

URL: <https://github.com/YeZhang125/cs6650-assignment1.git>

Client Architecture Overview

Key Components and Structure

The client implementation comprises the following essential classes:

1. MultiThreadedLiftRideClient.java

- **Purpose:** Acts as the program's main entry point, orchestrating the load testing process.
- **Functions:**
 - Configures parameters such as server URL, number of threads, and requests per thread.
 - Utilizes `ExecutorService` to manage concurrent execution.
 - Monitors thread completion and delegates task execution to `HTTPClientThread`.

2. HTTPClientThread.java

- **Purpose:** Handles HTTP request execution and logs responses from simulated clients.
- **Functions:**
 - Sends HTTP POST requests via `HttpClient`.
 - Logs request details, including timestamp, type, latency, response code, and throughput, to a CSV file.
 - Implements retry logic for failed requests.
 - Calculates and stores response times for further evaluation.

3. EventProducer.java

- **Purpose:** Generates randomized event data for HTTP POST requests, simulating skier lift events.
- **Functions:**
 - Produces synthetic data representing ski resort activities with randomized attributes (e.g., resort ID, season ID, skier ID).
 - Ensures varied request simulation to enhance realism in load testing.

4. LatencyComputationForClient2.java

- **Purpose:** Processes collected latency data post-test to derive performance metrics.
- **Functions:**
 - Reads logged data from CSV files.

- Computes statistics such as mean, median, minimum, maximum, and p99 latencies.
- Determines system throughput.

The server implementation comprises the following essential classes:

1. SkierServlet.java

Purpose:

- Handles HTTP POST requests for skier lift ride data, validates parameters, processes data, and returns appropriate responses.
- **Functions:**
 - Listens at /skiers/*, extracts parameters, and processes skier events.
 - Checks URL and request body for correct format and valid values. Returns 400 Bad Request on failure.
 - Reads JSON body and ensures required fields (liftID, time) are present.
 - Returns 201 Created on success or 400 Bad Request with error message on failure.

Project Structure and Class Interactions

Modules:

- **client1:** Houses the primary classes (`MultiThreadedLiftRideClient`, `HTTPClientThread`, `EventProducer`, `LatencyComputationForClient2`).
- **server:** Contains the servlet responsible for request handling.

Class Interactions:

- `MultiThreadedLiftRideClient` initiates and supervises multiple `HTTPClientThread` instances.
- Each `HTTPClientThread` collaborates with `EventProducer` to generate request data.
- After execution, `LatencyComputationForClient2` evaluates and reports system performance.

Throughput Performance Estimation Using Little's Law

Throughput Calculation:

Throughput is measured by analyzing the total test duration and the number of completed requests.

Given:

- **Total Requests** = 200,000
- **Request time per request** = 0.1 sec
- **Number of threads** = 500

Steps:

1. **Time to process all requests:**

Since we have 500 threads running in parallel, each thread can handle requests simultaneously. The total time to process all 200,000 requests would be:

Total time = Total Requests / Threads × Request time per request = Total time = $200,000 / 500 \times 0.1 = 400 \times 0.1 = 40$ seconds

2. **Throughput using Little's Law:**

We can now calculate the throughput. Throughput is the number of requests handled per second. In this case, it's simply:

Throughput = Total Requests / TotalTime = $200,000 / 40 = 5,000$ requests/sec

The estimated throughput is **5,000 requests per second**, as it takes 40 seconds to process all 200,000 requests with 500 threads, where each request takes 0.1 second.

This metric provides insight into the system's capacity and performance under different load conditions.

Screenshots

Client 1

```
===== Client 1 Output =====
Number of Threads: 500
Successful requests: 200000
Failed requests: 0
Total requests sent: 200000
Total response time: 41675 ms
Throughput: 4799.040191961608 requests per second
```

Client 2

```
===== Client 1 Output =====
Number of Threads: 500
Successful requests: 200000
Failed requests: 0
Total requests sent: 200000
Total response time: 41675 ms
Throughput: 4799.040191961608 requests per second

===== Client 2 Output =====
Response Time Metrics:
-----
Average: 99.720575 ms
Median: 96 ms
Minimum: 21 ms
Maximum: 2061 ms
99th Percentile: 371 ms

Throughput Metrics:
-----
Per-thread: 10.02802079711233 requests/sec
Estimated Total (500 threads): 5014.010398556165 requests/sec

Process finished with exit code 0
```

EC2 Instance

HTTP

http://44.243.12.46:8080/assignment1_war/skiers/10/seasons/2025/days/1/skiers/4

Save

POST

http://44.243.12.46:8080/assignment1_war/skiers/10/seasons/2025/days/1/skiers/4

Send

Params

Authorization

Headers (8)

Body

Scripts

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

Beautify

1

2

3

4

"time": 217,

"liftID": 21

Body

Cookies

Headers (5)

Test Results

201 Created

46 ms

222 B

{ JSON }

Preview

Visualize

1

2

3

{

"message": "Skier processed successfully"

}

aws

Search

[Option+S]

United States (Oregon)

voclabs/user3754450-zhang.ye5@northeastern.edu @ 2782-9776-1980

EC2

Instances

i-05ba64aa9ee4b963f

Alarm recommendations

3h

1d

1w

1h

UTC timezone

Add to dashboard

CPU utilization (%)

Percent

49.5

24.8

0

00:55

01:55

Network in (bytes)

Bytes

107M

53.4M

0

00:55

01:55

Network out (bytes)

Bytes

54.2M

27.1M

0

00:55

01:55

Network packets in (co...)

Count

566k

283k

0

00:55

01:55

Network packets out (...)

Count

371k

186k

0

00:55

01:55

Metadata no token (co...)

No unit

1

0.5

0

00:55

01:55

CPU credit usage (count)

Count

1.28

0.638

0

00:55

01:55

CPU credit balance (co...)

Count

36.8

18.4

0

00:55

01:55

Dashboard

EC2 Global View

Events

Instances

Instances

Instance Types

Launch Templates

Spot Requests

Savings Plans

Reserved Instances

Dedicated Hosts

Capacity Reservations

Images

AMIs

AMI Catalog

Elastic Block Store

Volumes

Snapshots

Lifecycle Manager

Network & Security

Security Groups

Elastic IPs

CloudShell

Feedback

© 2025, Amazon Web Services, Inc. or its affiliates.

Privacy

Terms

Cookie preferences

