Remote Scientific Research Project Report----field of Big Data

Name: Gao Hanyuan

Instructor: Fan Zhang

2020.04

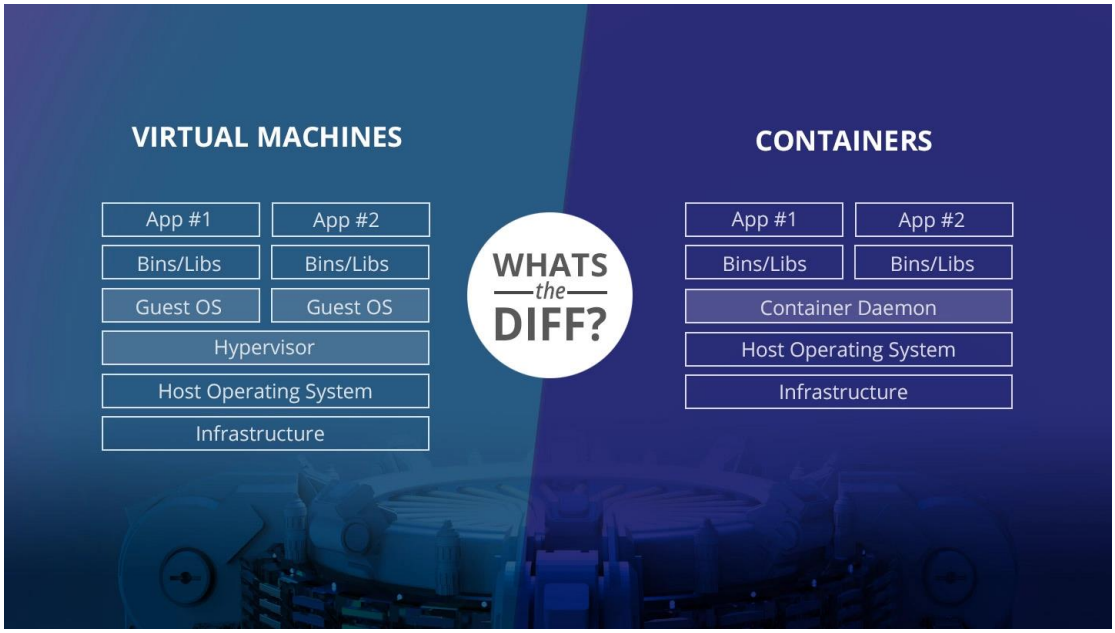
1. Background

1.1 Virtualization by Container

We are now familiar with Big Data, a large and complex data sets. As the machine rapidly develops from bottom to top, people as well as computers are facing more and more massive data. A question comes naturally that how to deal with and what can we get from these data, which are within the scope of data mining and learning. Another question lies on how to store and transfer the data, or at least how to deploy our result from the data easily, considering that the result can be an application. For the former, many try to build and optimize storage systems, like distributed database, and access them through network. The later question seems simple but complicated. When deploy our application and instrument, the environment can be quite different from that of developing. The software and hardware are strict with its operating environment, thus causing trouble when transfer from one system to another. As a result, developers must develop their product of different versions to fit in many current popular systems, which is common nowadays. Since we can not easily unite different systems, people came up with methods that allow an app to easily run in any system on any machine by creating a virtual environment, which the app originally runs in, on the machine. Traditional virtual machine set up an entire system and consume huge resources. Recent years, containerization becomes a rising used method gradually.

The notable difference between normal virtualization and containerization has been mentioned above. Traditional virtual machine set up an entire operating system and an entire hardware simulation on current system, occupying multiplied sources and almost paralyze the machine, especially in PC. Apparently, the virtual machine set up many replicas of part of OS kernel. Containerization makes use of this point and only virtualizes an environment. The environment uses existing functions of current OS. Software are packaged into a box. The box contains runtime dependencies what the current system lacks. All these runtime dependencies are managed by a middle layer above OS.

