Lab 2 & 1 final report

Student name: Aya Khaled Abu-Elhija.

Student ID: 11718412

Collage : IT

In this homework I have made a RESTful API , I had make many REST Requests through 2 different tools which are : browser & postman .

I recorded the responses I got in the attached images with explanations, which contains three web micro-serves. Every server resides on different machine.

I used **python** language by micro web framework **Flask**

But now I added three new books to the Bazar.com!

Because of the popularity of the book store I made rearchitecting to handle the higher in the workload ,and to improve request processing latency.

Replication:

I did a copy the (catalog server) to two new servers and also copied the (order server) to two new servers, so now I have 3 catalogs and 3 orders, but the front end server is not replicated, each one runs in a different port.

Consistency:

I implemented consistency by using http rest calls from the server which have a change in its database to the other two replicas.

Cache consistency needs to be addressed whenever a database entry is updated by buy requests or arrival of new stock of books, so I send invalidate request to the in-memory cache, which will cause the data for that item to be removed from the cache.

Load balance:

I used (round robin algorithm) by send each request from the front end server to one of the replicas by using 2 counters from 1 to 3 one for catalog server and the other one for order server. According to the counter value 1, 2 or 3 I send the request to a specific server. If catalog counter equal 1 I send the request to the first replica, but if it is 2 so I send the request to the second replica.
and because all the servers have the same capabilities I balanced the requests to 50% for each server.

How I run my Bazar.com?

I run all the servers each one on its machine (windows and VirtualBox Ubuntu system) on a different port of the other replica port and use the browser and Postman program as a client send request to the front end server, which will resend it to the needed server.

How does it is works?
1. Catalog server :

I ran 3 catalog servers on catalog server machine, each one on different port

3 replicas if catalog server

Catalog_1 on port 2000

```
 * Serving Flask app "catalog_1/catalogServer
.py" (lazy loading)
 * Environment: development
 * Debug mode: on
 * Running on http://0.0.0.0:2000/ (Press CTR
L+C to quit)
 * Restarting with stat
/home/aisha/.local/lib/python3.8/site-package
s/flask_marshmallow/__init__.py:26: UserWarni
ng: Flask-SQLAlchemy integration requires mar
shmallow-sqlalchemy to be installed.
  warnings.warn(
 * Debugger is active!
 * Debugger PIN: 349-933-185
/home/aisha/.local/lib/python3.8/site-package
s/flask_marshmallow/__init__.py:26: UserWarni
ng: Flask-SQLAlchemy integration requires mar
shmallow-sqlalchemy to be installed.
  warnings.warn(
[]
```

```
 * Serving Flask app "catalog_2/catalogServer.p
y" (lazy loading)
 * Environment: development
 * Debug mode: on
 * Running on http://0.0.0.0:3000/ (Press CTRL+
C to quit)
 * Restarting with stat
/home/aisha/.local/lib/python3.8/site-packages/
flask_marshmallow/__init__.py:26: UserWarning:
Flask-SQLAlchemy integration requires marshmall
ow-sqlalchemy to be installed.
  warnings.warn(
 * Debugger is active!
 * Debugger PIN: 349-933-185
/home/aisha/.local/lib/python3.8/site-packages/
flask_marshmallow/__init__.py:26: UserWarning:
Flask-SQLAlchemy integration requires marshmall
ow-sqlalchemy to be installed.
  warnings.warn(
```

```
 * Serving Flask app "catalog_3/catalogServer.
py" (lazy loading)
 * Environment: development
 * Debug mode: on
 * Running on http://0.0.0.0:4000/ (Press CTRL
+C to quit)
 * Restarting with stat
/home/aisha/.local/lib/python3.8/site-packages
/flask_marshmallow/__init__.py:26: UserWarning
: Flask-SQLAlchemy integration requires marshm
allow-sqlalchemy to be installed.
  warnings.warn(
 * Debugger is active!
 * Debugger PIN: 349-933-185
/home/aisha/.local/lib/python3.8/site-packages
/flask_marshmallow/__init__.py:26: UserWarning
: Flask-SQLAlchemy integration requires marshm
allow-sqlalchemy to be installed.
  warnings.warn(
```

2. **Order server** :

