Computer Engineering Department

Distributed operating system

Bookstore store

Turning Bazar to Amazon

Raghad Ramez ElTiti

11715213

Roaa Yahya Arafat

11819584

Program Design and How it works

The program consists of 3 servers (front-end server, catalog server, order server) and 1 client, the front-end server and the client using postman are on the same machine which the host machine, and both the order and catalog server are replicated on multiple machines.

The client sends a request to the front-end server, then the front-end send the request to the catalog server if the operation that the client needs is search or info, or to the order server if the operation is purchase, then the catalog request the order server to check if this book is over from store and depending on that the order will send update request to decrement the quantity of this book and add the book to the orders data list. If the book is over the server send failed message to the frontend, then to the client.

In-memory cache

We added in memory cache in the front-end server, so when client makes a read request, the server first checks the cache if the data is stored on it, if cache hits, then the server returns the data to client. However, if cache misses, then 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 (6 items) in the cache, and when the cache is full, we replace the older ones with the newer ones.

Cache consistency

When the client makes a purchase request so updating the data of the books, consistency is a must, the replicas send invalidate requests to the front-end server (where the cache is implemented). The invalidate request causes the item to be removed from the cache.

Replication

When client needs a service whether it was on the catalog or the order server, it first checks the Cache as explained before, if the desired service couldn't be found in cache, it commits these services on the original or replicated servers with a flag.

Which resemble the simplest way od replicating, to choose a different server every time a service is desired.

Replication consistency

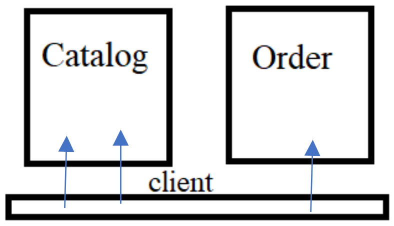
When the client makes a purchase request from one of the machines so updating the data of the books on this machine, the server send update request to the other machine to update its data.

The request is **get** request if the operation is search or info, but it is **put** request if the operation is purchase, and adding orders to the list is done internally.

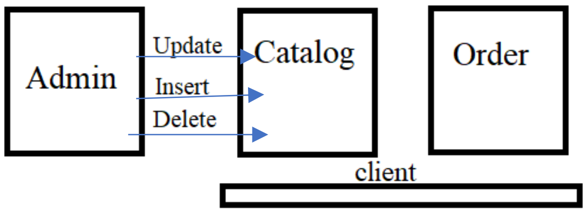
We used Python with Flask to implement the design. Also, we stored the catalog data and orders in csv files.

Possible improvements and extensions to your program

1. Make the client talks directly with the servers and no need to the front-end server to decrease the pressure on the pc, and speed up the operations.



2. Or we can replace it with admin server to change cost or make changes on data by sending requests to the catalog.



3. Implement cache consistency in a better way, to improve the response time, and speed up the operations.
4. Implement Replication better, by counting processes which are being executed in each server and check if the counter exceeded a specific number (threshold).

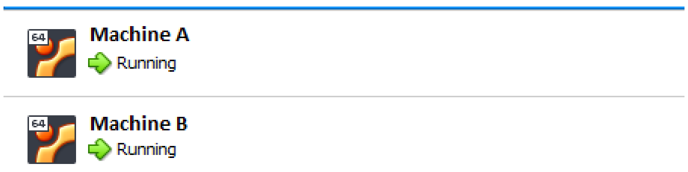
Design trade-offs considered and made.

Each request sent has a received response, when the client requests operation, the front-end server requests the servers, get response from them and return it to the client. Also the order send to the catalog request, get response from it and then returns it back to the front-end to return it to the client.

Adding cache needs cache consistency and make limitations on the number of items and also make replacement to the older items with the newer ones.

How to run the program

To run the program, we should run the two virtual machines and the host, and run postman program also on the host pc as a client to send the requests to these machines. Each machine has two replicas and data.

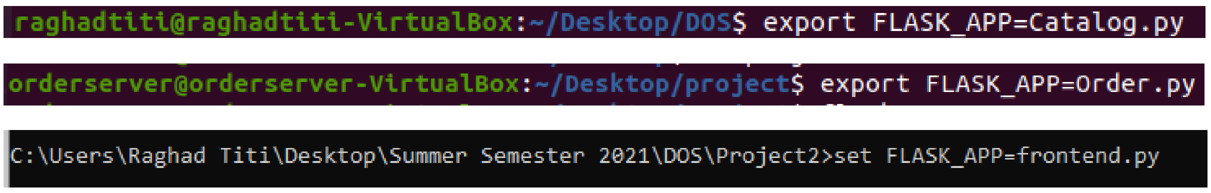

Machine A
Running
Machine B
Running

Each machine has an IP address, so by using the IP address starts with (192.168.1), with port 5000 and 5001 for the replicas we make the request with the suitable endpoint.

In my case the IPs are:

- 192.168.1.105: front-end server which the host
- 192.168.1.110: catalog server
- 192.168.1.109: order server

On the cmd terminal of each server we first export the application and then, run the server python file using flask run–host=0.0.0.0 to make it visible to all with port number.

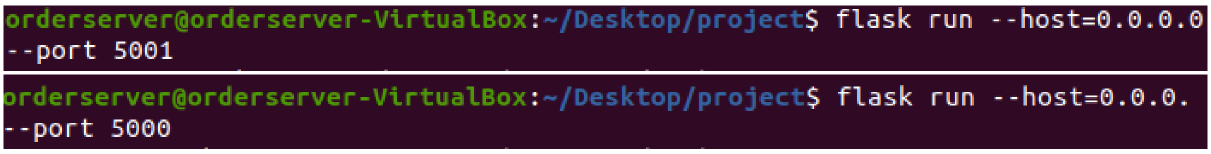


```
raghadtiti@raghadtiti-VirtualBox:~/Desktop/DOS$ export FLASK_APP=Catalog.py
```

```
orderserver@orderserver-VirtualBox:~/Desktop/project$ export FLASK_APP=Order.py
```

```
C:\Users\Raghad Titi\Desktop\Summer Semester 2021\DOS\Project2>set FLASK_APP=frontend.py
```

On each machine, we run the two servers (catalog and order) from different ports.



```
orderserver@orderserver-VirtualBox:~/Desktop/project$ flask run --host=0.0.0.0
--port 5001
```

```
orderserver@orderserver-VirtualBox:~/Desktop/project$ flask run --host=0.0.0.
--port 5000
```

When the servers are running, send the request from the postman and use the suitable method to get the needed json response, the output shown in the output folder.