Hindawi
Scientific Programming
Volume 2020, Article ID 6929750, 13 pages
https://doi.org/10.1155/2020/6929750

Research Article

A Microservice-Based Big Data Analysis Platform for Online Educational Applications

Kehua Miao, Jie Li, Wenxing Hong, and Mingtao Chen

Department of Automation Xiamen University, Xiamen 361005, Fujian, China

Correspondence should be addressed to Wenxing Hong; hwx@xmu.edu.cn

Received 9 February 2020; Accepted 19 May 2020; Published 3 June 2020

Academic Editor: Chenxi Huang

The booming development of data science and big data technology stacks has inspired continuous iterative updates of data science research or working methods. At present, the granularity of the labor division between data science and big data is more refined. Traditional work methods, from work infrastructure environment construction to data modelling and analysis of working methods, will greatly delay work and research efficiency. In this paper, we focus on the purpose of the current friendly collaboration of the data science team to build data science and big data analysis application platform based on microservices architecture for education or nonprofessional research field. In the environment based on microservices that facilitates updating the components of each component, the platform has a personal code experiment environment that integrates JupyterHub based on Spark and HDFS for multiuser use and a visualized modelling tools which follow the modular design of data science engineering based on Greenplum in-database analysis. The entire web service system is developed based on spring boot.

1. Introduction

In recent years, data science and big data technology stacks have achieved explosive growth. In data science, especially machine learning, it mainly benefits from the improvement of computing power, especially the rapid development of GPU (Graphics Processing Unit), and the popularity of deep learning [1]. In terms of big data technology stacks, new types of big data tools such as Spark have gradually replaced some components in the traditional Hadoop ecosystem [2], and the update iteration speed is persistent. Based on the rapid development of technology, the modularization and subdivision of work methods become more and more obvious. Traditional working methods, from setting up an experimental environment to data acquisition, data processing, modelling training, and data prediction, are often performed in a unified manner [3]. And this way of working is obviously not suitable for new data science research methods. At present, people pay more attention to teamwork, so infrastructure environment sharing, data sharing, and model sharing have become mainstream workflows. Of course, a personal workspace based on sharing conditions is also essential.

However, the development of big data [4] and the deployment of operating environments often require development, operation, and maintenance personnel with professional knowledge to build and maintain [5], which is more difficult for ordinary teams, especially novice or student team. People often waste a lot of time on infrastructure and environmental construction, and these troubles are often accompanied by huge operation and maintenance problems. People need a multiuser based infrastructure environment platform [6].

For data analysis of actual business scenarios, people's thinking is focused on not only code but also text, pictures, and even mathematical formulas. People need a working code area for a markdown-like environment [7]. The Jupyter Notebook provides such a working environment, but the Juypter Notebook is aimed at single-user members. For web-based platforms [8], JupyterHub is required [9]. For workers with big data needs, they also need to integrate the experimental environment based on Hadoop [10], Spark, and JupyterHub.

In data sharing and data stream processing, the security protection of data is often the most troublesome problem in the analysis of actual business scenarios. People often do not want to perform compatibility processing and desensitization of various data migrations. In team collaboration, the platform needs to be integrated with tools that can analyze data within the database. For team collaboration, data flow processing and model sharing are often accompanied. These abstract contents are often not suitable for collaborative analysis in a short time. People need a visualized modelling environment.

In this paper, we present Qunxian Platform. Qunxian Platform is a big data analysis platform based on microservices. It helps to realize the sharing of data, computing power, and infrastructure resource. Moreover, it enables users to use a more friendly environment to record and share the experimental process and the visualized model. It uses JupyterHub and visualized modelling as its two main applications. The platform's infrastructure environment components are the big data component and the data science environment. In order to easily adapt to the characteristics of rapid environmental changes, the platform uses microservices [11] as a technical dependency; in particular, it uses Docker [5, 12]. The environment's distributed file system uses the HDFS (Hadoop Distributed File System) [13]. The parallel computing architecture uses Apache Spark. Based on Spark and HDFS, Jupyter Notebook is used as the user's code engineering environment. And JupyterHub manages a multiuser notebook environment in a unified manner. In terms of visualized modelling, the Greenplum-based in-database analysis method is used to implement the modelling module of the visualized modelling application with the built-in MADlib algorithm and custom algorithm interface [14]. Because the entire platform system will be oriented to different users and different user-environments, users will share computing and data resources of the server. This system locates B/S (Browser/Server) type systems [15]. Based on the B/S type system, the platform is built using a framework based on spring boot and uses element-admin as a back-end front-end solution [16]. Moreover, it uses Greenplum and MADlib to implement in-database data calculation to protect the data, quickly use the data, and store the model. Based on Element-UI [17], the drag-and-drop method of the back-end module is realized [18], and the visualized data modelling is further realized.

