MIT Remote Research Project Report

Big Data

张立涵

Zhang Lihan

2019.8.13

Contents

1 Project Background

1.1 Big Data

Big data has been a hot professional and research direction in recent years. It refers to a collection of data that cannot be captured, managed, and processed with regular software tools over a certain period of time. It is a massive, high growth-rate diversified information assets that requires new processing models to have greater decision making, insight and process optimization capabilities. Big data technology refers to the ability to quickly obtain valuable information from various types of data. Technologies for big data, including massively parallel processing (MPP) databases, data mining grids, distributed file systems, distributed databases, cloud computing platforms, the Internet, and scalable storage systems. With the explosive development of computers, especially the Internet, we have indeed entered the era of big data.

So what is the role of big data? We have summarized the following 4 points.

First, the analysis of big data processing is becoming the node of the next generation of information technology convergence applications. Mobile Internet, Internet of Things, social networking, digital home, e-commerce, etc. are the application forms of next-generation information technologies that continue to generate big data. Cloud computing provides a storage and computing platform for these massive and diverse big data. By managing, processing, analyzing and optimizing data from different sources, the results are fed back into the above applications, which will create enormous economic and social value.

Second, big data is the new engine for the rapid growth of the information industry. New technologies, new products, new services, and new formats for the big data market will continue to emerge. In the field of hardware and integrated devices, big data will have an important impact on the chip and storage industry, and will also lead to integrated data storage processing servers, memory computing and other markets. In the software and services arena, big data will lead to the development of rapid data processing analysis, data mining technology and software products.

Third, the use of big data will be a key factor in improving core competitiveness. Decisions from all walks of life are shifting from "business-driven" to "data-driven". The analysis of big data enables retailers to grasp market dynamics and respond quickly in real time; it can provide decision support for merchants to develop more accurate and effective marketing strategies; it can help enterprises provide consumers with more timely and personalized services; In the field, the diagnostic accuracy and drug effectiveness can be improved; in the public sector, big data also plays an important role in promoting economic development and maintaining social stability.

Fourth, the methods and methods of scientific research in the era of big data will undergo major changes. For example, sample surveys are the basic research method of social sciences. In the era of big data, we can monitor and track the massive behavior data generated by the research objects on the Internet in real time, conduct mining analysis, reveal the regularity, and put forward research conclusions and countermeasures.

1.2 Project Description

This project will implement a handwritten digit recognition system. The system will implement the following features.

1. The user uploads the image to be recognized on the webpage.

2. The server side recognizes and returns the predict number

3. Save the image name, predict number, timestamp and other information in the database.

2 Tensorflow

2.1 Introduction

Tensorflow is a symbolic mathematics system based on dataflow programming. It is widely used in the programming implementation of various machine learning algorithms. Its predecessor is Google's neural network algorithm library DistBelief.

As a Google open source deep learning system, Tensorflow is based on a data flow graph processing framework. It is a very flexible framework that runs on single or multiple CPUs and GPUs on a PC or server, even on mobile devices. It can be used not only for neural network algorithm research, but also for ordinary machine learning algorithms. Tensorflow can run on a variety of hardware through support for threads, queues, and asynchronous calculations, and assigns operations to appropriate devices based on computational needs.

The TensorFlow calculation framework requires that all calculations be represented as graphs, and nodes are referred to as opera ops (abbreviation of operation). One operation can get zero or more Tensors, and after calculation, more Tensors can be generated. A Tensor is a multidimensional array. The calculation graph is submitted through the Session. A Session determines whether the operation in the graph should be calculated on the device, such as CPU or GPU. The result of the op operation is a numpy.ndarray array object in python.

2.2 Model Realization

In this project, we use tensorflow to train a handwritten digital picture recognition model. The training data set used by this model is MNIST. It is a very classic data set in the machine learning field, consisting of 60,000 training samples and 10,000 test samples, each of which is a 28 * 28 pixel grayscale handwritten digital image. It comes from the National Institute of Standards and Technology. The training set consists of numbers written by 250 different people, 50% of whom are high school students and 50% from Census Bureau staff. The test set is also the same proportion of handwritten digital data.

