Assignment 1

CS 6650, Fall 2023, Ximing Liang

GitHub Repository URL

https://github.com/ximing0116/CS6650-Assignment1

Client Design

The SimpleClient is designed to call an online server, sending multiple requests to save album details and retrieve them. Using tools like HttpURLConnection to connect and ObjectMapper to interpret server replies, it first conducts a series of "test" requests and then dispatches bulk requests in batches, with a pause between each batch. The program keeps track of its operation's duration and provides optional debug messages detailing its actions throughout the process.

Major classes

- **SimpleClient:** This is the main class that contains the core logic of the client operation. It's responsible for sending requests to a server, managing threads for those requests, and measuring the execution time.
- **AlbumResponse:** Nested within the SimpleClient class, it's a model class that like a data holder. It represents the response received from the server, capturing details like the album's ID and the size of its image.

Methods

- **main method:** This is the program starts and makes sure you give it the right number of instructions (4 arguments, threadGroupSize = N, numThreadGroups = N, delay = N (seconds), and IPAddr = server URL). It then sends out some "test" requests. After that, it sends out many more requests in groups and waits a bit between each group. At the end, it tells you when it started and stopped and how long it took.
- **sendRequests & sendPairRequest methods:** These are the "threads" that make server requests. One fetches a single album and repeatedly requests its details. The other sequentially retrieves multiple albums and their data one by one.
- **sendPostRequest & sendGetRequest methods:** These are the "protocols" that communicate with the server. One sends data to save an album, while the other retrieves album details from the server.
- **debugPrintln method:** This is a helper tool that displays program messages while it's active.

Special Points

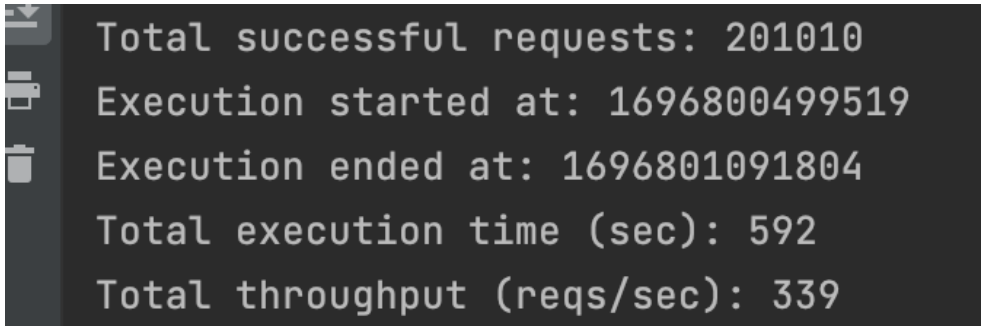
- **ExecutorService:** It control the threads, directing when they start and stop.
- **HttpURLConnection:** It's the connection method uses to communicate with the server.
- **ObjectMapper:** It translates server responses to a JSON string.

Client Part 1 Results

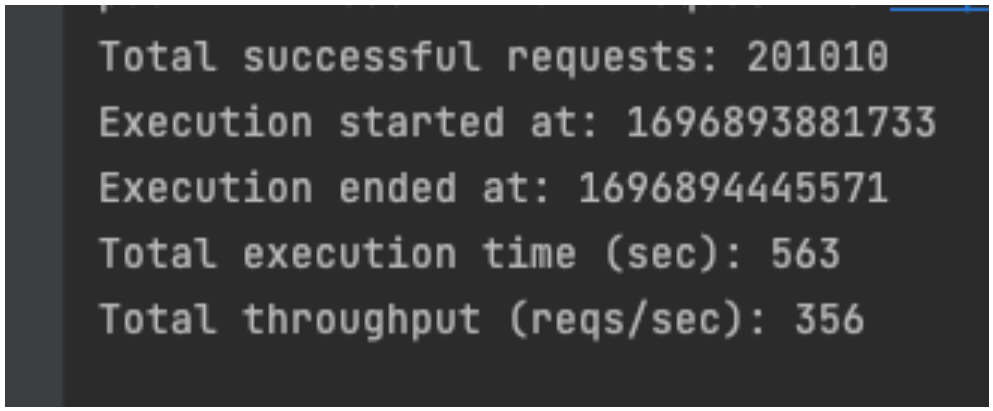
Scenarios	GO Throughput (reqs/sec)	Java Throughput (reqs/sec)
threadGroupSize = 10, numThreadGroups = 10, delay = 2	**339**	**356**
threadGroupSize = 10, numThreadGroups = 20, delay = 2	**335**	**349**
threadGroupSize = 10, numThreadGroups = 30, delay = 2	**281**	**338**

threadGroupSize = 10, numThreadGroups = 10, delay = 2

go server:

