COMPSCI 677 Spring 2022

Lab 2: Tiered Microservices-Based Toy Store

Team Members: Maoqin Zhu, Yixiang Zhang

Evaluation & Performance Measurement

In terms of screenshots of our functional tests & load tests, please check out the **"output" file** for details. In this document, we mainly focus on analyzing the results we have.

1. Load Test Results of Part 1 (on EdLab)

Our testing codes can automatically send **1000** HTTP GET or **100** HTTP POST at each terminal as you want. For average Query latency for each request, we should divide the total time by **1000** at first. For average Buy latency for each request, we should divide the total time by **100** at first. Here we vary the number of clients from 1 to 5 and measure the average latency as the load goes up.

Table1: Load test result of Part 1

Number of Clients	*Average Query Latency / s*	*Average Buy Latency / s*
1	*0.0189*	*0.0127*
2	*0.0200*	*0.0427*
3	*0.0210*	*0.0732*
4	*0.0215*	*0.0935*
5	*0.0226*	*0.1045*

Then we make a plot showing number of clients on the X-axis and response time/latency on the Y-axis.

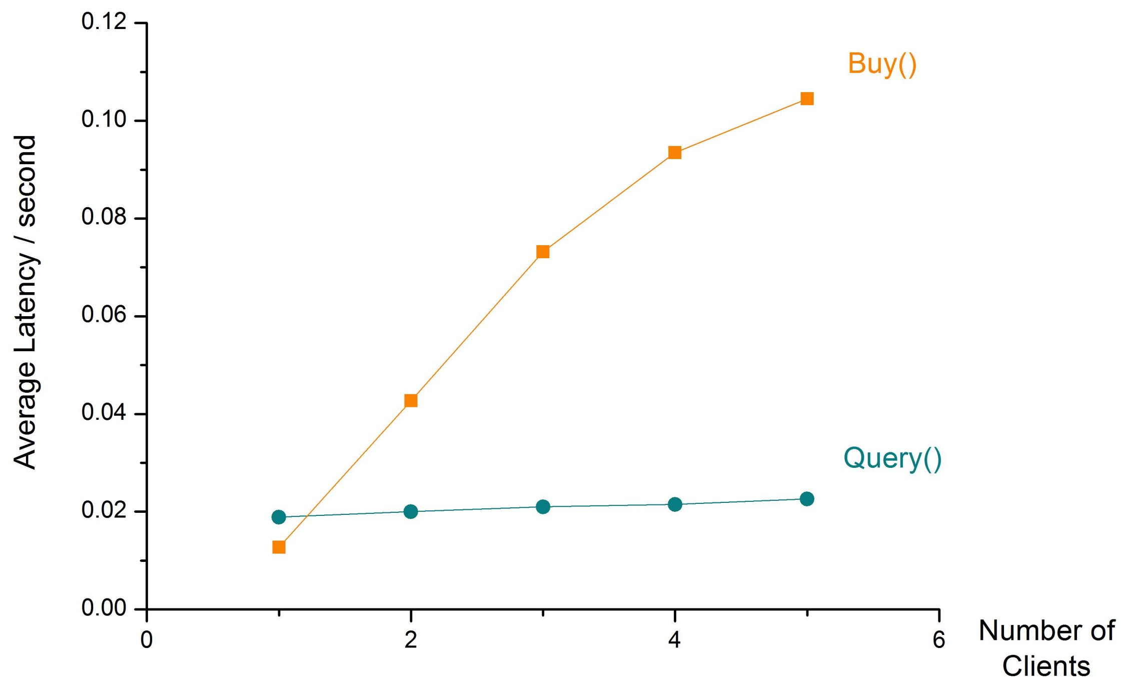

Figure1: Load test result of Part 1

2. Load Test Results of Part 2 (on Local Machine)

Our testing codes can automatically send **1000** HTTP GET or **100** HTTP POST at each terminal as you want. For average Query latency for each request, we should divide the total time by **1000** at first. For average Buy latency for each request, we should divide the total time by **100** at first. Here we vary the number of clients from 1 to 5 and measure the average latency as the load goes up.

Table2: Load test result of Part 2

Number of Clients	*Average Query Latency / s*	*Average Buy Latency / s*
1	*0.0084*	*0.0128*
2	*0.0137*	*0.0578*
3	*0.0191*	*0.0703*
4	*0.0231*	*0.0729*
5	*0.0353*	*0.0744*

Then we make a plot showing number of clients on the X-axis and response time/latency on the Y-axis.

