Github Data Analysis

Wide Project Proposal

YUAN Yingzhe
Computer Science and Enigeering
HKUST
Hongkong, China

GUO Jiong
Computer Science and Enigeering
HKUST
Hongkong, China

I. Introduction

Github is an open source platform for developers to share their codes and collaborate with others. In this project, we will use spark to construct a real-time processing framework, which aims at handling massive activity records and analyze data to find out patterns of users, repositories and languages.

II. Dataset

All the data we will use in this project are from Github. There is not an existing complete dataset to get enough well-structured data for our project, so we need to collect data and preprocess by ourselves. In this section, we will illustrate why we choose Github's data and what the data sources are.

A. Why Github

Github is the largest developer community and the largest platform for developers to share their codes and collaborate with others in the world. By analyzing the data from Github, we can get some insights about the development of open source projects, the development of programming languages, and the development of the developer community.

B. Source I: Open API

Github provides a series of open API for developers to access most of the informations on Github. This 'Github REST API [2]' is mainly used for creating integrations, retrieve data, and automate workflows, but we can use it to query specific data needed for our project.

There are mainly two main areas of data:

- **Users:** Basic open informations of user, such as id, name, email, following users, repositories owned, etc.
- **repositories:** Basic information of public repositories, such as id, name, owner, description, stars etc.

We will provide some examples results of API query in AppendixA.

There are limits of current official REST API:

- Some need authentication, which means we need to provide a valid token to access the data.
- Some API cannot be called too frequently, which means we need to wait for a while before we can call it again.
- Most of API need an ***id*** as its query parameter, so we need to get the id first before we can query the data, which means we need to construct a massive id list.

After considering the above limitations, we decide to use the Github Archive [1] as our main data source and use the REST API to get some additional data or search some specific content.

C. Source II: Github Archive

GH Archive [1] is a project to record the public GitHub timeline, archive it, and make it easily accessible for further analysis. In fact, it also based on Github's Open API, while only use the **activity** API:

```
https://api.github.com/events
```

GH Archive will crawl github's activity data in real time and sort them as json file in order of when they occurred. Activity archives are available starting from 2011-02-12, and are updated amost half day, so the total size of data is **TB** level.

Considering too large size of the data and the the earliest data using a different format, we will only use the latest part of them. In fact, the current amount data of data in an hour is about 700MB, so the latest data is surely enough for our project. The corresponding compression packages with specific time can be downloaded easily, and the minimal time interval of packages is one hour.

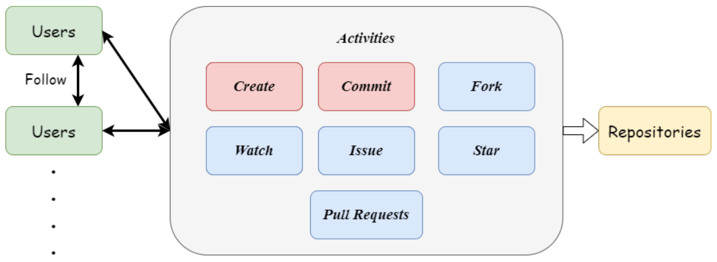

Fig. 1. This figure shows relation between user and repository and basic activities.

The archieved activity data contains almost all 20+ event types provided by Github, which ranges from new commits and fork events, to adding members to a project. We will only pay attention to some key event type, such as *commit*, *create*, etc.

We will give some example of the data format in AppendixB.

D. Summary

Based on almost all activity record, Github's Open API can help us to get almost all related information. For example, from **GH archive**, we know user a push a commit at 12:00 in repository A, then we can get almost all details of a and A by calling Github's API.

III. Task

Github has provided some basic data aggregation for user to explore. For example, in ***Trending [3]***, Github shows the most popular repositories in the specific time range, language and spoken language. In ***Topics [4]***, Github shows repositories' topics according to tags of repositories. However, these data analysis are very basic and not wide to cover most of the information in Github.