Containerization has many advantages. It packages an app into a single address access container. The lightweight of the container makes it able to start up an app fast, almost within seconds. Besides, the containers can be stored in registry and started by simple instructions. The deploy of the app has been hugely simplified wherever the app runs in. The sealed boxes are also easy to migrate and combine with each other. Nowadays, the tech has been widely used in software developing, running and connecting multiple applications, cloud computing, etc.



1.2 Frontier Research

Nowadays, there has been many developing tools and framework. The most known container virtual machine is docker. Docker is originally developed in Linux, but now accessible to most mainstream operating system. Many software has been packaged and pushed into registry of docker. Docker allows one to pull software images from registry then run a container from the image and build an image of your own program then push it to registry for others to use. It is quite convenient to deploy a program.

Another instrument useful for storage and deploy is Cassandra, a distributed NoSQL database system. Cassandra is the most successful system of all distributed database with its scalability reaching linear scale, meaning you can always improve its performance by adding a node into the existing cluster under current problem size. Coordinating with docker, Cassandra is easy to deploy on a large scale.

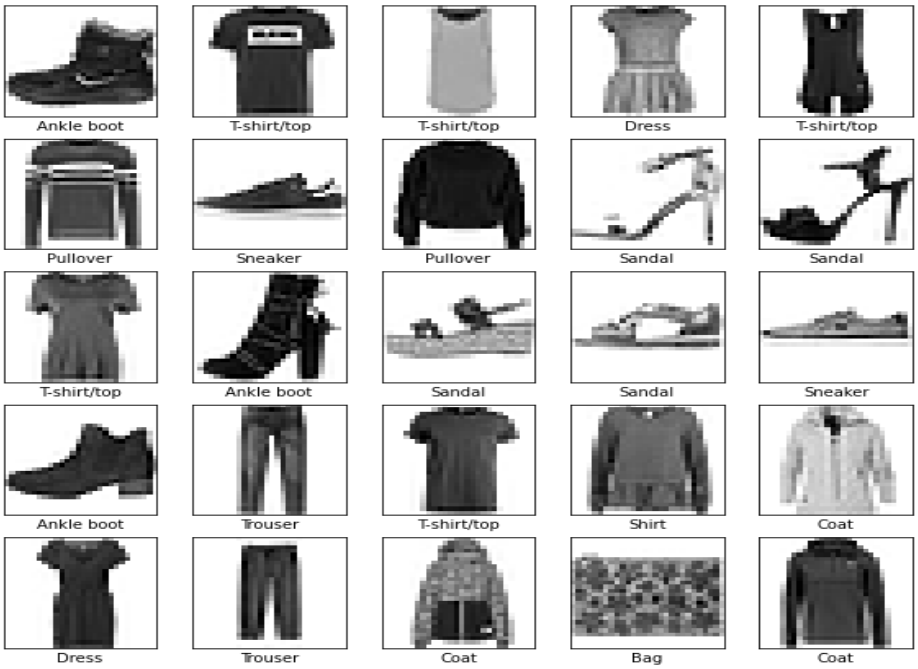
2. Project Description

2.1 Introduction

This project is a lab about bigdata virtualization and machine learning. The lab built a simple web app based on the framework of python & flask & Cassandra, which realized the classification of images of clothing. Then it built the app into a container using docker.

2.2 Application Layer

The application is a tutorial example from TensorFlow(https://tensorflow.google.cn/). The program uses the Fashion MNIST dataset to train a neural network model to classify images of clothing.