We import the data set into the program like this:

```
from tensorflow.examples.tutorials.mnist import input_data


mnist = input_data.read_data_sets('F:/MNIST_dataset', one_hot=True)
```

Then we use this training set to train the model:

```
sess = tf.InteractiveSession()

x = tf.placeholder(tf.float32, shape=[None, 784])

y_ = tf.placeholder(tf.float32, shape=[None, 10])
```

```
W = tf.Variable(tf.zeros([784, 10]))

b = tf.Variable(tf.zeros([10]))



sess.run(tf.global_variables_initializer())

y = tf.matmul(x, W) + b

cross_entropy = tf.reduce_mean(

      tf.nn.softmax_cross_entropy_with_logits(labels=y_, logits=y))



train_step=tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)



for j in range(1000):

   batch = mnist.train.next_batch(100)

   train_step.run(feed_dict={x: batch[0], y_: batch[1]})

correct_prediction=tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))



accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

#define weight

def weight_variable(shape):

   initial = tf.truncated_normal(shape, stddev=0.1)

   return tf.Variable(initial)



def bias_variable(shape):

   initial = tf.constant(0.1, shape=shape)
```

```
    return tf.Variable(initial)



def conv2d(x, W):

    return tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding='SAME')



def max_pool_2x2(x):

    return tf.nn.max_pool(x, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1],
padding='SAME')



W_conv1 = weight_variable([5, 5, 1, 32])

b_conv1 = bias_variable([32])

x_image = tf.reshape(x, [-1, 28, 28, 1])

h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1)

h_pool1 = max_pool_2x2(h_conv1)



W_conv2 = weight_variable([5, 5, 32, 64])

b_conv2 = bias_variable([64])

h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)

h_pool2 = max_pool_2x2(h_conv2)



W_fc1 = weight_variable([7 * 7 * 64, 1024])

b_fc1 = bias_variable([1024])
```

```
h_pool2_flat = tf.reshape(h_pool2, [-1, 7*7*64])

h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)

keep_prob = tf.placeholder(tf.float32)

h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob)

W_fc2 = weight_variable([1024, 10])

b_fc2 = bias_variable([10])



y_conv = tf.matmul(h_fc1_drop, W_fc2) + b_fc2

cross_entropy = tf.reduce_mean(

    tf.nn.softmax_cross_entropy_with_logits(labels=y_, logits=y_conv))

train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)

correct_prediction = tf.equal(tf.argmax(y_conv,1), tf.argmax(y_,1))

accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))



saver = tf.train.Saver()  # defaults to saving all variables



sess.run(tf.global_variables_initializer())

for i in range(20000):

  batch = mnist.train.next_batch(50)

  if i%100 == 0:

    train_accuracy = accuracy.eval(feed_dict={

        x:batch[0], y_: batch[1], keep_prob: 1.0})

    print("step %d, training accuracy %g"%(i, train_accuracy))
    train_step.run(feed_dict={x: batch[0], y_: batch[1], keep_prob: 0.5})



# save model

saver.save(sess, '. /model_2/model_2.ckpt')
```

The step size is 100. After running, you can see that the accuracy of the recognition is increasing, and the overall accuracy is about 92%. After the training, we save the model for later use.

3 Container Technology

3.1 Docker

Docker is a currently popular open source application container engine. It allows developers to package their applications and dependencies into a portable image and then publish it to any popular Linux or Windows machine, as well as virtualization.

Docker has several advantages over traditional virtualization.

First, it is quick for Docker to create, start, and delete containers. Docker’s startup is second level, and virtual machines usually take a few minutes. Docker's fast iteration means you can save a lot of time, whether it's development, testing or deployment.

Second, it takes up less resources. Docker is virtualized at the operating system level. Docker containers interact with the kernel with virtually no performance loss, and perform better than virtualization through the Hypervisor and kernel layers.

The third is lightweights. The docker architecture can share a single kernel and shared application libraries with minimal memory. In the same hardware environment, Docker runs far more mirrors than virtual machines, and the system utilization is very high.

There are three very important concepts in Docker: Image, Container, and Repository.

Image can be seen as a special file system. In addition to providing the programs, libraries, resources, configuration files, etc. which are required by the container to run, it also contains some configuration parameters (such as anonymous volumes , environment variables, users, etc.) prepared for the runtime. The image does not contain any dynamic data and its content will not be changed after it is built. It is a unified view of a bunch of read-only layers.

The Container is almost the same as the image, but the top level of the container is readable and writable. The relationship between images and containers is equivalent to the relationship between classes and objects in OOP.

The Repository is a place to centrally store images, similar to Git. The warehouse is divided into two types: public and private.

