An-Najah National University

Faculty of Engineering & Information
Technology Department of Computer Engineering

Distributed and Operating Systems

Lab 2: Turning Bazar to Amazon

Muna Awwad & Manar Arandi

System Design and How it works:

Our program consists of three machine first machine contain (frontend server) and postman, second machine (machine A) this machine contains (catalog and order server), third machine (machine B) contain (catalog and order servers).

the client send request to frontend server and determine the require verb based on operation (get, post), then frontend server send **GET request** to catalog server if the process is **info or search**, and send **PUT request** to order server if the process is **purchase.**

to perform purchase process, order server sends info request to catalog server in same machine to get amount related to the id specified on url request if amount not equal zero, then purchase operation succussing and recording in order database excel then the order server send update request to the catalog in the same machine, to decrease amount by one.
catalog server will send invalidate request to frontend server to delete item purchased from cache if exist, also send update2 request to catalog server in other machine to decrease amount from order database.

to perform info, search operation frontend server sends info or search request to catalog server and return the information about the item.

How did we achieve the Replication in system?

To deal with replication, the frontend server needs to implement a load balancing that takes each incoming request and sends it to one of the replicas we made it by declaring variable (catalogFlag ,orderFlag) so we do load balancing on a per-request basis.
we put order server, catalog server in machine A and the same servers we put in machine B.

How did we achieve consistency in the system?

In purchase request there is will be update on the book amount so to make the same change in different machine we send update2 request to the other machine.

Cache in the system:
We use caching in the python and integrated it into the front-end server process, when client makes a read (queries to catalog) request, the server first checks the cache if the data already exists in it, then the server returns the data to client from cache. else, the server requests the catalog server to take data from it and then store it in the cache. We add limitation on the number of items (4 items) in the cache, and when the cache is full, we replace older items with newer ones (LRU).

Cache consistency needs to be addressed whenever a database entry is updated by buy requests. Catalog server send invalidate requests to the in-memory in frontend server, the invalidate request causes the data for that item to be removed from the cache.

How to run the program

To run the program, we should run the machines and run postman program as a client to send the requests to these machines.
On the cmd terminal of each machine we run the server python file using
flask run–host=0.0.0.0
**adding --host=0.0.0.0 make the server publicly available which tells operating system to listen on all public Ips.
On each machine, we run the two servers (catalog and order) from different ports.
When the servers are running, send the request from the postman and use the correct method to get the result.

Note: we put all test requests and outputs as photos in result-img Folder in GitHub.

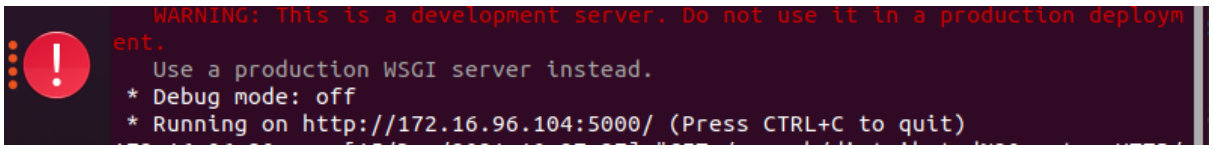
Frontend server run on ip: 172.16.96.30

```
D:\Manar\DOS\project\Dosproject2>flask run --host=0.0.0.0
 * Serving Flask app 'Frontend.py' (lazy loading)
 * 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 all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://172.16.96.30:5000/ (Press CTRL+C to quit)
```

catalog server on machineA run on ip: 172.16.96.99
to run order server in same machine we change the port

```
C:\Users\dell\Documents\Flaskapp\Dosproject2>python -m flask run --host=0.0.0.0
 * Serving Flask app 'catalog.py' (lazy loading)
 * 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 all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://172.16.96.99:5000/ (Press CTRL+C to quit)
```

catalog server run on machineB on ip : 172.16.96.104



Possible improvements and extensions to your program:

1.Insted of deleting item from cache we can edit it immediately or add it after delete it instead of waiting for client to request it again.

2.Deleting from cache after reach the limitation be by deleting the least popular item (the least requested item) instead of older one.