Ran 3 replicas of order server on their machine each one on a different port

Order_1 on port 2000

```
C:\Users\dell\Desktop\Lab2>set FLASK_APP=order_1/order.py

C:\Users\dell\Desktop\Lab2>set FLASK_ENV=development

C:\Users\dell\Desktop\Lab2>set FLASK_RUN_PORT=2000

C:\Users\dell\Desktop\Lab2>flask run --host=0.0.0.0
 * Serving Flask app 'order_1/order.py' (lazy loading)
 * Environment: development
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 878-000-355
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a produ
ction deployment.
 * Running on http://192.168.1.121:2000/ (Press CTRL+C to quit)
▯
```

Order_2 on port 3000

```
C:\Users\dell\Desktop\Lab2>set FLASK_RUN_PORT=3000

C:\Users\dell\Desktop\Lab2>flask run --host=0.0.0.0
 * Serving Flask app 'order_2/order.py' (lazy loading)
 * Environment: development
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 878-000-355
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a productio
n deployment.
 * Running on http://192.168.1.121:3000/ (Press CTRL+C to quit)
▯
```

Order_3 on port 6000

```
C:\Users\dell\Desktop\Lab2>set FLASK_RUN_PORT=6000

C:\Users\dell\Desktop\Lab2>flask run --host=0.0.0.0
 * Serving Flask app 'order_3/order.py' (lazy loading)
 * Environment: development
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 878-000-355
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a produc
tion deployment.
 * Running on http://192.168.1.121:6000/ (Press CTRL+C to quit)
```

3. **front-end server** :

Ran one front end server on port 5000

```
C:\Users\dell\Desktop\Lab2>set FLASK_RUN_PORT=5000

C:\Users\dell\Desktop\Lab2>flask run --host=0.0.0.0
 * Serving Flask app 'front-end/front-end server.py' (lazy loadin
g)
 * Environment: development
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 878-000-355
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a prod
uction deployment.
 * Running on http://192.168.1.121:5000/ (Press CTRL+C to quit)
```

Postman tests:

- I had add three new books to the store, so now I have 7 books

```
GET          ∨     http://192.168.1.121:5000/bazar/info/all

Params    Authorization    Headers (7)    Body    Pre-request Script    Tests    Settings

Body    Cookies    Headers (4)    Test Results                                    Status: 200 OK

Pretty    Raw    Preview    Visualize    JSON ∨    ⇄

  1  [
  2      {
  3          "id": 0,
  4          "price": 100.0,
  5          "quantity": 85,
  6          "title": "How to get a good grade in DOS in 40 minutes a day.",
  7          "topic": "distributed systems"
  8      },
  9      {
 10          "id": 1,
 11          "price": 22.35,
 12          "quantity": 60,
 13          "title": "RPCs for Noobs.",
 14          "topic": "distributed systems"
 15      },
 16      {
 17          "id": 2,
 18          "price": 50.0,
 19          "quantity": 8,
 20          "title": "Xen and the Art of Surviving Undergraduate School.",
 21          "topic": "undergraduate school"
 22      },
 23      {
 24          "id": 3,
 25          "price": 89.33,
 26          "quantity": 65,
 27          "title": "Cooking for the Impatient Undergrad.",
 28          "topic": "undergraduate school"
 29      },
 30      {
 31          "id": 4,
 32          "price": 70.0,
 33          "quantity": 40,
 34          "title": "How to finish Project 3 on time ",
 35          "topic": "new"
 36      },
 37      {
 38          "id": 5,
 39          "price": 100.0,
 40          "quantity": 80,
 41          "title": "Why theory classes are so hard ",
 42          "topic": "new"
 43      },
```

```
44 ∨    {
45          "id": 6,
46          "price": 700.0,
47          "quantity": 90,
48          "title": "Spring in the Pioneer Valley ",
49          "topic": "new"
50       }
51    ]
```

```
thorization    Headers (7)    Body    Pre-request Script    Tests    Settings

Cookies   Headers (4)   Test Results                          ⊕  Status: 200 OK   Time: 307 ms   Size: 296 B   Save Response

Pretty    Raw    Preview    Visualize    HTML ∨    ⇥                                    🗑 Q

1  {
      "price": 100.0,
      antity": 85,
           "How to get a good grade in DOS in 40 minutes a day.",
           ted systems"
```

