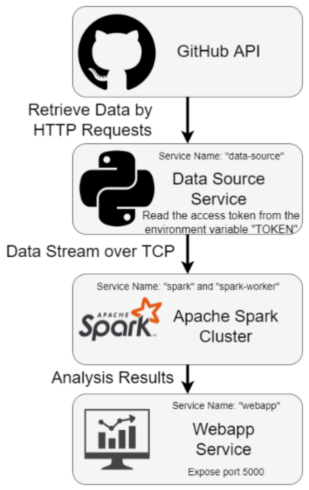
EECS 4415 Project 3 Report

Anuj Ramesh

April 11, 2021

Description of the System Architecture


GitHub API
Retrieve Data by HTTP Requests
Service Name: "data-source"
Data Source Service
Read the access token from the environment variable "TOKEN"
Data Stream over TCP
Service Name: "spark" and "spark-worker"
Apache Spark Cluster
Analysis Results
Service Name: "webapp"
Webapp Service
Expose port 5000

This application consists of four components, as illustrated in this diagram. Firstly, the Data Source Service makes a request to the GitHub API to retrieve repository data. The GitHub API sends it the repository data in JSON format. The Data Source Service then takes only the relevant information from the JSON, streams it over to the Apache Spark Cluster via TCP. The Apache Spark Cluster then analyzes this data and sends some key statistics over to the Webapp Service, which uses these statistics to graphical visualizations. Finally, the user can see these visualized statistics by visiting the webapp service through localhost:5000.

Data Source Service

In this application, the data source service is performed by data_source.py. This program listens to port 9999, waiting for another service to connect. Once a connection has been established, it makes a get request to the GitHub API every 15 seconds. It iterates through the JSON object it receives, and gets the repo's full name, its primary language, its star count, and its description. It then takes these values and puts it into a string separated by delimiters and ending with a newline character. It encodes the data and sends it over to spark application.

Spark Application

The spark application, performed by spark_app.py, gets a stream of data from the data source via TCP connection. It creates batches of 60 seconds, where the data is stored. socketTextStream separates each element of data based on newline characters.

It then maps each repo splitting each string by the delimiter, such that each of the repos' name, language, stars, and description can be separately identified. It then maps each repo into a key, value pair, where keys represent a tuple of the repo's name, language, stars, description, and value is 1. It then reduces by key based on if there are repeating repositories. After it is able to perform map reduce on the repos, and update state by key, it then converts the RDD into a dataframe to count repos, calculate average number of stars per language, etc. It passes these statistics over to the webapp service.

Webapp Service

The webapp service gets important repo data and stores it into a redis database. When a user tries to connect to localhost:5000, it gets this data, and visualizes this data.