It requires several steps to set up the model. The program loads the Fashion MINIST datasets and use it as an input of a training function from the TensorFlow model. The function receives images and the true class label as basic and automatically creates and trains a neural network to do classification.

```
# Build the Model
# Set up the layers
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(10)
])
# Compile the model
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(
                  from_logits=True),
              metrics=['accuracy'])
# Train the model
# Feed the training data to the model, set epoches=10
model.fit(train_images,
            train_labels,
            epochs=15)
```

After the training finished, the model can be used to do prediction. Given any images of clothing after pretreatment, the `predict` member function of the model can do predictions that what class the image maybe classified as. Finally, we save the model for further develop.

```
# Make predictions
predictions = probability_model.predict(test_images)
# Save the model
model.save('my_model.h5')
```

2.3 Web Framework: API

We use flask as our Web application framework. Flask is a lightweight web application framework written in Python. It uses a simple core. Extensions are used to add other functions. Flask does not have a database and form verification tool used by default.

In this application, briefly, the app receives an image from the front-end. On the back-end, the program uses the model built above and uses the image as an input to make prediction. Finally, return the prediction result to the front-end page.

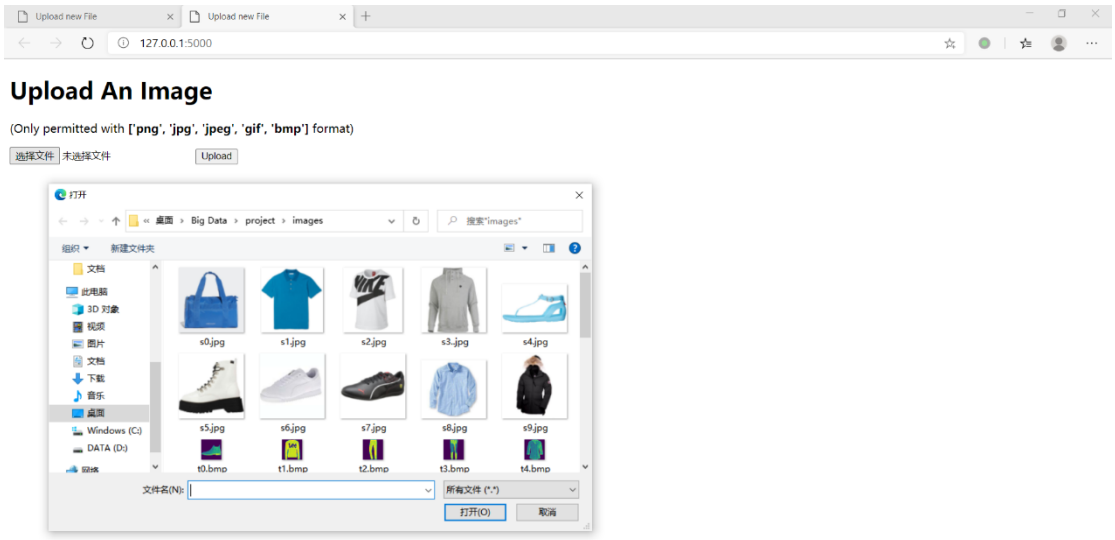
Flask is lightweight and convenient to do such simple framework. The program only needs to create an `app` object and define some page action to the object.