At first, I started with empty cache, so when I sent request for "info/0" the front end server resent the request to the one of the catalog replicas because the cache is empty and didn't have this book information.

```
.0
 * Serving Flask app 'front-end/front-end server.
py' (lazy loading)
 * Environment: development
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 878-000-355
 * Running on all addresses.
   WARNING: This is a development server. Do not
use it in a production deployment.
 * Running on http://192.168.1.121:5000/ (Press C
TRL+C to quit)
{'price': 100.0, 'quantity': 85, 'title': 'How to
 get a good grade in DOS in 40 minutes a day.', '
topic': 'distributed systems'}
192.168.1.121 - - [14/Aug/2021 16:46:27] "GET /ba
zar/info/0 HTTP/1.1" 200 -
```

front-end server

the request sent to the catalog_1 to read the information of the book with id = 0

The response returned from the catalog_1 to the front end server then to the client and stored a copy of the information on the cache..
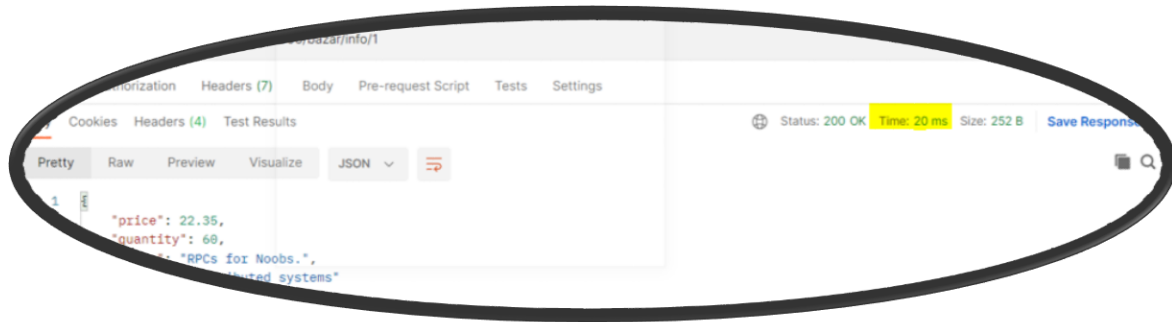
so when the client sent this request for the first time it needed 307 ms but when he sent it again it spend 32 ms, because the needed information stored in the front end cash, and the front end server returnd it without sending request to the catalog server.

[After cashing](#)
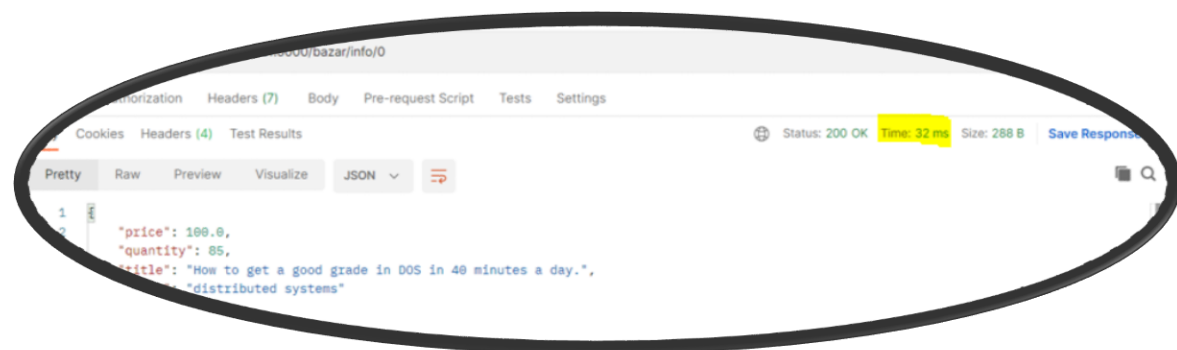
Test 2

As prevouse trial when I asked for information not found in the cache, front end server brought it from the catalog server, but here I noticed load balance, where the front end server sent the request to the second replica of the catalog server, stored its response in the cache for the next requests.

