 32, 64, 128 clients. Bonus points if you manage 256 or more.

Shows throughput/latency graphs as per assignment 2 and also write a short analysis comparing the performance you see here with Assignment 2 results on EC2.

Step 2

It's time to move to Google Cloud Platform (GCP) :). Cash in your free credits and get stuck in.

Google cloud has the Google Compute Engine. This is essentially the same as EC2. You build and deploy a VM and install tomcat/etc on it to run your server. Here's a guide to VM creation.

This should be a pretty simple exercise to get your server up and running. I find it easier to use than AWS (don't tell anyone at Amazon!!).

GCP also has Cloud SQL, the equivalent of AWS RDS with MySQL available. The documentation is here, Your code from AWS should work pretty much as is once you have created a MySQL instance. Here's a StackOverflow page that has a good example. This may also be useful.

Once this is all working, for performance testing you might want to use Google Cloud Load Balancing, specifically the HTTP(s) load balancer: Here's the guides on how to do this. I have not tried this and am not sure of how it is priced, so if you go down this path, let us know what you find (bonus participation points for Piazza posts). If it is painful/expensive, then just crank up the size of your single instance to as large as you can afford and test away.

Hand in the same throughput/latency graphs as per previous step, from 32-256 clients. How does the performance of GCP compare with AWS?

Bonus points if you can test up to 512 or 1024 clients. But watch costs.

Submission:

As in assignment 2, (both) submit (the same) pdf to blackboard with a URL to your repo and the following

Steps 1

- The plot and performance statistics showing the results for a test run with up to 128 threads.**(12 points)**
- Compare results with EC2 from assignment 2 **(3 points)**
- > than 128 clients **(Bonus 2 points)**

Step 2

- The plot and performance statistics showing the results for a test run with up to 256 threads.**(12 points)**
- Compare results with EC2 from assignment 2 **(3 points)**
- > than 256 clients **(Bonus 2 points)**

Deadline: Friday 23rd November (sometime :))