



Created by Team **Catalyst**

---

# Final Project Presentation

Linni Cai, Shengnan You, Furong Tian

April 26, 2022  
Northeastern University

## Project Overview:

- Client: run POST three times with varied day id from 1, 2, 3 with the following arguments
  - num\_threads : 256
  - num\_skiers: 20000
  - num\_lifts: 40
- Send data to server and publish data to queue
- Three consumers subscribe and receive data from queue, store the data into Redis storage
- Server implemented three GET requests



## Project Deployment:

- One Linux instance for server
- One Ubuntu instance for RabbitMQ
- Three Linux instances for consumers (skier, skier2, resort) with Redis



## Data Model for Redis Storage

- Use Pipeline to do transaction lock
- Resort: hash set
  - Key: a constant string NUM\_SKIERS
  - Field Name: ***resortID-seasonID-dayID***
  - Field Value: the number of skiers
- Skier: hash set
  - Key: a constant string TOTAL\_VERTICAL
  - Field Name: ***resortID-seasonID-dayID-skierID***
  - Field Value: the total vertical
- Skier2: hash set
  - Key: skierID
  - Field Name: ***seasonID***
  - Field Value: the total vertical



### 3 GET Requests:

- Skier Total Vertical:
  - **GET /skiers/{resortID}/seasons/{seasonID}/days/{dayID}/skiers/{skierID}**
  - get the total vertical for the skier for the specified ski day
- Skier Total Vertical Result List:
  - **GET /skiers/{skierID}/vertical**
  - get the total vertical for the skier for specified seasons at the specified resort
- Resort:
  - **GET /resorts/{resortID}/seasons/{seasonID}/day/{dayID}/skiers**
  - get number of unique skiers at resort/season/day



## Improvement in the future:

- Add configuration automation, such as IP address synchronize
- Try different data storage instead of Redis
- Try message queue in Redis instead of RabbitMQ



---

# Client POST Statistics



# Skier Total Vertical Result List Queue

## Data for Day 3

### ----- Client Post Statistics -----

#### ----- PART 1 -----

number of successful requests sent: 160420  
 number of unsuccessful requests: 0  
 the total run time for all phases to complete: 96045  
 the total throughput in requests per second: 1000

#### ----- PART 2 -----

mean response time (millisecs): 115  
 median response time (millisecs): 32  
 throughput: 1000  
 p99 (99th percentile) response time: 2021  
 min response time (millisecs): 12  
 max response time (millisecs): 9601

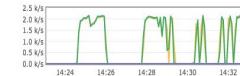
### Queue consumer\_skier\_2\_queue

#### Overview

##### Queued messages last ten minutes ?



##### Message rates last ten minutes ?



#### Details

Features: durable: true  
 Policy: Operator policy  
 Consumer capacity: 100%

Features	durable: true	State	idle	Consumers	512	Messages	0	Ready	0	Unacked	0	In memory	0	Persistent	0	Transient	Paged Out
Policy				Consumer capacity	100%	Message body bytes	0 B	0 B	0 B	0 B	0 B	0 B	0 B	0 B	0 B	0 B	0 B
Operator policy				Effective policy definition		Process memory	309 kB										

Messages: 0 B, Ready: 0 B, Unacked: 0 B, In memory: 0 B, Persistent: 0 B, Transient: 0 B, Paged Out: 0 B

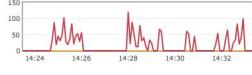
Message body bytes: 0 B, Process memory: 309 kB

## Skier Total Vertical Queue

### Queue consumer\_resort\_queue

#### Overview

##### Queued messages last ten minutes ?



##### Message rates last ten minutes ?



#### Details

Features: durable: true  
 Policy: Operator policy  
 Consumer capacity: 100%

Features	durable: true	State	idle	Consumers	512	Messages	0	Ready	0	Unacked	0	In memory	0	Persistent	0	Transient	Paged Out
Policy				Consumer capacity	100%	Message body bytes	0 B	0 B	0 B	0 B	0 B	0 B	0 B	0 B	0 B	0 B	0 B
Operator policy				Effective policy definition		Process memory	309 kB										

## Resort Queue

### Queue consumer\_skier\_queue

#### Overview

##### Queued messages last ten minutes ?



##### Message rates last ten minutes ?



#### Details

Features: durable: true  
 Policy: Operator policy  
 Consumer capacity: 100%

Features	durable: true	State	running	Consumers	512	Messages	0	Ready	0	Unacked	0	In memory	0	Persistent	0	Transient	Paged Out
Policy				Consumer capacity	100%	Message body bytes	0 B	0 B	0 B	0 B	0 B	0 B	0 B	0 B	0 B	0 B	0 B
Operator policy				Effective policy definition		Process memory	1.2 MB										