In this project, we will use spark to construct a real-time processing framework, which aims at handling massive activity records and analyze data to give a more comprehensive analysis of Github.

There are main tasks in this project:

A. Data Preprocess

1) **Collect** data. Crawl Github Archive [1] and use Github API [2] to jointly query more detailed information about activities according to indices in Github Archive. Though Github Archive update several times a day, we will simulate it to a real-time stream to process data in real-time.
2) **Preprocess** data by Spark. For complete each json record, split it into several parts, filter useful parts and store them in different tables.
3) Data **Persistence**. Design different schema and construct key meta tables to store necessary information, statistics result or some data not easily obtained directly.

B. Data Analysis

1) **Statistics**. We wiil try to construct a dashboard to show some basic statistics of Github, such as the repository with the highest frequency of commits, the most frequently commits time period, etc.
2) **Mining**. There are plenty of ***text*** data from repository's file itself, commit record, issues, etc. We will try to use NLP to extract some useful information from these text data, such as more accuratly topics of repositories, technical hotspots, etc.
3) **Prediction**. We will try to predict the future trend of Github, such as the number of repositories created daily in the future, trends of some programming languages, etc.
4) **Relations**. One of the main two parts of Github is **Users.** They connect by following each other or participating in the development of the same repository, for example, make pull requests, or give issues, etc. We will try to analyze and construct a social graph.

C. Data Visualization

1) **Dashboard**. We will try to construct a dashboard to watch the process of data process and show the results of data analysis by deploying a webpage.
2) **Interaction**. We will try to make the dashboard interactive, such as users can choose the time range of data to analyze, or choose the language to analyze, etc. Results of corresponding process not only comes from persistant database, but also from real-time stream produced by Spark.

IV. Technologies

The technologies we will use in this project can be divided into three parts roughly: big data process, data visualization and environment deploy. Some of them are not sure if they will be used.

The details are as followed:

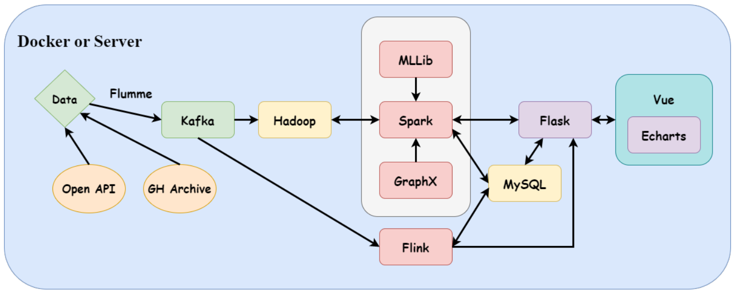

Fig. 2. This picture show the rough technologies based on Dataflow.

A. Big Data Process

1) **Spark**. As a computing engine, it will perform most of the data processing tasks.
2) **Flumme** As a data collector, it will collect data from Github Archive and Github API.
3) **Kafka**. As a message queue, it will simulate the real-time stream of Github Archive.
4) **HDFS**. As a distributed file system, it will store the intermediate data as file or some persistant data.
5) **Hive**. As a data warehouse, it will store the persistant data in tables.
6) **Flink**. As a stream processing engine, it will process the real-time stream.
7) **MLLib**. As a machine learning library, it will be used to extract features from data and finish some basic prediction tasks.
8) **GraphX**. As a graph processing library, it will be used to construct a social graph.
9) **HBase**. As a NoSQL database, it will be used to store some persistant data.
10) **SparkNLP**. As a open-source library, it will perform the complicated NLP tasks. We will explore how to integrate it with Spark, because it's not an official library of Apache.

B. Data Visualization

1) **Vue**. As a front-end framework, it will be used to construct the basic framework of dashboard.
2) **Echarts**. As a data visualization library, it will be used to visualize the results of data analysis.

