CSE 5306 – Section 4, Fall 2023
Project Assignment 1

Building a Dockerized gRPC-Backed Image Search Engine

Due: Oct. 18th, 23:59 CT

Two students can form a team and turn in **one** submission.

Please add the following statement and sign it at the beginning of your report. I have neither given nor received unauthorized assistance on this work. I will not post the project description and the solution online.

Sign: **Date:**

1. Introduction

The purpose of this project assignment is to become familiar with Docker containers and gRPC. It helps you gain a strong understanding of these essential technologies in modern software development.

In particular, you will build a multi-threaded server and a client for a simple image search engine. You will use gRPC for communication between the server and client, and both will be containerized using Docker.

You can use **any** programming language to implement this project.

2. Background and Review

2.1 Search Engine

A search engine is a software application or online service that allows users to search for and retrieve information from the vast amount of data available on the internet or within a specific database.

As introduced in class, “applications can often be constructed from roughly three different pieces: a part that handles interaction with a user or some external application, a part that operates on a database or file system, and a middle part that generally contains the core functionality of the application” [1]. Those three parts are data-interface layer, data layer, and processing layer.

Figure 1 shows the simplified organization of an Internet search engine into traditional three application layers. If the engine is for housing, “someone can provide descriptors such as a city or region, a price range, the type of house, etc. The back end consists of a huge database of houses currently for sale. The processing layer does nothing else but transform the provided descriptors into a collection of database queries, retrieves the answers and post-processes these answers by, for example, ranking the output by relevance and subsequently generating an HTML page” [1].

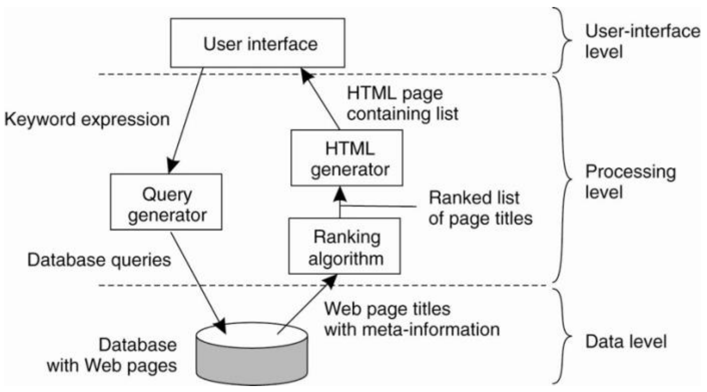

Figure 1. The simplified organization of an Internet search engine into traditional three application layers. [1].

Image Search Engine. An image search engine is a specialized type of search engine designed for users to search for and retrieve images on the Internet. Instead of relying on text-based queries, image search engines allow users to search for images or photos using visual cues, such as description of images or images themselves.

2.2 Multithreaded Server

“A server having more than one thread is known as Multithreaded Server. When a client sends the request, a thread is generated through which a user can communicate with the server. We need to generate multiple threads to accept multiple requests from multiple clients at the same time.” [2]

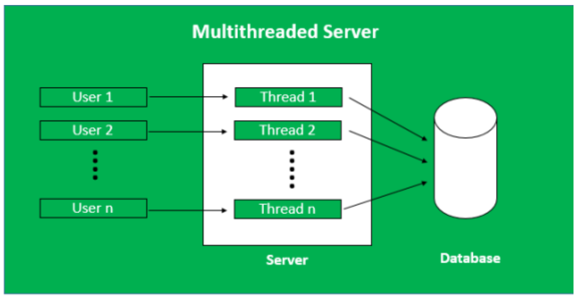

Figure 2. Simplified architecture of Multithreaded Server [2].

2.3 gRPC

gRPC stands for "Google Remote Procedure Call". It is an open-source remote procedure call (RPC) framework developed by Google. It is designed to facilitate efficient and high-performance communication between distributed systems.

It provides support for multiple programming languages, including C++, Python, Java, Go, Ruby, C#, Node.js, and more. This means you can write clients and servers in different languages that can communicate seamlessly.