Messages: 0 B, Ready: 0 B, Unacked: 0 B, In memory: 0 B, Persistent: 0 B, Transient: 0 B, Paged Out: 0 B

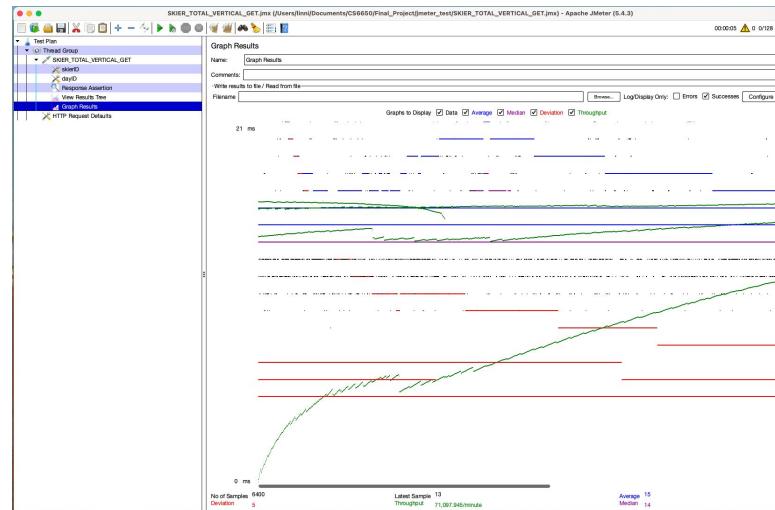
Message body bytes: 0 B, Process memory: 1.2 MB

# JMeter Test Result



# Skier Total Vertical GET

The screenshot shows the Apache JMeter interface with a test plan named "SKIER\_TOTAL\_VERTICAL\_GET.jmx". The plan includes a Thread Group with 100 threads, each performing a "SKIER\_TOTAL\_VERTICAL\_GET" sampler. A "View Results Tree" listener is active, showing the results of these samplers. Other listeners like "JSON Path Tester" and "Graph Results" are also present in the left sidebar.



Aggregate Report														00:00:05	0 / 0/128													
Thread Group		Aggregate Report																										
Name:	Aggregate Report																											
Comments:																												
-Write results to file / Read from file—																												
Filename														Browse...	Log/Display Only:	<input type="checkbox"/> Errors	<input type="checkbox"/> Successes	<input type="checkbox"/> Configure										
Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB/sec	Sent KB/sec																
SKIER_TOTAL_VERTICAL...	6400	13	13	16	19	28	10	222	0.00%	1137.6/sec	214.58	192.67																
TOTAL	6400	13	13	16	19	28	10	222	0.00%	1137.6/sec	214.58	192.67																

# Skier Total Vertical Result List GET

JMeter Test Plan (Users/linn/Documents/C56850/Final\_Project/meter\_test/SKIER\_TOTAL\_VERTICAL\_LIST\_GET.jmx) - Apache JMeter [5.4.3]

View Results Tree

Name: View Results Tree

Comments:

-Write results to file / Read from file

Filename: [ ] Browse... Log/Display Only: [ ] Errors [ ] Successes [ ] Configure

Search [ ] Case sensitive [ ] Regular Exp [ ] Search [ ] Reset

JSON Path Test

```
1. { "results": [
  2.   {
  3.     "sessionID": "2019",
  4.     "totalVertical": 200
  5.   }
]
```