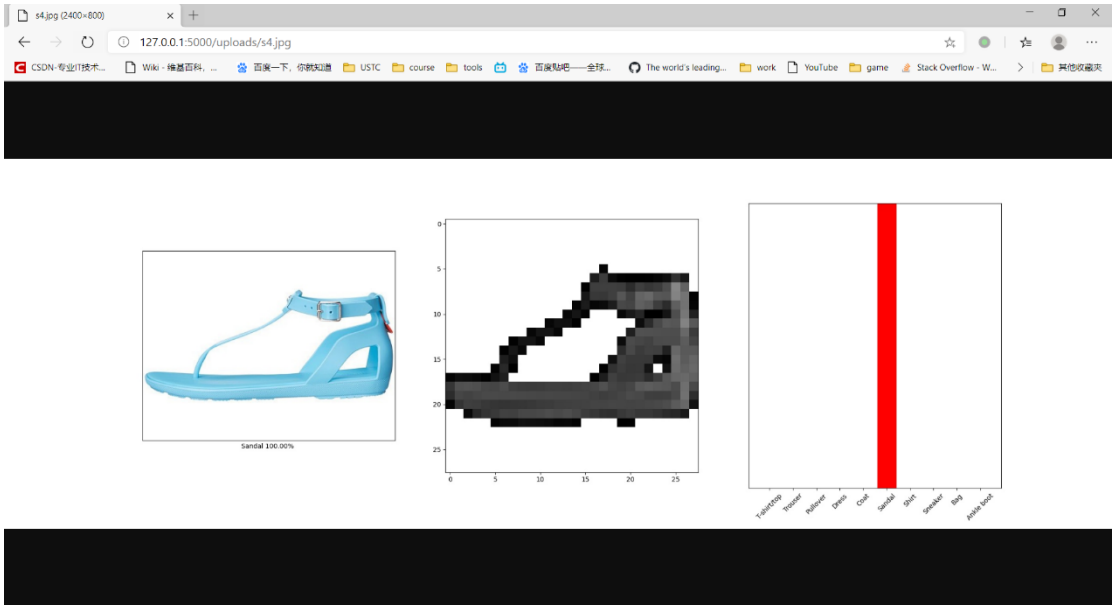
```
from flask import Flask
app = Flask(__name__)

@app.route('/', methods=['GET'])
def hello_world ():
    return 'Hello World!'
```

Of course, the actions are more complex than the function above. We need to write a front-end page contain a `form` item. Then return the page when `get` method is triggered. We also need to write a function to receive file with method of `post`. Flask has relevant built-in functions to implement file receiving and sending. Then weed function to load the model we saved above and use it with the image to do prediction. Finally, we return the result back to the font-end.

Flask app can be run by several simple instructions, but it is not run by python directly. If not be open to public, the app will run at the local address 127.0.0.1, port 5000. We can visit the app when it is running.

Here are some demos of final effect.





2.4 Web Framework: Database Connection

We use Cassandra as our bottom layer system, for its light weight and good distributability. More convenient, Cassandra has been packaged into docker. We can get and run it with few commands.

```
docker pull Cassandra

docker run --name some-cassandra --network some-network -d cassandra:
tag

docker run -it --network some-network --rm cassandra cqlsh some-
cassandra
```

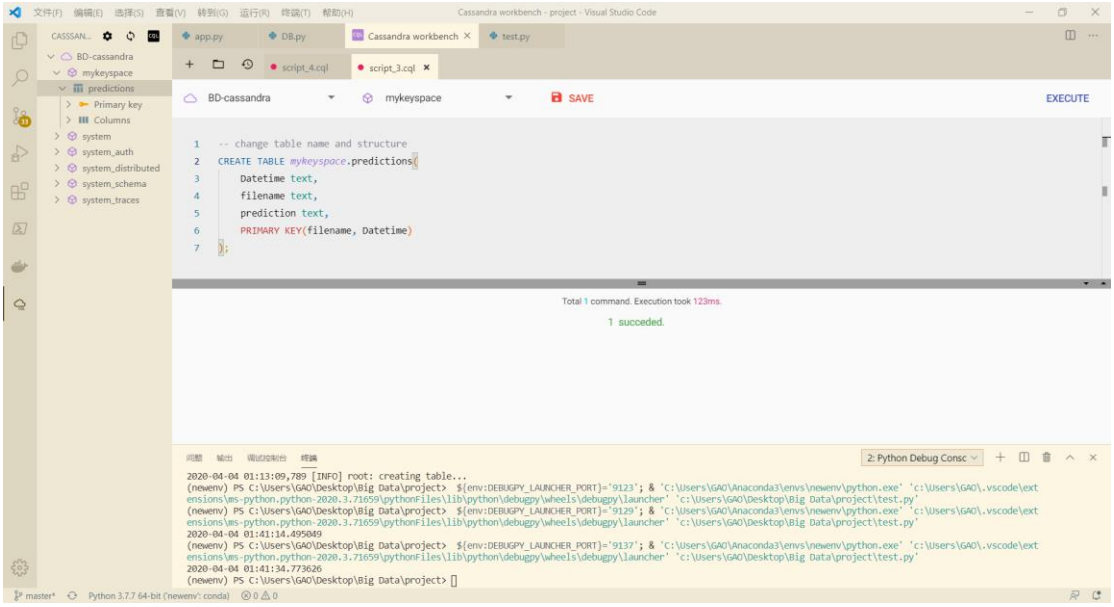
The database server has been run in a container. We can connect to it through the address of the container by means of cassandra-driver, a model of python. Cassandra-driver offers many useful interface to connect to cassandra server and do query operations.

