

Distributed Operating Systems

Turning the Bazar into an Amazon: Replication, Caching and Consistency

Wafaa Dwikat & Israa Wadi

12/25/2021

As we solve in Lab 1, the program was built using Flask with SQLite. Every server runs on a different machine and the communication between them is done by HTTP requests and responses.

In this Lab, we add three new books, and according to the popularity of the site, we will resign it using replication, caching, load balance, and consistency to improve request processing latency.

🔧 Replication:

For replication, we repeated (copied) both catalog server and order server into three servers. But the front one is still a single copy. Each one of the replicated servers runs on the same machine with different ports.

🔧 Consistency:

For consistency, we use HTTP calls from the server that have a change or update in its database to the other replicated servers. For cache, if the entry which will be modified in the cache, we will update the value of it.

🔧 Load balance:

For this part, we use round-robin algorithm to achieve this aim. To do that we use two counters, one for catalog server and the other for order server. Which count from 1 to 3 and according to these counters the server will send the request to the corresponding replicated server.

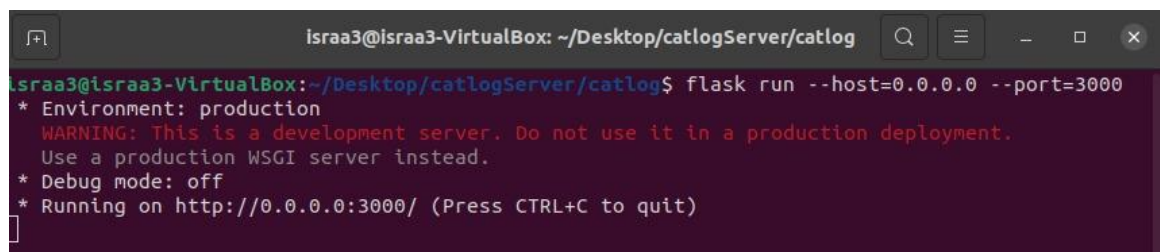
Running Bazar.com:

In order to start this program, we run the entire server on its machine, each one on a different port. And use Postman to send the request to the front server to test it.

How does it work?

First, we ran the three catalog servers on catalog server machine, each one on different port.

🔧 CatalogServer1 on port 3000

A terminal window titled 'israa3@israa3-VirtualBox: ~/Desktop/catlogServer/catlog' with search, menu, and window control icons. The terminal shows the command 'flask run --host=0.0.0.0 --port=3000' being executed. The output includes: '* Environment: production', 'WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.', '* Debug mode: off', and '* Running on http://0.0.0.0:3000/ (Press CTRL+C to quit)'.

```
israa3@israa3-VirtualBox: ~/Desktop/catlogServer/catlog
israa3@israa3-VirtualBox:~/Desktop/catlogServer/catlog$ flask run --host=0.0.0.0 --port=3000
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:3000/ (Press CTRL+C to quit)
```

🚦 CatalogServer2 on port 4000

```
israa3@israa3-VirtualBox: ~/Desktop/catalogServer2/catlog
israa3@israa3-VirtualBox:~/Desktop/catalogServer2/catlog$ flask run --host=0.0.0.0 --port=4000
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:4000/ (Press CTRL+C to quit)
```

🚦 CatalogServer3 on port 5000

```
israa3@israa3-VirtualBox: ~/Desktop/catalogServer3/catlog
israa3@israa3-VirtualBox:~/Desktop/catalogServer3/catlog$ flask run --host=0.0.0.0 --port=5000
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

Second, we ran the three catalog servers on order server machine, each one on different port.

🚦 OrdreServer on port 3000