Without cache
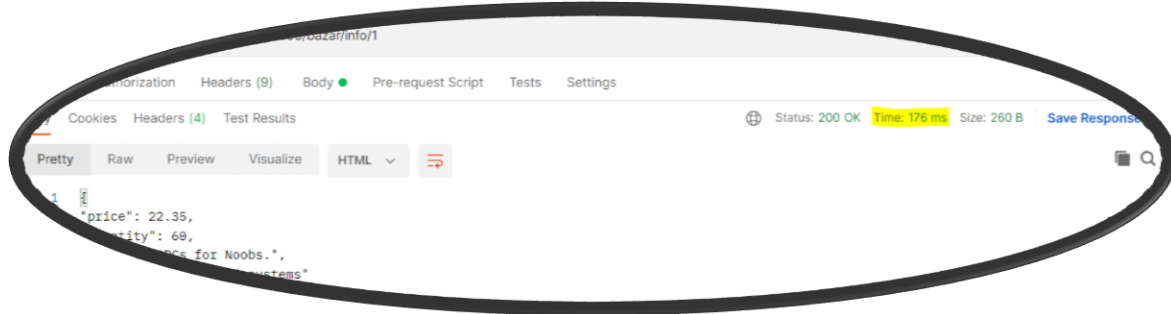
Front end server sent request to catalog_2


```
* Debugger is active!
* Debugger PIN: 878-000-355
* Running on all addresses.
  WARNING: This is a development server. Do not
use it in a production deployment.
* Running on http://192.168.1.121:5000/ (Press C
TRL+C to quit)
{'price': 100.0, 'quantity': 85, 'title': 'How to
 get a good grade in DOS in 40 minutes a day.', '
topic': 'distributed systems'}
192.168.1.121 - - [14/Aug/2021 16:46:27] "GET /ba
zar/info/0 HTTP/1.1" 200 -
192.168.1.121 - - [14/Aug/2021 16:47:55] "GET /ba
zar/info/0 HTTP/1.1" 200 -
{'price': 22.35, 'quantity': 60, 'title': 'RPCs f
or Noobs.', 'topic': 'distributed systems'}
192.168.1.121 - - [14/Aug/2021 16:49:07] "GET /ba
zar/info/1 HTTP/1.1" 200 -
```

The response brought from catalog_2 (load balance) and stored in the cache memory
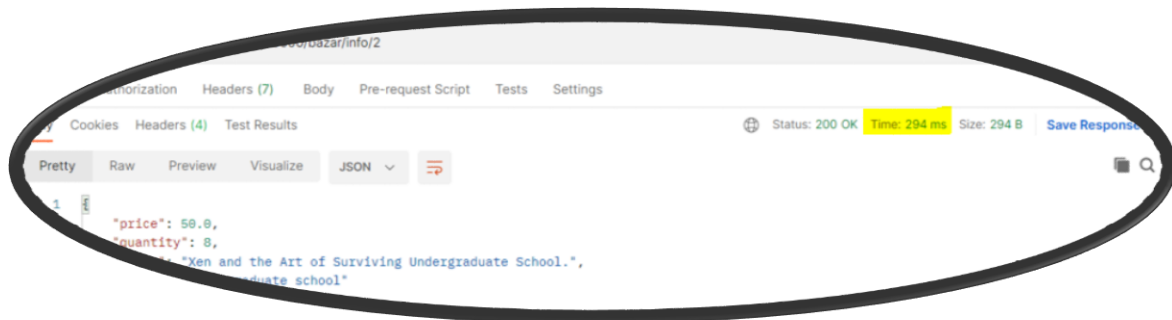
With cache

Test 3

Here the request sent from client to the front end server..

Then to catalog_3(load balance).

Response returned from the catalog to the front end server, stored in the cache and returned the response to the client..

The time was 294 ms.

Front end server

```
 * Running on http://192.168.1.121:5000/ (Press C
TRL+C to quit)
{'price': 100.0, 'quantity': 85, 'title': 'How to
 get a good grade in DOS in 40 minutes a day.', '
topic': 'distributed systems'}
192.168.1.121 - - [14/Aug/2021 16:46:27] "GET /ba
zar/info/0 HTTP/1.1" 200 -
192.168.1.121 - - [14/Aug/2021 16:47:55] "GET /ba
zar/info/0 HTTP/1.1" 200 -
{'price': 22.35, 'quantity': 60, 'title': 'RPCs f
or Noobs.', 'topic': 'distributed systems'}
192.168.1.121 - - [14/Aug/2021 16:49:07] "GET /ba
zar/info/1 HTTP/1.1" 200 -
{'price': 50.0, 'quantity': 8, 'title': 'Xen and
the Art of Surviving Undergraduate School.', 'top
ic': 'undergraduate school'}
192.168.1.121 - - [14/Aug/2021 16:51:25] "GET /ba
zar/info/2 HTTP/1.1" 200 -
```