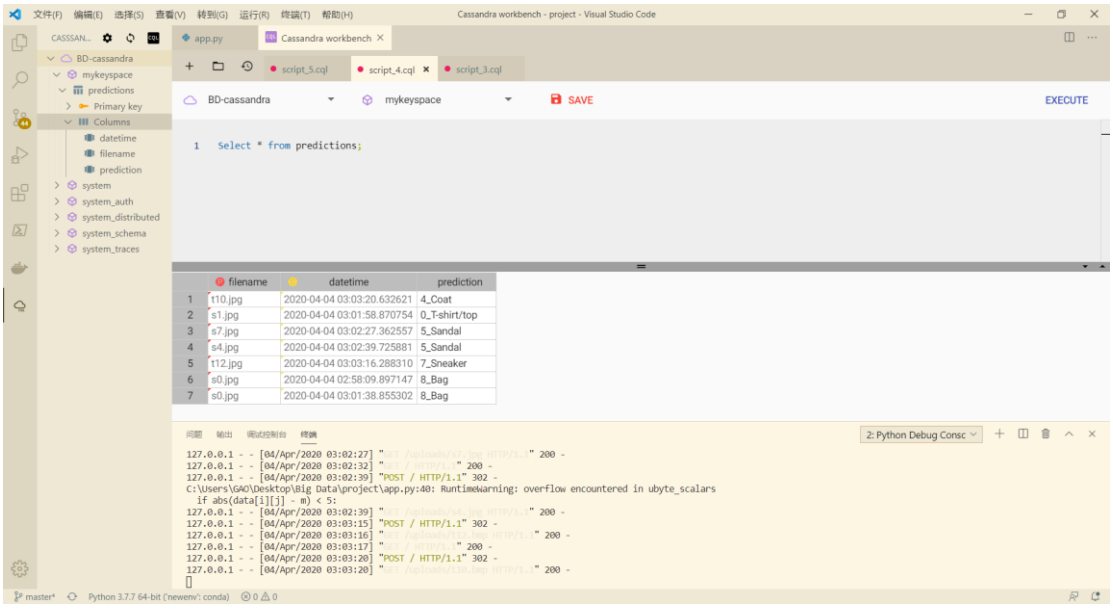
```
def createKeySpace():
    cluster = Cluster(contact_points=IP_ADRESS, port=9042)
    session = cluster.connect()

    log.info("Creating keyspace...")
    try:
        session.execute("""
            CREATE KEYSPACE %s
            WITH replication = { 'class': 'SimpleStrategy', 'replicatio
                n_factor': '2' }
            """ % KEYSPACE)

        log.info("setting keyspace...")
        session.set_keyspace(KEYSPACE)

        log.info("creating table...")
        session.execute("""
            CREATE TABLE mytable (
                mykey text,
                col1 text,
                col2 text,
                PRIMARY KEY (mykey, col1)
            )
            """)
    except Exception as e:
        log.error("Unable to create keyspace")
        log.error(e)
```