```
israa2@israa2-VirtualBox: ~/Desktop/OrderServer/order
israa2@israa2-VirtualBox:~/Desktop/OrderServer/order$ flask run --host=0.0.0.0 --port=3000
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:3000/ (Press CTRL+C to quit)
```

🚦 OrdreServer on port 4000

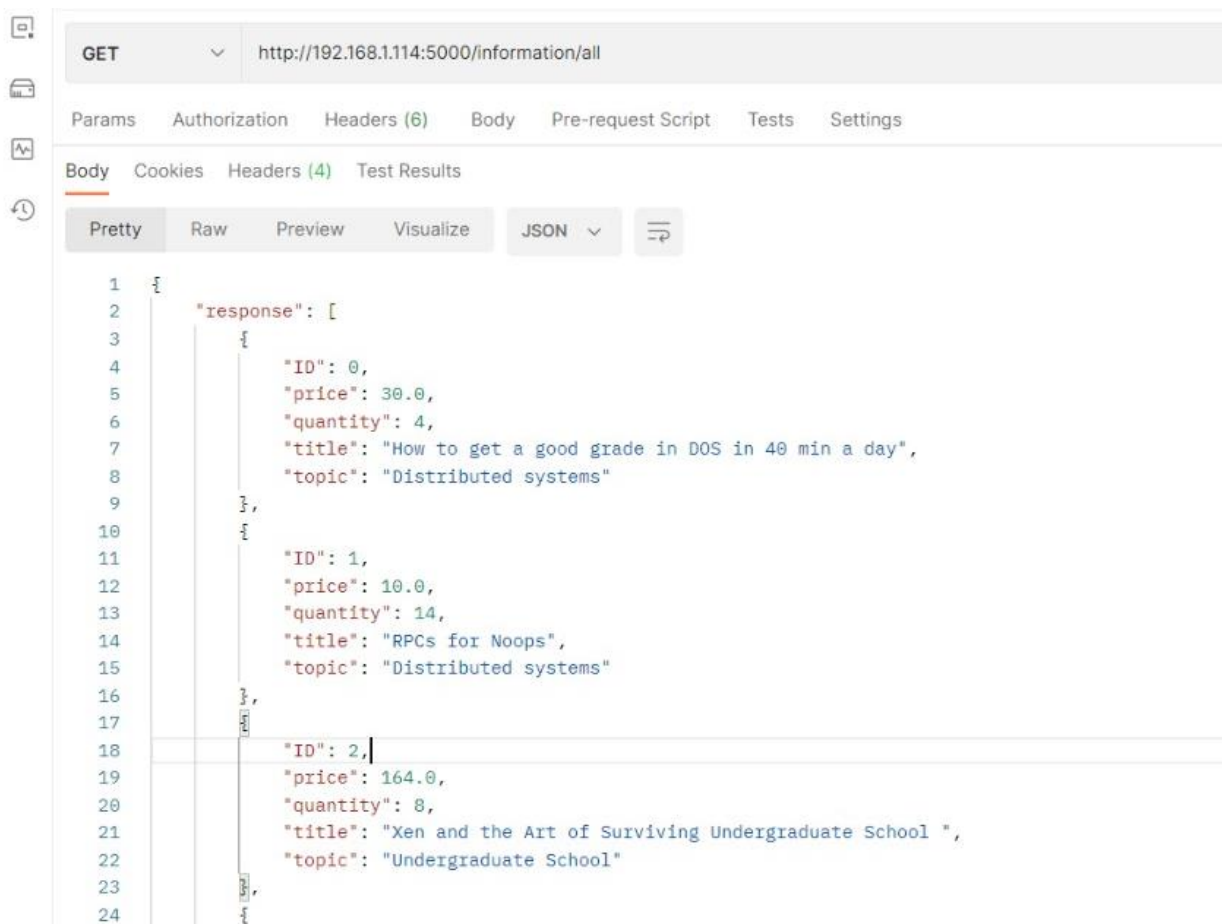
```
israa2@israa2-VirtualBox: ~/Desktop/OrderServer2/order
israa2@israa2-VirtualBox:~/Desktop/OrderServer2/order$ flask run --host=0.0.0.0 --port=4000
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:4000/ (Press CTRL+C to quit)
```

🚀 OrdreServer on port 5000

```
israa2@israa2-VirtualBox: ~/Desktop/OrderServer3/order
israa2@israa2-VirtualBox:~/Desktop/OrderServer3/order$ flask run --host=0.0.0.0 --port=5000
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

And then we use postman as client.

🚀 We add three new books to the store, so now we have 7 books



```
GET http://192.168.1.114:5000/information/all

Body
Pretty Raw Preview Visualize JSON

1 {
2   "response": [
3     {
4       "ID": 0,
5       "price": 30.0,
6       "quantity": 4,
7       "title": "How to get a good grade in DOS in 40 min a day",
8       "topic": "Distributed systems"
9     },
10    {
11      "ID": 1,
12      "price": 10.0,
13      "quantity": 14,
14      "title": "RPCs for Noops",
15      "topic": "Distributed systems"
16    },
17    {
18      "ID": 2,
19      "price": 164.0,
20      "quantity": 8,
21      "title": "Xen and the Art of Surviving Undergraduate School ",
22      "topic": "Undergraduate School"
23    }
24  ]
25 }
```

```

24
25     "ID": 3,
26     "price": 110.0,
27     "quantity": 18,
28     "title": "Cooking for the Impatient Undergrad",
29     "topic": "Undergraduate School"
30 },
31 {
32     "ID": 4,
33     "price": 190.0,
34     "quantity": 3,
35     "title": "How to finish Project 3 on time",
36     "topic": "new catalog"
37 },
38 {
39     "ID": 5,
40     "price": 36.0,
41     "quantity": 7,
42     "title": "Why theory classes are so hard",
43     "topic": "new catalog"
44 },
45 }

