CS6650: Building Scalable Distributed Systems
Northeastern University
Fall 2023
Prof. Gorton

Assignment 2

Shangli Yu

1. *URL for your code repo.*

   https://github.com/uppb/cs6650_assignment2

2. *A short description of your data model (5 points) - Please state size of image used if not using the stock image, and also Database/File storage solution.*

   I am using MySQL for this assignment. My table stores the image as binary, and the album profile as a json in string format. Specifically, the SQL query used to create the table was

   ```
   CREATE TABLE albums (
       id INT AUTO_INCREMENT PRIMARY KEY,
       image_content BLOB,
       profile_data TEXT
   );
   ```

   I used an auto-increment integer id as the primary key. I used a BLOB type to store the image. A blob supports at most 65,565 bytes. Since I'm using the image given, which is 3000ish bytes, using a blob type is more than enough. The album profile was converted to a json string using the toString method defined in the API and stored as a TEXT type.

3. *Output windows for the 3 client configuration tests run against a single server/DB (5 points)*

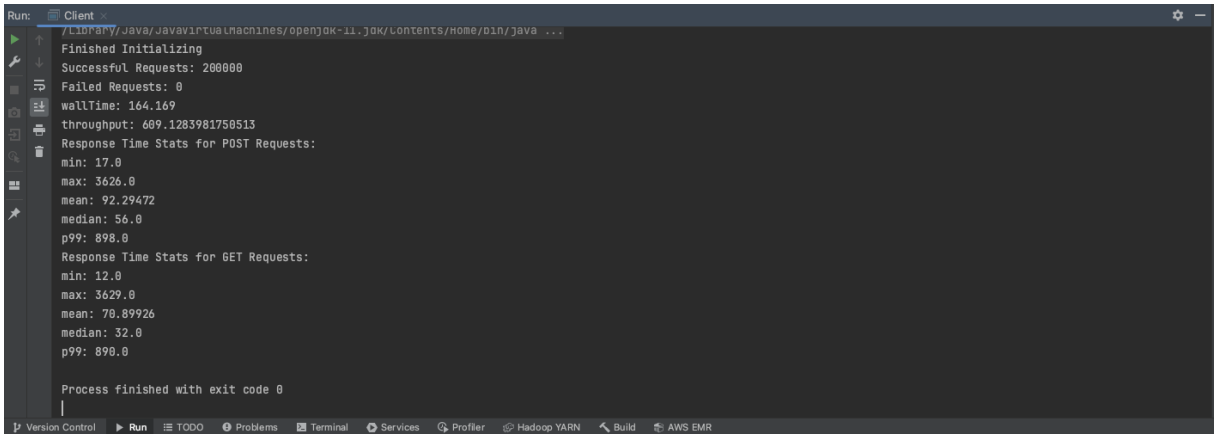


Figure 1: Single Servlet - threadGroupSize = 10, numThreadGroups = 10, delay = 2

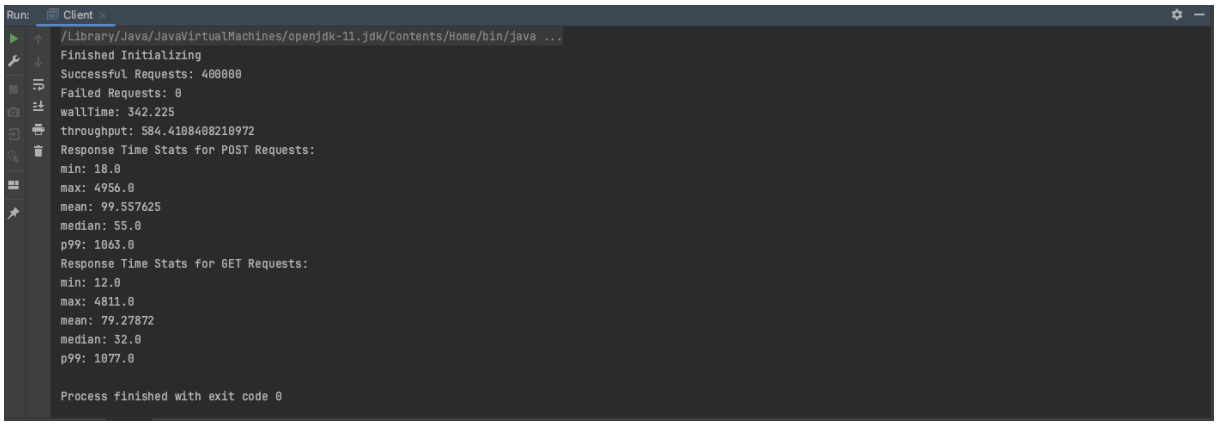


Figure 2: Single Servlet - threadGroupSize = 10, numThreadGroups = 20, delay = 2

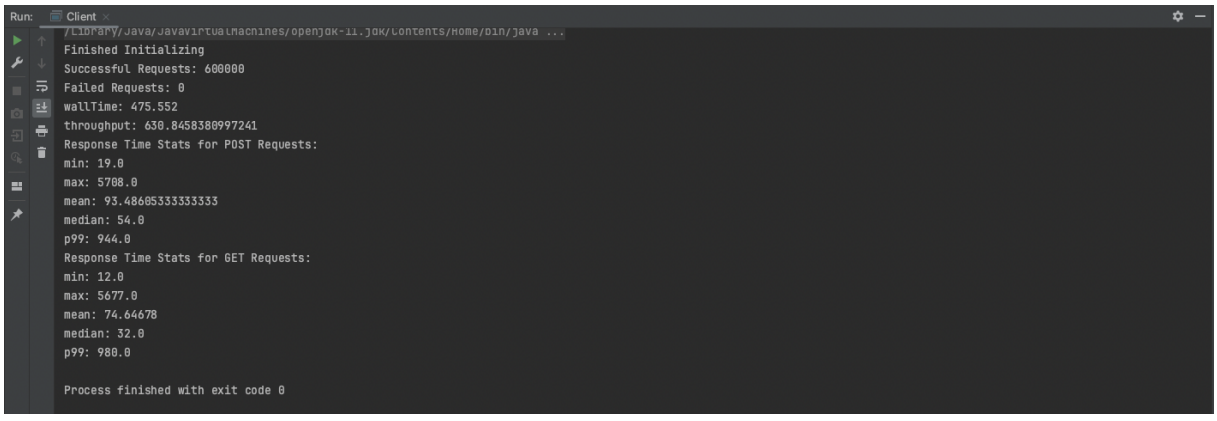


Figure 3: Single Servlet - threadGroupSize = 10, numThreadGroups = 30, delay = 2

		Single Servlet		
Configuration		10/10/2	10/20/2	10/30/2
Wall Time		164	342	476
Throughput		609	584	631
Post	Min	17	18	19
	Max	3626	4956	5708
	Mean	92	100	93
	Median	56	55	54
	P99	898	1063	944
Get	Min	12	12	12
	Max	3629	4811	5677
	Mean	71	79	75
	Median	32	32	32
	P99	890	1077	980

Table 1: Statistics for single servlet

4. *Output windows for the 3 client configuration tests run against a two load balanced servers/DB (15 points)*

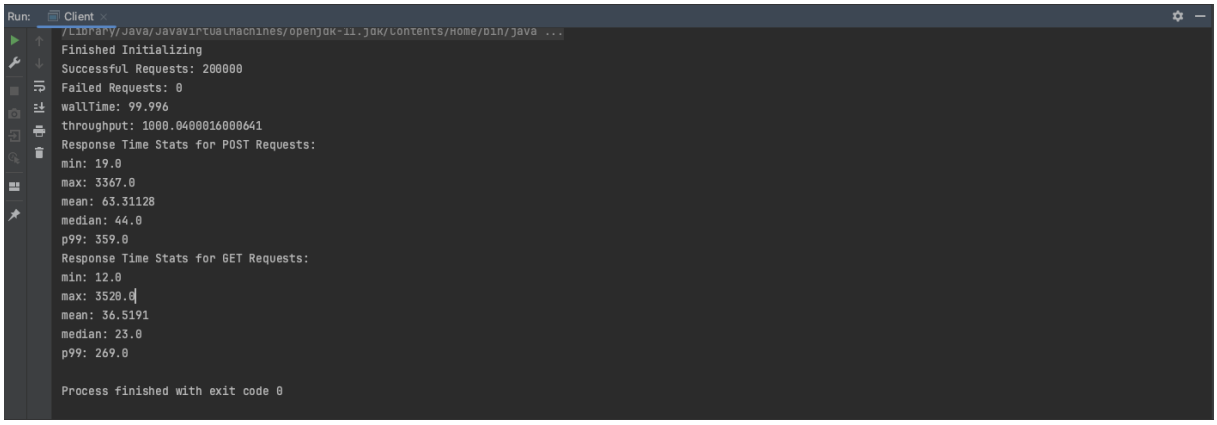


Figure 4: Two Load Balanced Servlet - threadGroupSize = 10, numThreadGroups = 10, delay = 2

