**Distributed Operating Systems**

Spring 2020

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

Student: **Raghad M. Abu Lebdeh**

Student No.: **11610455**

I have started building the design by creating 3 different VMs on Oracle VM VirtualBox. The codes were completely written using Java on eclipse IDE. The design depends on Spark dependencies for apache maven project as you can see in the pom.xml file for each server alongside other dependencies that was needed for the whole design to be done as **org.apache.poi** for managing excel (.xlsx) files which I have considered to be the DataBase that was Shared between the catalog and order server and my own computer (to share the folder a command line was used every time we open the VM for the backend servers which is *sudo mount -t vboxsf Shared ~/Desktop/SharedFolders*. As shared is the folder name on my computer and SharedFolders for the VMS). The DataBase were two excel files one named Books2 for the original servers and the other Books for the clone servers.

The process follows one sequence of events where a user starts by writing a request as HTTP request (*e.g.: http:// [IP Address]:[Port]/lookup/id*) and as we were asked to use a VMs the IP address is configuration on the private network from a DHCP server integrated into VirtualBox which will differ from the host address (*e.g.: 192.168.1.125*) and to know the IP for the VM, I have used *ifconfig* command line for the five VMs. Next, once the user hit enter, the HTTP request will be caught by the front-end server which will filter the requests and route them to the right back-end server. In the back-end servers, each server can access the database and do the request as it was programmed for it to do (more explaining will be below on how it’s done and how the consistency between the DBs is made)

The front-end server is the last server that we run once we want to run the whole system because we want to make sure that the ports numbers it has are the right ones so we assign the backend servers ports, run them then run the front end.

The code of the front-end server contains a Round-Robin algorithm for load balancing and in-memory-cache integrated.

The back-end server (catalog) which is responsible for lookup and search requests has a class for handling the management of the database (excel files) where it only reads as requested from the database.

The back-end server (order) is responsible for one request which is the buy request where we will write on the database (modify it). It’s also where we keep the consistency between the two databases as explained in the code.

To clearly run my program:

1- Please run the backend servers first after giving them available ports but before doing that make sure to change the path directory for the excel files in each four servers. (for the Catalog server, you need to change the path in the server code and the ExeleDataReader code).

2- Change the IP addresses and ports number for the back-end servers in the front end server code in Round Robin function (the first four are for the Catalog server and it’s clone, the below two for the Order server and it’) then give the front end server available port number then run it.

3- Use your browser or PostmanCanary to write the HTTP request as the example above.

Side note: as explained above, the IP addresses should be known in numbers. ifconfig is a the command used to know the current machine IP address.

**Performance results of your measurements/experiments:**

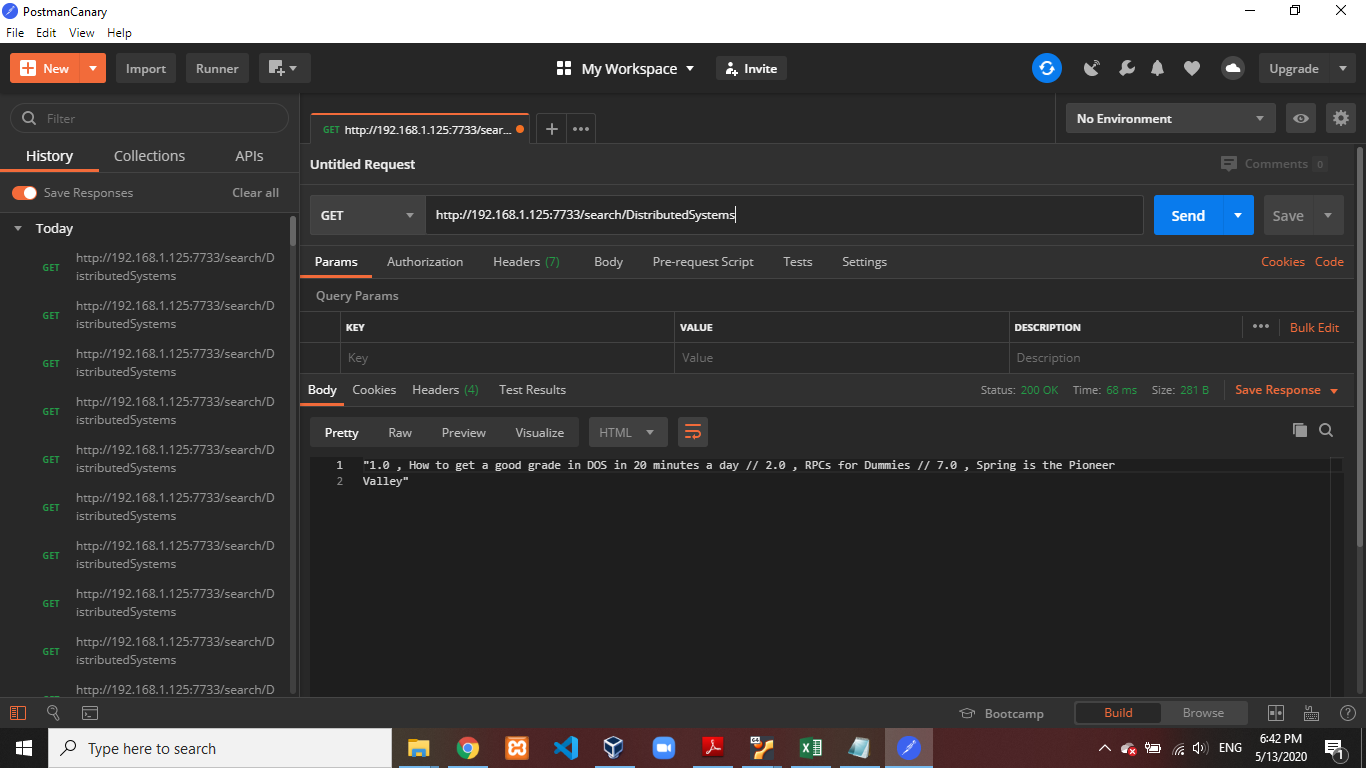
The following table will illustrate the response time of the requests.