```

```

45     {
46         "ID": 6,
47         "price": 100.0,
48         "quantity": 48,
49         "title": "Spring in the Pioneer Valley",
50         "topic": "new catalog"
51     }
52 ]
53 }

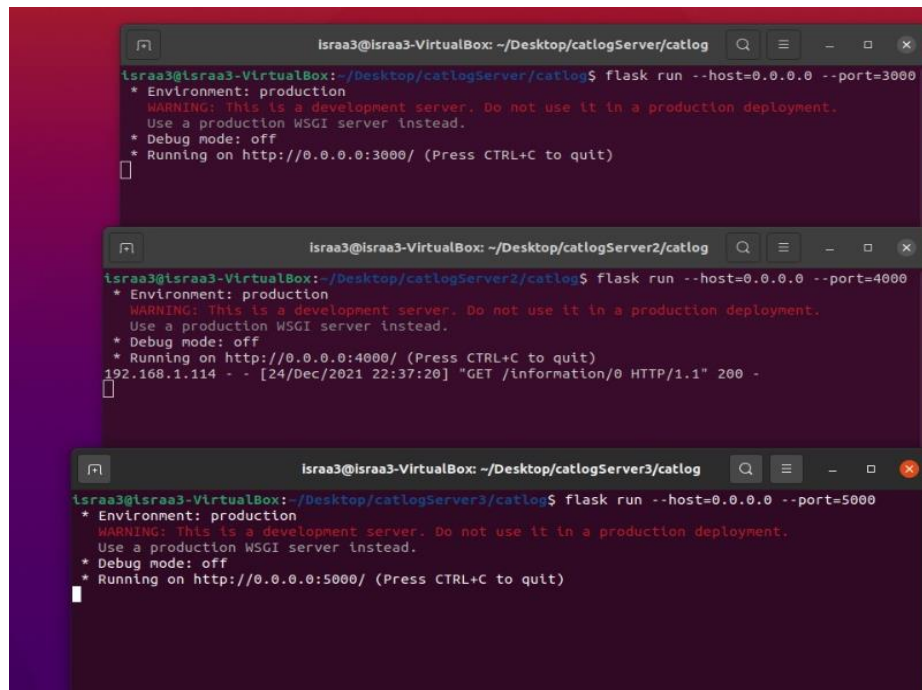
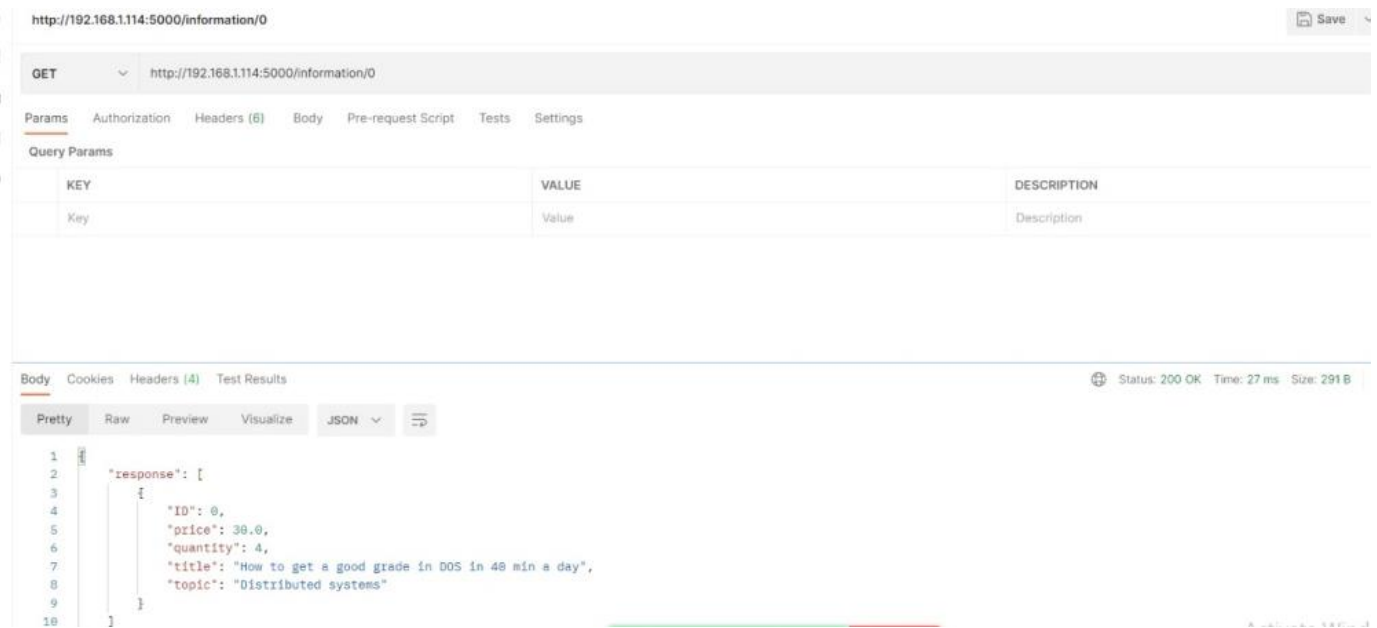
```

Cache and load balance:

Trial 1:

Before caching.

