International Core Journal of Engineering Volume 7 Issue 5, 2021
ISSN: 2414-1895 DOI: 10.6919/ICJE.202105_7(5).0056

Research on Training Environment of ML based on Microservice

Jingyi He[1,a], Xichuan Hu[1,b], Zezhong Chen[2,c]

[1]School of Information Engineering, Shanghai Maritime University, Shanghai 201306, China;

[2]School of Software Engineering, East China Normal University, Shanghai 200062, China.

[a]hejy123@qq.com, [b]huxc@shmtu.edu.cn, [c]czz890@qq.com

Abstract

The training environment of ML(Machine Learning) based on Microservice integrates the user operation interface, data storage and related computing resources through a highly decoupled fine-grained service interface. It strives to add new computing services when the consumption of computing resources increases rapidly. In this way, there is no obvious change in the operating efficiency of the training task. When the code operating environment has not been fully established or the environmental requirements have changed, the system can seamlessly switch the operating environment. When the number of tasks is large or the demand for computing resources is large, as long as there are enough execution machines, the system can still make all tasks run normally. The system design mainly considers the application of Python language, the communication method between services is gRPC, and the user interface is rendered to the browser. The system has obvious practical significance for solving the problems of complicated environment configuration and relatively insufficient local computing resources in the process of machine learning training.

Keywords

Microservices; Machine Learning; Data Storage; Computing Resources; gRPC.

1. Introduction

Some machine learning beginners often find it more cumbersome and prone to errors when configuring development environments, especially when configuring multiple development environments, which affects the effect of training and learning. When applying a machine learning development platform, not only the hardware environments be analyzed and the corresponding installation package must be installed, but also a lot of software libraries must be installed. The algorithm of machine learning is more complicated, and the environment configuration requirements are also relatively high, and ordinary personal computers often cannot meet the requirements.

Compared with the local code compilation environment, the online compiler is remote and lightweight. Use the browser to open the corresponding webpage, the learner can edit the code online, run it online and get the result faster. It is also convenient to switch between different compilers. There are many online compilers that can adapt to various programming languages, such as techio[1], codingground[2], and rextester[3]. Although these compilers can be used as tools for learning programming languages, they cannot debug machine learning applications due to the lack of relevant high-level tool libraries. Constructing an online machine learning training system based on the microservice architecture has obvious significance for improving the learning performance of machine learning beginners.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202105_7(5).0056

2. Web technology for microservices

2.1 Web technology based on Django

Many popular web technologies are often combined with multiple technical frameworks, and technical applications are more complex, such as JAVA Web, ASP.NET, etc. Django is a typical representative in the Python-based Web technology framework, which is used to implement the user operation panel of the machine learning training system. Because the B/S model has the characteristics of cross-platform and cloud storage, the system can be used on a terminal with a browser without ever installing a development environment, so the front-end display page design of the machine learning training system is based on the B/S model with BootStrap framework technology.

2.2 Microservice background technology

Microservice architecture has the advantages of distributed storage, high availability, scalability, and intelligent operation and maintenance [4-6]. Service collaboration is the key consideration of microservice architecture, which is more advanced than monolithic architecture and service-oriented architecture.

There are many popular microservice architectures. Domestically, there are Alibaba's Dubbo[7], Sina Weibo's Motan[8] and Tencent's Tars[9], and foreign countries include Pivotal's Spring Cloud[10], Google's gRPC[11], and Facebook's Thrift[12], etc. Because gRPC supports cross-language operations and is open source, it is selected as the RPC framework of the online machine learning training system.

The development technology used by each service in the microservice architecture may not be exactly the same and can run independently, which will produce a diversified operating environment. Because Docker is a virtualization technology with a sandbox mechanism and supports diversified operating environments, Docker technology can be used to deploy different services. Use multiple containers in the server and use Docker images with different versions of software installed, where image A is Tensorflow 1.6 and image B is Tensorflow 2.3. During the training, according to the user's version selection, it can be passed to different containers, and the execution result is returned to the front end. The system not only has various services, but also considers the communication between services, including information interaction mode, communication protocol, implementation framework, message format, etc.

2.2.1 RESTful protocol

RESTful is a commonly used communication protocol for web development. RESTful API makes it very convenient and intuitive to call resources, and it also reduces the complexity of the service. State transformation in REST is a set of architectural constraints and principles, with obvious architectural styles. A software architecture that conforms to the REST architectural style can be considered RESTful. The RESTful software architecture emphasizes the use of URLs to locate resources; resource operations are performed through performance, resources can be XML, JSON, binary files, etc.; the client uses 4 types of HTTP verbs (GET, POST, PUT, DELETE) to make server-side performance Layer state transformation. The input and output interfaces involved in the user operation interface of the online machine learning training system based on the microservice architecture all have a RESTful style[13].

2.2.2 RPC protocol

The remote procedure call RPC service allows a program to call a class method or function in another address space. RPC is a protocol hidden at the bottom of the network. It can call remote programs like local services, so that the system throughput can be improved without high coding costs. There are many kinds of microservice RPC frameworks.

