# ECE 568 Engineering Robust Server Software

Homework 4: Exchange Matching Server

By Prathikshaa Rangarajan (pr109), Yanjia Zhao (yz476)

## Scalability Test:

The project is built in python and uses its inbuilt threads. This enforces the constraint of the Global Interpreter Lock (GIL). This is inherently less scalable than other options such as using processes or using different compilers/libraries or switching to other languages such as C/C++, Go, etc. which may be more friendly to parallel programming for optimized performance.

For scalability testing we have written a python script scalability_test.py. This script calls functions that randomly generate create and transaction requests. Each create request contains one random account and one random position creation request. Each of these may be a success or a failure and the system responds accordingly.Each transaction request contains one random order creation, one random query and one random cancel request. Each of these may be a success or a failure and the system responds accordingly.

## Experimentation Methodology:

The scalability test was performed by running the python script described above. It generates 1000 requests per each run. The program is run ten times to measure average values and deviation of the performance metrics. The performance metrics include latency and throughput. The script measures time per each request as well as the total time taken in order to measure the throughput and latency respectively.

The latency lies over an overall range of 0-60ms in our experimentation. The histograms show the number of requests with latencies in 10ms ranges in 0-60 ms inclusive. The ranges are [0, 10], (10, 20], (20, 30], (30, 40], (40, 50] and (50, 60].

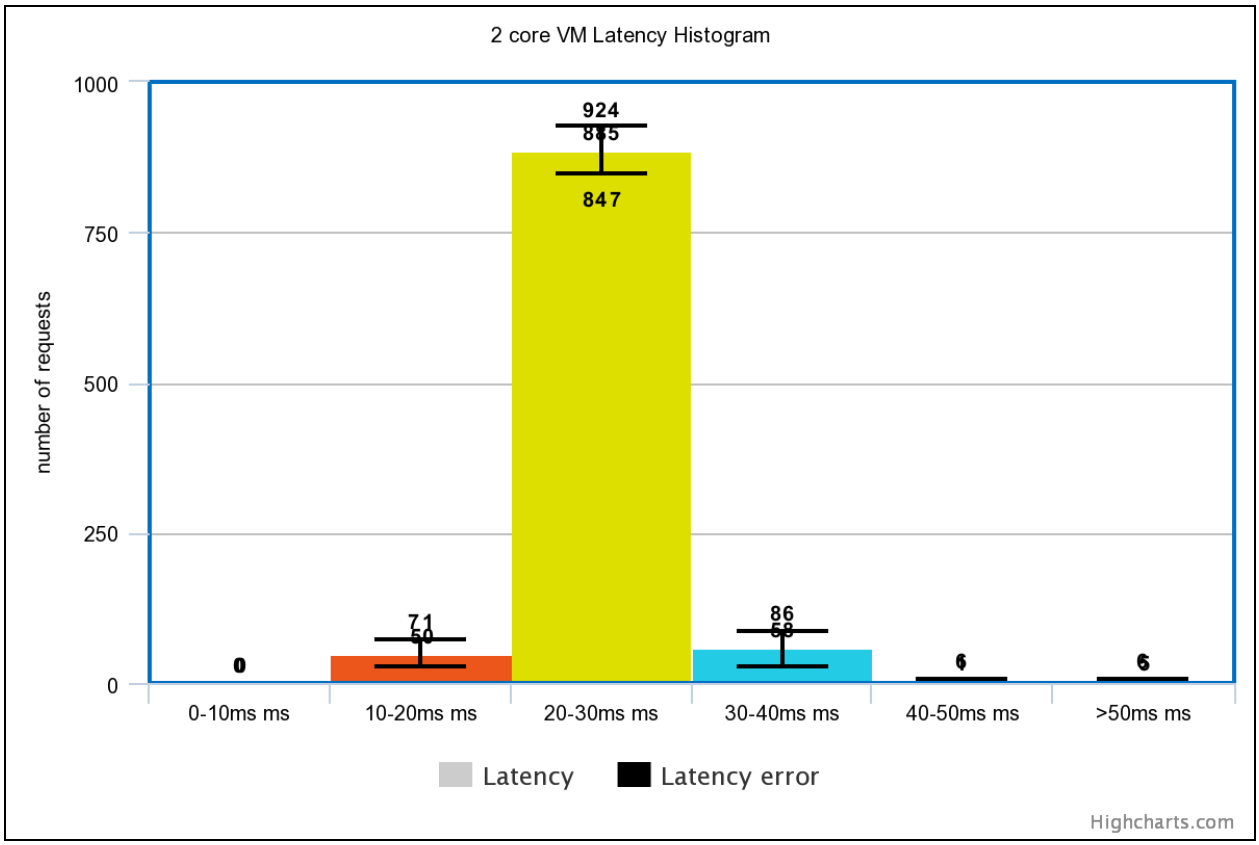
The scalability testing was performed on two different systems with specifications as described below. The latency and throughput were better on VM 2 which had more compute resources available.