This trial will be the start of the program, that is mean the cache is empty. So the time that the request takes is shown in the figure below which is 27ms. Also, we display the three replication order servers to shoe that one of them work at once according to the round-robin algorithm in the front server.



The response returned from the catalogServer1 to the front end server then to the client will check if there is a copy of the information on the cache.

After caching.

The response returned from cache memory since the information was get to the server before this request. And the response time will be less than the first trial which is 10ms.

GET http://192.168.1.114:5000/information/0 Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

| KEY | VALUE | DESCRIPTION | ... | Bulk Edit |
|-----|-------|-------------|-----|-----------|
| Key | Value | Description | | |

Body Cookies Headers (4) Test Results Status: 200 OK Time: 10 ms Size: 289 B Save Response

Pretty Raw Preview Visualize JSON

```

1  {
2    "response": [
3      {
4        "ID": "0",
5        "price": "38.0",
6        "quantity": "4",
7        "title": "How to get a good grade in DOS in 48 min a day",
8        "topic": "Distributed systems"
9      }
10   ]
11 }

```

```

israa3@israa3-VirtualBox: ~/Desktop/catlogServer/catlog
israa3@israa3-VirtualBox:~/Desktop/catlogServer/catlog$ flask run --host=0.0.0.0 --port=3000
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:3000/ (Press CTRL+C to quit)

israa3@israa3-VirtualBox: ~/Desktop/catlogServer2/catlog
israa3@israa3-VirtualBox:~/Desktop/catlogServer2/catlog$ flask run --host=0.0.0.0 --port=4000
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:4000/ (Press CTRL+C to quit)
192.168.1.114 - - [24/Dec/2021 22:37:20] "GET /information/0 HTTP/1.1" 200 -

israa3@israa3-VirtualBox: ~/Desktop/catlogServer3/catlog
israa3@israa3-VirtualBox:~/Desktop/catlogServer3/catlog$ flask run --host=0.0.0.0 --port=5000
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)

```

The response brought from the cache memory

Trial 2:

Without caching:

As previous trial when we asked for information not found in the cache, front end server brought it from the catalog server, but here we noticed load balance.

GET http://192.168.1.114:5000/information/1 Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

| KEY | VALUE | DESCRIPTION | ... | Bulk Edit |
|-----|-------|-------------|-----|-----------|
| Key | Value | Description | | |

Body Cookies Headers (4) Test Results Status: 200 OK Time: 222 ms Size: 261 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "response": [
3     {
4       "ID": 1,
5       "price": 100.0,
6       "quantity": 14,
7       "title": "RPCs for Noops",
8       "topic": "Distributed systems"
9     }
10  ]
11 }
```

Activate Windows
Go to Settings to activate Windows.

The time in this trial is 222ms and the load balancing is shown in the figure below (the response will be back from server number 2)

```
israa3@israa3-VirtualBox: ~/Desktop/catlogServer/catlog
israa3@israa3-VirtualBox:~/Desktop/catlogServer/catlog$ flask run --host=0.0.0.0 --port=3000
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:3000/ (Press CTRL+C to quit)

israa3@israa3-VirtualBox: ~/Desktop/catlogServer2/catlog
israa3@israa3-VirtualBox:~/Desktop/catlogServer2/catlog$ flask run --host=0.0.0.0 --port=4000
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:4000/ (Press CTRL+C to quit)
192.168.1.114 - - [24/Dec/2021 22:37:20] "GET /information/0 HTTP/1.1" 200 -

israa3@israa3-VirtualBox: ~/Desktop/catlogServer3/catlog
israa3@israa3-VirtualBox:~/Desktop/catlogServer3/catlog$ flask run --host=0.0.0.0 --port=5000
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
192.168.1.114 - - [24/Dec/2021 23:05:57] "GET /information/1 HTTP/1.1" 200 -
```


With caching:

The response time with caching is 17ms which is better than the time without caching.

http://192.168.1.114:5000/information/1

GET http://192.168.1.114:5000/information/1

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

| KEY | VALUE | DESCRIPTION |
|-----|-------|-------------|
| Key | Value | Description |

Body Cookies Headers (4) Test Results Status: 200 OK Time: 17 ms Size: 259 B Save Response

Pretty Raw Preview Visualize JSON

```
1  {
2    "response": [
3      {
4        "ID": "1",
5        "price": "100.0",
6        "quantity": "14",
7        "title": "RPCs for Noops",
8        "topic": "Distributed systems"
9      }
10   ]
11 }
```

Activate Windows
Go to Settings to activate Windows.

The back servers do not receive any request.