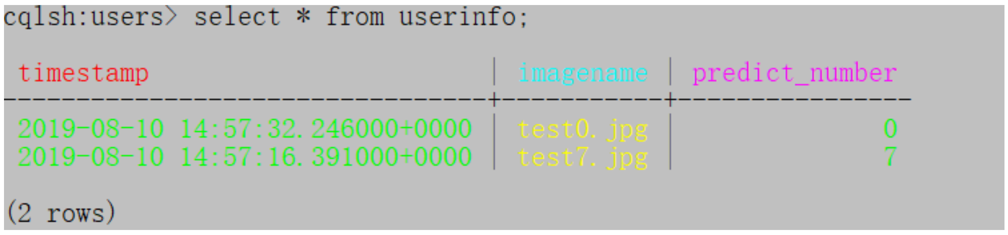
Docker uses the C/S architecture. The Docker client interacts with the Docker server, which is responsible for building, running, and distributing Docker images. The Docker client and server can run on a single machine or communicate with a remote Docker server via a RESTful, stock or network interface.

3.2 Docker in Project

The system described in this report also uses docker to store our programs and data. There are mainly two containers.

The first one is the container under the official image of Cassandra. The container implements a Cassandra database. Cassandra is an open source distributed NOSQL database system similar to Google's Big Table. In fact, Cassandra is not a database, but a distributed network service composed of a bunch of database nodes. It has the characteristics of flexible mode, scalability, and multiple data centers. It is a good choice for the storage of big data.

In the container, we use the CQL language to create the corresponding key spaces and tables. The table structure is shown below.



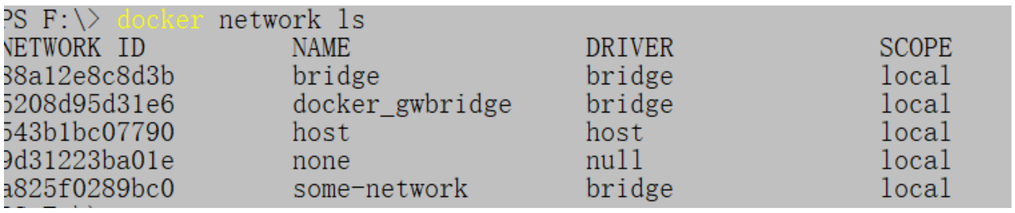
```
cqlsh:users> select * from userinfo;

 timestamp                        | imagename | predict_number
---------------------------------+-----------+----------------
 2019-08-10 14:57:32.246000+0000 | test0.jpg |              0
 2019-08-10 14:57:16.391000+0000 | test7.jpg |              7

(2 rows)
```

The table name is ‘UserInfo’. Cassandra supports more data types than traditional MySQL databases, such as the timestamp type. The primary key is set to be (timestamp, imagename). ‘predict_number’ represents the number recognized by the model.

The ip address of the container is 172.18.0.3, and the network environment is some-network bridged network which is shown in the following figure.



```
PS F:\> docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
38a12e8c8d3b        bridge              bridge              local
5208d95d31e6        docker_gwbridge     bridge              local
543b1bc07790        host                host                local
9d31223ba01e        none                null                local
a825f0289bc0        some-network        bridge              local
```

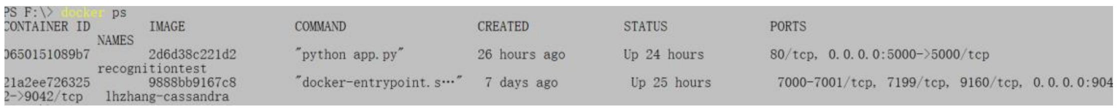
The port is 9042 for later communication.

The second is the container under the image recognitiontest:v1.0.1 created by myself using the Dockerfile. The container is responsible for communicating with the front-end user through the flask structure, reading the picture to be tested, judging with the previously trained model, and returning the result number to the user. At the same time, the container also communicates with the Cassandra container to write the picture information into the database. Therefore, the container is also under the 'some-network' network in order to communicate with the Cassandra container. Therefore, in the program cqloperation.py responsible for communication, the parameter used in the function to connect the Cassandra should be written as '172.18.0.3'. The port is 5000:5000, and the front-end page connects to the container by accessing http://localhost:5000/.

