Assignment 1 Write Up

CS 6650, Spring 2022
Sean Stevens

Github Repository URL

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

Client Design

My design for the client portion of this assignment heavily relies on Java's synchronization aids to ensure race conditions and deadlock do not occur. The Java CountDownLatch, ExecutorService, and AtomicInteger classes are used to keep thread contention to a minimum so that request throughput is maximized.

Major Classes

- ***Args:*** A class to parse command line arguments and convert values to the types expected by the main program. Uses the JCommander command line parser library to parse the arguments and has implementations for custom validators and converters.
- ***Client:*** The "entry point" class for the program that contains the main functionality of the program. Creates instances of other classes to parse arguments, generates synchronization aids, executes phases with their specific parameters, and prints out the final report with run statistics.
- ***DataProcessor:*** Holds data (in the form of a list of LatencyRecord items) and allows synchronized access for threads to add their request data to the list. Has methods to get statistics related to the run performance, including: mean response time, median response time, max and min response times, and the 99th percentile response time.
- ***LatencyRecord:*** A POJO class that serves as a container for request data. Has fields for the start time of the request, the latency, the request type, and the response code.
- ***LatencyTest:*** A utility class that generates some sample statistics for single-threaded performance of API calls. Used to establish a baseline for response latency and expected throughput.
- ***PhaseOptions:*** Another POJO class that holds the options/parameters relevant to a particular phase of the program. Allows a decoupled way to execute a phase without needing the main method to pass many parameters around to its helper methods.
- ***WorkerRunnable:*** The Runnable task that is instantiated with different parameters when added to the fixed size thread pool for execution. The runnable performs the request with the specified bounds, retries failed requests up to 5 times, and updates the synchronization aids while adding the generated request data to the DataProcessor instance.

Little's Law Predictions

Below are some Little's Law predictions for 32, 64, 128, and 256 thread runs with 20,000 skiers and 40 lifts. The predictions are used as a comparison metric to measure how efficient the client is at sending requests (given a theoretical server with infinite request processing capacity). **Note:** The mean response time is derived from the client's own data output.

Little's Law:

$$L = A \, x \, W$$

Where L is the number of concurrent requests (threads), A is the number of requests entering and leaving the system, and W is the average time each request spends in the system (response time).

Threads	**Mean Response Time** (ms)	**Throughput** (reqs/sec)
32	35	914.28
64	35	1828.57
128	36	3657.14
256	37	7314.28

Packages

As mentioned above, this client implementation takes advantage of a few packages for ease of development. For command line argument parsing, the JCommander library is used as it provides good options, customization, and flexibility that can easily support this project's needs. Additionally, the Apache Commons CSV library provides a framework to print CSV files to disk while keeping client code relatively uncluttered. Finally, the Swagger auto-generated client library is included to allow for thread-safe API calls and relatively simple API set up code.

Client Part 1 Results

32 Threads:

```
------ STARTING RUN: 32 threads, 20000 skiers, 40 lifts, 10 runs ------

Total successes: 160003
Total failures: 4
Time elapsed (sec): 332.468
Total throughput (reqs/sec): 481.25834666795
```

64 Threads:

```
------ STARTING RUN: 64 threads, 20000 skiers, 40 lifts, 10 runs ------

Total successes: 159814
Total failures: 0
Time elapsed (sec): 170.996
Total throughput (reqs/sec): 934.6066574656717

Process finished with exit code 0
```

128 Threads:

```
------ STARTING RUN: 128 threads, 20000 skiers, 40 lifts, 10 runs ------

Total successes: 159820
Total failures: 5
Time elapsed (sec): 97.444
Total throughput (reqs/sec): 1640.1215056853166

Process finished with exit code 0
```

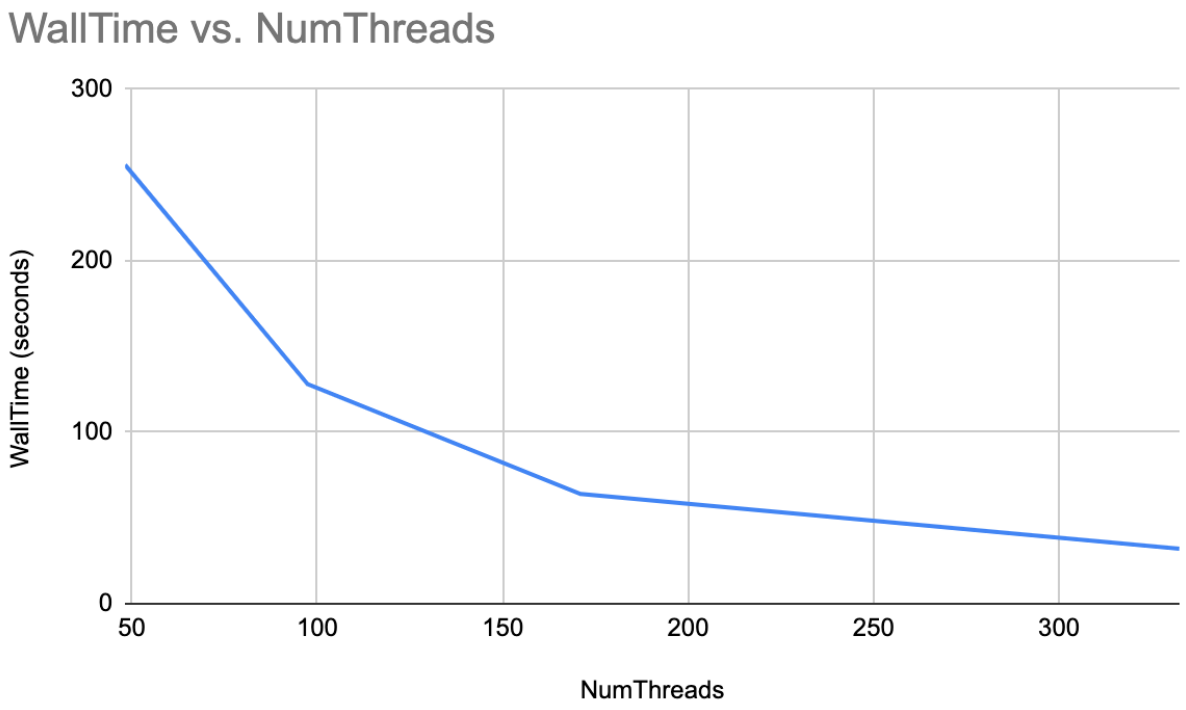
256 Threads:

```
------ STARTING RUN: 256 threads, 20000 skiers, 40 lifts, 10 runs ------

Total successes: 159769
Total failures: 0
Time elapsed (sec): 48.183
Total throughput (reqs/sec): 3315.8790444762676

Process finished with exit code 0
```