Sent to catalog_3

```
 * Running on http://0.0.0.0:2000/ (Press CTR
L+C to quit)
 * Restarting with stat
/home/aisha/.local/lib/python3.8/site-package
s/flask_marshmallow/__init__.py:26: UserWarni
ng: Flask-SQLAlchemy integration requires mar
shmallow-sqlalchemy to be installed.
  warnings.warn(
 * Debugger is active!
 * Debugger PIN: 349-933-185
/home/aisha/.local/lib/python3.8/site-package
s/flask_marshmallow/__init__.py:26: UserWarni
ng: Flask-SQLAlchemy integration requires mar
shmallow-sqlalchemy to be installed.
  warnings.warn(
192.168.1.121 - - [14/Aug/2021 16:37:58] "GET
 /bazar/info/all HTTP/1.1" 200 -
192.168.1.121 - - [14/Aug/2021 16:46:27] "GET
 /bazar/info/0 HTTP/1.1" 200 -
```

```
 * Environment: development
 * Debug mode: on
 * Running on http://0.0.0.0:3000/ (Press CTRL+
C to quit)
 * Restarting with stat
/home/aisha/.local/lib/python3.8/site-packages/
flask_marshmallow/__init__.py:26: UserWarning:
Flask-SQLAlchemy integration requires marshmall
ow-sqlalchemy to be installed.
  warnings.warn(
 * Debugger is active!
 * Debugger PIN: 349-933-185
/home/aisha/.local/lib/python3.8/site-packages/
flask_marshmallow/__init__.py:26: UserWarning:
Flask-SQLAlchemy integration requires marshmall
ow-sqlalchemy to be installed.
  warnings.warn(
192.168.1.121 - - [14/Aug/2021 16:49:07] "GET /
bazar/info/1 HTTP/1.1" 200 -
```

```
 * Environment: development
 * Debug mode: on
 * Running on http://0.0.0.0:4000/ (Press CTRL
+C to quit)
 * Restarting with stat
/home/aisha/.local/lib/python3.8/site-packages
/flask_marshmallow/__init__.py:26: UserWarning
: Flask-SQLAlchemy integration requires marshm
allow-sqlalchemy to be installed.
  warnings.warn(
 * Debugger is active!
 * Debugger PIN: 349-933-185
/home/aisha/.local/lib/python3.8/site-packages
/flask_marshmallow/__init__.py:26: UserWarning
: Flask-SQLAlchemy integration requires marshm
allow-sqlalchemy to be installed.
  warnings.warn(
192.168.1.121 - - [14/Aug/2021 16:51:25] "GET
/bazar/info/2 HTTP/1.1" 200 -
```

Second request got the response from the cache with 21 ms.
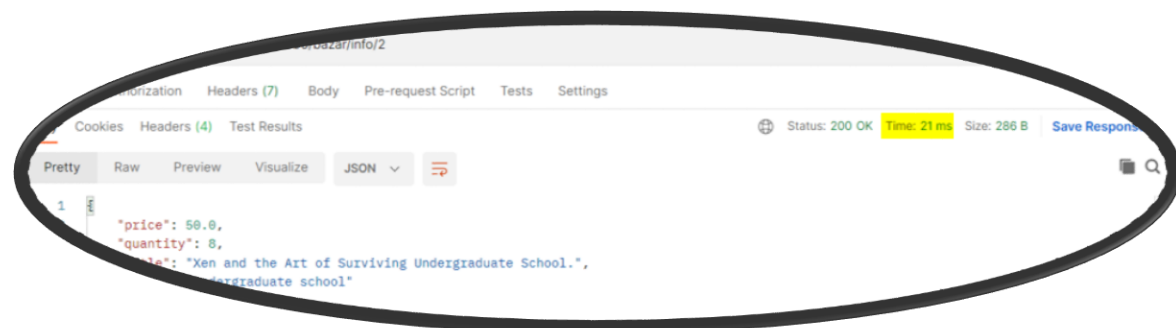
The cache has limited size, less than the catalog database size. So when the cache is full and I need to store new entry inside it I use LRU algorithm to replace it with one of the entries inside the cache.

Consistency

When I update information in the catalog database with "PUT" or "POST" url calls, the same update sent to the other replicas by using http rest calls, and I send invalidate request to the cache memory, which will cause to delete the information about the changed book.

