Big-data Project Report

Zirui Deng

1. Introduction

This project deploys MNIST application into the container. Users submit digital pictures with handwritten characters, and the python program first recognizes the pictures and then returns the recognized numbers. Every time a user submits an image and it is identified in MNIST, the information is recorded in Cassandra.

The web service is created using Flask, and the prediction model is the Softmax model from the tensorflow tutorial. The prediction model is exported using tensorflow and loaded when a Docker Container is built under app.py. The online database is achieved using Cassandra through Docker.

2. Terminology

MNIST is a classic introductory demo for deep learning. It consists of 60,000 training pictures and 10,000 test pictures, each of which is 28*28 in size (as shown below), and is made of black and white (where the black is a floating point number of 0-1, the deeper the black is, the closer the value is to 1).

Tensorflow is an open source software library using data flow graphs for numerical computation. Nodes represent mathematical operations in graphs, while edges represent multi-dimensional arrays of data that are interconnected among nodes, i.e. tensors. Its flexible architecture allows you to deploy computing on a variety of platforms.

Flask is a lightweight Web application framework written in Python. Its WSGI toolbox uses Werkzeug and its template engine uses Jinja2. Flask uses BSD authorization. Flask is also known as the "microframework" because it uses a simple core and extensions to add other functions.

Docker is an open source application container engine that allows developers to package their applications and dependencies into a portable container and then publish them to any popular Linux machine, as well as virtualization. Containers are completely sandboxed and do not have any interfaces with each other.

Apache Cassandra is an open source distributed database management system designed to handle large amounts of data on many commercial servers, providing high availability without failure. It provides powerful support for clusters spanning multiple data centers with asynchronous, unowned replication, allowing all clients to perform low-latency operations.

Cassandra Query Language, or CQL, is a declarative language that enables users to query Cassandra using a language similar to SQL. CQL was introduced in Cassandra version 0.8 and is now the preferred way of retrieving data from Cassandra. A major benefit of CQL is its similarity to SQL and thus helps lower the Cassandra learning curve.

A RESTful API is an application program interface API that uses HTTP requests to manipulate data. A RESTful API is based onrepresentational state transfer REST technology, an architectural style

and approach to communications often used in web services development.

3. Implementation

The mnist deep learning model used in this project uses softmax Regression, referring to the official code of tensorflow on github, its final correct rate is around 91%. We need to save the model after training. First we should define a saver, then use the function saver.save to save the model to a predefined path. After saving the model, we also need to load the model into use when using the model for prediction.

We then need to connect to Cassandra first, and then use it as a database to store the user's submitted images, recognized text and timestamp information. We need to start the Cassandra container and CQLSH, and build namespaces and tables through the command line:

First, start a Cassandra server instance with the following command (which I named ll-cassandra). After running, it will run a cassandra container named ll-cassandra, and the version is latest:

$ docker run --name ll-cassandra -p 9042:9042 -d cassandra:latest

Enter the command "$ docker ps" to get the Cassandra container instance. As shown below, you can see that the ll-cassandra container is running, and the port mapping has been successful--'0.0.0.0:9042->9042/tcp'.

Then, we can use the following command entry Docker adds for linked containers ll-cassandra and runs cqlsh (Cassandra Query Language Shell) , allowing you to execute CQL statements against your database instance:

$ docker run -it --link ll-cassandra:cassandra --rm cassandra cqlsh Cassandra

After the connection is successful, the following will occur:

```
Connected to Test Cluster at cassandra:9042.
[cqlsh 5.0.1 | Cassandra 3.11.3 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh>
```

To build a docker container, we need python files, requirements.txt and Dockerfile. The python file is a web application. Requirements.txt is used to record all dependencies and their exact version numbers for new environment deployment. In this project, the contents of requirements.txt are as follows:

Flask
numpy
pillow
datetime
cassandra-driver

tensorflow

Dockerfile is a text file that contains a strip of instructions. Each instruction builds a layer. Therefore, the content of each instruction is to describe how the layer should be constructed, and customize the configuration and files added to each layer.

Then run the build command to create a Docker image:

$ docker build --tag=big-data .