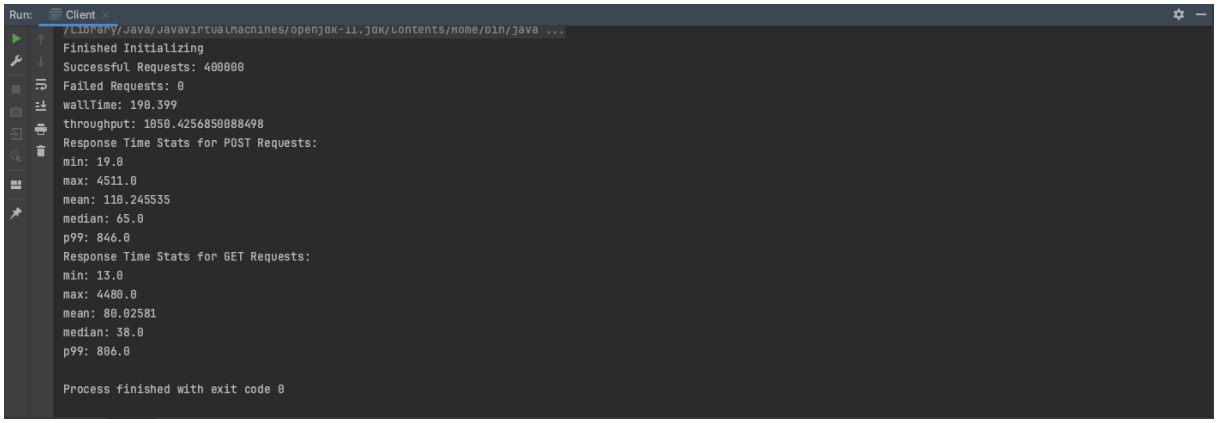


Figure 5: Two Load Balanced Servlet - threadGroupSize = 10, numThreadGroups = 20, delay = 2

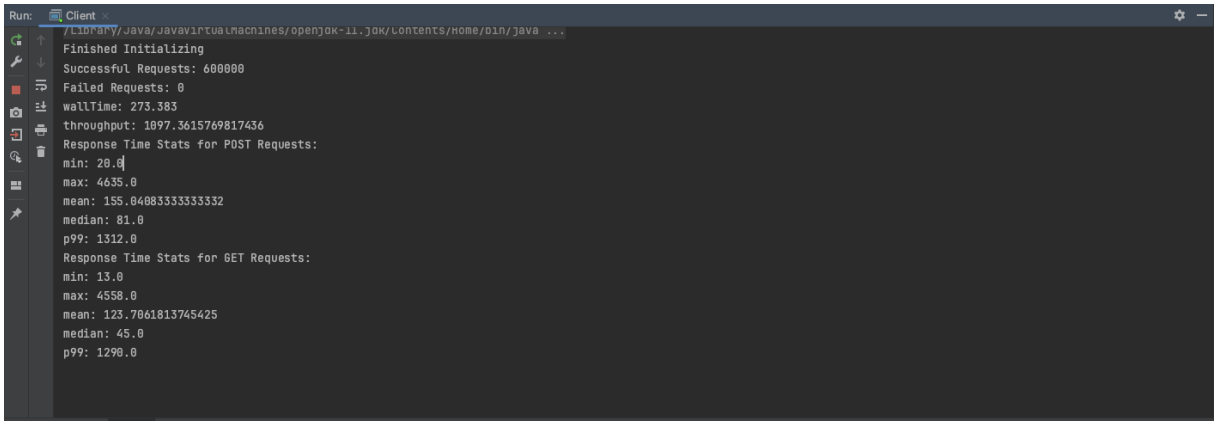


Figure 6: Two Load Balanced Servlet - threadGroupSize = 10, numThreadGroups = 30, delay = 2

