Distributed Operating System

Bazar.com: A Multi-tier Online Book Store Report

Sabah Tartir

Rawan Abo-Dyak

Supervisor : Dr. Samer Arandi

5th May, 2022

Introduction

As we did in Lab 1, the program was created with Flask and SQLite. Every server runs on a distinct system, and they communicate using HTTP requests and responses(in a restful API).

In this Lab, we add three new books and based on the popularity of the site, we will resign it by using replication, caching, load balancing, and consistency to reduce request processing time (latency).

What did we do?

1. **Replication:** for this part, we make 2 replicas (2 copies) for the catalog server and the order server but the front server remains the same with a single server. Each replica run at a different port in the same machine
2. **Consistency:** to achieve this part we create a simple cache class with LRU policy replacement, and we used HTTP calls from the server that have an update in its database to the other replicated servers. And for cache, it will delete any updated value.
3. **Load Balancing:** for this part, we used a round-robin algorithm, we made 2 counters, one for the catalog server and one for the order server, that can be set to 1 or 2 to represent which Replica is going to respond. in addition to declaring a port variable to select port 4000 or 5000 for each server.

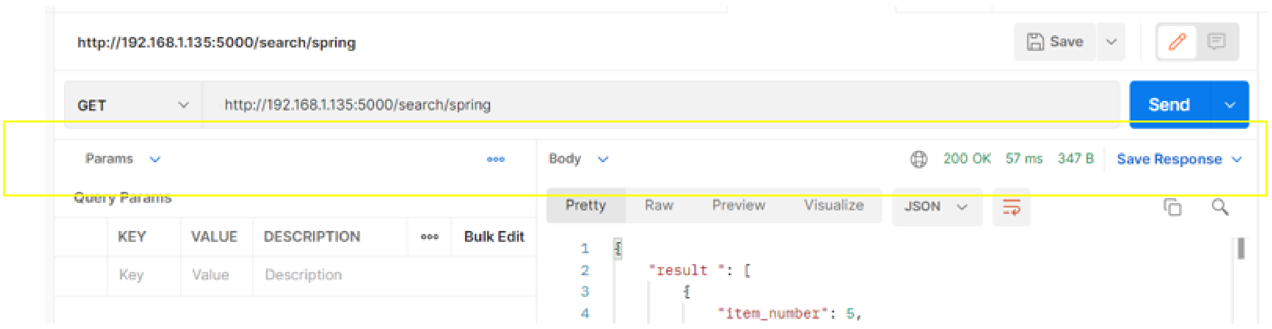
How does the program run?

To achieve this part, we used postman to send requests and show responses on the front side, for others, we run servers on its machine, each replica on a different port.

Experimental Evaluation and Measurements:

☐ **Compute the average response time (query/buy) of your new systems. What is the response time with and without caching? How much does caching help?**

We made 3 trials and got these results: (we can find them from the results in the OUTPUT pdf file)


http://192.168.1.135:5000/search/spring

GET http://192.168.1.135:5000/search/spring

Body 200 OK 57 ms 347 B

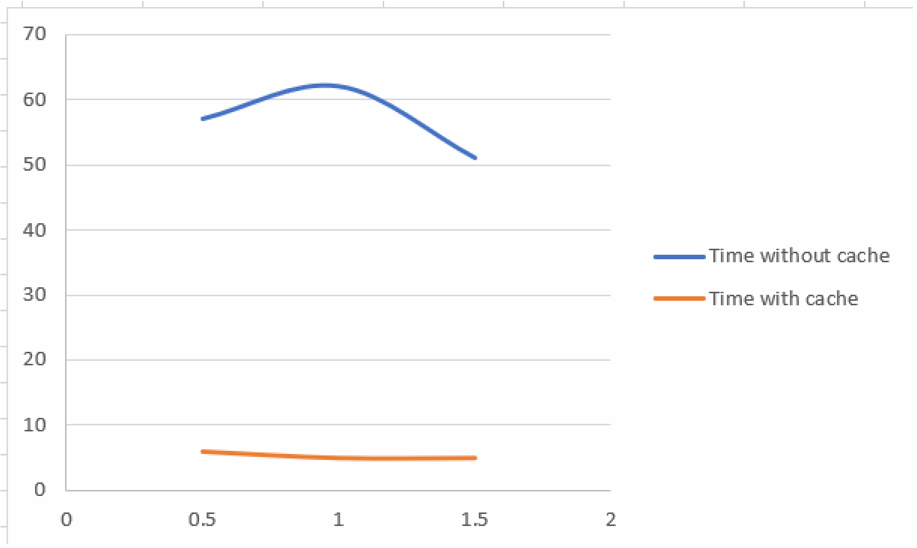
```
{
    "result ": [
        {
            "item_number": 5,
```

	Time without cache	Time with cache
Trial 1	57ms	6ms
Trial2	62ms	5ms
Trial 3	51ms	5ms

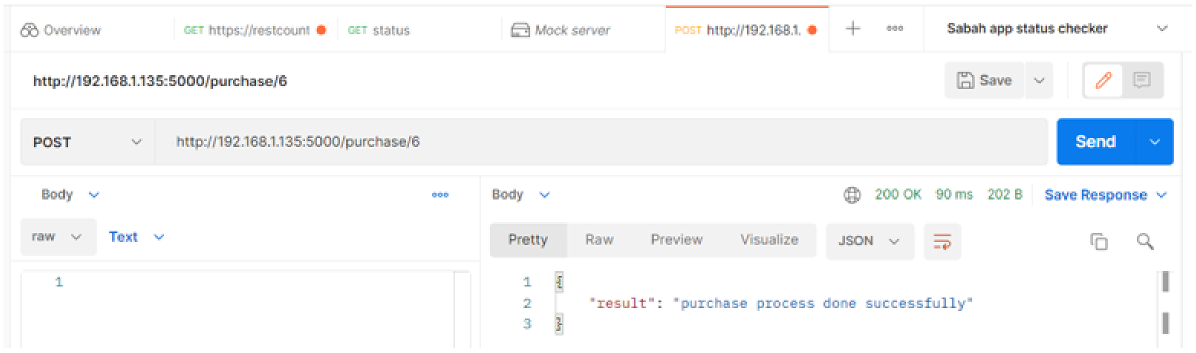
Average response time without cache = (57+62+51)/3 =56.67 ms

Average response time with cache = (6+5+5)/3 = 5.33ms

Performance: 56.67/5.33=10.63 it helps with 10 times and a half for better performance.



- [ ] **Construct a simple experiment that issues orders or catalog updates (i.e., database writes) to invalidate the cache and maintain cache consistency. What are the overhead of cache consistency operations? What is the latency of a subsequent request that sees a cache miss?**



http://192.168.1.135:5000/purchase/6

POST http://192.168.1.135:5000/purchase/6

200 OK 90 ms 202 B

```
{
    "result": "purchase process done successfully"
}
```

For this part, in the purchase process, for example, it took 90ms to update the price, because we need to send an update request to the cache in the front server (delete_from_cache()) function and update requests to the other catalog replicas, so it will spend a lot of time.

For the overhead, it's from buying a book and update operation took a long time because it needs cache consistency.

The latency will increase when there is a cache miss because it must fetch it from back servers which will cost a long time to do.