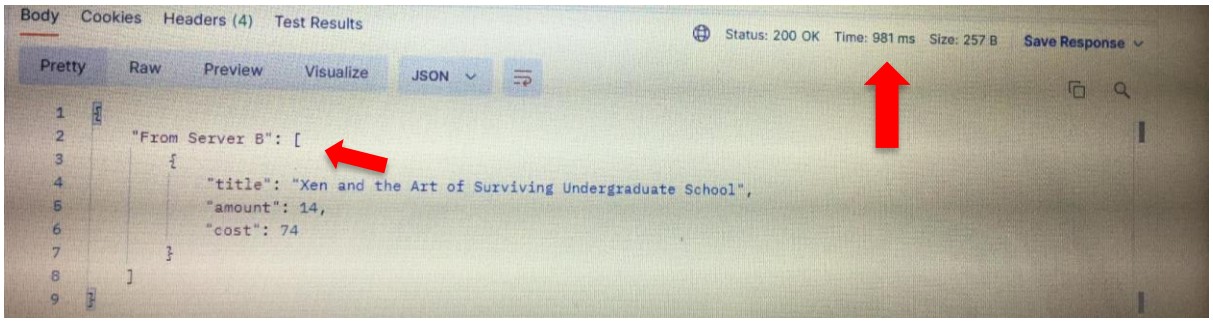
Constrains: no big constrain faced us.

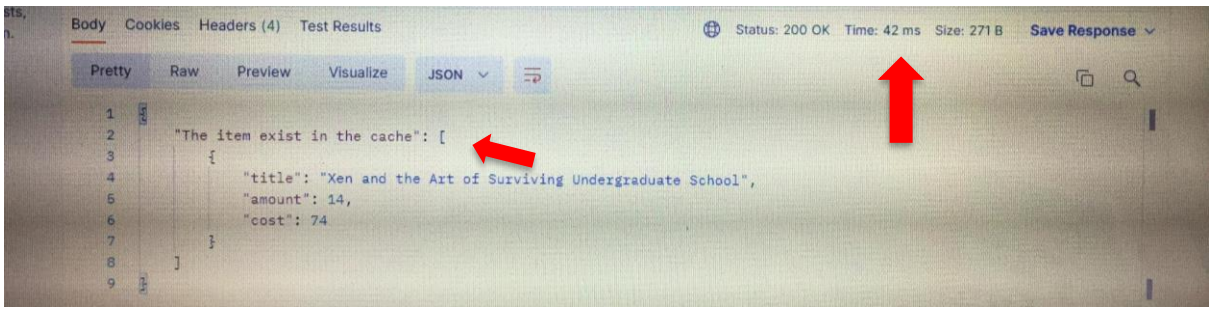
Experimental Evaluation and Measurements

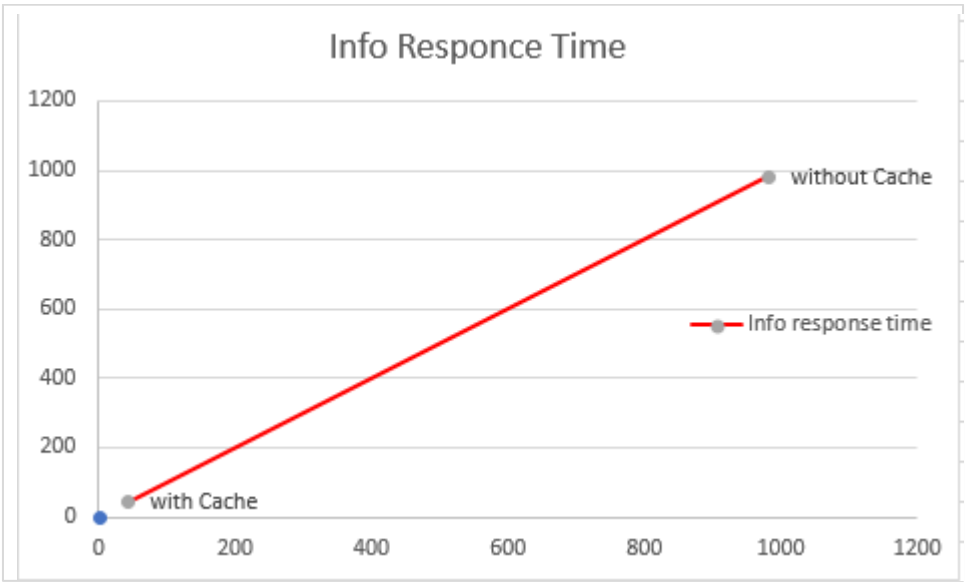
Q1: Compute the average response time (query/buy) of your new systems. What is the response time with and without caching? How much does caching help?