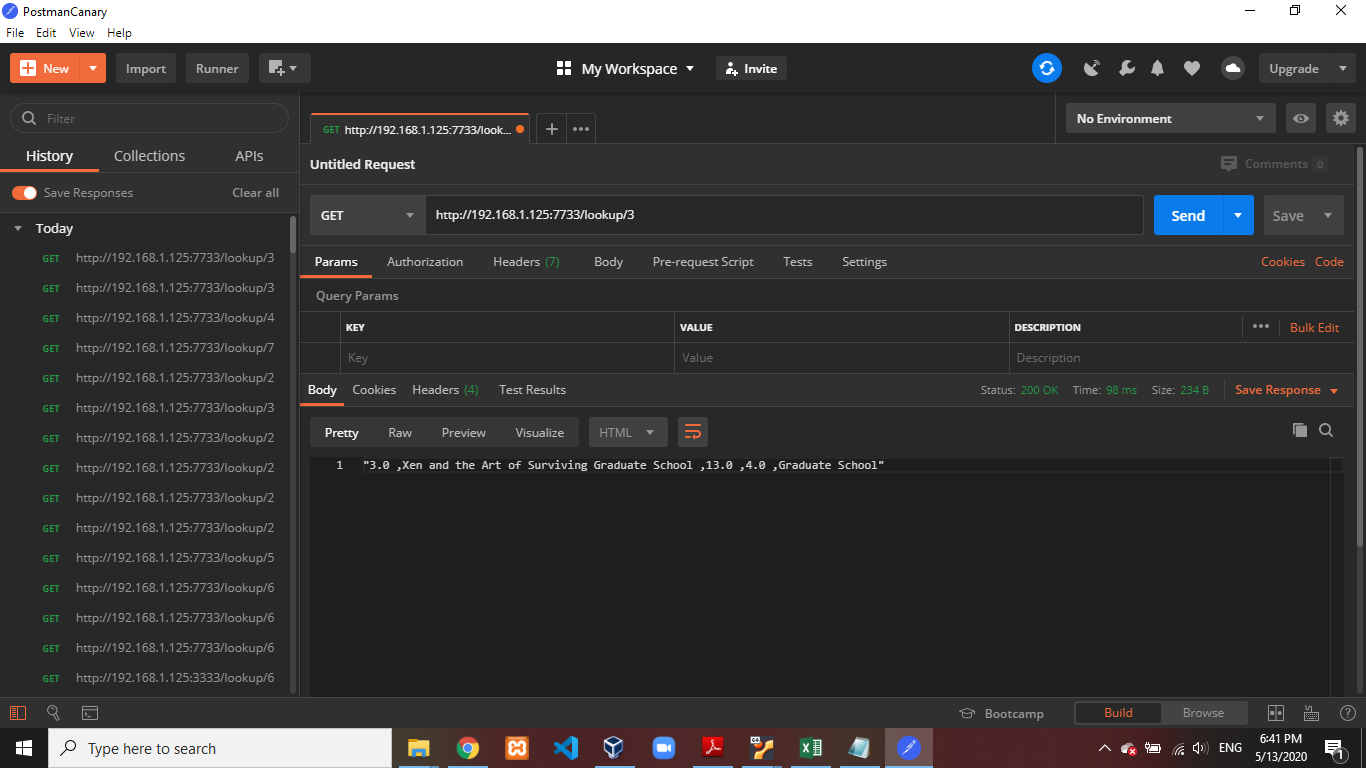
|  |  |  |  |
| --- | --- | --- | --- |
| Request type | Without Caching | With Caching ( request first time) | With Caching ( after a while) |
| Look up | 4.61 seconds | 3.30 seconds | 334 ms |
| Search | 190 ms | 180ms | 173ms |