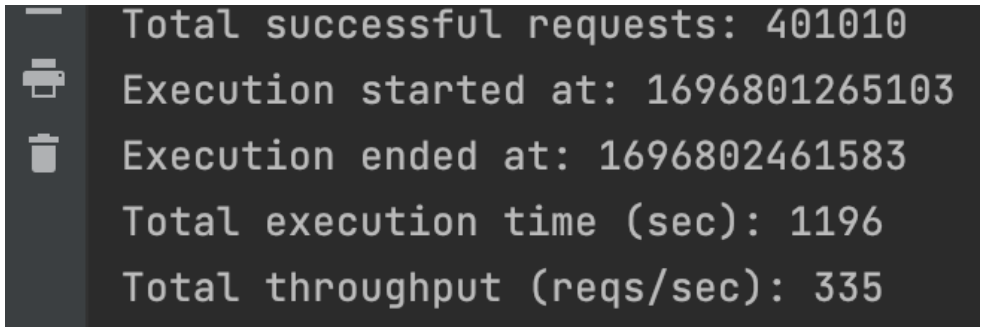


java server:

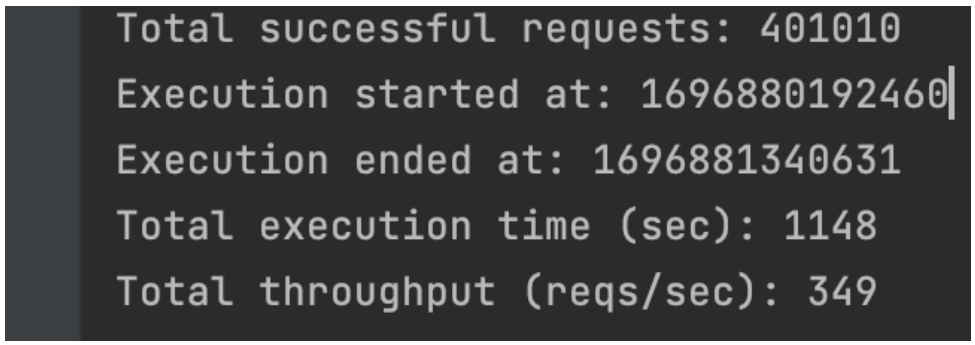


threadGroupSize = 10, numThreadGroups = 20, delay = 2

go server:

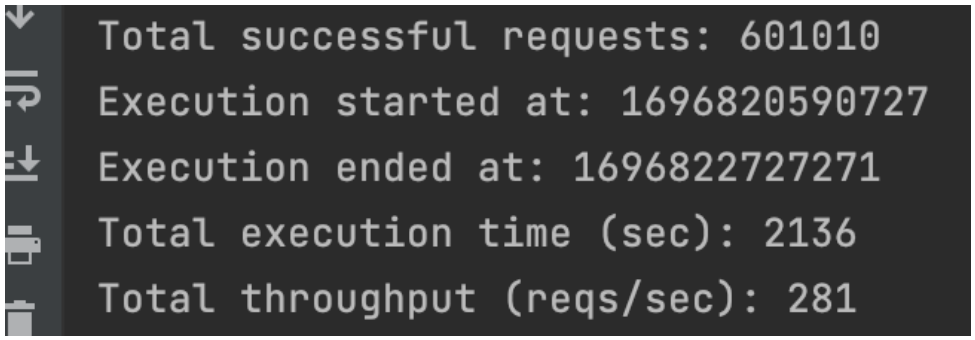


java server:

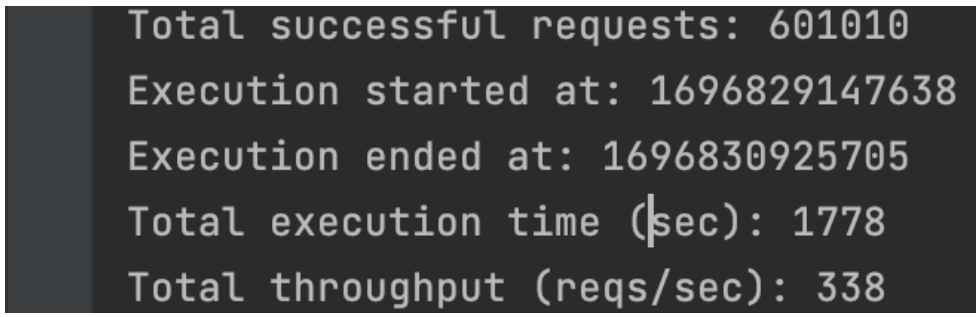


threadGroupSize = 10, numThreadGroups = 30, delay = 2

go server:



java server:

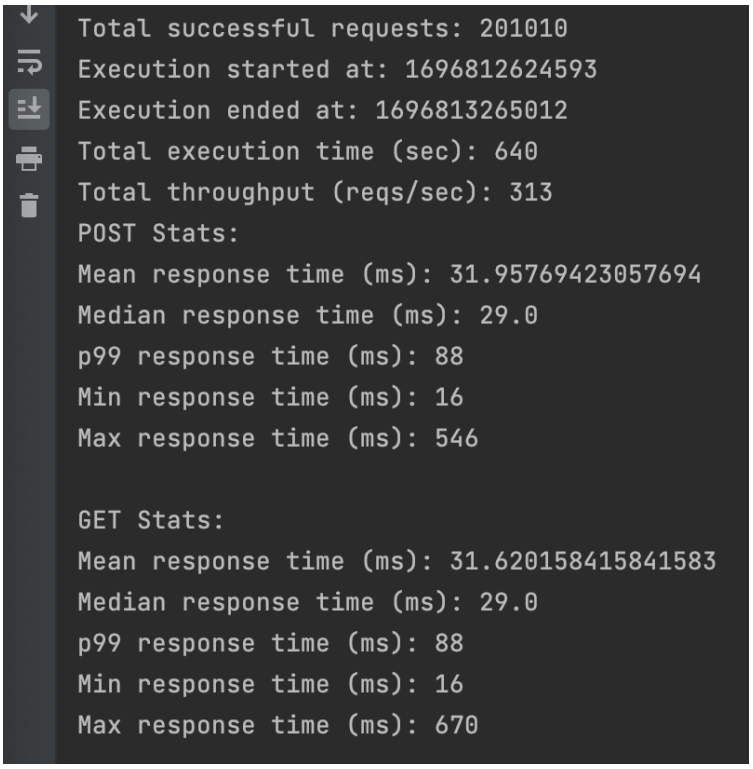


Client Part 2 Results

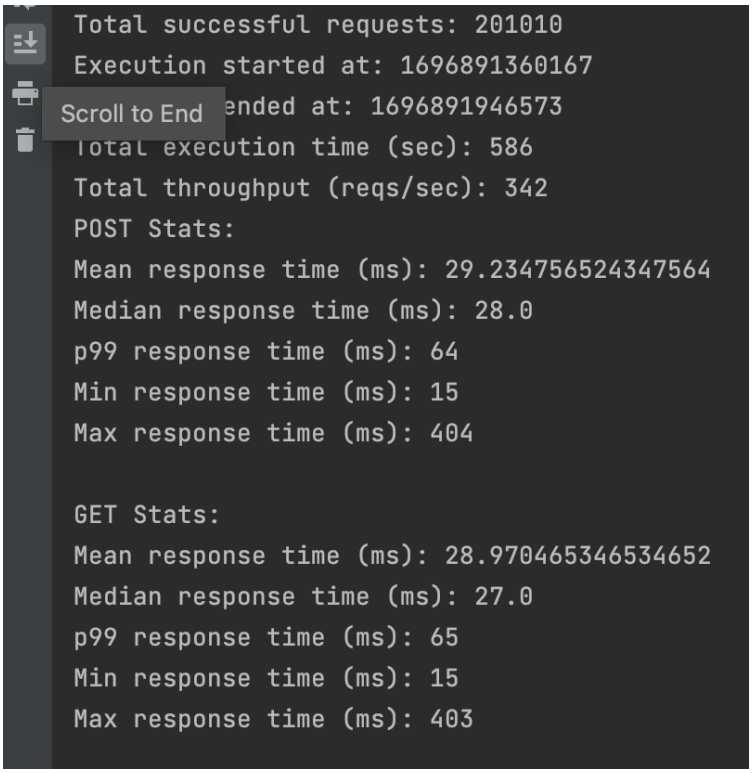
Scenarios	GO Throughput (reqs/sec)	Java Throughput (reqs/sec)
threadGroupSize = 10, numThreadGroups = 10, delay = 2	**313**	**342**
threadGroupSize = 10, numThreadGroups = 20, delay = 2	**321**	**342**
threadGroupSize = 10, numThreadGroups = 30, delay = 2	**281**	**337**

threadGroupSize = 10, numThreadGroups = 10, delay = 2

go server:

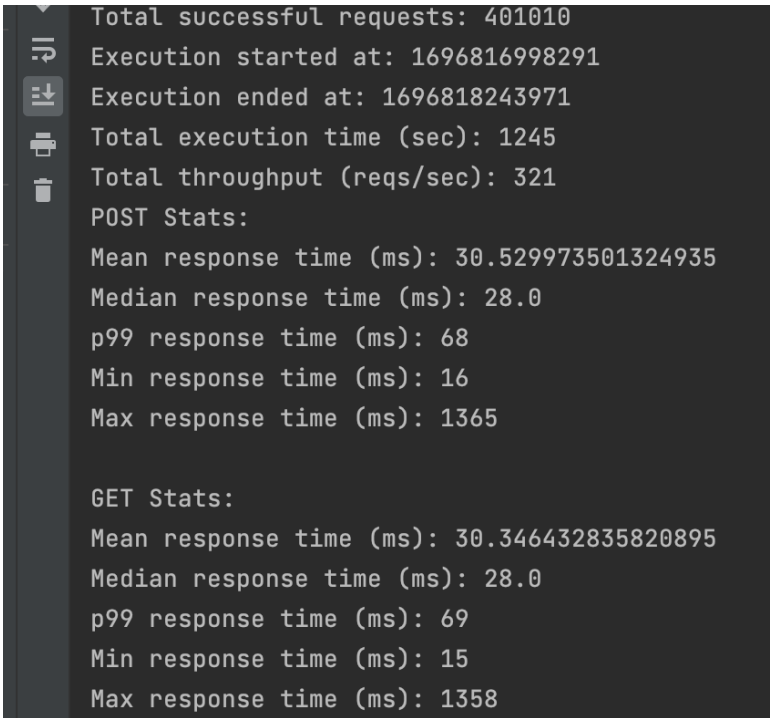


java server:

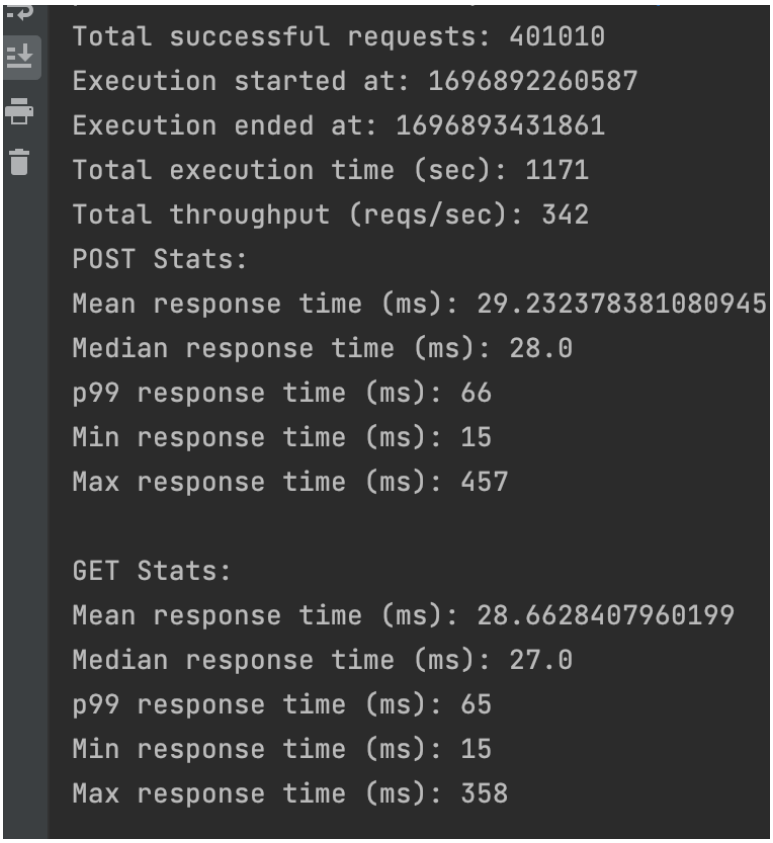


threadGroupSize = 10, numThreadGroups = 20, delay = 2

go server:

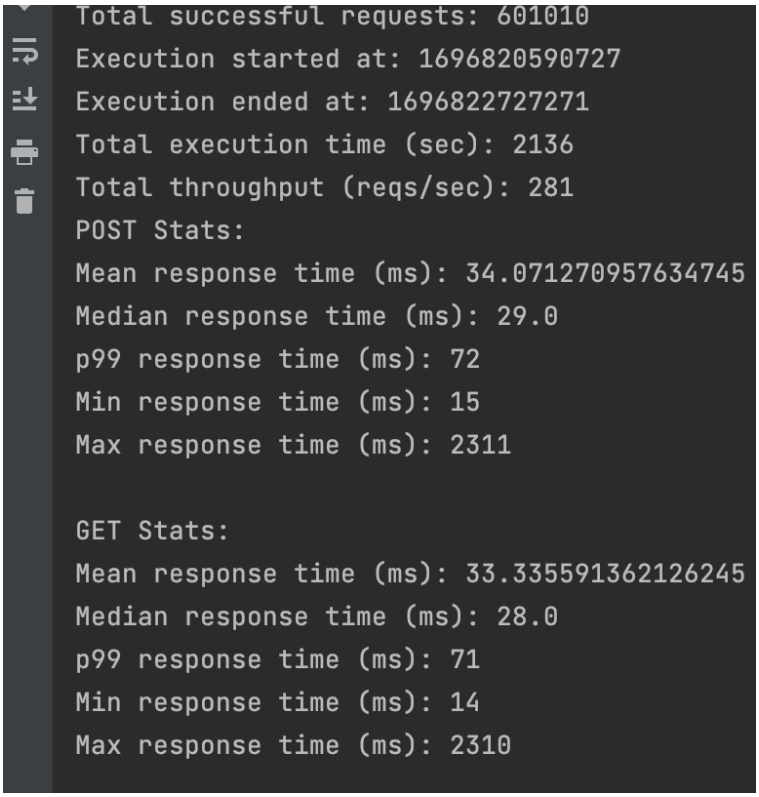


java server:

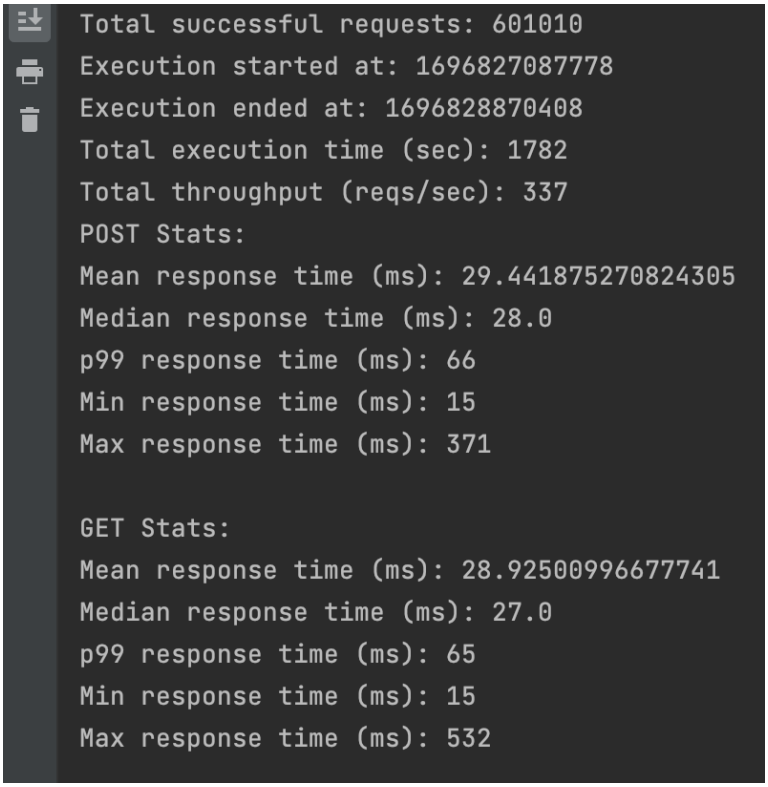


threadGroupSize = 10, numThreadGroups = 30, delay = 2

go server:



java server:



Throughput over time