Then run the app in the background, in detached mode:

$ docker run -d -p 4000:80 big-data

After that, we need to upload our built image and run it somewhere else. Now we push the container to registries as I want to deploy containers to production. First, we should log in. Run the docker tag image with your username, repository, and tag names so that the image uploads to your desired destination. Then upload the tagged image to the repository. The syntax of the command is:

$ docker tag big-data kendeng1127/project-bigdata:bigdata
$ docker push kendeng1127/project-bigdata:bigdata

From now on, you can use docker run and run my app on any machine with this command:

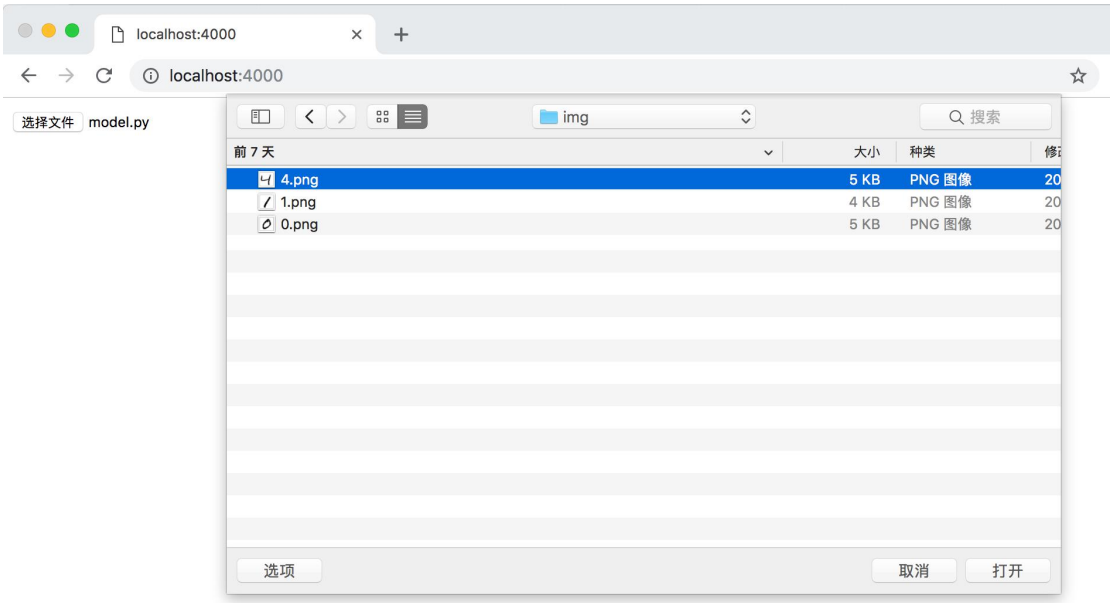
$ docker run -p 4000:80 kendeng1127/project-bigdata:bigdata

4. Project Display

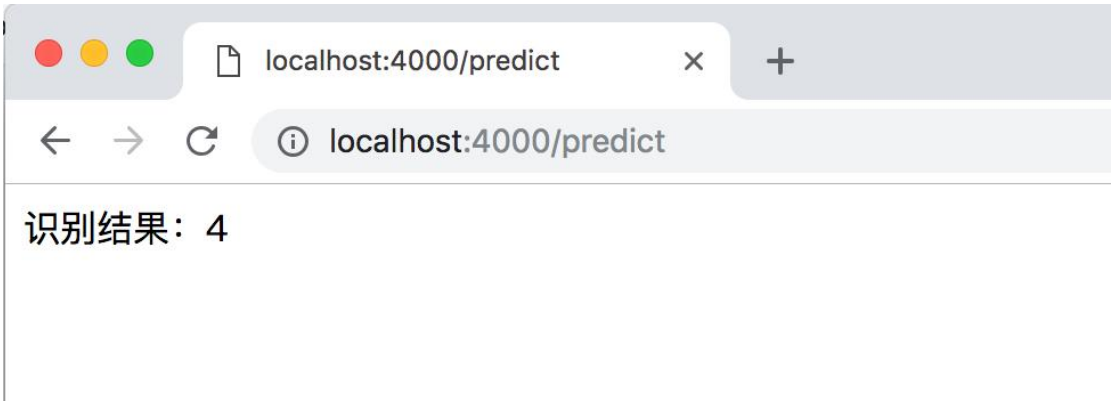
First, start our cassandra container. The standard Cassandra ports are exposed, so container linking makes the Cassandra instance available to other application containers. Start my application container like this in order to link it to the Cassandra container:

$ docker run --name ll-mnistproject --link ll-cassandra:cassandra -p 4000:80 ll-bigdata

We can use Web to submit images. The app is running on localhost:4000.



Then using the model of mnist, it will return the predict result.



5. Conclusion

In this study, I was able to learn substantial and important knowledge of Docker and Cassandra along with other essential skills and techniques and applied them to real practice. These two tools that I singled out are to me quite convenient and efficient to use, and I believe they will be of great help to my future endeavors.

6. Reference

http://docs.jinkan.org/docs/flask/patterns/fileuploads.html

http://abiasforaction.net/a-practical-introduction-to-cassandra-query-language/

https://hub.docker.com/_/cassandra/

https://docs.docker.com/get-started/

http://flask.pocoo.org/docs/0.12/quickstart/#a-minimal-application

https://academy.datastax.com/courses