The remaining part of this paper is arranged in the following format. Section 2 presents related works in this field. In Section 3, we visit the platform's architecture without the app layer and support layer. Then, we describe the two web-based applications in Section 4. After that, we give our experiment configuration on Google cloud from Section 5. And finally, we conclude with Section 6.

2. Related Works

As described in the introduction, there is a certain demand to get advanced tools for storing and sharing some kinds of environment for data science and big data research, which is configured by specific engineering staff on cloud computing infrastructure for individuals and research teams. A lot of systems have been developed in recent years and solved some of the problems mentioned above, including big data environment infrastructure constructing [4, 10, 19], microservice-based platform [5, 11, 20], and online educational programming platform [21–23]. However, several issues remain, which are not addressed well by existing approaches and systems.

The first issue is supporting a platform which is based on data science computing and big data ecosystem environment as online services. The use of the service-oriented approach can enable wide-scale sharing and deployment of the new data model as well as automation of scientific tasks and composition of applications into new services [24]. However, the existing web-based toolkits or platforms either do not construct a stable big data ecosystem environment or provide the raw and cumbersome command shell, which is unfriendly for students to getting started with.

The second issue is teamwork corporation based on big data environments platform. The existing online programming platforms always are designed for private use [21, 24, 25]. However, the existing platform usually lacks personal volume storage for a long time as a personal application. Moreover, the personal application does not agree with teamwork nowadays. In this way, to share personal ideas in the teamwork, especially for students in an educational way [21], we need to share the visualized model without the raw code only.

Qunxian addresses the described issues by relying on microservice. On the infrastructure building, we implement the docker-compose service. On the web service building, we implement the spring boot.

Spring framework offers flexibility to all the configure beans in many ways such as annotations and XML [26]. With the number of features being increased, the complexity also gets increased and configuring spring applications becomes error-prone and tedious [27]. Spring boot helps to address the complexity of configuration. Also, spring boot is much more helpful to integrate the microservice applications [28].

To help to build the concept of bringing the science to data, on the web front-end, users will try to manipulate the data in a visualized way. To make it happen, we implement the vue.js to help the front-end communicate with the back-end. Technically, vue.js is a progressive framework that helps developers to build a user interface for website development. It is designed to be applied layers by layers from the bottom-up [17]. In particular, it focuses more on the view layers. These features help to integrate many modern toolchains and various supporting class libraries. Also, it helps to reduce the difficulty of web development. To build a production-ready front-end solution [29], we integrate the element-admin, which is based on Vue and uses the UI Toolkit Element-UI. Vue-Element-Admin is a back-end front-end solution based on Vue and Element-UI. It implements the latest front-end technology stack, built-in i18 international solutions [30], dynamic routing, permission verification, and refined typical business models and provides rich functional components, which can help web

developers quickly build enterprise-level product prototypes. All the tools and frameworks help to build a user-friendly web-based platform.

3. The Platform's Architecture

Qunxian follows the Platform as a Service (PaaS) and Figure 1 gives a high-level overview of the Qunxian platform architecture, which is described in the sequel.

3.1. Hardware Layer

3.1.1. Cloud-Based Big Data Platform. Cloud computing is the provision of computing services providing rapid innovation, elastic resources, and economies of scale through the cloud. For cloud services, web developers usually pay only how much they use, which helps reduce operating costs, enables the infrastructure to operate more efficiently, and adjusts the use of services based on changes in business needs [31]. Its simplicity is the type of computing that depends on shared computing resources instead of local servers. Qunxian Platform decided to take the cloud computing service as the hardware layer and it would get the below-mentioned benefits:

(i) It greatly reduces IT and labor costs

(ii) It is more scalable and offers better and secure storage

(iii) Collaboration and effective communication platforms are provided

(iv) Best work practices and flexibility are received

(v) Access to automatic updates for your IT requirements is included

Nowadays, for cloud computing services, Google Cloud Platform (GCP), Amazon Web Services (AWS), and Microsoft Azure hold a ruling position among the cloud companies. Qunxian chose the Google Cloud Engine, the module of GCP, as the hardware service.

