Evaluating Serverless Machine Learning Performance on Google Cloud Run

Prerana Khatiwada, Pranjal Dhakal
Preranak@udel.edu, dpranjal@udel.edu

Summary

Through this project, we aim to understand microservice architecture, which is an increasingly popular approach to software development. We aim to learn about containerization using Dockers and the different features offered by the Google Cloud Platform. In this project, we will create a machine learning Python Flask application for image classification, containerize it, and deploy the containerized application on the Google Cloud Platform server. We will expose endpoints that the client applications can use to send requests. We will stress test the machine learning application and determine metrics like response latency, cold start time, memory and CPU utilization, and crash frequency. We will also understand the auto-scaling feature offered by cloud-run.

Keywords: Microservice, Flask, Cloud, Deploy, Machine Learning

Introduction

Serverless computing platforms eliminate and minimize the need for expensive onsite hardware, software, and storage infrastructure. These platforms run containerized applications, which are very cost-effective. Therefore, it's important to know how to package such applications as containers and to be familiar with some of the technical terms we'll come across while working on this project. This section provides a more detailed description of the terms that are mentioned in the summary and are discussed further in the rest of the section. A container is an executable entity consisting of software code along with the operating system libraries and dependencies required to run the code. Containerization is the process of creating containers [1]. Containers are portable and are extensively used for developing modern cloud applications. The host machine (server) where the containers are executed has an open-source runtime engine like the Docker runtime engine installed on its operating system [2]. The host's operating system services are shared by the containers. Docker uses resource isolation in the OS kernel to run multiple containers on the same OS, and these containers are well-suited to microservices, which is the traditional approach known as the monolith architecture of software development that involves developing solutions as one large system [3]. The components in the monolith architecture are tightly coupled. So, making minor changes is a very involved and lengthy process. The microservices architecture, in contrast, divides the application into smaller functional units, making them loosely coupled. These functional units or services are developed, containerized, and deployed independently. If one of the services requires scaling, it can easily be done without scaling the entire application, leading to efficient resource utilization.

Microservices communicate with each other using REST Application Programming Interfaces (APIs) [4]. REST stands for Representational State Transfer. REST APIs use HTTP requests and can be developed using any programming language. To develop and deploy highly scalable containerized applications on a serverless platform [5], we will use Google Cloud Run, a service provided by Google. Developers can build and run applications without having to manage servers. It performs automatic scaling depending on the request traffic. During the idle period, it even scales down to 0 and only charges for the resources we have used. It also provides logging and monitoring features.

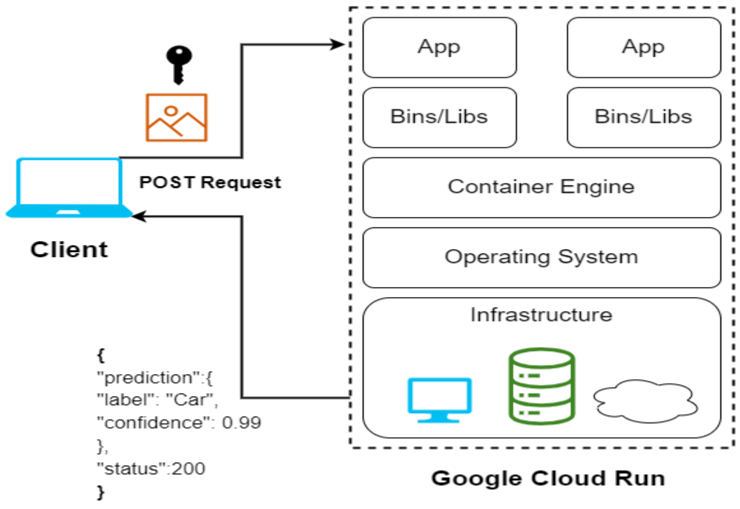
Related Work

"Cloud services" are gradually becoming an important component for businesses of all sizes [6]. This allows them to store data on a third-party cloud service like Microsoft Azure or Amazon Web Services. Microservices are a relatively recent concept in the software development world. James Lewis first proposed the notion at a conference in San Francisco in 2012 [7]. Netflix was one of the first companies to use the concept on a big scale, referring to it as "fine-grained service-oriented architecture" [8]. Microservices are built on these earlier techniques, and some argue that microservices and service-oriented architecture are interchangeable terms. On the other hand, microservices are smaller and do not share data storage [9].

Proposed work

In this project, we will write a Flask application that will take care of the server-side processing. Flask is a web microframework developed in Python. Our application will provide REST APIs that the client devices can use to send requests with proper authentication. This app will authenticate the request and, if valid, accept images from the user. The app will run a TensorFlow MobileNetV2 Image Classification model [10]. This model can classify images into one of the 1000 categories.

The Flask application is then containerized using Docker. Deploying a Flask application with Docker will allow us to replicate the application across different servers with minimal reconfiguration.



```
{
"prediction":{
"label": "Car",
"confidence": 0.99
},
"status":200
}
```

