MIT Remote Course: Big Data Course Report

Zhe Luan
Advised by Dr. Fan Zhang
September 16, 2019

Course Summary

As Big Data has recently become an indispensable approach in both the academy and the industry of all the fields that are related to analysis and utilization of information, I as a B.S. in computer science think it is beneficial for my future career as either a researcher or an engineer, to be exposure to the widely used Big Data knowledge and techniques. Dr. Zhang's remote course is a great experience for me not only because of the knowledge he presented but also the opportunity to practice these techniques along with my programming skills in the hand-written digit recognition webservice project.

After a quick recap of basic knowledge of the programming language, developing environment and subversion control tool we used in the lecture and the project, Dr. Zhang introduced the idea of container techniques and the thriving implementation and platform Docker. Unlike virtual machines obtain full access to resources of the host machine and operating system through its hypervisor and the guest operating system, container shares only a portion of the kernels of the host operating system without guest operating system. Container use the union file system and stack file system to organize the dependency installation so that it can minimize the storage usage. In my opinion it seems the container engine is a generalization and a development of the idea of java virtual machine. It's more portable and lite-weighted than a virtual machine and more flexible than java virtual machine. The lite-weighted feature is achieved by better resources usage strategy than virtual machine and the flexible feature due to not imitated to Java. Developers can have the applications have any kind of dependency if necessary, without being worry about environment of new colleagues and the clients. By using the container engine, users are provided environments which are exactly the same as that of the developers' expectation. As long as the tasks are not handling a server with multiple virtual machines running, container can provide a more convenient solutions of

deploying applications for both developers and clients. Also wrapping up all the run time dependency in a docker image greatly improve the efficiency of the development procedure in the version control of the libraries and packages.

Flask was also discussed in the lectures as a great start point of webservice microservice framework for deploying service easily online. Incorporating with the idea of RESTful, we can develop our API for the any project very quickly. However due to lack of experience in programming web application, it took me incredibly more time than my expectation to figuring out the structure and appropriate usage of request in flask framework during the development of the hand-written digit recognition webservice project. Moreover, getting use to deploying service to web could be useful for my future career, since it's a better way to have my products or results got more exposure.

We discussed the difference between traditional relational database and NoSQL database. We also had a touch of the model and architecture of the NoSQL database. Then we went through Cassandra, as an example of modern Non-relational distributed database system. The traditional relational database cannot satisfy the requirements for the reliability and fast-response when manipulating bigdata. Cassandra partitioned the data records onto multiple clusters and nodes for fast response. By monitoring timestamps and consistency level of data replicas Cassandra ensure the correctness of the data cross the distributed system without reducing the concurrency of the system. The most impressive design of NoSQL database for me is the idea that by efficient usage of computing resources and storage capacity of distributed system to avoid join and union operations.

In the last part of the course, I gained understanding of MapReduce model and its implementation Hadoop and Spark. MapReduce is model for cluster-computing. As its name, MapReduce has two phases. Map phase converts the dataset into a map and hashes and partitions it onto reducers. Reduce phase process data partitioned at each reducer and reduce the result together. It's noticeable that Cassandra has the similar partition procedure as MapReduce. From my understanding, appropriate partition strategy is essential to scalability of the distributed cloud computing system.

We talked about Spark as a better data process framework than Hadoop and its modules and components. Due to the limited amount of time of this course and project, it is not possible to have experience of Spark, I will try to get started with Spark by myself. We not only discussed were all data processing frameworks, but also data visualization tools. Dr. Zhang introduce and demonstrate AmChart. I had working with terminals simple charts and plots for four years, I think this is a good chance to learn how to present my outcome in a better way.

Dr. Zhang wrapped up the course with an overview of architecture of Big Data technology. As the foundation of the cloud-computing, server providers like AWS constructed the infrastructure of Bigdata. As leveling up in the pyramid of Bigdata ecosystem we have provisioning service to do infrastructure management, run time service to provide base of executing and storage, then we get service management and scheduling and finally we have application products at the top serve for various purposes.