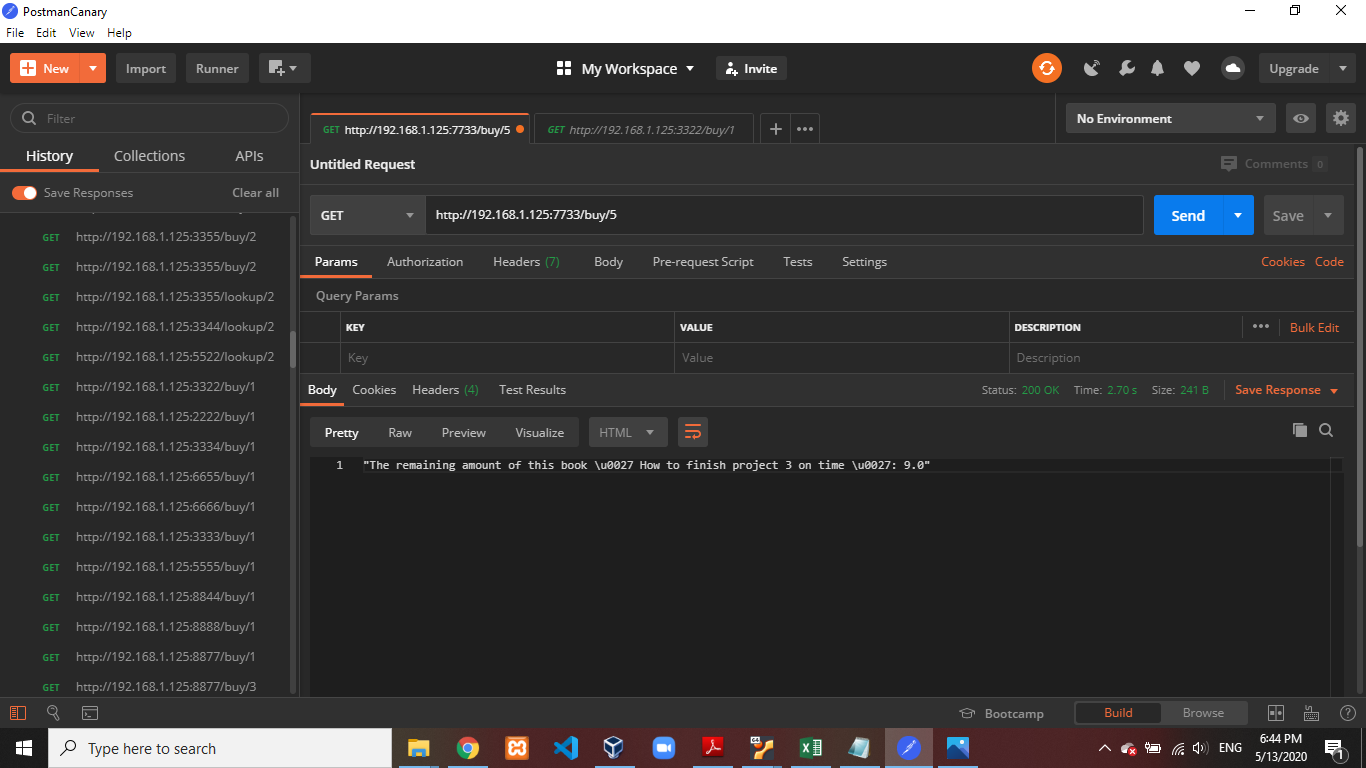
As we can see from the above table, caching will reduce the response time as it’s supposed to because caching will remove the process of routing the request to another server and waiting for the server to process the request and send back the requested data as the data will be saved in the front end server from the start and the front end server will give the requested data immediately.

Doing an updates or orders will automatically remove data from the cache (invalidates it by removing it) so we will not have an actual cache consistency price but we will have to write the data once again to cache if the cache is not full. if the cache is full and we cant write on it, the price will be as if there’s no caching at all for that request on that ID that was removed.

**Some Images of the output:**

****

****

****