```
Israa3@Israa3-VirtualBox: ~/Desktop/catlogServer/catlog
Israa3@Israa3-VirtualBox:~/Desktop/catlogServer/catlog$ flask run --host=0.0.0.0 --port=3000
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:3000/ (Press CTRL+C to quit)

Israa3@Israa3-VirtualBox: ~/Desktop/catlogServer2/catlog
Israa3@Israa3-VirtualBox:~/Desktop/catlogServer2/catlog$ flask run --host=0.0.0.0 --port=4000
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:4000/ (Press CTRL+C to quit)
192.168.1.114 - - [24/Dec/2021 22:37:20] "GET /information/0 HTTP/1.1" 200 -

Israa3@Israa3-VirtualBox: ~/Desktop/catlogServer3/catlog
Israa3@Israa3-VirtualBox:~/Desktop/catlogServer3/catlog$ flask run --host=0.0.0.0 --port=5000
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
192.168.1.114 - - [24/Dec/2021 23:05:57] "GET /information/1 HTTP/1.1" 200 -
```

Trial 3:

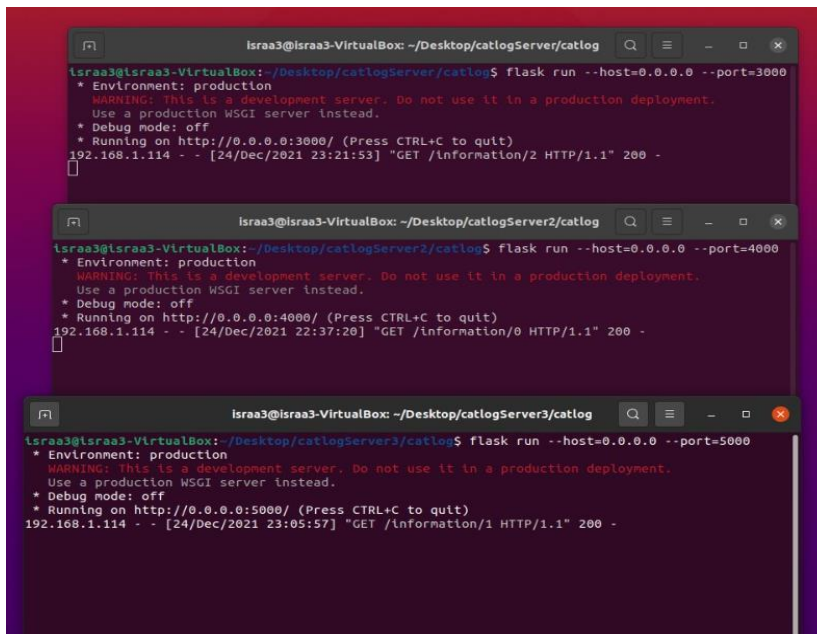
Without caching.

As previous trial when we asked for information not found in the cache, front end server brought it from the catalog server, but here we noticed load balance.

The screenshot shows a REST client interface with a GET request to `http://192.168.1.114:5000/information/2`. The response status is 200 OK, with a time of 78 ms and a size of 297 B. The response body is a JSON object:

```
{
  "response": [
    {
      "ID": 2,
      "price": 164.8,
      "quantity": 8,
      "title": "Xen and the Art of Surviving Undergraduate School ",
      "topic": "Undergraduate School"
    }
  ]
}
```

The time in this trial is 78ms and the load balancing is shown in the figure below (the response will be back from server number 3)



With caching:

The response time with caching is 15ms which is better than the time without caching.

The screenshot shows a web browser interface with a GET request to `http://192.168.1.114:5000/information/2`. The response status is 200 OK, with a time of 15 ms and a size of 295 B. The response body is displayed in JSON format, showing a list of items with details like ID, price, quantity, title, and topic. The response is cached, as indicated by the status bar.

| KEY | VALUE | DESCRIPTION |
|-----|-------|-------------|
| Key | Value | Description |

```
{
  "response": [
    {
      "ID": "2",
      "price": "164.0",
      "quantity": "8",
      "title": "Xen and the Art of Surviving Undergraduate School ",
      "topic": "Undergraduate School"
    }
  ]
}
```

The back servers do not receive any request.

The screenshot shows three terminal windows running Flask servers. The first window (port 3000) shows a GET request to `/information/2` with a 200 status. The second window (port 4000) shows a GET request to `/information/0` with a 200 status. The third window (port 5000) shows a GET request to `/information/1` with a 200 status. All three servers are running on the same host (192.168.1.114) and are not receiving any requests from the browser.