Google's gRPC framework is an open source high-performance framework based on the http2.0 protocol, and its efficiency is several times higher than that of the RESTful framework based on the http1.1 protocol. To transfer method calls, call parameters, response parameters, etc. between the two

ISSN: 2414-1895 DOI: 10.6919/ICJE.202105_7(5).0056

servers, these parameters need to be serialized. gRPC uses the protocol buffer syntax (proto file). Through the proto syntax, the method to be called, parameters, and response format can be defined, which can easily complete remote method calls, and is very conducive to extending and updating parameters. The online machine learning training system based on the microservice architecture uses gRPC remote calls for the management of tasks, execution machines and other related data. Taking adding a new task as an example, the proto content involved is roughly as follows:

```
service TaskMng {
  rpc insert_idle_task (insert_idle_task_Request)
  returns (insert_idle_task_Reply) {}
}

message insert_idle_task_Request {
  string data_set = 1;
  string code_content = 2;
  string code_tag = 3;
}

message insert_idle_task_Reply {
  string task_num = 1;
}
```

Figure 1. Add new task proto code

The proto content of the new task is mainly divided into two parts: operation function interface and input and output parameter format. The operation function interface is insert*idle*task, the input parameter format is defined as insert*idle*task*Request, and the output return value format is defined as insert*idle*task*Reply.

3. System Use Case Model

The use case model of the system mainly includes three types of users: tourists, registered users, and administrators.

(1) Tourist: The tourist has not registered and logged in. You can use the public case and data set that comes with the system with the corresponding default operating environment running case, or you can write your own code, select the public data set, select the operating environment to run the code, and view the running logs and reports.

(2) Registered user: already registered and logged in. Not only can you use the above functions, you can also upload private data sets and management tasks, such as deleting tasks and filing tasks as cases.

(3) Administrator: Unified management of users, cases, tasks, execution machines and data sets.

An use case diagram for tourists is shown in Figure 2.

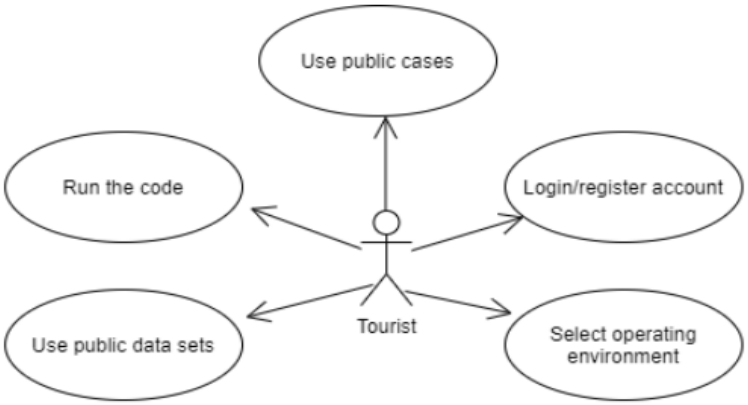

Figure 2. Tourist use case diagram

ISSN: 2414-1895 DOI: 10.6919/ICJE.202105_7(5).0056

An use case diagram of a registered user is shown in Figure 3.

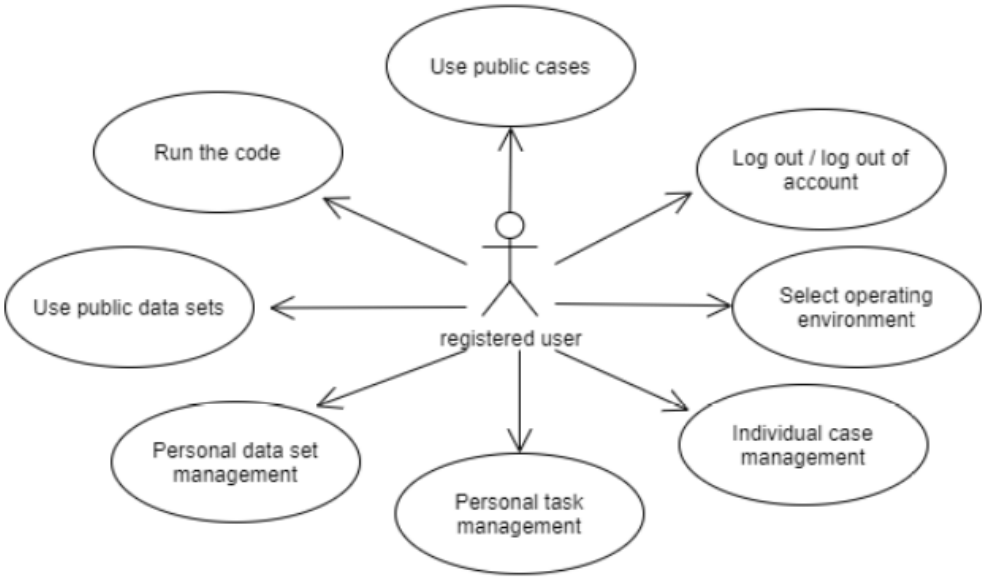

Figure 3. Use case diagram for registered users

The administrator use case diagram is shown in Figure 4.