3.1.2. Docker. Based on cloud service, we decided to use Docker as a deployment methodology to manage the hardware layer efficiently. All individual modules (i.e., big data components, such as Hadoop and Spark, web architecture, and JupyterHub service concerned) rely on different operating environments, such as different Java versions. To coordinate them smoothly, without any compatibility problem, we apply the Docker container to separate them into individuals. Also, as the foundation of microservice nowadays, Docker helps to simplify the operation and maintenance work, which is the main work after the first time to deployment because of the frequent updating. By the benefit of the Docker container, it is an easy case to update the individuals that needed to be updated instead of all the system architecture. Also, the Dockerfile and YAML files help to record the deployment process for the reference of next time deployment.

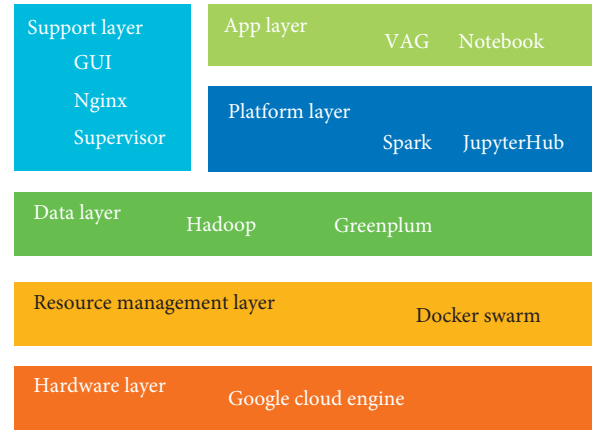


Figure 1: High-level Qunxian platform architecture.

3.2. Resource Management Layer. With the concern of the development stage and the number of users, we do not apply Kubernetes, which is the most popular Docker orchestration tool, to deploy Docker containers on a cluster. We apply Docker Swarm, which is the default orchestration tool of Docker, to construct our Resource Management Layer. On top of Docker Swarm, we implement Docker Compose as a collection of communicating containers that can be scaled and scheduled dynamically [32].

3.3. Data Layer. In the data layer, we deploy the HDFS, the Hadoop distributed file system, and Greenplum, a massively parallel Postgres for analytics relational database [19]. HDFS and Greenplum work with Spark, set at the platform layer, independently.

3.4. Platform Layer. In the platform layer, we deploy the JupyterHub, which is a multiuser server for Jupyter notebooks, as the back-end with containerized big data environment based on Docker.

Table 1 gives Docker components of the platform, which build up the basis big data ecosystem environment of Qunxian.

4. Two Web-Based Applications

Based on the platform layer, we introduce the app layer, which consists of two web-based applications. We describe them as follows.

4.1. Online Programming Application. For students to get the most of Qunxian's high-performance computing (HPC) resource, Qunxian applied JupyterHub, the platform layer's component, the data science, and big data ecosystem environment as a web-based application with the help of Docker.

Based on the platform layer, a default way of working with Apache Spark is to launch a cumbersome command shell from the terminal, which makes it very hard to present information. To go beyond that and make data analysis more shareable and reusable, we choose Jupyter Notebook.

Table 1: Big data Docker component.

Docker name	Base images	Features
Hadoop-Base	Debian	Hadoop basis with OPENJDK
Datanode	Hadoop-Base	HDFS slave server
Namenode	Hadoop-Base	HDFS master server
Spark-Base	python	Monitor the containers and Spark Base
Spark-Master	Spark-Base	Put Spark under supervision
Spark-Worker	Spark-Base	Expose ports for each application

Jupyter Notebook, which spawns by JupyterHub, provides an easy way to use interactive programmatic interface which is accessible from the browser that alleviates the burden for students to implement their Scala or python job via cumbersome command-line methods.

