No Brains ML

Rohan Jayaram
Thejas Kiran
Vinay Kumar

INTRODUCTION

The **goal** of this project is to automate the iterative tasks involved in creating machine learning models using distributed systems for multiple services like pre-processing, training, and forecasting results. Our proposed application – 'No Brains ML' will accept input data sets from the users, and then utilize those data sets to build relevant models and provide the best performing model in the output to the user.

Traditional machine learning methodologies involve a variety of time-consuming procedures, such as data cleaning, pre-preprocessing, training, and evaluating several models. Our suggested concept is intriguing since we intend to streamline this method to automate these procedures and give the user a trained model that may subsequently be optimized to meet their demands. Additionally, I would like to explore different cloud platforms like Google BigTable, Google Cloud Storage, Redis, and different orchestration software like Docker and Kubernetes because I believe that working with these technologies will prove beneficial to me while employing MLOps techniques.

SCOPE OF THE PROJECT

Our suggested solution will make extensive use of the concepts and technologies discussed in class. In addition, we will study how these technologies might be applied to more cohesive products. Our proposed solution is ambitious, as we will be leveraging six different technologies to create a unified product which is not only capable of processing data and running several models, but also capable of providing the user with the best model.

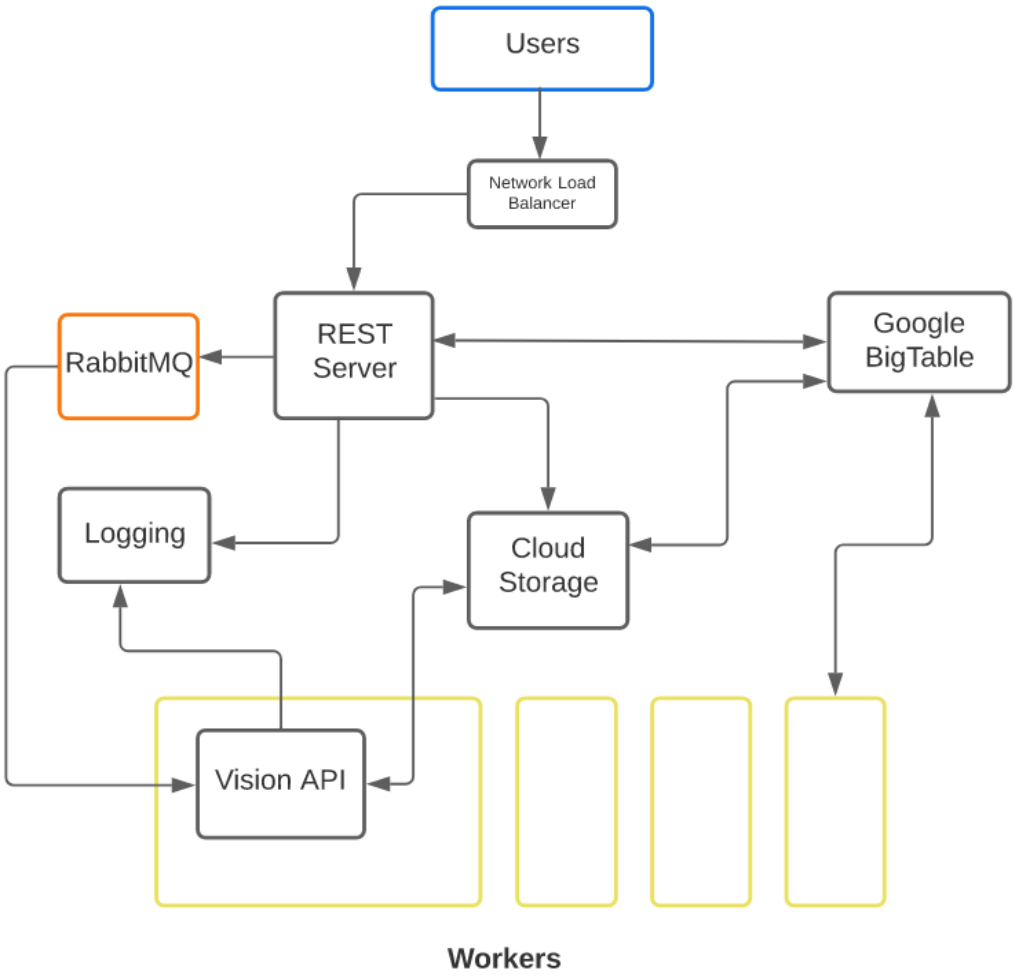
TECHNICAL AMMUNITION

To achieve our goal, we are planning to use the technologies mentioned below to build our application:

- Google BigTable
- RabbitMQ
- Flask

- Kubernetes
- Google Cloud Storage
- REST API
- Message Marshalling/encoding

ARCHITECTURE



INTERACTION

Our implementation design would require us to have a way to distribute user requests by sending them to a network load balancer. Our application can train on user data, provide predictions, and maintain a history of all prior user activities and requests. This is done by

sending these user requests to the REST Server. For user authentication, we plan to validate credentials against the ones stored in the Google BigTable, which is a NoSQL database. Upon authentication, a user will be able to upload files which will be stored and accessed from Google Cloud Storage; the user requests will be relayed as messages to our backend servers using RabbitMQ. A database will be set up to store intermediate operation results on the user data before the best model parameters are presented to the user.

We are planning to use the Flask Framework to facilitate the handling of API requests to our backend server. Finally, to manage, scale, and maintain the software the entire application will be containerized and orchestrated on the Kubernetes engine.

DEBUG MECHANISM

To ensure that our application runs smoothly and is working as expected, we are planning to debug it using below mentioned mechanisms -

- We are planning to connect our application to WebSocket via Postman to ensure that all requests are correctly forwarded.
- Test several plugins to enrich our application and at the same time better the authentication.
- Check the efficiency of messaging services by forwarding the messages to actual backend services.
- Additionally, if time permits, we will be using Chaos Monkey – a tool implemented by Netflix which is responsible for randomly terminating instances to ensure services are resilient to instance failures.

PROJECT REQUIREMENTS

Our proposed application covers six of the components present in the Data Centre Scale Computing project requirements (The link for these requirements can be found here)

- We are planning on building multiple API interfaces so that the user can interact with the interface seamlessly.
- To ensure secure transfer, all file queries requested by the user will be passed to the backend server using message marshalling techniques.
- We are planning to store user's authentication information in key-value pairs and their history will be stored in a database.
- The files which are given as an input by the users will be stored in a storage device like S3.
- Furthermore, we are going to host this whole application on a virtual machine.

Our proposed idea meets all the projects requirements as we will be using six different cloud technologies – REST API, Message Marshalling/Encoding techniques, Key-value pairs, Databases, Cloud Storage Devices and Virtual Machines.

REFERENCES

1. How to Draw Useful Technical Architecture Diagrams
2. Redis
3. Chaos Monkey
4. MlLib Spark
5. Google Cloud Documentation