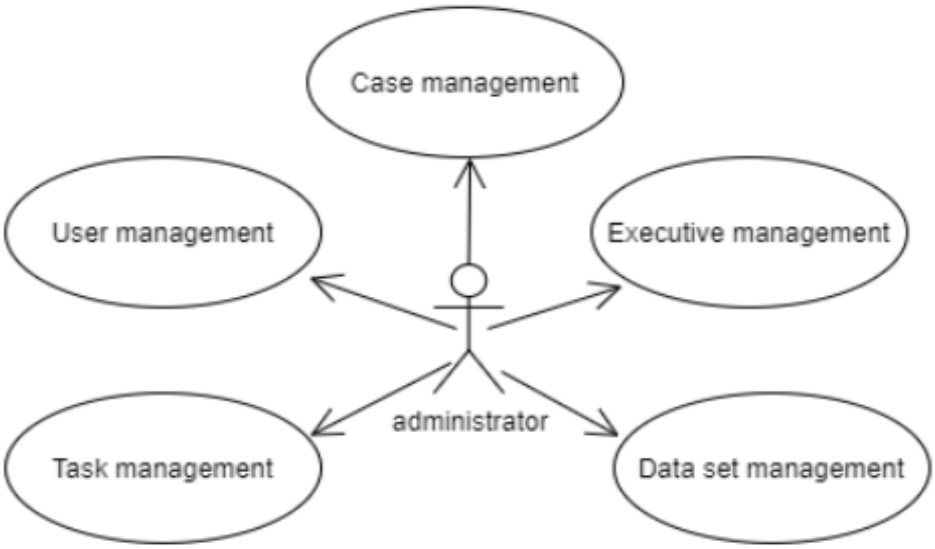

Figure 4. Administrator use case diagram

4. System design

4.1 System structure

Under the microservice architecture, the process split mode is adopted, and various services are organized based on the tasks that need to be performed, and the task data changes as the basis, and the appropriate service boundaries are determined, and the overall system is split into data set management and task management, Execution machine management and many other microservices. This split method ensures that each service focuses on its own responsibility area and reduces the coupling between services. The code size of each microservice is relatively small, and it is relatively easy to maintain the microservice [14-18]. When the system is deployed and operated, each microservice can be deployed independently, and the system can dynamically adjust the number of instances of the service according to the actual workload of the service. Therefore, the machine learning online training system with microservice architecture is more conducive to ensuring the reasonable use of hardware resources.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202105_7(5).0056

The services are not directly deployed on the server operating system, but are deployed into the container using the Docker engine. At the same time, the shared folder function of Docker is used to interact with the file system of the operating system to facilitate file sharing.

All service interfaces are registered to a unified service registry, and the service caller obtains the service interface by reading the service registry, which increases the portability and dynamic fault tolerance of the service. Part of the service data management is more complicated, and the database can be used to manage the data.

Currently, the microservice gateway is set to interact with the WEB interface to facilitate the management and invocation of the service interface. At the same time, when the platform is expanded, you don't need to care about each service, only the API provided by the microservice gateway.

The overall architecture of the machine learning online training system can be shown in Figure 5:

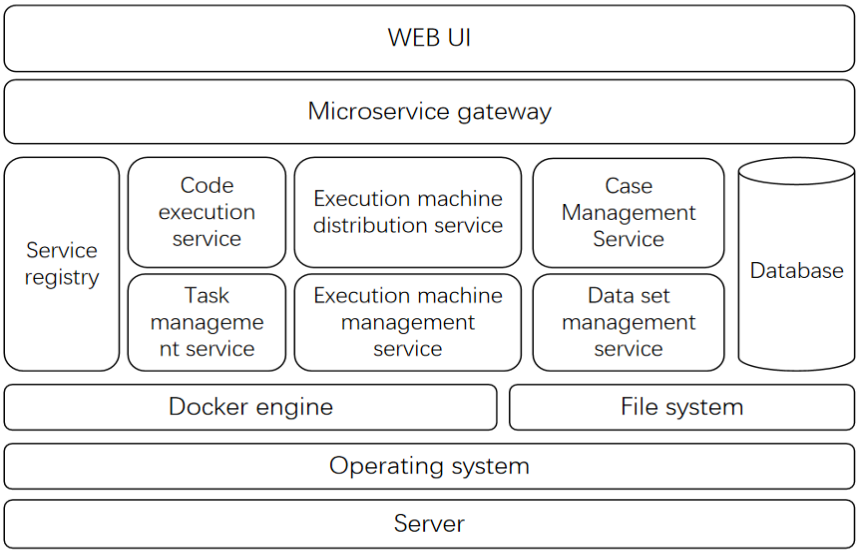

Figure 5. System architecture diagram

4.2 Module design

4.2.1 Service

The services involved in the system mainly include:

(1) User data management service: maintain user information and user operation data (cases, data sets).

(2) Task management service: maintain all users' tasks and task states (initial state, waiting state, execution state, stop state and completion state).

(3) Data storage service: Provide data set storage and CRUD (Create, Retrieve, Update, Delete) operations.

(4) Execution machine: A computer or container installed with a machine learning code operating environment will actively register with the execution machine management service and incorporate it into the entire system by starting the computing service. The execution machine can be deployed according to the scale of the actual number of users to ensure the reasonable use of hardware resources.

(5) Execution machine management service: realizes the registration and discovery of execution machines, and allocates execution machine computing resources to tasks through scheduling algorithms.