```
israa3@israa3-VirtualBox: ~/Desktop/catlogServer/catlog
israa3@israa3-VirtualBox:~/Desktop/catlogServer/catlog$ flask run --host=0.0.0.0 --port=3000
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:3000/ (Press CTRL+C to quit)
192.168.1.114 - - [24/Dec/2021 23:21:53] "GET /information/2 HTTP/1.1" 200 -

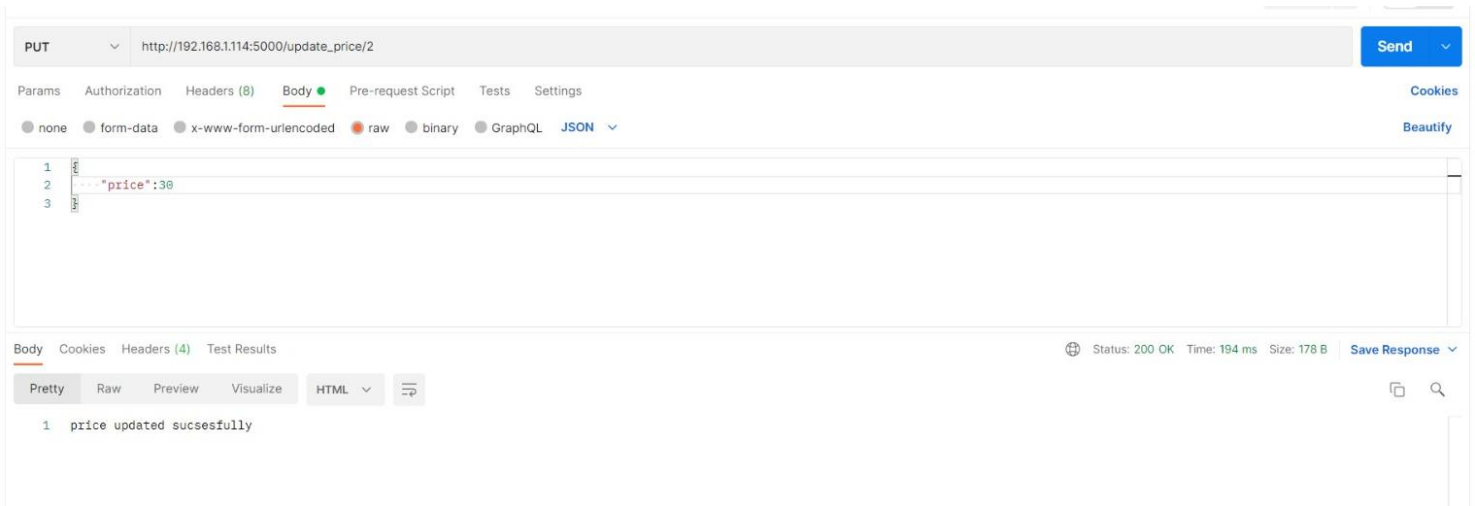
israa3@israa3-VirtualBox: ~/Desktop/catlogServer2/catlog
israa3@israa3-VirtualBox:~/Desktop/catlogServer2/catlog$ flask run --host=0.0.0.0 --port=4000
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:4000/ (Press CTRL+C to quit)
192.168.1.114 - - [24/Dec/2021 22:37:20] "GET /information/0 HTTP/1.1" 200 -