JSON Path Expression [ ] Test

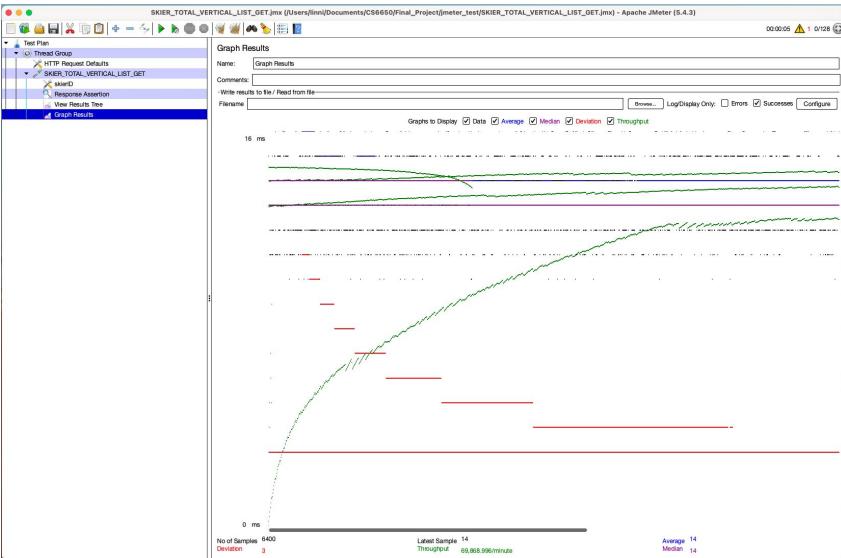
SKIER TOTAL\_VERTICAL\_LIST\_GET

skierID

Response Assertion

View Results Tree

Graph Results



Aggregate Report

00:00:05 1/ 0:28

Test Plan

- Thread Group
  - HTTP Request Defaults
  - SKIER\_TOTAL\_VERTICAL\_LIST\_GET
  - skierID
  - Response Assertion
  - View Results Tree
  - Graph Results

Aggregate Report

Name: Aggregate Report

Comments:

-Write results to file / Read from file

Filename: [ ] Browse... Log/Display Only: [ ] Errors [ ] Successes [ ] Configure

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB/sec	Sent KB/sec
SKIER_TOTAL_VERTICAL_LIST_GET	6400	13	13	16	19	27	9	146	17.06%	1151.3/sec	245.81	168.53
TOTAL	6400	13	13	16	19	27	9	146	17.06%	1151.3/sec	245.81	168.53

# Resort GET

RESORT\_GET.jmx (Users/linni/Documents/C58650/Final\_Project/jmeter\_test/RESORT\_GET.jmx) - Apache JMeter (5.4.3)

00:00:06 0 0/128

Test Plan

- Thread Group
  - HTTP Request Defaults
  - RESORT\_GET
    - dayID
    - Response Assertion
    - View Results Tree
    - Graph Results

View Results Tree

Name: View Results Tree

Comments: -Write results to file / Read from file-

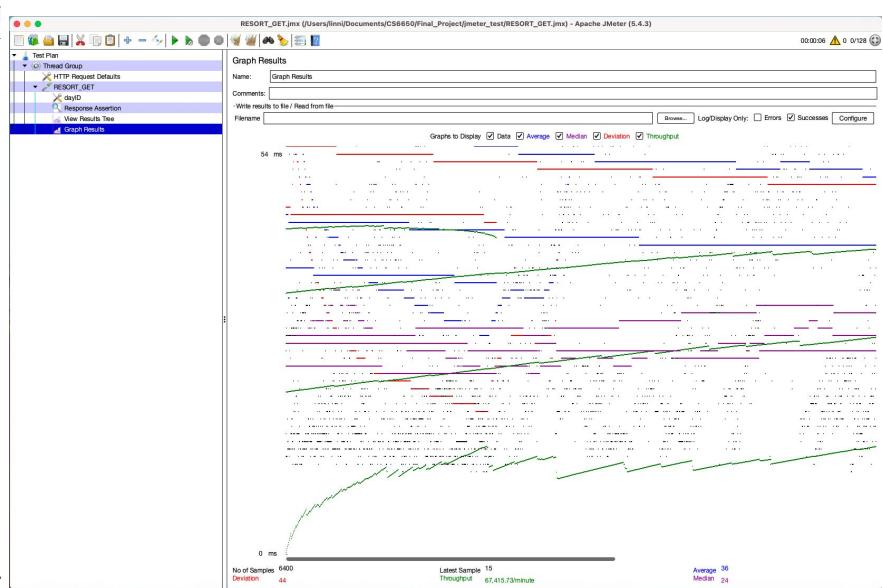
Filename:  Browse... LogDisplay Only:  Errors  Successes  Configure

Search:  Case sensitive  Regular exp  Search  Reset

JSON Path Tester

```
1
2   "resortName": "Mission Ridge",
3   "numSkiers": 161816
4 }
```

JSON Path Expression:  Test



00:00:05 0 0/128

Test Plan

- Thread Group
  - HTTP Request Defaults
  - RESORT\_GET
    - dayID
    - Response Assertion
    - View Results Tree
    - Graph Results
    - Aggregate Graph
    - Aggregate Report

Aggregate Report

Name: Aggregate Report

Comments: -Write results to file / Read from file-

Filename:  Browse... LogDisplay Only:  Errors  Successes  Configure

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB/sec	Sent KB/sec
RESORT_GET	6400	13	13	16	19	26	9	52	0.00%	1159.8/sec	179.77	190.29
TOTAL	6400	13	13	16	19	26	9	52	0.00%	1159.8/sec	179.77	190.29

Thank you