Here I sent request to update price of book1 (write operation)
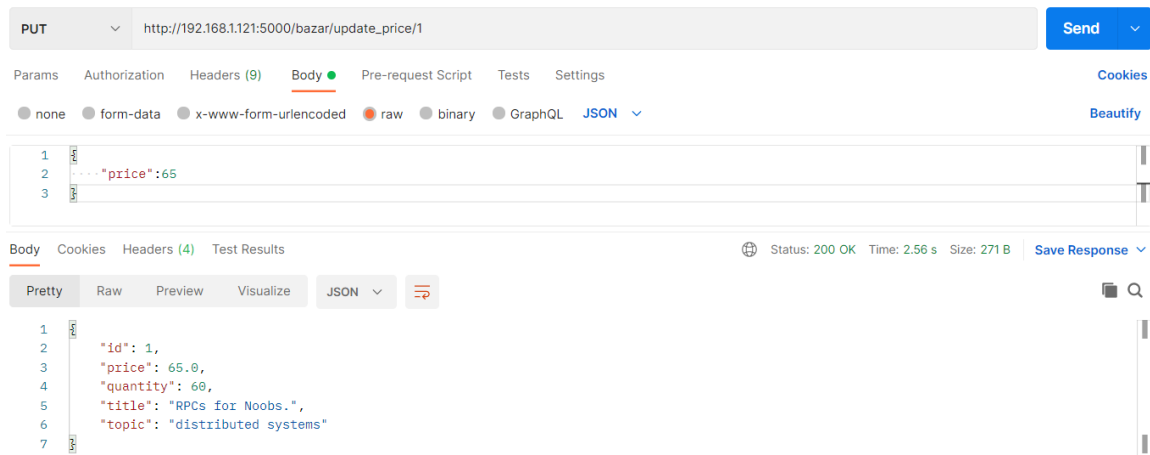
So I delete the entry of this book from the cache by sending invalidate request from the catalog to the cache

The request went to catalog_1, but then requests sent from it to the other replicas to update the same information on their databases.

Here if I got the book 1 information from any catalog server, it will be the same.

And the same if we did buy operation on order server, it will store the new order in the all replicas databases, and update book quantity on all the catalog replicas.

- 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
Test 1	307ms	32ms
T2	176ms	20ms
T3	294ms	21ms

عنوان المخطط
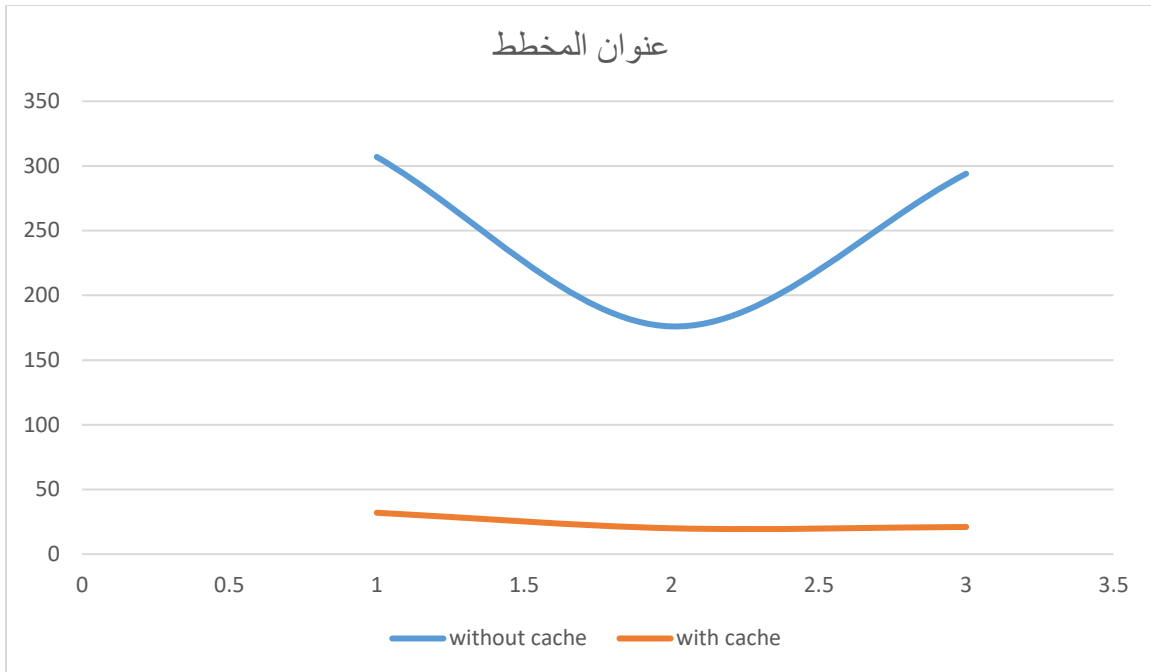
average response time without cache = (307+176+294)/3   = 259ms

average response time with cache = (32+20+21)/3 = 24.3ms

Performance :259/24.33 = 10.6 so the new performance 10.6* old performance

Cache increase decrease the latency time so I get the data faster.