operation	without cache	with cache
info	981ms	42ms
search	1481ms	16ms

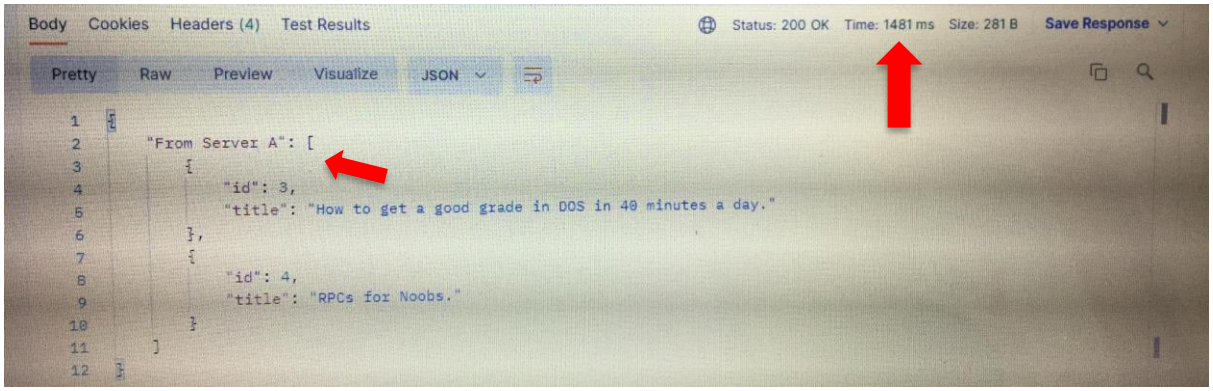
Info operation:

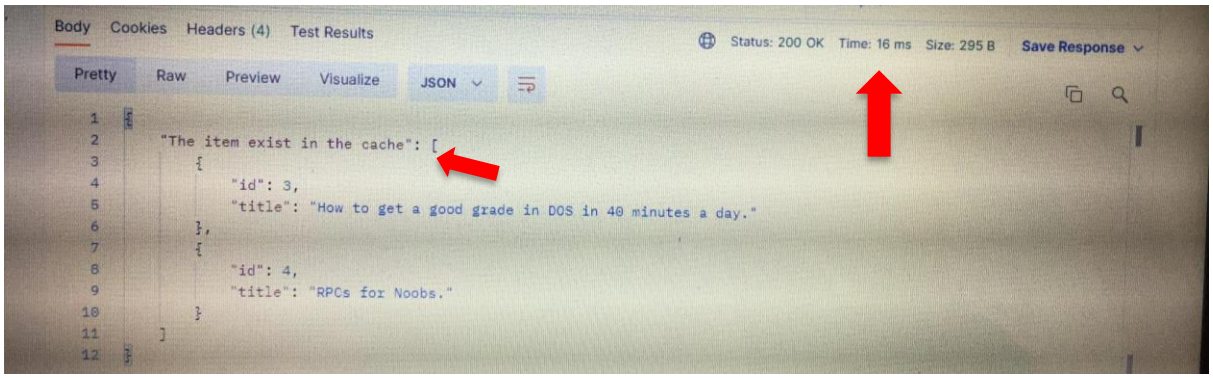






Search operation:





From the above result, caching speed up the operation, because it reduces the need to access servers.

Q2: What is the overhead of cache consistency operations?

Consistency required more response time, because purchase operation with cache consistency required more request, from order to catalog and then to frontend server to implement the consistency and delete this item from cache.

Without consistency	With consistency
0.982sec	2.01sec

Q3: What is the latency of a subsequent request that sees a cache miss?

In cache miss, the response time will be close to using server without cache.

GitHub Link: https://github.com/ManarArandy/Dosproject2.git