Figure 2 gives a high-level overview of online programming applications on the app layer based on JupyterHub and Docker. It allows the Authenticator to be a GitHub user, which helps the team to cooperate with their work based on the project. Docker Spawner spawns single-user Jupyter Notebook servers in separate Docker containers based on their separate Docker volume, which helps to persist every user's notebook directories. Also, for the persistence of JupyterHub data, we deploy a single Docker volume on the host.

4.2. Visualized Modelling Application. Visualized modelling application is a data science modelling tool, which helps students to understand the high-level machine learning pipeline designed by teachers or other researchers. We apply the classic machine learning module, which includes data processing, feature extraction, and model selection. Students can explore their data on this tool without building their data research way from scratch. Also, for research way, we developed a new module for students and researchers to explore their new idea on model building.

Figure 3 gives a pipeline case on the web-based visualized modelling application. It is a simple case for students to understand and repeat the experiment on each part. From model constructing to model deployment, every single part in data science can be selected from the given choice or build new components based on the pipeline.

5. Experiments

We performed an experimental evaluation of Qunxian with all server-side platform components being deployed on Google Cloud Engine. The boot disk is based on CentOS 7 OS image with the standard persistent disk of 2TB. And the machine type is n1-standard-2 with 64vCPU, 240 GB memory. We add tags and firewall rules to allow specific network traffic from the Internet, including MySQL service [33], Greenplum service, web service, and Hadoop service on the VPC (Virtual Private Cloud) network/Firewall rules in Google cloud platform. A minimal tuning of the system and the Nginx web proxy server [10] was done to support a high number of concurrent connections. In the following part, we will first describe the service deployment and then give two examples of the main applications on the platform.

5.1. JupyterHub Service. The JupyterHub module, on the app layer, is based on the data layer. To integrate with the HDFS and Spark, we do this part of the experiment by docker-compose. And JupyterHub-docker-compose-example.yml file shows part of the configuration. In this paper, we do not show the details. To make the data sustainable of JupyterHub, we choose the PostgreSQL as a database. In the yml file, we leave out some detailed information, such as the environment variable, volumes, and networks' configuration. (Algorithm 1).

5.2. Database Service

5.2.1. MySQL Service. For the web service, we use MySQL service to help build safety information storage, especially the data of users. We pull the MySQL:5.6 image from the DockerHub to build the service. After giving birth to a new container from the image-registry, we make a SQL script to build the ZDSW database, which includes system, applications, users, and other entities' information. In Figure 4, we show part of the physical data model from the database, which is one of SQL script execution results.

5.2.2. Greenplum Service. To put the in-database data processing in force, we ask for the help of Greenplum, which is able to integrate with the MADlib extension without any compatible problem. And MADlib is an open-source library for in-database data processing. Moreover, Greenplum is a relational data warehouse, which follows massively parallel processor architecture. It helps us to make full use of a big data environment. Due to its PostgreSQL kernel, we take advantage of the development platform tool, pgAdmin4, to integrate the database in the front-end on a specific web port. In this experiment, we try to make it easy to repeat. So, we apply the pseudo distribution in a single node with the help of Docker and docker-compose. We prepare a gpdb-docker-compose-example.yml file, which is shown below, for configuration reference. To make it successful, a startGPDB.sh file, which helps Docker to start the database, the base image, and other configuration should be prepared. (Algorithm 2).

The following part shows an example of using pgadmin4 to access Greenplum:

(1) Prerequisites

(2) Starting Docker-compose

(3) Configure Greenplum

(4) Configure pgadmin4

At the prerequisites part, we install the docker-compose and then pull the images of pgadmin4 and GPDB 5.x OSS. Prepare a script, such as docker-compose-pgadmin4.yml,