israa3@israa3-VirtualBox: ~/Desktop/catlogServer3/catlog
israa3@israa3-VirtualBox:~/Desktop/catlogServer3/catlog$ flask run --host=0.0.0.0 --port=5000
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
192.168.1.114 - - [24/Dec/2021 23:05:57] "GET /information/1 HTTP/1.1" 200 -
```

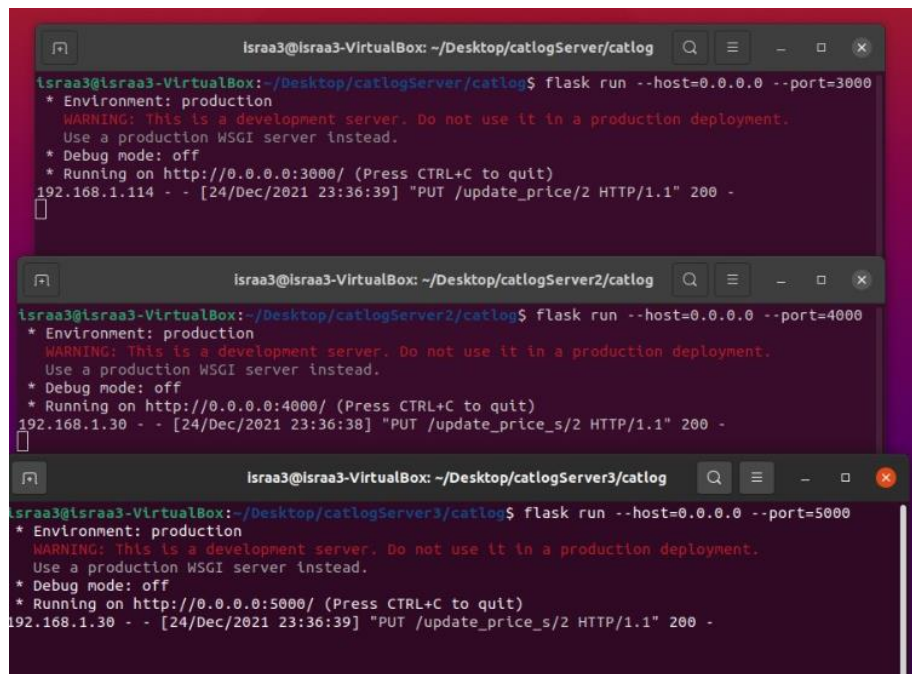
The cache has limited size, less than the catalog database size. So when the cache is full and we need to store new entry inside it we delete the maximum ID from the cache and replace it with the new entry.

Consistency

When we update information in the catalog database, the same update sent to the other replicas from the server that receives the update request, and we send invalidate request to the cache memory to update the cache. Here we sent request to update the price of a book.



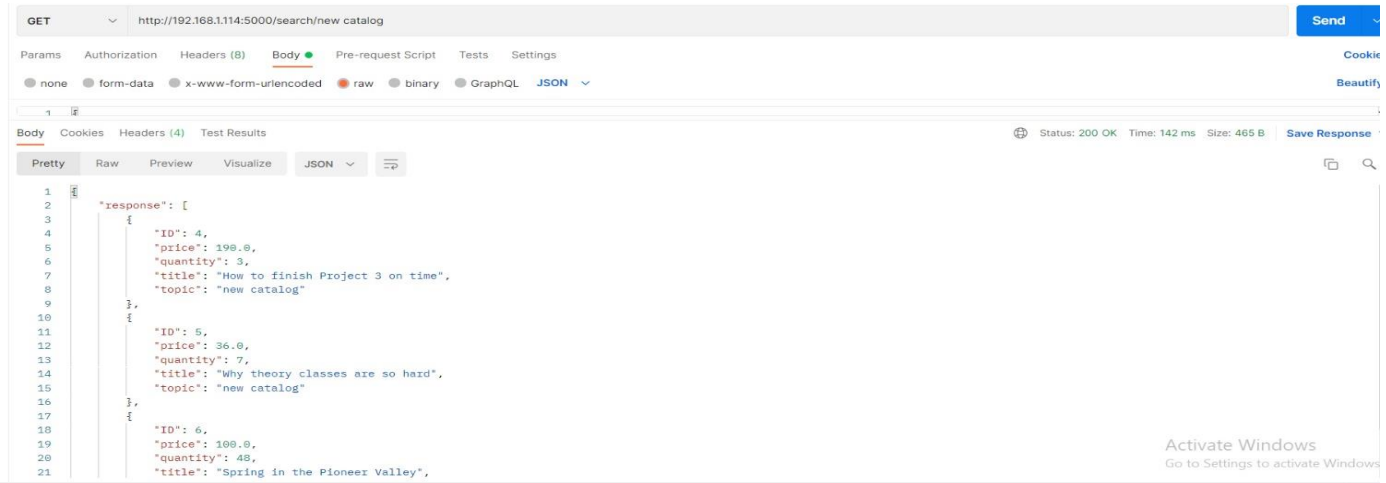
The request went to catalogServer2, but then requests sent from it to the other replicas to update their databases.



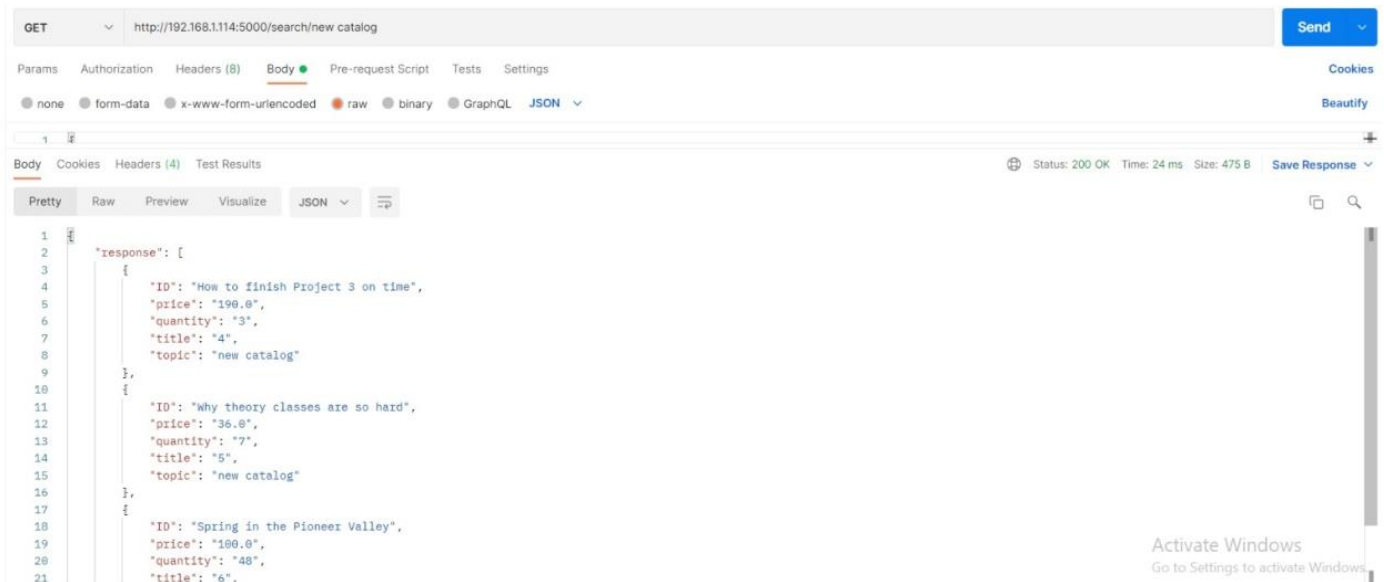
Search operation

Also we use cache for search operation. According to what we do to get information of a specific book, we do that again for search method. So, if the books of the topics in the cache, it will return the books that stored in the cache. But if not, it will send a request to the catalog server to get them and store them in the cache.