Search operation

Also I used cache for search operation (read operation). For example, when I searched on a topic as a client and I got a response, returned the books from the catalog, and the books stored on the cache, so when I need to get information about one of them I got it from the cache memory with low latency. When the cache is full I will replace the new books with the least ordered book in the cache.

Starting with empty cache, we searched on "distributed systems" books, I get it from the catalog database and stored it in the cache database, it spent 127ms.

```
...121:5000/bazar/search/distributed%20systems

eaders (9)   Body ●   Pre-req.   Tests   Settings

                                    200 OK  127 ms  460 B   Save Re

etty    Raw      Preview    Visualize    JSON ∨    ⇥

1   [
2       {
3           "id": 0,
4           "price": 100.0,
5           "quantity": 85,
6           "title": "How to get a good grade in DOS in 40 minutes a day.",
7           "topic": "distributed systems"
8       },
        {
            "id": 1,
            "price": 22.35,
            "quantity": 60,
            "title": "RPCs for Noobs.",
            "": "distributed systems"
```
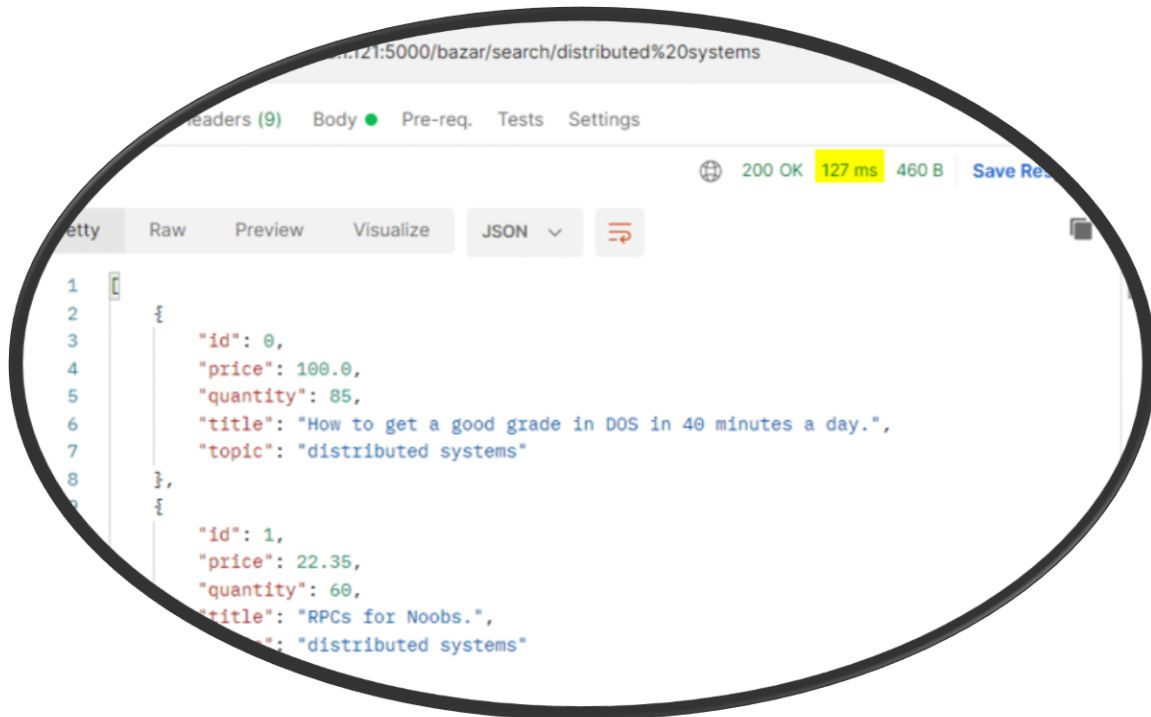
And we searched on "undergraduate school"

```
...121:5000/bazar/search/undergraduate%20school

eaders (9)   Body ●   Pre-req.   Tests   Settings

                                    200 OK  86 ms  480 B   Save Re

etty    Raw      Preview    Visualize    HTML ∨    ⇥

1   [
2   {
3   "id": 2,
4   "price": 50.0,
5   "quantity": 8,
6   "title": "Xen and the Art of Surviving Undergraduate School.",
7   "topic": "undergraduate school"
8   },
    {
    "id": 3,
    price": 89.33,
    ntity": 65,
        "Cooking for the Impatient Undergrad.",
        ergraduate school"
```
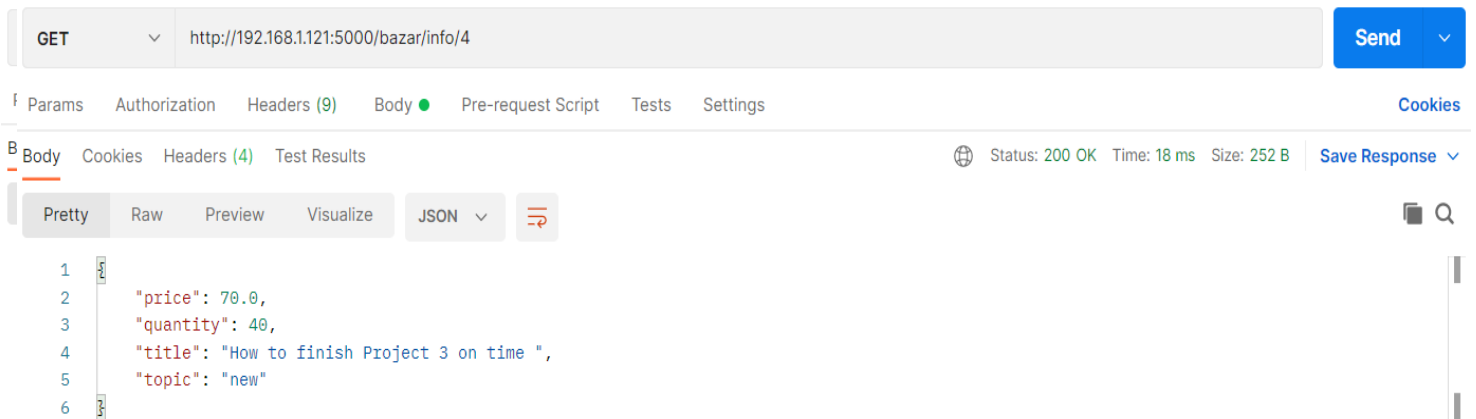
But the cache size is 5, and when we sent search requests of 2 different topics, cache now has 4 entries so there is a space for 1 book now and if I want to search of topic 3 "new" there are 3 new books come to the cache, so I had replaced the three new books with the least ordered ones.