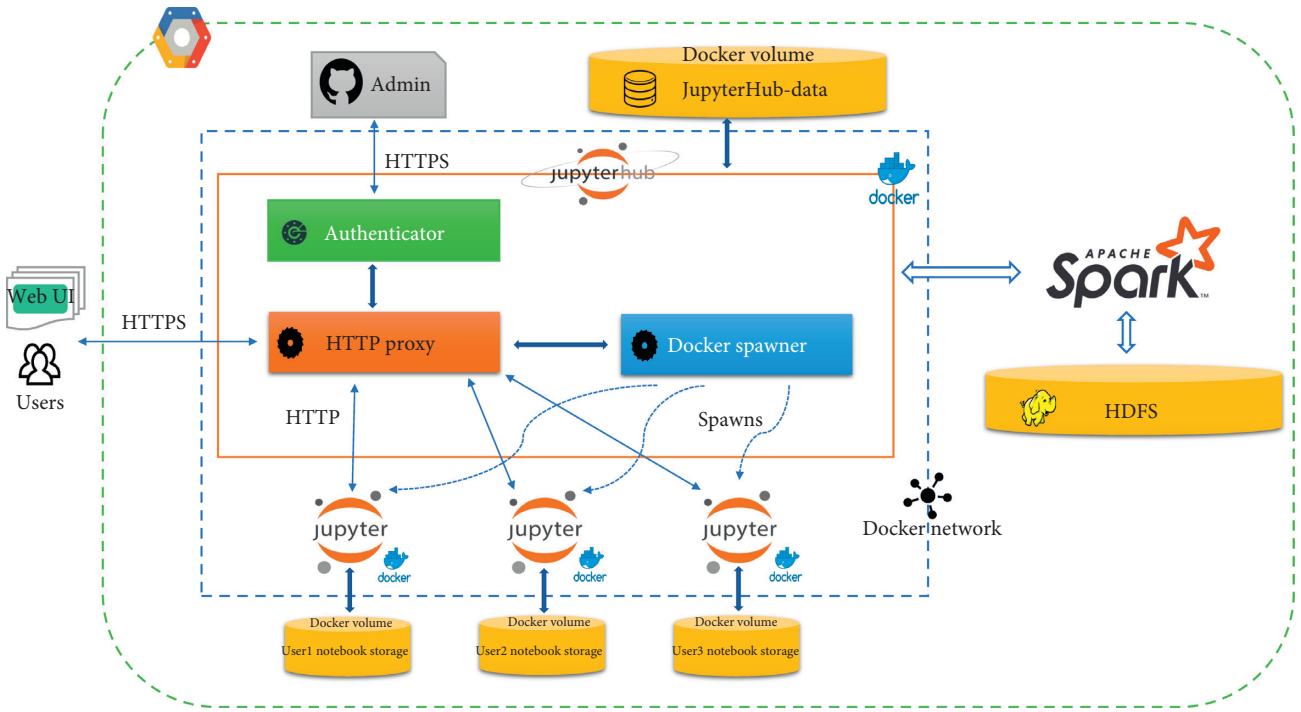


Figure 2: Overview of platform layer architecture.

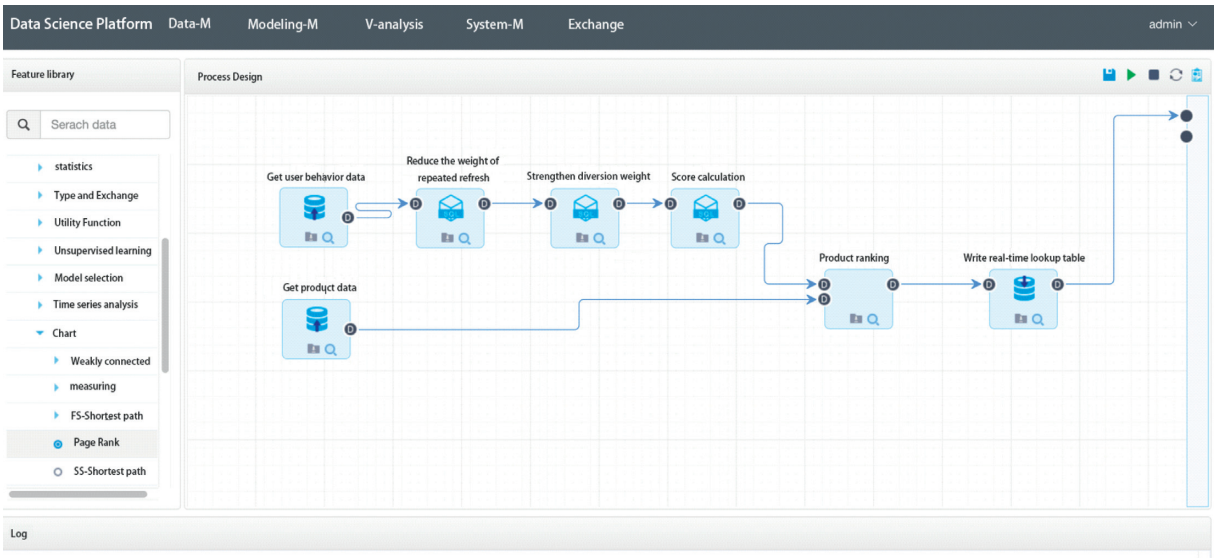


Figure 3: A pipeline case on the web-based visualized modelling application.

which configures the Greenplum and pgadmin4 well, to integrate them successfully.