4 Project Display

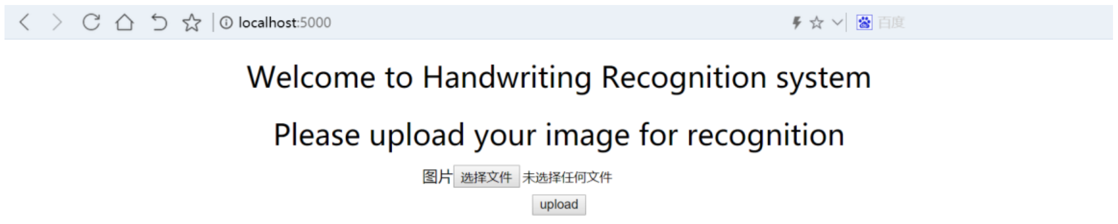
4.1 System Flow

First, we start the two containers in turn and check the states of them to make sure that they are running:



```
PS F:\> docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
                NAMES
0650151089b7        2d6d38c221d2        "python app.py"          26 hours ago        Up 24 hours         80/tcp, 0.0.0.0:5000->5000/tcp
                recognitiontest
21a2ee726325        9888bb9167c8        "docker-entrypoint.s…"   7 days ago          Up 25 hours         7000-7001/tcp, 7199/tcp, 9160/tcp, 0.0.0.0:904
2->9042/tcp    lhzhang-cassandra
```

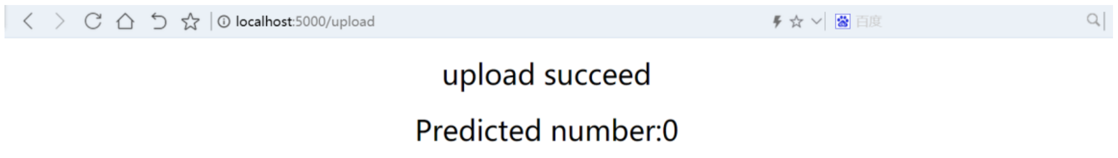
Now we visit the browser with the address: http://localhost:5000/ to enter the initial page of the system:



Welcome to Handwriting Recognition system

Please upload your image for recognition

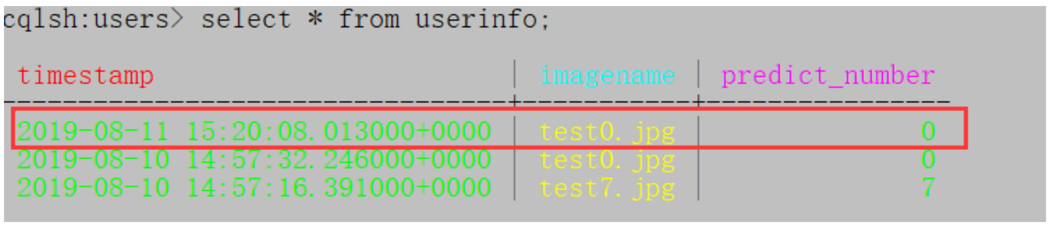
Click the ‘select file’ button, select the image to be recognized, and then click ‘upload’ button to upload, the system will jump to the following page:



upload succeed

Predicted number:0

This indicates that the upload was successful and the number recognized by the model is 0. At this point, visit the Cassandra container, you can see that a new record has been added to the UserInfo table:



```
cqlsh:users> select * from userinfo;

 timestamp                          | imagename | predict_number
------------------------------------+-----------+----------------
 2019-08-11 15:20:08.013000+0000 | test0.jpg |              0
 2019-08-10 14:57:32.246000+0000 | test0.jpg |              0
 2019-08-10 14:57:16.391000+0000 | test7.jpg |              7
```

This shows that the system is running normally and the function is fully implemented. At this point, the user can return to the start page and upload a new picture again.

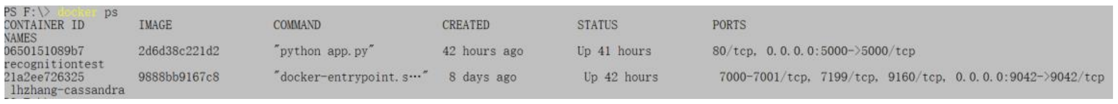
4.2 Difficluties & Solutions

In the project, I have encountered many difficulties. Here are a few of the most difficult and typical difficulties I have met.

The first one is the combing of ideas. I was confused when I got the project request, because I have never learned anything about Docker, Tensorflow, Cassandra, etc., so I didn't know how to build the container at first. After learning for a while, I gradually got my own understanding: because each container can only load one base image, the initial idea of putting all the programs in one container is not realistic. Later, I divided the project into two parts: First, the backend Cassandra database. This container has a relatively simple function. It is only responsible for accepting information and writing to the database. The second is the main program container, for which I select tensorflow as the base image. The container acts as a host. On the one hand, it exposes the port to the front-end user, uploads images. On the other hand, it communicates with the Cassandra container and writes information into the database.