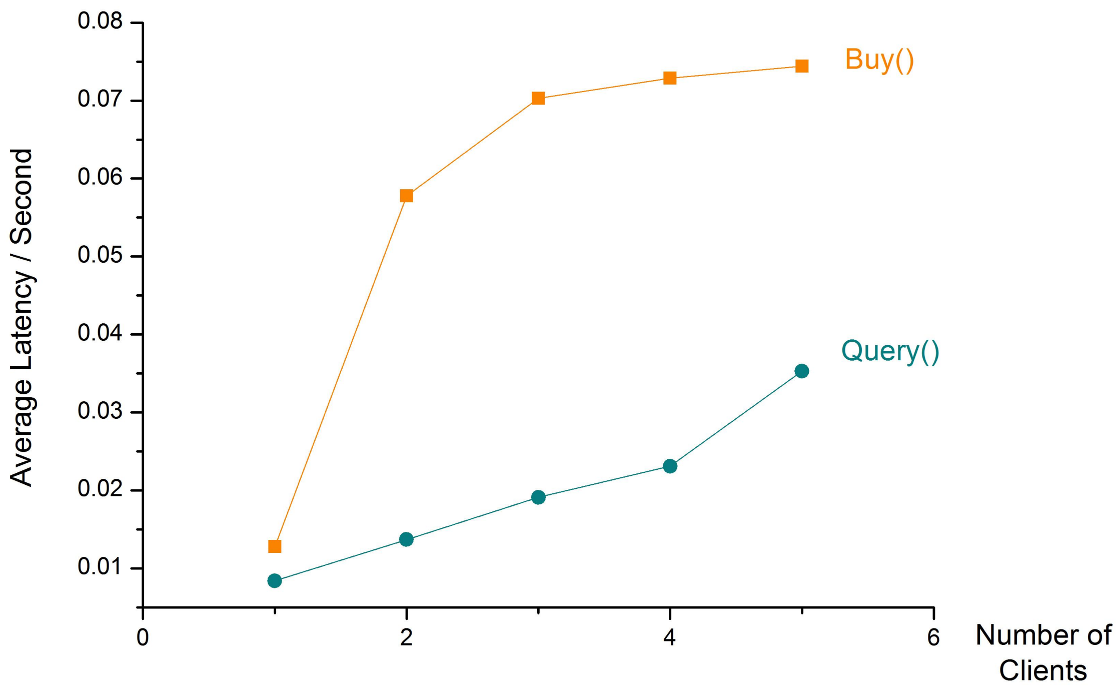

Figure2: Load test result of Part 2

3. Questions in Part 3

In this section, let's analyze load test results by answering following questions.

Problem 1

Does the latency of the application change with and without Docker containers? Did virtualization add any overheads?

Solution:

Yes, the latency of the application will change with & without Docker containers, and also virtualization actually added overheads.

Since we cannot containerize our application using Docker on remote EdLab server. In order to find out if virtualization add any overheads, we did **an extra local query load test without containers** (You can find the result screenshots of this extra load test in output file in Part2). Here we compare the numerical results with that in part 2.

Table3: Local query load test result with & without containers

Number of Clients	*Local Latency With Container / second*	*Local Latency Without Containers / second*
1	*0.0084*	*0.0078*
2	*0.0137*	*0.0092*
3	*0.0191*	*0.0136*
4	*0.0231*	*0.0176*
5	*0.0353*	*0.0221*

Problem 2

How does the latency of the query compare to buy? Since buy requests involve all three micro-services, while query requests only involve two micro-services, does it impact the observed latency?

Solution:

As **table 1**, **figure 1**, **table 2** and **figure 2** shown above, the latency of the Query is distinctively smaller than the latency of Buy, especially when there are multiple concurrent clients requests.

In terms of why the latency of the Query is smaller, one of the reasons is that Buy requests involve all three microservices, while query requests only involve two microservices. Hence Buy takes more communication time between micro-services which actually can be observed in our automatic functional test cases. (You can find the result screenshots in output file)

Another reason is that we implemented the catalog server as an in memory but persistent on disk database. In other words, in our application, it is sort of like the Redis. Hence it can be used as a cache to store data in memory for a speed up. In this case, all the Query can be faster without I/O operations. However, since the persistent strategy we use is writing the data into disk every time we modify the data in memory, for Buy method, the lock contention is more severe to solve the concurrency & synchronization problems.

Problem 3

How does the latency change as the number of clients change? Does it change for different types of requests?

Solution:

As the number of clients grows, the latency seen by clients would increase too (especially the latency of Buy). Vary the number of clients, for different types of requests, we can provide an analysis as follows:

In part 1, as the **figure 1** shows, when the number of clients increases from 1 to 5, the average Query latency recorded a slow growth from 0.0189 seconds to 0.0226 seconds. However, the average Buy latency recorded an obvious growth from 0.0127 seconds to 0.1045 seconds.

In part 2, as the **figure 2** shows, when the number of clients increases from 1 to 5, the average Query latency recorded a slow growth from 0.0084 seconds to 0.0353 seconds. However, the average Buy latency recorded an obvious growth from 0.0128 seconds to 0.0744 seconds.

At both of **figure 1** and **figure 2** , we could say the latency curve of Buy is above the latency curve of Query.