After the database preparation, as the MySQL service, we need to execute a SQL script. The script is to prepare a data set to test the in-database machine learning.

Figure 5 shows the result of script result on the web-based pgadmin4 client dashboard.

5.3. Web Service. In the web-based project part, we apply spring boot [34] and webpack frameworks [35] to build the platform. The flowing part describes the details of the front and back-end service applications.

5.3.1. Spring Boot Back-End. On the back-end of web service, we build a Java project based on spring boot. The project includes three parts: data-control, resource, and web-app. The data-control helps the visualized Analysis-tools workflow going. The resource connects the back-end and front-end. Web-app is an essential part of the webpack project.

```
version: '3'
services:
  hub-db:
    image: postgres
    container_name: Jupyterhub-db
    restart: always
    env_file:
      -Jupyterhub/secrets/postgres.env
    volumes:
      –"db : ${DB_VOLUME_CONTAINER}"
  hub:
    depends_on:
      –hub-db
    build:
      context:./Jupyterhub
      dockerfile: Dockerfile
      args:
      Jupyterhub_VERSION: ${Jupyterhub_VERSION}
    restart: always
    image: Jupyterhub
    container_name: Jupyterhub
    ports:
      –"443:443"
      –"7070:7070"
    links:
      -hub-db
    env_file:
      -Jupyterhub/secrets/postgres.env
      -Jupyterhub/secrets/oauth.env
  command: >
    Jupyterhub-f/srv/Jupyterhub/Jupyterhub_config.py
  spark-master:
  build:
      context: /spark-master
      dockerfile: Dockerfile
  image: spark-master
  container_name: spark-master
  hostname: spark-master
  ports:
      –"8585 : 8080"
      –"7077 : 7077"
  volumes:
      -/mnt/spark-apps : /opt/spark-apps
      -/mnt/spark-data : /opt/spark-data
  spark-worker-1:
  spark-worker-2:
  spark-worker-3:
  namenode:
    image: hadoop-namenode : 2.0.0-hadoop3.1.1-java8
    container_name: namenode
  volumes:
      -"namenode:/hadoop/dfs/name"
  env_file:
      -/hadoop.env
  datanode:
    image: hadoop-datanode : 2.0.0-hadoop3.1.1-java8
```

Algorithm 1: JupyterHub-docker-compose-example.yml.