Starting without any caching the response time is 142ms.

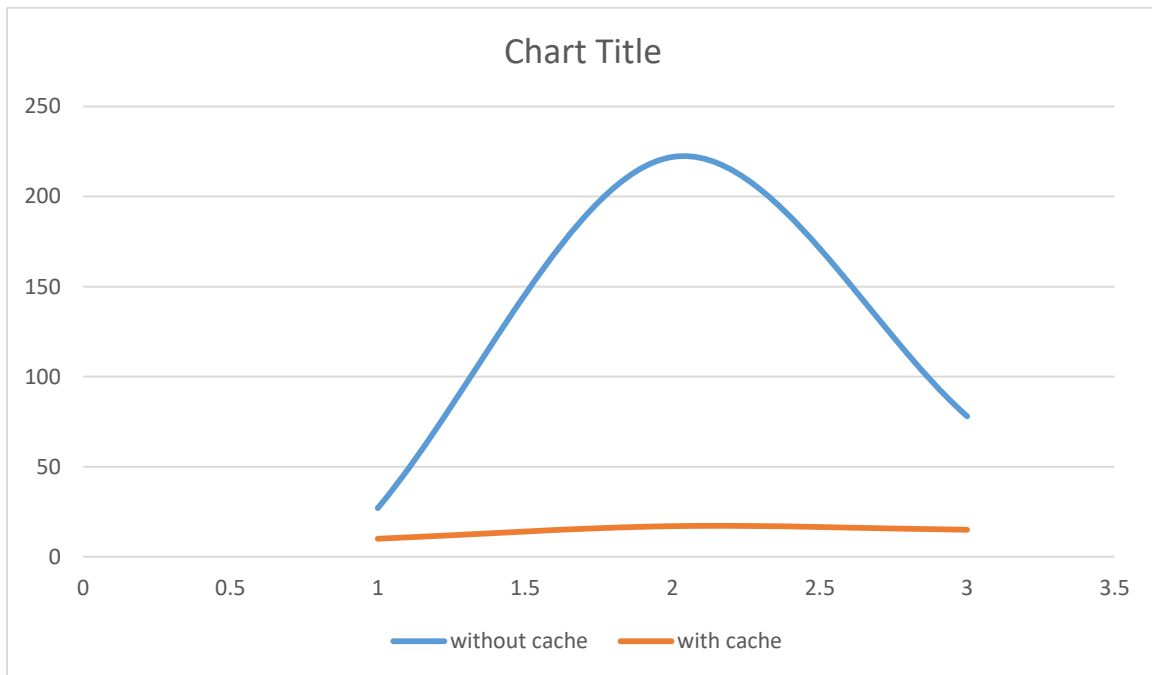


And with caching the time will be 24ms.



- ✚ 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?

| | Time without cache | Time with cache |
|---------|--------------------|-----------------|
| Trial 1 | 27ms | 10ms |
| Trial 2 | 222ms | 17ms |
| Trial 3 | 78ms | 15ms |



$$\begin{aligned}\text{Average response time without cache} &= (27+222+78)/3 \\ &= 109\text{ms}\end{aligned}$$

$$\begin{aligned}\text{Average response time with cache} &= (10+17+15)/3 \\ &= 14\text{ms}\end{aligned}$$

$$\text{Performance: } 109/14 = 7.78$$

- ✚ Construct a simple experiment that issues orders or catalog updates (i.e., database writes) to invalidate the cache and maintain cache consistency.

For update price operation, the time needed was 194ms because we need to send update request to the cache and update requests to the other catalog replicas, so it will spend a lot of time.

- ✚ What are the overhead of cache consistency operations?

Buy a book and update operation took long time, because it needs cache consistency.

- ✚ What is the latency of a subsequent request that sees a cache miss?

The latency increased when there is a cache miss, because it must fetch it from back servers which will cost a lot of time to do it.