		Two Load Balanced Servlets		
Configuration		10/10/2	10/20/2	10/30/2
Wall Time		100	190	273
Throughput		1000	1050	1097
Post	Min	19	19	20
	Max	3367	4511	4635
	Mean	63	110	155
	Median	44	65	81
	P99	359	846	1312
Get	Min	12	13	13
	Max	3520	4480	4558
	Mean	37	80	124
	Median	23	38	45
	P99	269	806	1290

Table 2: Statistics for two load balanced servers

5. *Output window for optimized server configuration for client with 30 Thread Groups. Briefly describe what configuration changes you made and what % throughput improvement you achieved (15 points)*

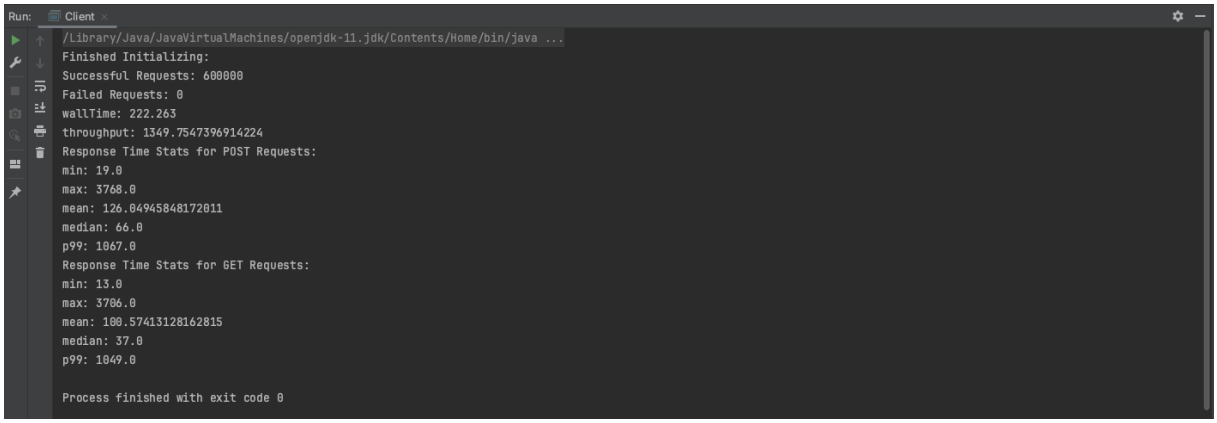
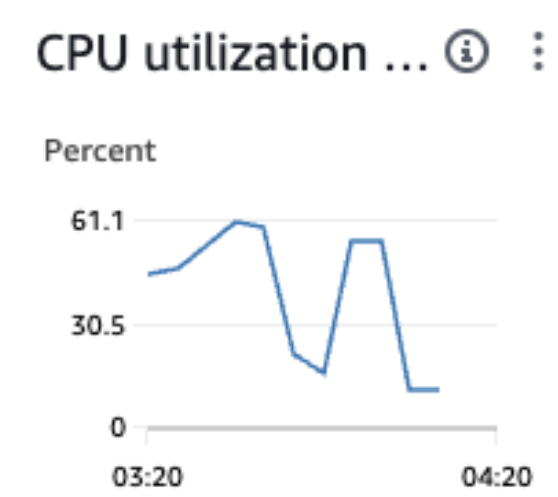


Figure 7: Optimized Load Balanced Servlets - threadGroupSize = 10, numThreadGroups = 30, delay = 2

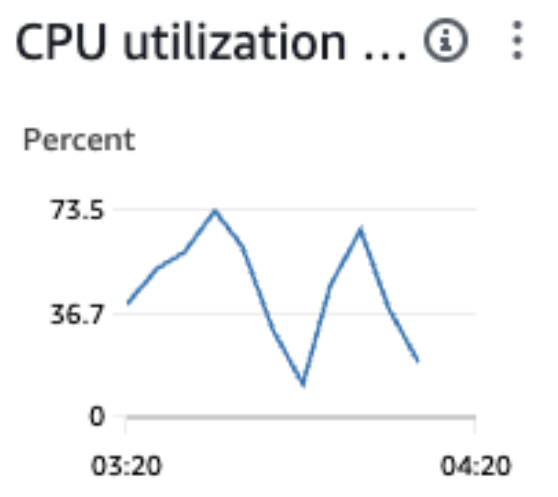
		Optimized Servlets
Configuration		10/30/2
Wall Time		222.263
Throughput		1350
Post	Min	19
	Max	3768
	Mean	126
	Median	66
	P99	1067
Get	Min	13
	Max	3706
	Mean	101
	Median	37
	P99	1049