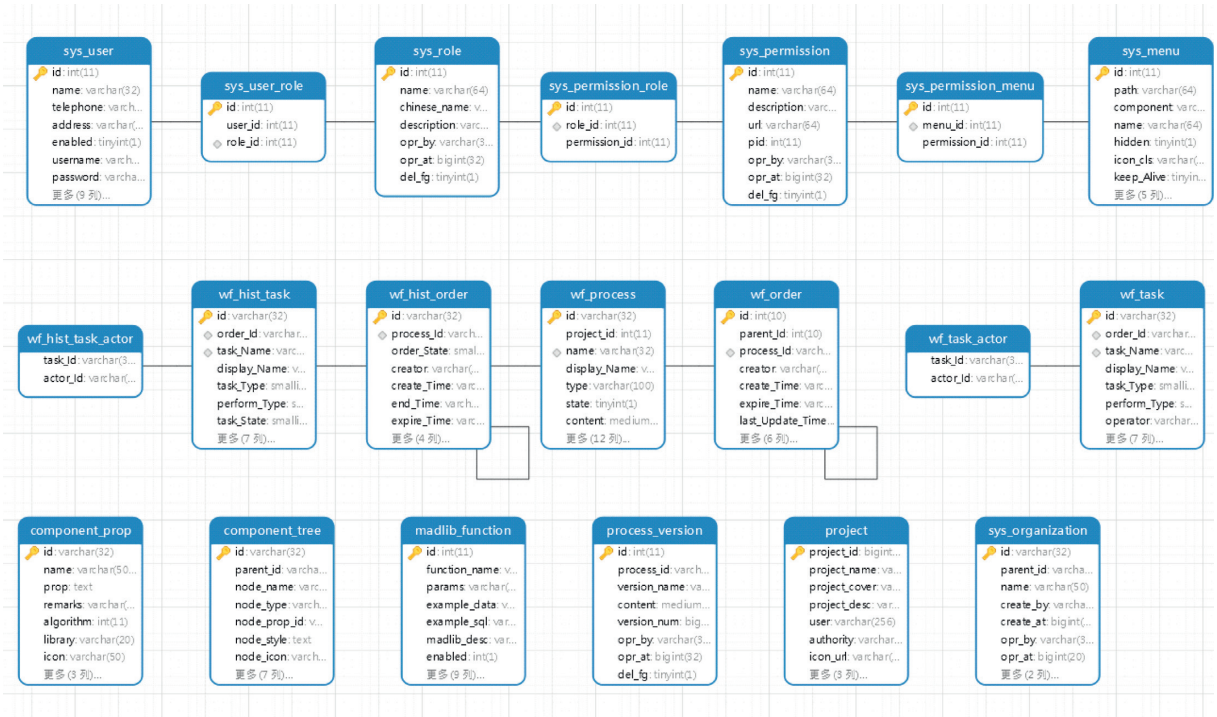


Figure 4: Physical data model (partial).

```
version: '3'
services:
  pgadmin4:
    image: pgadmin4
hostname: "pgadmin4"
    environment:
      -PGADMIN_DEFAULT_e-Mail = xxx@domain.com
      -PGADMIN_DEFAULT_PASSWORD = SecretPassword
    ports:
      -"80 : 80"
    volumes:
      -.. : /code
    tty: true
    privileged: true
  networks:
    mynetwork:
    aliases:
      -pgadmin4
  gpdb5:
    image: "gpdb5oss"
    command: bin/bash-c "startGPDB.sh && bin/bash"
    hostname: "gpdbsne"
    container_name: gpdbsne
    ports:
      -"5432 : 5432"
      -"5005 : 5005"
      -"5010 : 5010"
    expose:
      -"5432"
    volumes:
```

Algorithm 2: Continued.

```
      -..:/code
    privileged: true
    tty: true
    networks:
    mynetwork:
    aliases:
      -gpdbsne
      -gpdbsne.localdomain.com
networks:
    mynetwork:
    driver: bridge
```

Algorithm 2: gpdb-docker-compose-example.yml.

