Geni Project Image Recognition

Chi Zhang, Di Zhu
<czhang1, zhudi>@bu.edu

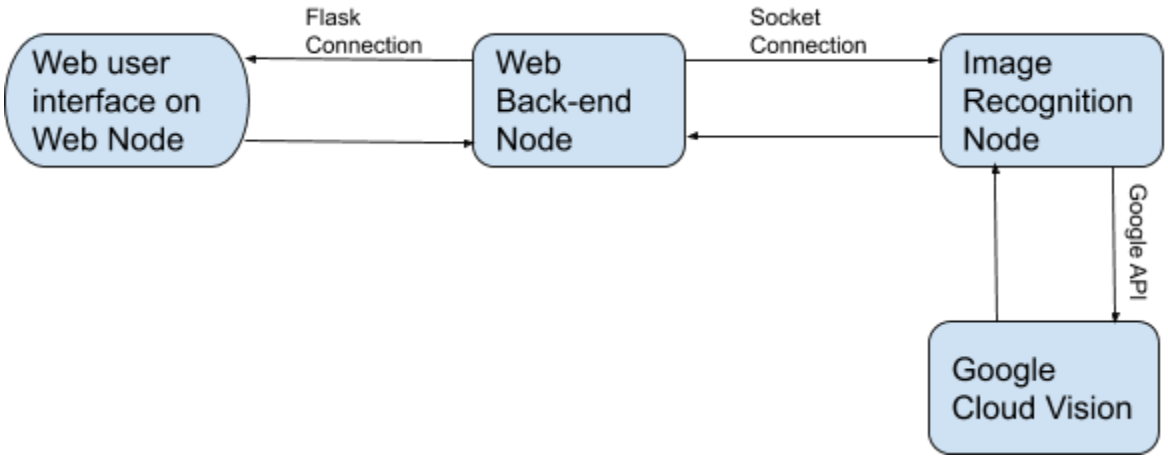
Geni public link of our website: http://128.163.232.71/mainPage
Github link: https://github.com/zhangchi1/ImageRecognitionWeb

Introduction

In this project, we implemented an image recognition service. Users can open the website and upload an image, then the backend server will run the image recognition algorithm using Google Vision API. Finally, the website will display the top three predicted class labels with probabilities of the uploaded image, as well as the RTT and throughput for measuring the network statistics. The main learning outcome of this project is to be able to establish a stable and fast communication between two nodes. More specifically, we created two server nodes on the GENI Portal, one for the web server, and the other one for the image recognition model. In the web server node, we installed the Apache2 http server. Then we applied Flask to send and receive data from Python back-end and html front-end webpage. After receiving the image uploaded by the user, we established a new socket connection between the Python back-end node and the Image Recognition node. In the image Recognition node, the server established a new socket connection using multi thread socket, when it received a new image from the Web node. Finally, the web Node received the predicted labels, and used Flask to post prediction response on the website.

Experimental Methodology

Architecture Diagram



1. Web user interface: we hosted the website on the webNode server, with server address: "http://128.163.232.71:80" via port 80
2. Web back-end Node: we created the Python back-end to send and receive data with the website using Flask. Then it sent image and received labels response from the Recognition Node using multi-threaded socket connection.
3. Image Recognition Node: after uploading the image, the image recognition server received the image through the socket. Then, this node used Google Cloud Vision API to analyze the image and send back the label response to the web Node. This socket will remain open until the response is received by the web Node.
4. Google Cloud Vision: we applied Google Cloud to analyze the image labels.

The maximum image size to upload is 10 MB, because of the payload the Google Cloud Vision API. In addition, our website should allow at least 10 users to upload images simultaneously, since we implemented multi-threaded socket. The server will automatically terminate after encountering 10 errors.

Results

Usage Instructions

1. Running environment: the server end of our application needs to be run on machines with Python 3.5 or higher version.
2. Download our code and configuration files from https://github.com/zhangchi1/ImageRecognitionWeb or
   download our files with git commands
   git clone https://github.com/zhangchi1/ImageRecognitionWeb.git
3. Setting up Geni servers by creating a slice and adding resources with the Rspec file in our project repository '*rspec.xml*'. Click 'Reserve Resources' button and wait until the nodes are ready. You will see two nodes: webNode and RecogNode.
4. Log on the RecogNode. Upload '*rcgnode_setup.sh*' onto the RecogNode and run it. This script will install relevant packages and download our source files. After setup finishes, you will see a folder named '*rcgNode*'. Navigate into '*rcgNode*' folder and use command: '*sudo python3 server.py'* to start the server.
5. Log on the WebNode. Upload 'webNode_setup.sh' onto the webNode and run it "sudo webNode_setup.sh". This script will install Apache2 for an HTTP server and other required packages. Navigate to the http root directory using "cd /var/www/html", and make sure you put all files in the webNode repo under this directory. After setup is complete, use command '*sudo python3 main.py'* to start the webNode server.
6. Open http://128.163.232.71/mainPage with your Google Chrome browser and start using the website.