Plot of Thread Count and Wall Time:



Client Part 2 Results

32 Threads:

```
------ STARTING RUN: 32 threads, 20000 skiers, 40 lifts, 10 runs ------

------ RUN STATISTICS ------

Total successes: 160003
Total failures: 8
Time elapsed (sec): 372.596
Total throughput (reqs/sec): 429.42758

Max response time: 10003
Min response time: 23
Mean response time: 42.63411890432533
Median response time: 40
99th percentile response time: 81

Process finished with exit code 0
```

64 Threads:

```
------ STARTING RUN: 64 threads, 20000 skiers, 40 lifts, 10 runs ------

------ RUN STATISTICS ------

Total successes: 159814
Total failures: 18
Time elapsed (sec): 222.775
Total throughput (reqs/sec): 717.37854

Max response time: 10016
Min response time: 23
Mean response time: 55.55526427749137
Median response time: 48
99th percentile response time: 119

Process finished with exit code 0
```

128 Threads:

```
------ STARTING RUN: 128 threads, 20000 skiers, 40 lifts, 10 runs ------

------ RUN STATISTICS ------

Total successes: 159820
Total failures: 2
Time elapsed (sec): 143.026
Total throughput (reqs/sec): 1117.4192

Max response time: 8973
Min response time: 26
Mean response time: 60.41134512144761
Median response time: 57
99th percentile response time: 125

Process finished with exit code 0
```

256 Threads:

```
------ STARTING RUN: 256 threads, 20000 skiers, 40 lifts, 10 runs ------

------ RUN STATISTICS ------

Total successes: 159769
Total failures: 85
Time elapsed (sec): 73.971
Total throughput (reqs/sec): 2159.887

Max response time: 10116
Min response time: 26
Mean response time: 56.658744854679895
Median response time: 50
99th percentile response time: 95

Process finished with exit code 0
```

Plot of Thread Count, Mean Response Time, and Throughput:

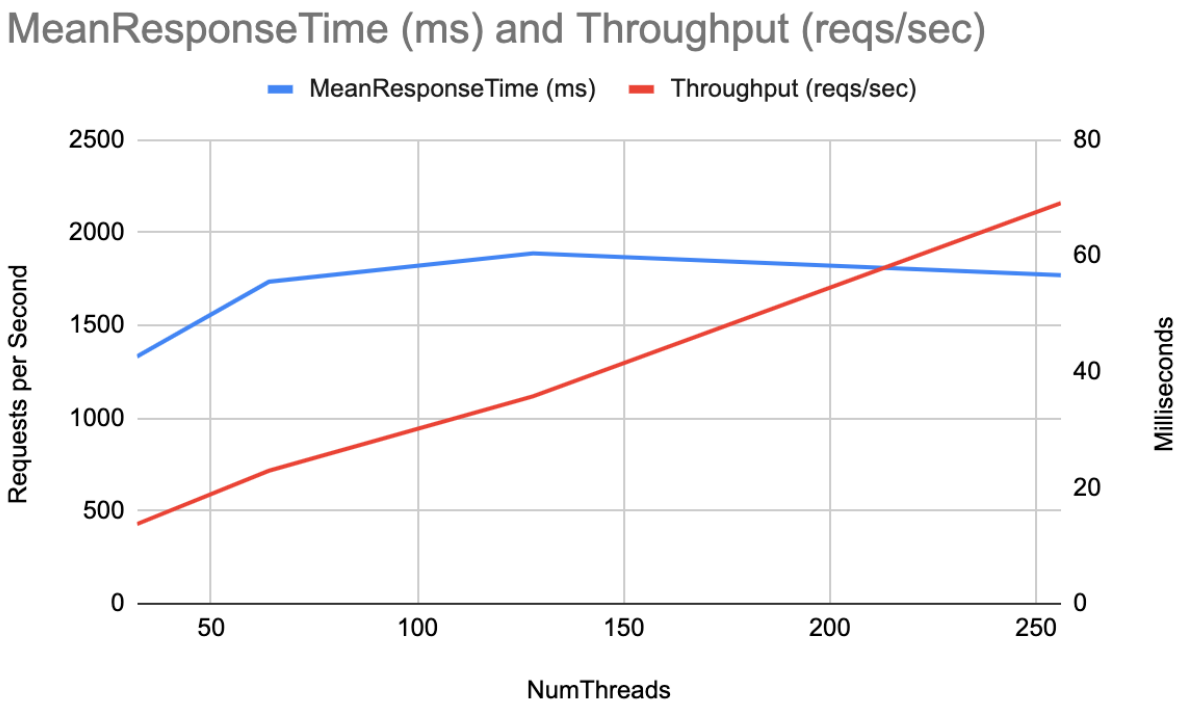


Chart of Latencies Over Time