C. Environment Deploy

1) **Docker**. Though the final application will be deployed on several cloud clusters. we may consider to deploy test environment on a powerful single machine locally by using Docker. By self-define images, it will help us construct the whole system easily and quickly.
2) **Kubernetes**. As a container orchestration engine, it will be used to manage the containers. This will be used as a backup to set up the test system.

V. Summary

Based on Github's activity data, we will use Spark and corresponding tools to construct a real-time processing framework, which will be deployed on a cloud cluster. We will show more comprehensive analysis of Github by using this framework.

References

[1] https://www.gharchive.org/
[2] https://docs.github.com/en/rest?apiVersion=2022-11-28
[3] https://github.com/trending
[4] https://github.com/topics

Appendix A
Github REST API Example

There are some API's calling examples.

A. Get All Followers of a User

```
https://api.github.com/users/mateiz/followers
```

```
[
    {
        "login": "bennettandrews",
        "id": 1143,
        "node_id": "MDQ6VXNlcjExNDM=",
        "avatar_url": "https://avatars.githubusercontent.com/u/1143?v=4",
        "gravatar_id": "",
        "url": "https://api.github.com/users/bennettandrews",
        "html_url": "https://github.com/bennettandrews",
        ...
    },
    ...
]
```

B. Get Repository's Detailed Information

```
https://api.github.com/repos/apache/spark
```

```
{
    "id": 17165658,
    "node_id": "MDEwOlJlcG9zaXRvcnkxNzE2NTY1OA==",
    "name": "spark",
    "full_name": "apache/spark",
    "private": false,
    "owner": {
        "login": "apache",
        "id": 47359,
        ...
    }
    ...
}
```

C. Get All Commits of a Repository

```
https://api.github.com/repos/apache/spark/commits
```

```
[
    {
        "sha": "0fde146e8676ab9a4aeafebb1684eb7a44660524",

        "commit": {
            "author": {
                "name": "Juliusz Sompolski",
                "email": "julek@databricks.com",
                "date": "2023-03-24T01:56:26Z"
            },
            "committer": {
                "name": "Hyukjin Kwon",
                "email": "gurwls223@apache.org",
                "date": "2023-03-24T01:56:26Z"
            },
            "message": "...",
            "tree": {
                "sha": "308faa10a134402eb4e0796c54b4e4dd000fc7dc"
            },
            ...
        },
        "author": {
            "login": "juliuszsompolski",
            "id": 25019163,
            "node_id": "MDQ6VXNlcjI1MDE5MTYz",
            ...
        },
        "committer": {
            "login": "HyukjinKwon",
            "id": 6477701,
            ...
        },
        "parents": [{}]
    },
    ...
]
```

Appendix B
GH Archive

One single json record from the GH Archive.

```
[
    {
            "id": "2489651051",
            "type": "PushEvent",
            "actor": {
                "id": 3854017,
                "login": "rspt",
                "gravatar_id": "",
                "url":
↪   "https://api.github.com/users/rspt",
                "avatar_url":
↪   "https://avatars.githubusercontent.com/u/3854017?"
            },
            "repo": {
                "id": 28671719,
                "name": "rspt/rspt-theme",
                "url":
↪   "https://api.github.com/repos/rspt/rspt-theme"
            },
            "payload": {
                "push_id": 536863970,
                "size": 1,
                "distinct_size": 1,
                "ref": "refs/heads/master",
                "head":
↪   "6b089eb4a43f728f0a594388092f480f2ecacfcd",
                "before":
↪   "437c03652caa0bc4a7554b18d5c0a394c2f3d326",
                "commits": [
                    {
                        "sha":
↪   "6b089eb4a43f728f0a594388092f480f2ecacfcd",
                        "author": {
                            "email":
↪   "5c682c2d1ec4073e277f9ba9f4bdf07e5794dabe@rspt.ch",
                            "name": "rspt"
                        },
                        "message": "Fix main header
↪   height on mobile",
                        "distinct": true
                    }
                ]
            },
            "public": true,
            "created_at": "2015-01-01T15:00:01Z"
        },
        ...
    ]
```