The task management service and the execution machine management service use RPC to communicate. As shown in Figure 6, the communication using the gRPC framework includes: registering tasks, returning to the task list, assigning executors, discovering idle tasks, executing tasks, discovering wait tasks, service registration, and service discovery.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202105_7(5).0056

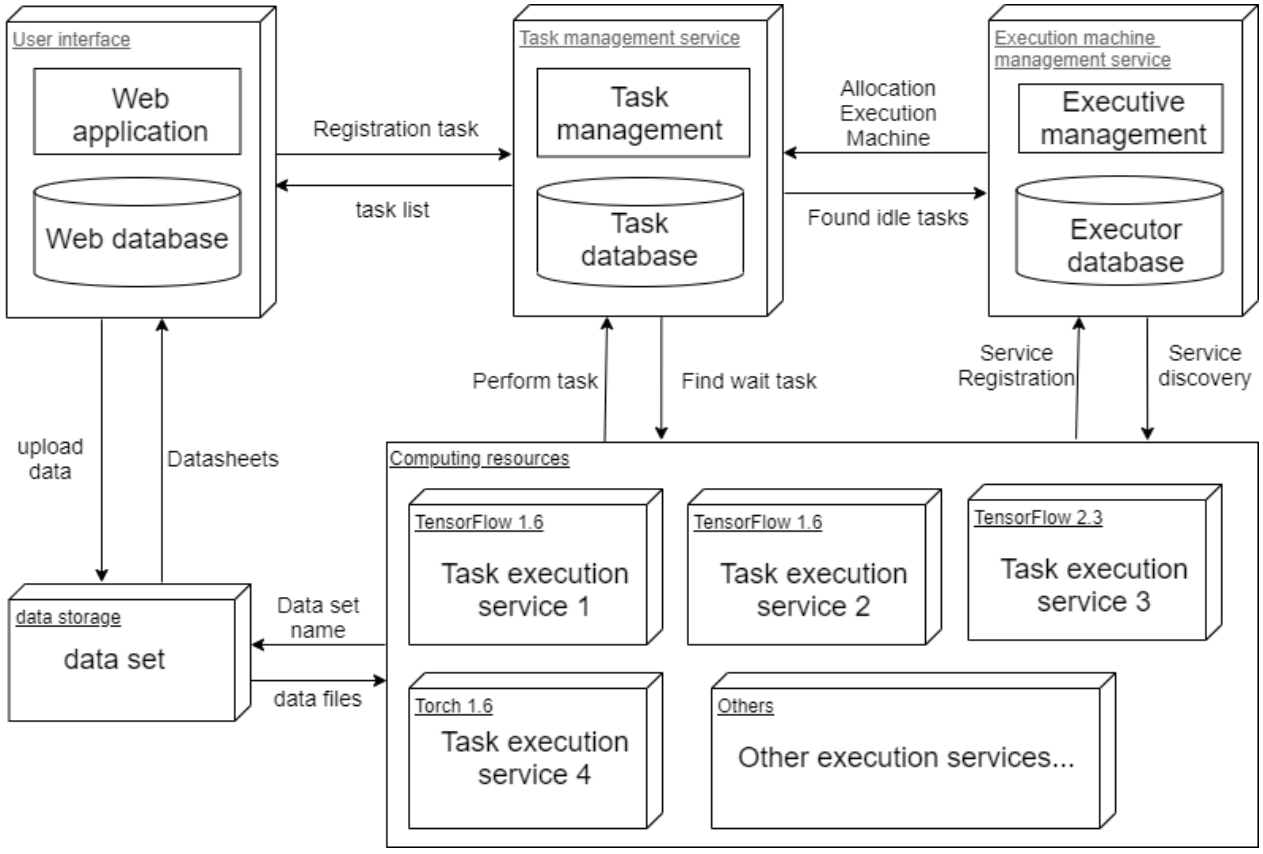

Figure 6. Relationship diagram between system services

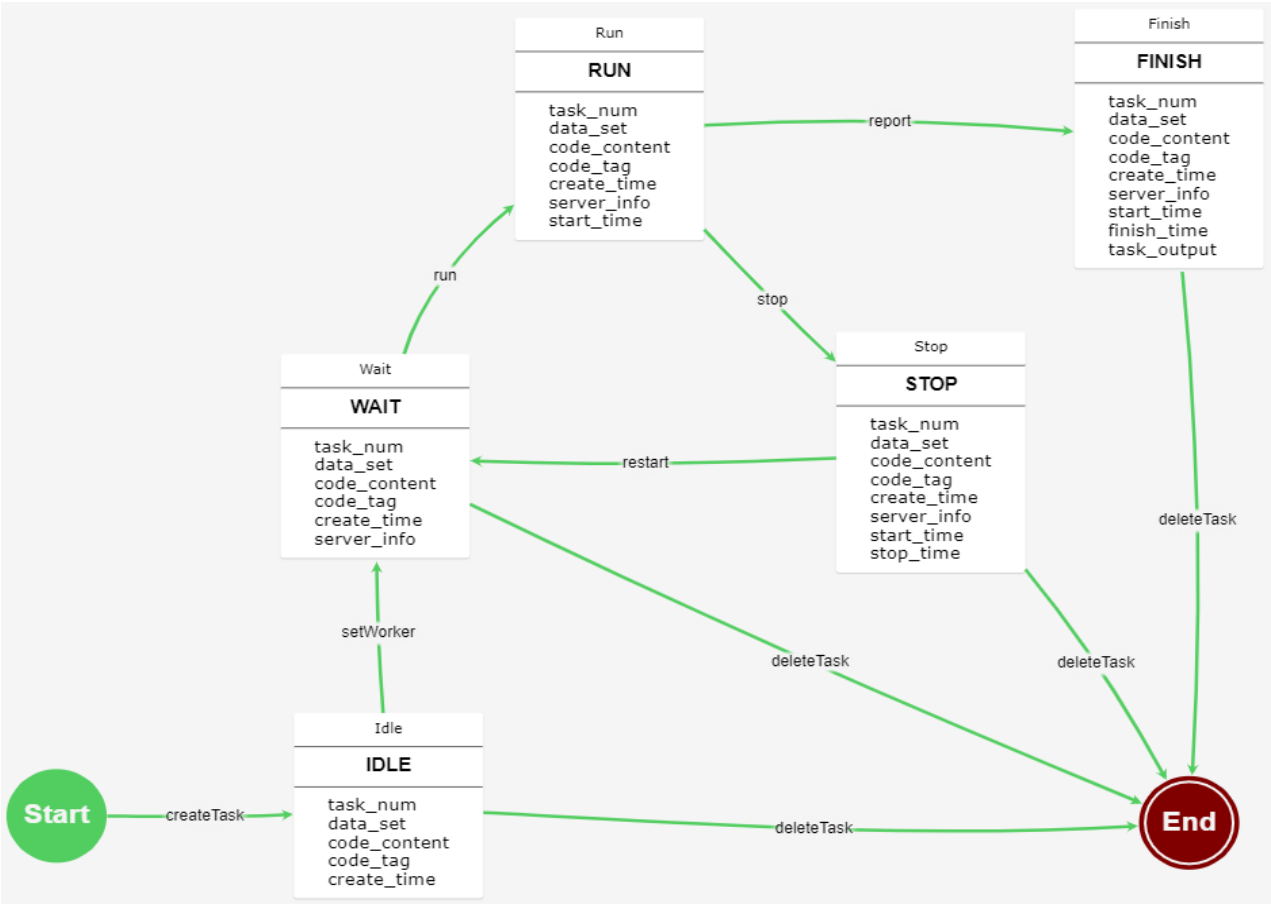

Figure 7. State machine diagram of machine learning computing task

ISSN: 2414-1895 DOI: 10.6919/ICJE.202105_7(5).0056

4.2.2 Task execution

The system's task flow is as follows.

(1) The user sets up the task data (code, data set and operating environment) and clicks to run. At this time, the task management service creates an initial state task.

(2) The executor management service detects the task in the initial state, and assigns an executor to the task to make the task wait.

(3) The executor detects its own task and starts to execute the task. At this time, the task is in the execution state. After the task is executed, the executor stores the running data, and the task is in the completed state at this time.

(4) If the task is not finished, it is forcibly stopped by the user, and the task is stopped at this time.

(5) The user deletes the task, and the life cycle of the task ends.

The task status change of the above operation process is shown in Figure 7.

The meanings of the task state attribute fields in the state machine diagram are shown in Table 1:

Table 1. Meaning of task status attribute fields

Task attributes	Description
task_num	Task number
data_set	Data set
code_content	Code content
code_tag	Code label (operating environment)
create_time	Task creation time
server_info	Execution machine information
start_time	Start running time
stop_time	Outage time
finish_time	End of run time
task_output	Task execution report

4.3 Development process

4.3.1 Iterative development

Compared with the traditional waterfall development model, the iterative development model only designs and implements a part of the product at a time, and gradually completes the integrated product. Each design and implementation phase is called an iteration. Iterative development has the advantages of reducing risks, obtaining early user feedback, continuous testing and integration, using changes, and improving reusability. Each iteration step is similar to the waterfall model, as shown in Figure 8:

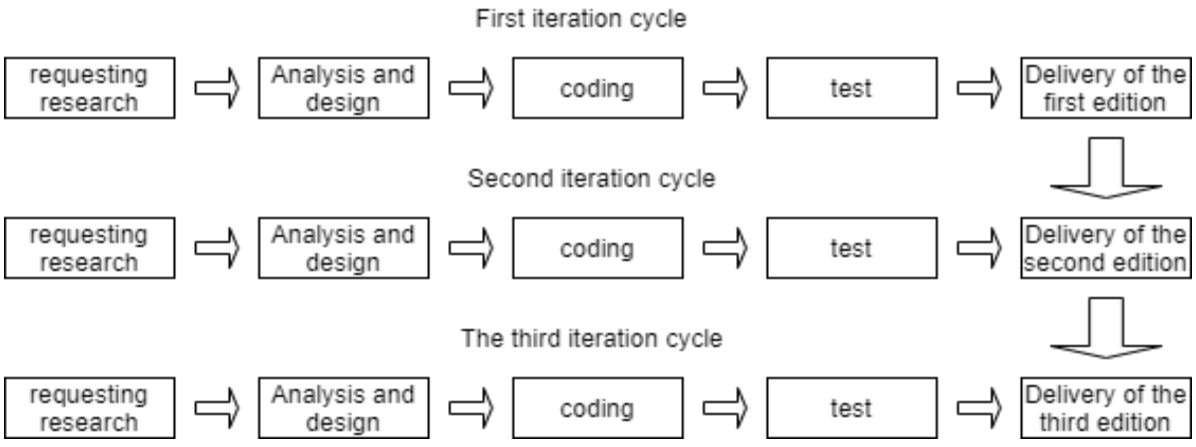

Figure 8. Schematic diagram of iterative development

ISSN: 2414-1895 DOI: 10.6919/ICJE.202105_7(5).0056

The machine learning training environment has undergone three iterations of development. Iteration 1, to realize the display function of typical cases in the textbook; Iteration 2, to increase the registration and login function, users can upload their own data sets for experiments, to increase the function of online writing and running code; Iteration 3, to increase the function of switching the operating environment version.

4.3.2 Tool-assisted development

The complexity of microservice architecture application development and deployment is far greater than that of monolithic applications, and it is difficult to complete manual configuration management by operation and maintenance personnel alone. DevOps advocates the use of automated task processing to achieve software delivery and infrastructure update, which is a necessary condition for microservice architecture application development and operation and maintenance. This project adopts the development process of DevOps concept to develop, realize the automated tools for development and deployment, and achieve the purpose of improving the efficiency of research and development.

When using gRPC in the development process, some common codes generated by proto are depended on in multiple services, usually one place is modified. In order to ensure the unity of the code, every module involved must be changed, which is very energy intensive. And error-prone. Therefore, the development of automated tool assistance can solve this problem, achieve the same code, modify one place, and take effect everywhere, reducing a lot of manual operations and the probability of errors. All in all, software engineers can use auxiliary tools to devote their energy to writing business interface definition files, and the two operations of generating common serialization code and copying code are completed by the auxiliary tool with one click, as shown in Figure 9:

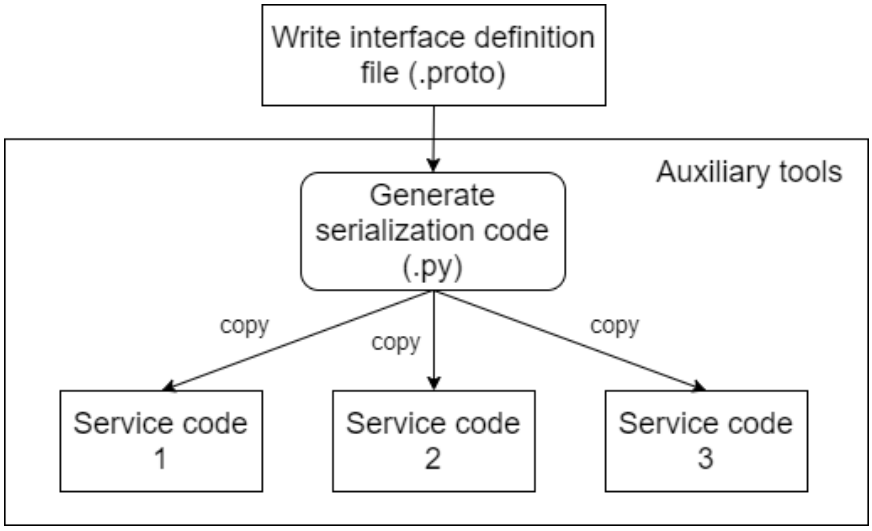

Figure 9. Schematic diagram of auxiliary tools

During the testing process of the machine learning online training system, since the system involves multiple service operations, it is quite cumbersome to manually start each time, so an automated script is also developed to start all services with one click to achieve the purpose of unified management of services.

5. Instance

The execution process of the machine learning online training system can be divided into the preparation phase and the execution phase. The preparation phase mainly includes the setting of three contents, namely the execution environment, data set and program code. These three contents can be set manually or through The case is automatically imported. After clicking the execute button, the

ISSN: 2414-1895 DOI: 10.6919/ICJE.202105_7(5).0056

system generates a task and enters the execution phase. Usually the tasks in the execution phase mainly go through four processes, namely Idle, Wait, Run and Finish. The following describes the detailed execution process of the tasks in the preparation phase and the execution phase.