The second difficulty is the configuration of the environment. In the beginning I used the ‘tensorflow/tensorflow’ image, in which Flask (responsible for interaction with the front-end page) and cassandra-driver (interacting with Cassandra) were installed via the RUN command in the Dockerfile. However, after the container starts, it will report an error and exit immediately. By looking at the container’s logs and reviewing the data, I found that the ‘tensorflow/tensorflow’ image uses python 2.7 by default, which is not compatible with Flask. So I use the image ‘tensorflow/tensorflow:latest-py3’ instead. Later, I found that the flask was abnormal and the container could not be started normally. After checking the data, I found that it did not match the newly installed python 3. Statement must be added before the program like this: #/usr/bin/env python3. Finally the container can be used normally. At this point, the overall environment of the first container was successfully built.

The third difficulty is the configuration of the port for communications between the containers. Because we should specify a port not only in the flask program but also in the moment when you create the two containers, so there is always an error at first. After several attempts, the configuration was finally successful. That is, the Cassandra container port is 9042->9042, and the main function container (flask) port is 5000->5000. The port settings should be consistent in the function and when creating the container, as shown in the following figure:



```
PS F:\> docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
NAMES
0650151089b7        2d6d38c221d2        "python app.py"          42 hours ago        Up 41 hours         80/tcp, 0.0.0.0:5000->5000/tcp
recognitiontest
21a2ee726325        9888bb9167c8        "docker-entrypoint.s…"   8 days ago          Up 42 hours          7000-7001/tcp, 7199/tcp, 9160/tcp, 0.0.0.0:9042->9042/tcp
 lhzhang-cassandra
```

After this setting, the flask can interact with the front page, but it cannot connect to the Cassandra container. After some queries I found that communication between the two containers must be under the same network. So the two containers are placed under the ‘some-network’ bridge network, and finally the connection is successful.

There are also some small "pits". For example, I am using the docker for windows, which is operated under the powershell command line. When you want to modify the file in the image, use the "docker run -t -i xxx (image name) /bin/bash" command to create a container into the linux shell to modify. Then if you want to create a new container, you must add the parameter "python app.py" to the "docker run" command to run the container directly. Otherwise, you will enter /bin/bash by default. Another point is that, when using the os.path.join function to store the file setting path, the input address string in the function should be preceded by 'r', such as r'/app/uploadimages', otherwise the python program string recognition will be wrong.

4.3 Summary

Through this project, I have acquired many new knowledge and skills.

The first is to learn a new programming language: Python. This language is flexible, powerful, and easy to learn. Most of the programming languages I've learned are compiling languages, such as C, C++, Java, etc. As an explanatory language, Python brings a new feeling to me for the whole programming idea.

The second is to master the flask framework and RESTful software architecture style, and understand how to provide your services to front-end users. I have a little experience in RESTful style before, so this part is not difficult for me. By the way, The entire flask framework and the Java Spring framework have some similarities, like they both use a request-response model.

Then there is Docker, a powerful container technology. Although I only know a few of them, I can already appreciate the power and convenience of this tool. Because of the experience of using virtual machines in the past, I can see the gap between the two. Using docker also enhances my ability to use the shell.

Later, I learned about the storage and processing of big data, with a focus on the Cassandra database. Previously, there were some doubts when I was learning traditional relational database courses. Data such as pictures and real-time data streams were not well stored in traditional databases. After understanding the NOSQL type database, I finally find my answer. After simple use, I found that there are many differences with traditional databases. In particular, many operations in the SQL language cannot be directly implemented in CQL. Regarding the processing of big data, we focus on the MapReduce programming model. The model was mentioned before when it came to learning parallel courses, but it was only used to understand parallel thoughts and was not associated with using it to process big data. In the process of processing big data, we must always have the idea of parallelism and distribution. It is obviously not enough to focus on one machine.

Finally, I learned about the Tensorflow deep learning system. This is a combination of big data and machine learning since a large amount of data is inevitably generated in the machine learning process and the training of the model also requires a large amount of data support. From the simplest linear regression to the convolutional neural network, I have made some simple understandings.

Throughout the project, my programming skills were exercised, knowing how to design and build a complete project and choose the right framework and tools to implement it. This is the ability that I was extremely lacking before. In addition, through this project, I also learned some of the latest and most popular programming ideas and tools, which are difficult to obtain in the school curriculum.

Overall, this project is full of rewards for me. I must thank Doc. Zhang Fan for his patient and helpful instructions and guidence.