Client Design Overview

This client is designed to generate load for a distributed ski lift system by sending multiple POST requests that record lift ride events. The client operates in a Multithreaded manner, creating a large number of threads that execute HTTP POST requests concurrently. The design is modular, making it flexible and efficient for testing system throughput and scalability.

Client Part 1:

LiftRideEvent is an object class that includes all lift ride information such as resortId, seasonId, dayId, skierId, liftId, and time.

EventGenerator class generates random LiftRideEvent objects and places them into the event queue. Each event consists of parameters like skier ID, resort ID, season simulating a lift ride by a skier.

ServerClient is a helper class that encapsulates the HTTP client logic. It sends POST requests to the ski lift system server and records the response HTTP response code.

MultithreadedClient class orchestrates the multithreaded operation. It manages the initialization of threads, sends requests, and handles the lifecycle of each thread. This class includes EventSender (Runnable): Each thread in the MultithreadedClient creates an instance of EventSender, which is responsible for taking events from the queue and sending them via HTTP POST to the server.

SinglethreadedClient class is a test class that sends 10000 requests from a single thread to estimate latency.

Client Part 2:

Added calculateStats( ) in MultithreadedClient to calculate the mean response time, the median response time, the p99 (99th percentile) response time, the max response time, and the min response time. Also utilized CSVWriter to generate a CSV file

Little's Law Throughput Prediction

In this client design, throughput is influenced by the number of concurrent threads and the response latency. According to Little’s Law, “The long-term average number of customers in a stable system N is equal to the long-term average effective arrival rate, λ, multiplied by the average time a customer spends in the system, W; or expressed algebraically: N = λW.”

We can use Little’s Law in distributed systems to relate the total number of users/requests, server throughput & the average request latency.

Given the design where we use 32 threads sending requests concurrently, the expected throughput depends on the average response time. If each request takes an average of W milliseconds, the predicted throughput would be roughly: *32 / W* requests per second.

Little’s Law predictions based on 32 threads which process 200K lift ride events are as followed:

N (Number of Thread)	λ (Throughput)	W (Average Response Time)
32	46000	0.695652173913043 (ms)