5.1 User Interface

The user operation interface of the machine learning online training system is mainly divided into two functional areas, the left and the middle are the functional areas of the preparation phase, and the task list on the far right is the functional area of the execution phase. As shown in Figure 10:

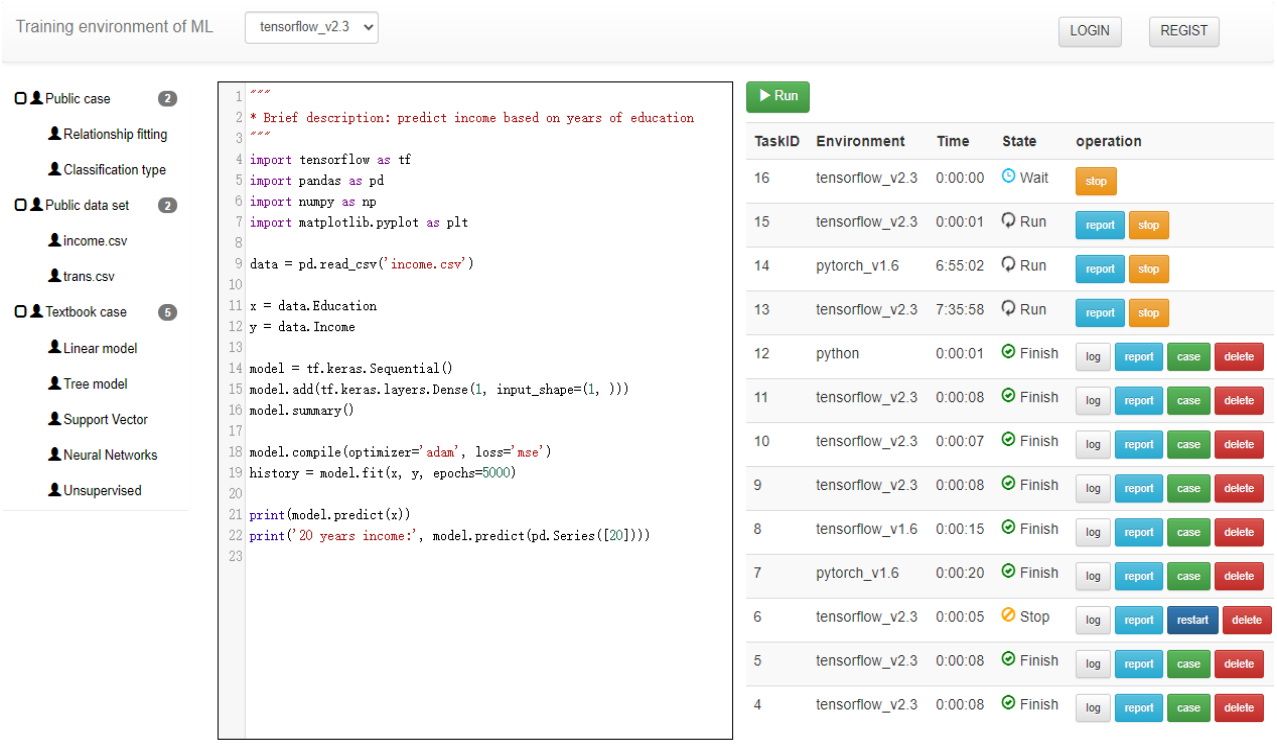

Figure 10. The user interface of the machine learning online training system

In the preparation phase functional area, the top is the execution environment selection, you can choose one of the four environments of tensorflow*v1.6*, *tensorflow*v2.3, pytorch_v1.6, and python; the left side is the list of cases and data sets, you can select the case to be executed and the The data set on which the case is executed; the middle part is the code display and editing area, where the case code can be displayed or customized code can be edited.

In the execution phase of the functional area, there is an execution button at the top. Click the "Run Code" button, and the system will create a new task, which will be displayed at the top of the task list. Each row in the task list represents a task, which displays the basic information, status, and operable items of the task. The basic task information includes task number, task environment, and time consumed for task execution. The operable items include viewing task running logs, viewing and downloading tasks Working directory files, stop tasks, restart tasks, archive tasks and delete tasks. Registered users can also convert tasks into private cases.

5.2 Operation process

The process of using this system can be divided into preparation phase and execution phase. The preparation phase mainly sets the data, and the execution phase mainly checks the task execution progress and execution results.

Take a case of inferring personal income based on years of education to introduce the core use process of the machine learning training environment.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202105_7(5).0056

5.2.1 Preparation phase

(1) Select the execution environment

Select the tensorflow*v2.3 environment from the drop-down menu in the upper left corner, because the experimental code in this example is planned to be developed based on the tensorflow*v2.3 environment.

(2) Set up the data set

In the public data set node on the left, select the income.csv data set. At this time, the data file can be used in the code. The data will be copied to the code execution path during the code execution. Therefore, the relative path can be passed in the code Read the data. E.g:

data = pd.read_csv('income.csv')

(3) Write code

In the central code editing area, write the code as follows:

```
"""
* Brief description: predict income based on years of education
"""
import tensorflow as tf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

data = pd.read_csv('income.csv')

x = data.Education
y = data.Income

model = tf.keras.Sequential()
model.add(tf.keras.layers.Dense(1, input_shape=(1, )))
model.summary()

model.compile(optimizer='adam', loss='mse')
history = model.fit(x, y, epochs=5000)

print(model.predict(x))
print('20 years income:', model.predict(pd.Series([20])))
```

Figure 11. Example of writing code for the central region

5.2.2 Execution phase

After preparing the data, click the "Run Code" button, the system will collect the data needed for a task, here is a triple, including (execution environment, data set, program code), and then use this triple data to create A newest task is displayed at the top of the task list.

