Distributed Operating Systems

Final project - Part 1

Project Report

Student Name: Duaa Braik

Student ID: 11820272

Submitted to: Dr. Samer Arandi

Introduction:

This part of the project aims to implement a RESTful microservices for a bookstore in a distributed design, using a lightweight framework, so I used Flask/python framework with sqlite database to achieve the required work.

The design of the system mainly consists of 3 servers: frontend server, catalog server and order server, and is clearly shown in figure 1:

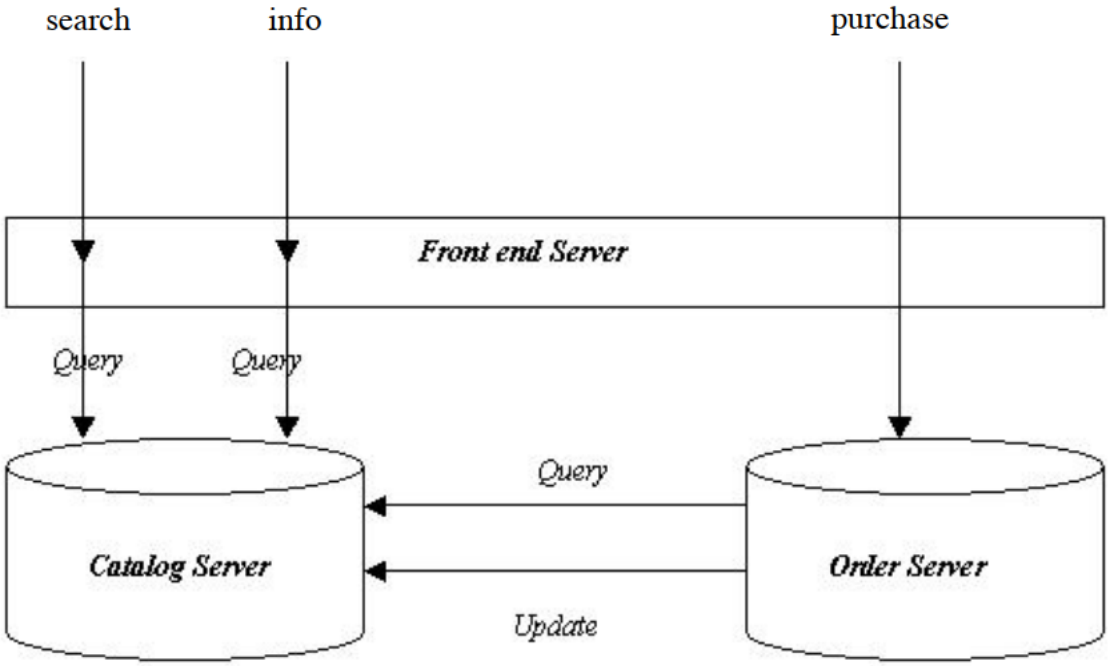

Figure 1

All requests are first sent to the frontend server, then it creates requests to the catalog server or order server according to the request sent from the client.
The client can do multiple operations: search, info and purchase.

1) **Search:** to search for a specific book or subject.

   The client types the following URL:

   http://IP_of_frontendserver/search/distributed%20systems

   The frontend server sends a GET request to the catalog server to make the following query:

   ```
   SELECT * FROM book where topic = (data sent from client)
   ```

   And return a response of the existing books, as shown in figure 2:

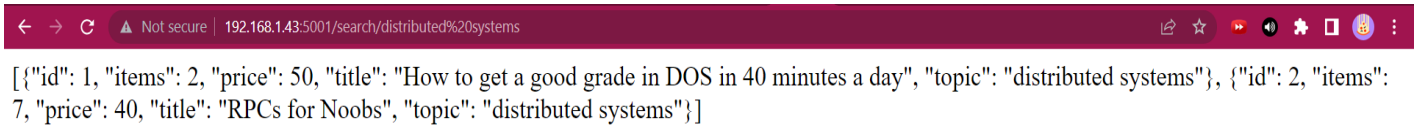

Figure 2

2) **Info:** the client specifies an item number to have detailed data about the book using the following URL:

   http://IP_of_frontendserver/info/1

   The frontend server sends a GET request to the catalog server to make the following query:

   ```
   SELECT * FROM book where id = (data sent from client)
   ```

And return a response of the existing books, as shown in figure 3:

Figure 3

3) **Purchase:** the client specifies an item number to order the book using the following URL:

http://IP_of_frontendserver/purchase/3

The frontend server sends a POST request to the order server, order server first makes sure that the book is available using a query to the catalog server with a GET request, then if the book is available it sends a post request to the catalog server to decrement the number of items remaining for this book using the following query:

```
UPDATE book SET items=items-1 WHERE id= (data sent from client)
```

And returns a response to the client as shown in figure 4:

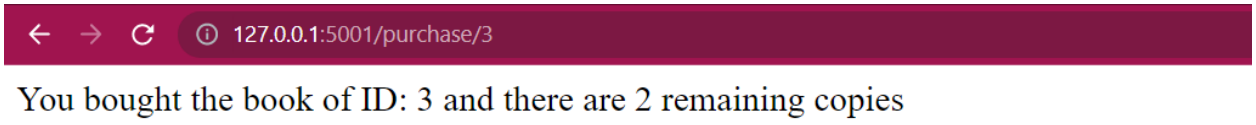

Figure 4

Design tradeoffs:

One of the trade-offs of this design is that there's some kind of overhead in the system because of the communication between 2 of the microservices (catalog and order), each time a request is sent to the order server requires 2 requests to the catalog server.

Possible improvements and extensions:

The system could be improved by having a copy of the database for the order server to implement some sort of decentralization and avoid single point of failure.

How to run the program:

Each microservice is set up and deployed on a separate virtual machine, and all VMs are assigned to be in the same host network to have the ability to communicate between each other. After setup is done, we use the host machine as client to send requests to the frontend server, the system uses GUI that enables the client to interact with and perform operations.