Protocol Buffers. Protocol Buffers (Protobuf) is a language-neutral, platform-neutral extensible mechanism for serializing structured data. It allows you to define the structure of your messages and service methods in a language-agnostic way. gRPC uses Protobuf as its interface definition language (IDL) to define data structures and services.

You can learn more about Protobuf on its website: https://protobuf.dev/overview/.

3. Assignment

Q0. (bonus: 10pts) Create a project timeline with your teammate and email it to the TA, Mr. Ali Amir Imran (aliamir.imran@uta.edu) by **Sep. 20th**. Send a concise weekly email to the TA, providing updates on progress and any adjustments to the timeline, **every Wednesday**. Mention your team Id in the email subject, e.g., “CSE 5306, Team 0”.

Q1. (10 pts) Select **two** programming languages and follow their language-specific guides (https://docs.docker.com/language/) to get yourself familiar with how to containerize an application using Docker.

- Python: https://docs.docker.com/language/python/
- Java: https://docs.docker.com/language/java/
- Golang: https://docs.docker.com/language/golang/

For each language, you should complete at least three tasks outlined in the guide, i.e., “Build images”, “Run containers”, and “Deploy your app”. You should capture screenshots for every step (commands and corresponding output) specified in the guide and include them in the report.

Please preface each command with the “whoami;” command to specify the student performing the action. For example, when building the docker image, you should run the command:

```
$ whoami; docker build --tag python-docker .
```

Q2. (10 pts) Select two programming languages and follow their language-specific guides (https://grpc.io/docs/languages/) to get yourself quickly started with gRPC.

- Python: https://grpc.io/docs/languages/python/quickstart/
- Java: https://grpc.io/docs/languages/java/quickstart/
- Golang: https://grpc.io/docs/languages/go/quickstart/

For each language, you should complete all the tasks outlined in the guide. You should capture screenshots for every step (commands and corresponding output) specified in the guide and include them in the report.

Please preface each command with the "whoami;" command to specify the student performing the action.

Q3. (20 pts) Implement a **multi-threaded server** for the image search engine and containerize it using Docker.

In this project assignment, **the server only needs to randomly return one image that contains the object of the object class given by the user.** Specifically, your image search engine only needs to support some basic functionalities shown in Figure 1:

1. Users are allowed to search only with the object class as the keyword, e.g., "dog".
2. The ranking algorithm in the search engine randomly selects a single image that matches the keyword in the "database" and sends it back to the client.
3. The HTML generator is not required. Instead, the server directly sends the selected image to the client.

For the "database", instead of storing images in a traditional database, you can create a main directory with multiple sub-directories, each for a specific object class. You can name each sub-directory using its corresponding object class.

Remember to containerize your multi-threaded server, following the guide in Q1.

Q4. (20 pts) Implement communication between the server and client using gRPC and containerize them using Docker.

You should build on top of Q3. The general steps involved are as follows.

First, define your services and message types using Protobuf by writing a .proto file that specifies the service methods and the structure of input and output messages. The message sent from the client to the server is the keyword. The message sent from the server to the client is the image.

Second, use the Protobuf compiler, protoc, to compile your .proto file into language-specific code. This will generate code for client and server stubs as well as message classes in the programming language of your choice.

Third, write the server and client code using the service methods defined in the .proto file.

Remember to containerize your multi-threaded server and client separately.

Q5. (5 pts) Test your server-client implementation.

You should design 5 different test cases and document them in the final report.

4. Turning in Your Solution

You should make sure that your deliverables include source code of client-side and server-side programs, a README file, and a report. You should put all the required files into a zipped folder and upload it to Canvas.

In the README file (5 pts), you should include explains:

- How to compile and run your program;
- Anything unusual about your solution that the TA should know;
- **Any external sources referenced while working on your solution.**

In the report, you should make sure you clearly list your names and student IDs. You should clearly mention which student worked on which part of the project.

NOTE: The late penalty is 20% per day.

References

[1] Andrew S. Tanenbaum and Maarten Van Steen, Distributed Systems: Principles and Paradigms (4th Edition)

[2] https://www.geeksforgeeks.org/multithreaded-servers-in-java/