Figure 1: Proposed Model

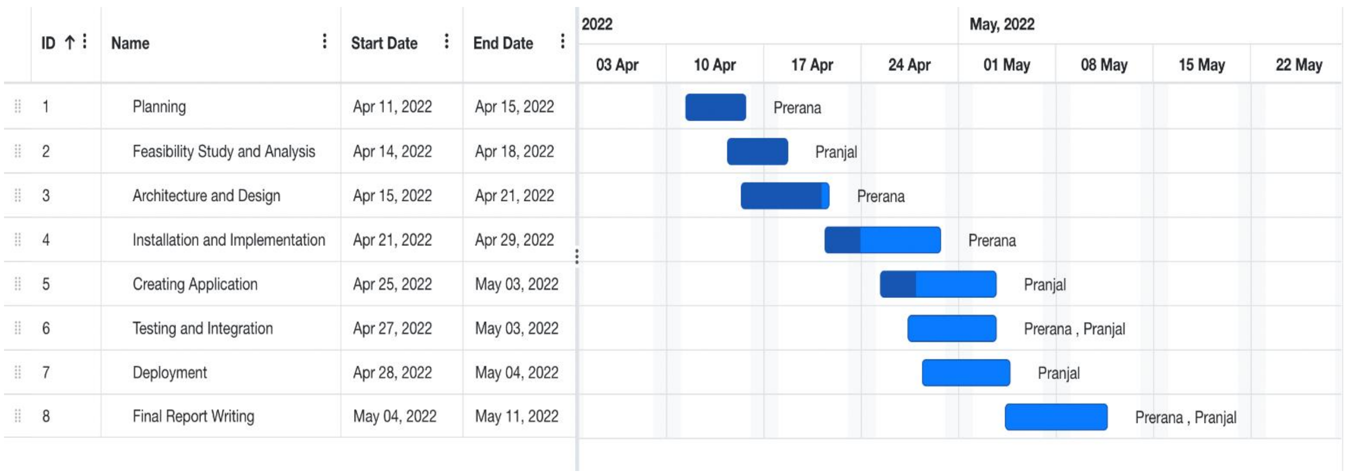
The containerized application will then be deployed on Google's Cloud Run platform. In this project, we intend to see and analyze how Google Cloud Run makes app development simpler and faster. We want to develop and deploy highly scalable containerized applications on a fully managed serverless platform. The software and packages used for this project are Python, Locust (https://locust.io/), Docker, Docker Hub, and Google Cloud.

Evaluation

To simulate different load scenarios and stress test the server, we will use the Locust API. Locust is a scriptable and scalable performance testing tool written in Python. We will generate metrics like response latency, cold start time, memory and CPU utilization, and crash frequency. We will be learning how the Cloud Run platform performs load balancing and handles scaling.

Project Timeline

The following is the timeline for our project.



ID	Name	Start Date	End Date	Assigned
1	Planning	Apr 11, 2022	Apr 15, 2022	Prerana
2	Feasibility Study and Analysis	Apr 14, 2022	Apr 18, 2022	Pranjal
3	Architecture and Design	Apr 15, 2022	Apr 21, 2022	Prerana
4	Installation and Implementation	Apr 21, 2022	Apr 29, 2022	Prerana
5	Creating Application	Apr 25, 2022	May 03, 2022	Pranjal
6	Testing and Integration	Apr 27, 2022	May 03, 2022	Prerana , Pranjal
7	Deployment	Apr 28, 2022	May 04, 2022	Pranjal
8	Final Report Writing	May 04, 2022	May 11, 2022	Prerana , Pranjal

Team Composition and Responsibilities

1. Pranjal Dhakal, *1st Year Ph.D. Student (CS),* **Skills:** *Python, Flask, Docker, Kubernetes*
2. Prerana Khatiwada, *1st Year Ph.D. Student (CS),* **Skills***: Python, Flask, Docker, Git*

Responsibilities

- Feasibility Study and Project Specifications - Prerana Khatiwada
- Architecture and Planning - Pranjal Dhakal
- Install Docker and necessary Packages - Prerana Khatiwada
- Creating a Flask Python Application - Pranjal Dhakal
- Creating a Docker file to Dockerize the Application - Prerana Khatiwada
- Deployed app on Google's Cloud Run platform - Pranjal Dhakal
- Testing and troubleshooting – Both

References

[1] By: IBM Cloud Education. (n.d.). *Containerization*. IBM. Retrieved April 22, 2022, from https://www.ibm.com/cloud/learn/containerization

[2] Bigelow, S. J., & Courtemanche, M. (2020, March 17). *What is Docker and how does it work?* SearchITOperations. Retrieved April 22, 2022, from https://www.techtarget.com/searchitoperations/definition/Docker

[3] Google. (n.d.). *Microservices architecture on Google Cloud | Google Cloud blog*. Google. Retrieved April 22, 2022, from https://cloud.google.com/blog/topics/developers-practitioners/microservices-architecture-google-cloud

[4] By: IBM Cloud Education. (n.d.). *Rest-apis*. IBM. Retrieved April 22, 2022, from https://www.ibm.com/cloud/learn/rest-apis

[5] Google. (n.d.). *Cloud run: Container to production in seconds | google cloud*. Google. Retrieved April 22, 2022, from https://cloud.google.com/run

[6] A. Islam, M. Irfan, K. Mohiuddin, and H. Al-Kabashi, "Cloud: The Global Transformation," in 2013 International Conference on Cloud & Ubiquitous Computing & Emerging Technologies, 2013.

[7] James Lewis, "Micro Services: Java, the Unix Way," 2012. [Online]. Available: https://www.infoq.com/presentations/Micro-Services

[8] Rick Osowski, "Introduction to microservices IBM Developer," 2017. [Online]. Available: https://developer.ibm.com/tutorials/cl-ibm-cloud-microservices-in-action-part-1-trs/ #netflix-streaming-the-birth-of-popularized-microservices

[9] Z. Xiao, I. Wijegunaratne, and X. Qiang, "Reflections on SOA and Microservices," in 2016 4th International Conference on Enterprise Systems (ES), 2016.

[10] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). MobileNetV2: Inverted residuals and linear bottlenecks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/cvpr.2018.00474