The 2 core VM with 2GB RAM *(VM1)* shows a mode latency in the range of (20, 30] ms. The 4 core VM with 8GB RAM *(VM2)* showed a mode latency in the range of (10, 20] ms. The average throughput on *VM1* is 39.29 requests per second. The average throughput on *VM2* is 49.41 requests per second.

# Test VM1:

## Specification:

2GB RAM
2 Cores




# Throughput:

Total Latency for 10 runs each time with 1000 randomly generated requests:
(See scalability_test.py)

Total Latency (ms) for 10 runs:

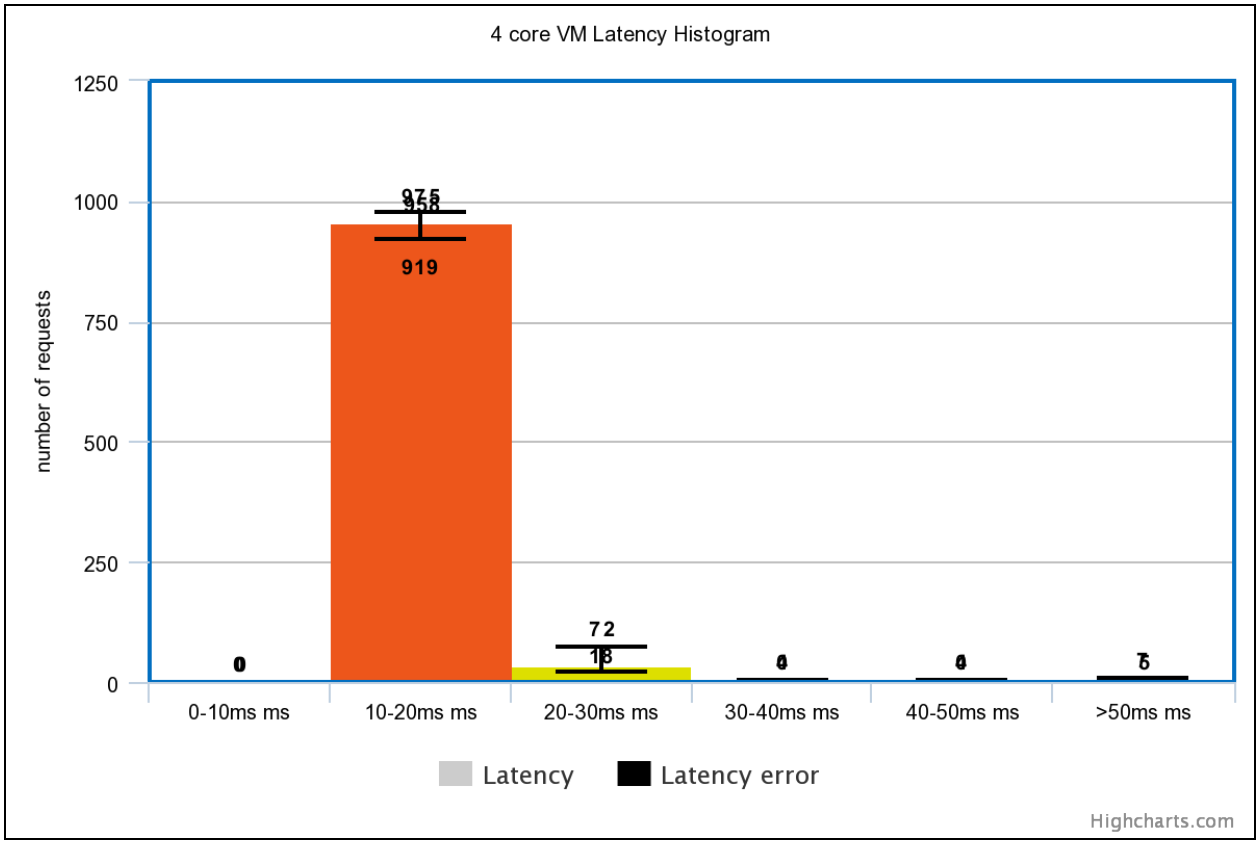
| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 28846 | 28498 | 27847 | 27762 | 28426 | 28858 | 27886 | 28767 | 28664 | 27789 |

Total requests per attempt = 1000
Avg. total latency = 28.334 sec
Avg Throughput: 1000/28.334 = 39.29 requests per second

# Test VM2:

## Specification:

8GB RAM
4 Cores




# Throughput:

Total Latency for 10 runs each time with 1000 randomly generated requests:
(See scalability_test.py)

Total Latency (ms) for 10 runs:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 20034 | 20260 | 21308 | 20128 | 19970 | 20137 | 19948 | 20545 | 20123 | 19914 |

Total requests per attempt = 1000
Avg. total latency = 20.237 sec
Avg Throughput: 1000/20.237 = 49.41 requests per second