Analysis

Image recognition analysis

Our system is an image recognition system, so the first thing we would like to see is whether our system could correctly identify what the queried image is about.

Probability	0.9789	0.587	0.8858
Label	Hair	Actor	Eyebrow

Probability	0.8964	0.9523	0.8747
Label	Hill	Cloud	Sunset

Probability	0.8293	0.9765	0.9744
Label	Carnivore	Canidae	Puppy

Probability	0.9848	0.9805	0.8737
Label	Amusement ride	Amusement park	Recreation

Probability	0.966	0.5917	0.9575
Label	Apple	Vegan nutrition	Fruit

Probability	0.9752	0.8732	0.9333
Label	Fruit	Plant	Food

We tried six images on our system with different themes (human, natural views, animal, huge object, small object and multiple objects). To avoid bias, we chose images that the theme object is emphasized in the center with some background (that is, not blank background). We can see that our system is good at identifying animals and single object. Humans will be identified as body parts like hair or eyebrow. The overall results are satisfying.

RTT and throughput analysis

The previous analysis has shown that our image recognition system gives out satisfying image recognitions. Another thing we are curious about is how long does it take our system to finish the entire image recognition process because nobody wants to wait for a very long time. Our metric of maximum acceptable waiting time is 7 seconds.

To test this, we measured RTTs and throughputs for different sizes of image files. The images we used for testing vary from icons of 8000 bytes to 4k image with a size of 2.4MB. The results are presented in the table below. We can see that as the file size gets larger, RTT and throughput increase accordingly. Our system takes 4.19 seconds to process a 2.4MB size of 4k image and this is a satisfying results tous. No data loss happened during testing.

Image File Size (byte)	RTT (second)	Throughput (byte / second)
7990	0.5575089454650879	8247
32355	0.9015896320343018	32660
85820	1.2576348781585693	85966
199817	1.7542827129364014	200044

437711	1.9089233875274658	437943
2522908	4.194766283035278	2523237

Conclusion

In this project, we implemented an image recognition system with an architecture of two nodes and Google Cloud Vision as our image recognition technique. Our system gives out satisfying results and is able to accurately identify the main theme of an image. We also measured the relationship of RTT and throughput with respect to image size. It turns out that as image size increases, RTT and throughput increase as well. The total response time is not too long (within 5 seconds) and users won't have to wait for a very long time for results to come back.

Division of Labor

Chi Zhang: Implementation of the html page that functions as user's interface to the system and the implementation of python socket (client) on the user interface node. Report writing.
Di Zhu: Implementation of python socket (server) on the image recognition node and its connection to image recognition API (Google Cloud Vision). Report writing.

Reference:

Server Addresses

Node #1:

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	RecogNode	pc2	2019-12-25T23:59:59.000Z	emulab-xen	RecogNode.finalProject.ch-geni-net.lan.sdn.uky.edu
Login	ssh psohal@pcvm2-11.lan.sdn.uky.edu ssh czhang1@pcvm2-11.lan.sdn.uky.edu ssh matta@pcvm2-11.lan.sdn.uky.edu ssh zhudi@pcvm2-11.lan.sdn.uky.edu				

Interfaces		MAC	Layer 3
interface-0	pc2:lo0	02ecc1488d59	ipv4: 192.168.1.201

Node #2:

Status	Client ID	Component ID	Expiration	Type	Hostname
CHANGING	webNode	pc2	2019-12-25T23:59:59.000Z	emulab-xen	webNode.finalProject.ch-geni-net.lan.sdn.uky.edu
Login	ssh psohal@pcvm2-12.lan.sdn.uky.edu ssh czhang1@pcvm2-12.lan.sdn.uky.edu ssh matta@pcvm2-12.lan.sdn.uky.edu ssh zhudi@pcvm2-12.lan.sdn.uky.edu				

Interfaces		MAC	Layer 3
interface-1	pc2:lo0	0236fa92c70e	ipv4: 192.168.1.200

Link #1:

Client ID	Endpoint #0	Endpoint #1
link-0	interface-0	interface-1