Other than the course content, Dr. Zhang often provide extra topic for further study which are quite helpful for me.

As a conclusion, after this course, on the hand I learned most recent and widely used knowledge and techniques in Bigdata field, on the other hand I strengthened my understanding and sharpen my skills in software development, database and parallel computing through both the lecture and the project.

Project Report

Introduction

Deep learning frameworks make it possible for everyone to build and use power deep learning model to perform various tasks in fields like computer vision and natural language processing. But building and training the deep learning model is not applicable for ordinary client other than developers. Deploying service that can access to the deep learning model through microservice on web is a great choice. Speaking of developing, setting up the working environment can introduce great

complicity for both developers and researchers. The container techniques presented by Dr. Zhang in the course is the most recent and widely used solution to this issue. This project builds a hand digit recognition service using, Flask, docker container, TensorFlow and Apache Cassandra.

Methodology

The hand-written digit recognition service will be deployed to web through flask microservice framework in the RESTful API fashion. The user can use curl command to upload images to get recognized by using the RESTful API of this service. The server side of this application is consisting of a flask sever application, a digit recognizer module and a database module. The digit recognizer module is a deep convolutional neural network model built in TensorFlow was trained with mnist dataset of handwritten digit images. The trained model parameters were saved to checkpoint files so that every time the server launches, the digit recognizer module can directly load the model without re-training. The database module connects to an Apache Cassandra cluster to provide non-relational database service to store the digit recognition request records. After the user uploaded image was recognized by the server, the server saves the image file to filesystem and store the image file name, recognition result and the service request timestamp to the database. For portability and convenience, the server application will be wrapped into a docker image so that no needs for environment setting for development and client-using in the future.

Experiment

In this part, the implementation choices made during the developing will be discussed.

For the reusability, the recognizer module has two versions. One version is for local environment, it uses OpenCV library for image preprocessing, and has a main function. Another version which use Pillow instead of OpenCV is for a light-weighted docker container environment. Pillow has sufficient functionality for the current image processing procedure and by using it can reduce the size of the docker image. It requires the docker image use Linux OS as the base image to use OpenCV

library. Since there was a great internet connection issue occurred during the development and Linux OS images are relatively huge in this condition, building docker image is quite time consuming. But for local environment, developers may have already set OpenCV library up and using OpenCV can fulfill further development requirements.

The image pre-processing procedure includes converting the user uploaded image to grayscale, resizing the image to mnist format which is 28 by 28 and reshape the matrix represent the image to (1,784) tensor format.

In the database module, Cassandra cluster has a keyspace for the service and only one table to record the digit recognition request. The column family of the table is the (filename, recognition result, request timestamp) with request timestamp as primary key. Setting timestamp as primary key for better query performance if more nodes and clusters are added to the database in the future. Considering the future development, a second table should be added to the keyspace which is image by user. This table is in the consideration of performance of query so that query for request logs of a user does not requires searching across node.

For dock container part, as it has been mentioned above, there are two possible approach to struct an applicable docker image for the server application. One is using official docker image of TensorFlow as base image, while another is using Linux OS like Ubuntu as base image. By using TensorFlow image, there is no need to install a guest OS in the container which is storage efficient. But the built-in version of python in this image is 2.7 which is going to lose the support from Python team and may cause some minor fix in the code, since the code was programmed in Anaconda Python3 environment. In exchange of huge size, Ubuntu OS provide a more familiar environment in the container. As it was stated before, due to the internet connection issue, the current version docker image is using TensorFlow image as base image and it will switch to Ubuntu image when the internet connection issue resolved.

Conclusion

During the development of this project, the knowledge and techniques obtained from the lectures was practiced and implemented as modules of the webservice of the digit recognition. This is not only a chance to working on the trending technology of Big Data, but also an introduction to an advanced approach of software and service development.