So if I do requests for books 3 and 4 again their will ordered more than others on the cache.

| GET | ∨ | http://192.168.1.121:5000/bazar/info/4 | | Send | ∨ |

Params   Authorization   Headers (9)   Body ●   Pre-request Script   Tests   Settings   Cookies

Body   Cookies   Headers (4)   Test Results   ⊕ Status: 200 OK   Time: 18 ms   Size: 252 B   Save Response ∨

Pretty   Raw   Preview   Visualize   JSON ∨

```
1  {
2      "price": 70.0,
3      "quantity": 40,
4      "title": "How to finish Project 3 on time ",
5      "topic": "new"
6  }
```

Notice the low latency, because I got this information from the cache.

Books 0, 1 and 2 are least ordered so when I search of "new" topic book it will replace them with 0, 1 and 2 books on the cache.


```
3      "id": 4,
4      "price": 70.0,
5      "quantity": 40,
6      "title": "How to finish Project 3 on time ",
7      "topic": "new"
8  },
9  {
10     "id": 5,
11     "price": 100.0,
12     "quantity": 80,
       "title": "Why theory classes are so hard ",
       "topic": "new"
   },
   {
       "id": 6,
       "price": 700.0,
       "...ty": 90,
       ...ing in the Pioneer Valley ",
```

Status: 200 OK   Time: 215 ms   Size: 556

So now if I need book 0, 1 and 2 information the front end server will bring it for the client from the catalog server with low latency not from the cache!!

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

- In the previous example of consistency, the time I needed was 2.56s to finish the update price operation, because I need to send delete request to the cache and update requests to the other catalog replicas, so it will spend a lot of time.

- Buy or update operation took long time, because it needs cache consistency.

- The latency increased because of cache miss, for example when we got the data from the cache it spent 20ms, but when there is a cache miss it spent 176ms to finish the operation.

Dockerization

For each server, a Dockerfile file is included in order to be able to create docker images for each server. It is set up to use the latest alpine image as a base, install python and pip on it, expose the needed ports and set the start-up command for each server.

I downloaded the *memcached* docker image using this link :

https://hub.docker.com/_/memcached