Table 3: Statistics for Optimized Servlets

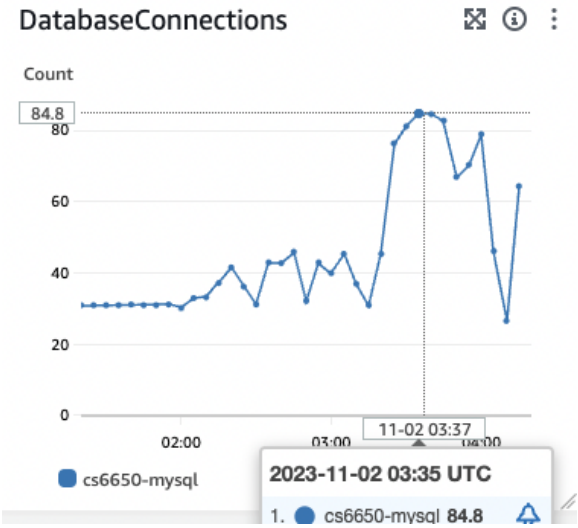
After analyzing the graphs shown in the monitoring tools of AWS, I found out that the cpu utilization for one of the instance reached 61% and the other reached 73%. Although the utilization is high, they never reached above 75% so I do not believe the servlets to be the bottleneck. Then I checked the rds instance and found out that the maximum concurrent connections established to be 84 connections, which is close to the maximum of 90 connections as I set for the connection pool. On the other hand, the cpu utilization for the rds instance reached 87%.



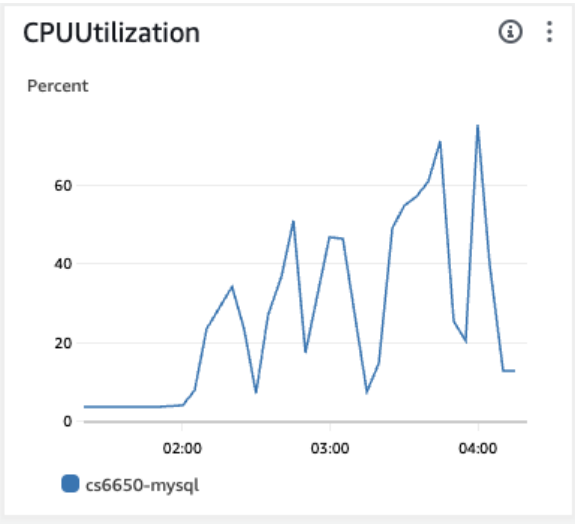
(a) CPU Utilization for 1st Instance



(b) CPU Utilization for 2nd Instance



(c) RDS Connection Count



(d) CPU Utilization for RDS Instance

Figure 8: A figure with four subfigures

Therefore I changed the instance type from the free tier db.t3.micro to db.t3.small. The results show that there was about 23% in the throughput.

		Single Servlet			Two Load Balanced Servlets			Optimized Servlets
Configuration		10/10/2	10/20/2	10/30/2	10/10/2	10/20/2	10/30/2	10/30/2
Wall Time		164	342	476	100	190	273	222.263
Throughput		609	584	631	1000	1050	1097	1350
Post	Min	17	18	19	19	19	20	19
	Max	3626	4956	5708	3367	4511	4635	3768
	Mean	92	100	93	63	110	155	126
	Median	56	55	54	44	65	81	66
	P99	898	1063	944	359	846	1312	1067
Get	Min	12	12	12	12	13	13	13
	Max	3629	4811	5677	3520	4480	4558	3706
	Mean	71	79	75	37	80	124	101
	Median	32	32	32	23	38	45	37
	P99	890	1077	980	269	806	1290	1049

Table 4: Statistics for all runs