

# Northeastern University - Seattle



**CS6650 Building Scalable Distributed Systems**  
**Professor Ian Gorton**

# Building Scalable Distributed Systems

---

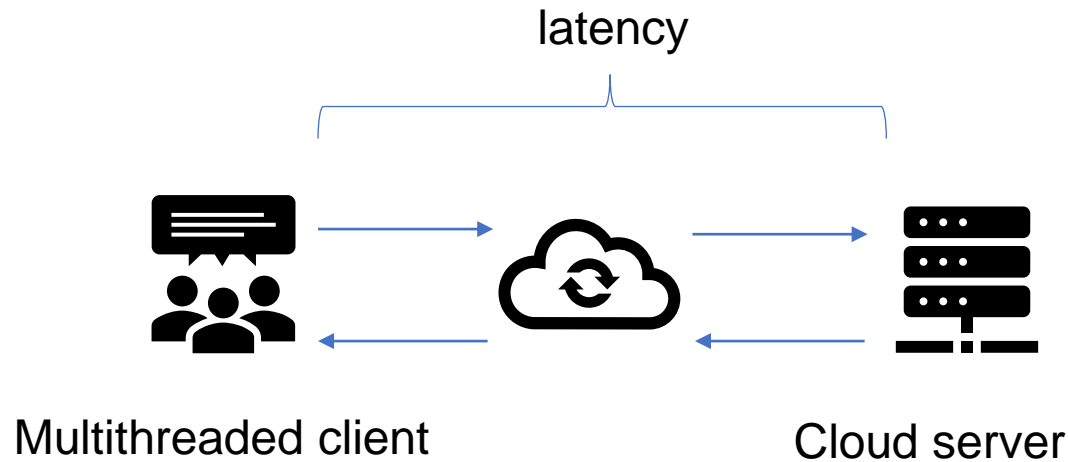
Assignment 1 Overview





Building the Client...

# Overall Aim



- Learning Objectives:
  - Write a complex multithreaded Java program
  - Implement a simple HTTP API
  - Configure and deploy a cloud based server
  - Implement a simple Java servlet
  - Learn how to analyze application performance



# Step 1 – Build the Server

- API specified in Swagger
- Server implemented as a Java servlet running in Tomcat
- Each API
  - Validates inputs
  - Returns a valid response
  - That's it 😊
- Test APIs with Postman

```
1 openapi: 3.0.0
2 info:
3   description: |
4     This is a simple supermarket example interface
5     for CS6650 at Northeastern University
6   version: "1.0.0"
7   title: GianTigle Supermarkets
8   contact:
9     email: i.gorton@northeastern.edu
10  license:
11    name: Apache 2.0
12    url: 'http://www.apache.org/licenses/LICENSE-2.0.html'
13  tags:
14    - name: purchase
15      description: a customer purchase
16    - name: store
17      description: purchases made at each store
18    - name: customer
19      description: supermarket customers
20    - name: stock
21      description: items in stock at each store.
22  paths:
23    /purchase/{storeID}/customer/{custID}/date/{date}:
24      post:
25        tags:
26          - purchase
27        summary: A customer purchase from a store
28        operationId: newPurchase
29        parameters:
30          - name: storeID
31            in: path
32            description: ID of the store the purchase
33              takes place at
34            required: true
35            schema:
36              type: string
```

Last Saved: 9:28:19 am - Jan 19, 2021

Read Only

purchase a customer purchase

**POST** /purchase  
/{storeID}  
/customer/{custID}  
/date/{date}

A customer purchase  
from a store

store purchases made at each store

customer supermarket customers

stock items in stock at each store.

Schemas

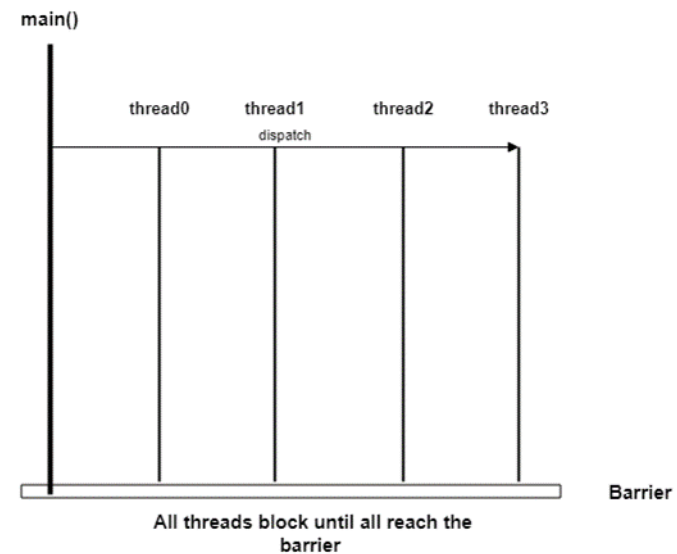
responseMsg {  
 message string  
}

Purchase {



## Step 2 – Build the Client (Simple)

- Build a multithreaded Java client to measure server performance
  - 3 phases to warmup and cooldown server
  - Max 256 threads
  - Write-heavy workload specified in assignment
- Test with 32, 64, 128, 256 threads
- Measure total 'wall time' performance and throughput for a test



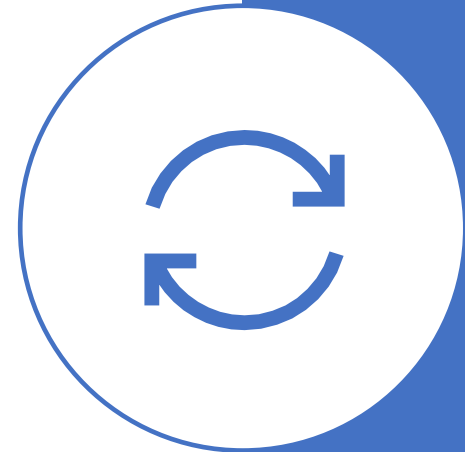
## Stage 2 Results

- Server effectively does nothing
- Client threads send requests and do nothing
- Therefore you should get the highest possible throughput 😊😊😊
- Things to consider:
  - Execute client and server as close together as possible
    - Same data center?
    - Laptop -> data center distance minimized
  - Throughput is requests/second



# Stage 3 – Instrument Client

- We want to measure and record the latency for every request
- We can use this to get insights into how the system is performing
- Every request latency needs to be stored for post-processing to calculate:
  - Mean, median, p99, throughput
- Want to achieve this with as little overhead as possible
  - Eg compare throughput with 32 threads from Stage 2 to instrumented version in Stage 3.
  - The difference is the code you have added - the instrumentation
  - Should be small (<5%)





# Things to Consider in Stage 3

- Client has limited heap space
- You are capturing 1000s of latencies – this has to be fast
  - Thread local operations are fast
- Avoid unnecessary serialization
- Collections.sort() – scalable?
- Many charting options:
  - <http://www.jfree.org/jfreechart/>



# Summary



There's lots of  
cloud/tomcat  
infrastructure  
to get working

Easy  
once  
you  
know  
how



Look critically at your  
results – we will ;)



Good luck!!