At this time, the first line of the task list displays the execution progress of the task. When the task is just created, the task is in the Idle state. After multiple service processing, the task becomes the Finish state or Stop state. The processing process is as follows:

Step 1: The executor management service recognizes the task in the Idle state, selects a suitable executor according to the execution environment information in the triplet, and binds the executor information to the task. At this time, the task is in the Wait state, Waiting for execution by the executor, which contains a four-tuple information (execution environment, data set, program code, executor information).

ISSN: 2414-1895 DOI: 10.6919/ICJE.202105_7(5).0056

Step 2: The executor recognizes the task in the Wait state. If the executor information in the task quadruple is consistent with the machine, the task is read and the task is processed. In this example, the execution opportunity creates a folder as the working directory. First download income.csv to the working directory, then write the code into the file main.py, and finally execute the main.py program. If the program is executed normally, the task is converted to the Finish state; if the task is terminated by the user or the task execution times out during the execution of the task, the task is converted to the Stop state.

step 3: the user can view the running log through the log button in the task list on the right side of the interface; through the report button, view all file information in the work space of the second step, which can be downloaded from the platform to the local if necessary, as shown in Figure 12. Show. If the execution task is a typical case, the registered user can convert the task into a private case through the archive button in the task list; if the task is of no value, you can choose to delete the task.

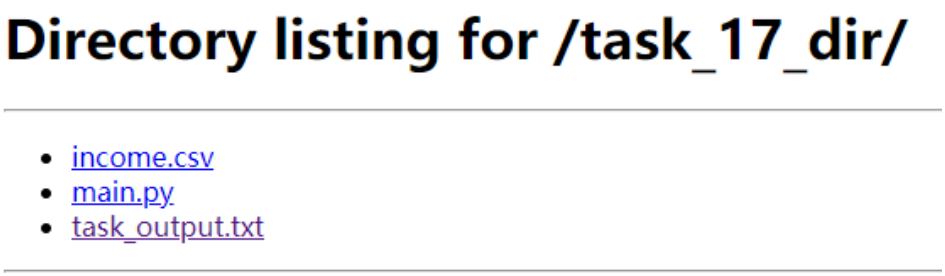

Figure 12. View interface of task execution report

6. Conclusion

The machine learning online training system based on the microservice architecture realizes online execution of machine learning cases, application service discovery and registration, inter-service calling, and configuration center three microservice features to realize core functions. It will make full use of the advantages of microservices, link tracking, service fusing, degradation, and current limiting to optimize the system and improve the overall robustness and maintainability of the system.

References

[1] Tech. io. techio Home[EB/OL]. https://tech.io/snippet,2020-11-12.

[2] Tutorials Point. tutorialspoint Home[EB/OL]. https://www.tutorialspoint.com/codingground.htm, 2020-11-12.

[3] Rextester.rextester Home[EB/OL]. https://rextester.com, 2020-11-12.

[4] Feng Zhiyong, Xu Yanwei, Xue Xiao, et al. Current status and prospects of the development of microservice technology [J]. Computer Research and Development, 2020, 57(5): 1103-1122. DOI: 10.7544/issn1000-1239.2020.20190460.

[5] Feng Hongwei. SOA collaborative software meets the new challenge of informatization[J].The Window of World, 2005(10):42-43.

[6] Zposion. The difference between SOA architecture and microservice architecture [OL].(2018-06-06) [2020-11-12].https://blog.csdn.net/zpoison/article/details/80729052.

[7] McCormac A, Calic D, Parsons K, et al. The effect of resil- ience and job stress on information security awareness[J]. Infor- mation and Computer Security,2018,26(3):277-289.

[8] motan user development guide[EB/OL]. https://blog.csdn.net/haponchang/article/details/94442955,2020-12-2.

[9] TARS-Focus on Microservice Ecosystem[EB/OL]. https://tarscloud.org/,2020-11-16.

[10] Spring Cloud Home[EB/OL]. https://spring.io/projects/spring-cloud.2020-11-12.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202105_7(5).0056

[11] Google. gRPC–A high-performance, open source universal RPC[EB/OL]. https://www.grpc.io/, 2020-11-12.

[12] Apache. Apache Thrift - Home[EB/OL].https://thrift.apache.org/,2020-11-12.

[13] Dai Fei, Liu Guozhi, Li Zhang et al. Microservice technology: architecture, communication and challenges[J]. Journal of Applied Sciences, 2020, Volume 38(5):761-778.

[14] Yin Jialing, Xia Fan, Gu Hang et al. Design and implementation of graduate student training system based on microservices[J]. Journal of East China Normal University (Natural Science Edition), 2019, (4):83-96.

[15] Gadgil H, Fox G, Pallickara S, et al. Scalable, fault-tolerant management in a service oriented architecture [C]//Proc of Int Symp on High Performance Distributed Computing. New York:ACM, 2007:1-15.

[16] Nadareishvili I, Mitra R, McLarty M, et al. Microservice Architecture: Aligning Principles, Practices, and Culture[M]. Sebastopol, CA: O'Reilly Media,2016.

[17] Yang Junwei, Ji Xin, Hu Qiangxin. Electric power cloud service platform based on micro service architecture[J]. Electric Power ICT, 2017,15(1):8-12.

[18] Fan Jing, Xiong Lirong, Xu Cong. Research and application of data integration in distributed enterprise service bus platform[J]. Computer Science, 2014,41(2):212-220