Of course, the database only needs to be created once. Once the database is created, we can add more functions in `app` to connect to it and insert data to it.

```
@app.route('/uploads/<filename>')
def uploaded_file(filename):
    pr = '\'' + predicting(filename) + '\''
    na = filename.rsplit('.', 1)[0].lower() + '.jpg'
    dt =  '\'' + str(datetime.datetime.now()) + '\''
    session.execute('INSERT INTO {} (datetime, filename, prediction)
        VALUES ({}, {}, {})'.format(TABLES[0], dt, '\''+na+'\'', pr))
    # return prediction result to the front-end
    return send_from_directory(app.config['UPLOAD_FOLDER'], na)
```

We only insert some basic information to the database, like the name of the image, the prediction of the image and timestamp. We can use some useful tools like cassandra workbench to help operate on cassandra. VS code has a plugin tool:





2.5 Build and run an image in docker

At this part, we package our app into an image and run a container in docker. Since we have already run the database server in docker, there will be two containers communicating with each other in the end.

The key to build a docker image is to write `dockerfile`.

```
# bulid an image based on an basic image
FROM python:3.7

# set the workplace
WORKDIR /app

# copy files
ADD . /app

# install dependencies
RUN pip install -r requirements.txt

EXPOSE 5000

# Run app.py when the container launches
CMD ["export", "FLASK_APP=hello.py"]
CMD ["flask", "run"]
```

Use docker command to build and run your images. Docker will run the instructions in dockerfile one by one and build an image.

```
docker build -t bdroject:latest .
```

Notice that `requirement.txt` contains all dependencies of python models. This file can be easily build by running a pip cmd in the virtual environment of your project:

```
pip freezing>requirements.txt
```

Then launch a container of an image. The CMD part of dockerfile will be carried out at this moment. We need to run our flask app, thus two cmd are executed.

```
export FLASK_APP=hello.py
flask run
```

Finally, the application run in a container!

3. Summary

I participated this remote project in spring 2020. This is the first time I took a project in such a unique way, but soon after I was used to this kind of online learning as well as working exactly because of the coronavirus disease, causing suspension of schools. Though the time I spend to take online lessons with Fan Zhang, our instructor, was limited. I've learned a lot from the short period of time.

We've carried out lessons for four times lasting about one month. Mr. Zhang brought us carefully prepared courses. He introduced many frontier research and newest instruments. As an undergraduate student majored in computer science, what I learned from college were very foundational knowledge, theoretical but unpractical. I hardly be in touch with these commonly used and newest skills in actual work and research. The project became my navigator. Of course, I just grasp only a tip of the iceberg because of the limited time. I was introduced of python——most popular object oriented programming language. I've learned C before and self-learnt python a little, but in this project, I realized why people see it as the most powerful language for data processing and machine learning. I was told to use Git & GitHub——useful version control system and registry. We learned the basic of TensorFlow, a machine learning integrated library. We learned a new web framework called flask, a new database called Cassandra and most importantly, we learned to use docker, a containerization virtual machine. Besides, Mr. Zhang present us more popular and mainstream tools for big data. I realize that new technique updates quite fast and our job will be accompanied with continuously learning. While we build our foundation, we also need to catch up with new things.

The project was descripted as above. I made a simple web application about classification and run it in a docker container. It was simple but not that simple because I almost knew about nothing at the beginning. Incredibly, all these things are made by myself. I just contact Mr. Zhang and other fellows online and our chat record can be scroll through within a second. Almost all stuff I did were by self-learning on the internet. I kept learning from the internet, reading after the documents and searching for tutorials. When I grow from a kinda noob to a slightly better fresh, I gained great reward and satisfaction looking back to myself. I've met many problems and get stuck in troubles for long, but we all know there is nothing more wonderful than solving a long-standing puzzle.

Finally, I give my heartfelt gratitude to Fan Zhang. He is the conductor of this project. He offered us students an outline and organized the process without mistakes. We did this project through our work time, thus all of us sacrificed for this work. I gained valuable experience from the intensive developing process. Besides, thanks to my group members. The communication with them brought me positive inspiration.