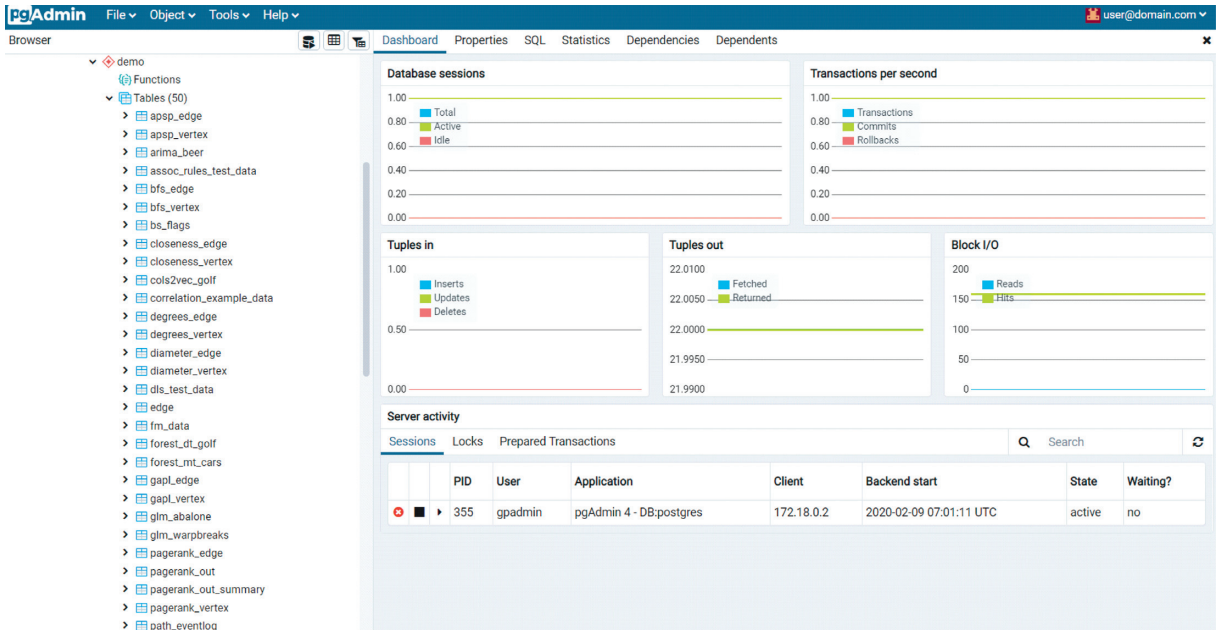


Figure 5: The web-based pgadmin4 client dashboard.

5.3.2. Element-UI-Admin Front-End. To reduce the time of web page development, we apply Element-UI to build an HTML, CSS, and JavaScript project for developing responsive [36], projects on the web. On the foundational of the UI-tool kit, we implement the open-source, Element-UI-Admin, to manage the front-web way of data management.

5.3.3. Deployment. After building the project and configuring the application properties, which is the configuration of database service and the features of low coupling and high cohesion, we deploy it with a special port on the Google Cloud Server Engine.

5.4. Online Programming Experiment on Spark and Hadoop. Since our educational application is based on Spark, Hadoop, and JupyterHub, we will do our experiment on Jupyter Notebook, spawned by JupyterHub on the website interface produced by Qunxian.

Figure 6 shows the SparkContext job and Hadoop job on the Jupyter Notebook, which is exposed by web port. And we can find the Spark job more directly on the port by HTTP way. Figure 7 shows the jobs works.

5.5. Visualized Modelling Application Experiment Example. Since the visualized modelling application is based on Greenplum database and the environment-friendly experiment experience with pure python.

In the experiment, we need to transform the algorithms to the format of pl/python [37], which is shown in Figure 8. To make it successful, the data running environment needs to be initialized to ensure that the environment dependencies required for the experiment are already available on the web platform. If there are missing experimental related environments, the server needs to install the relevant dependencies first. And the web back-end management system should grants experimental users environment

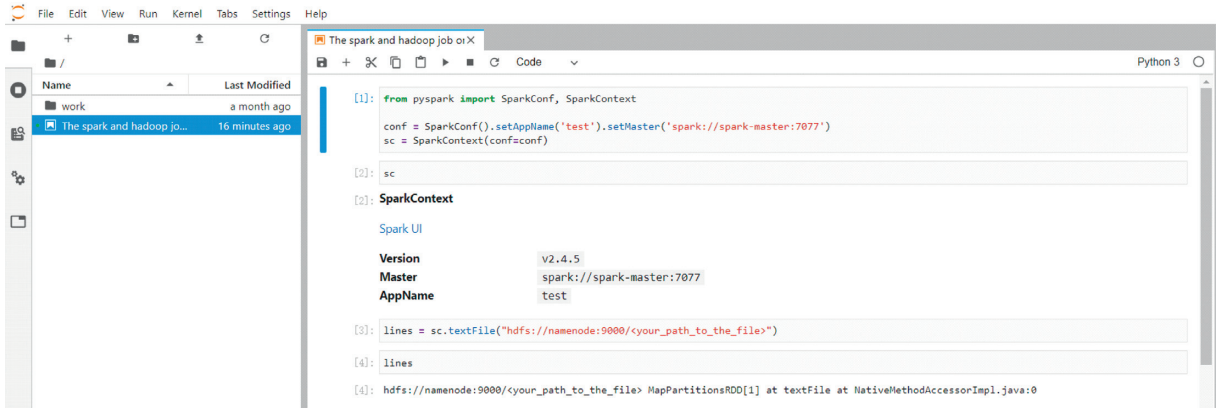


Figure 6: The Spark and Hadoop job on Jupyternotebook.

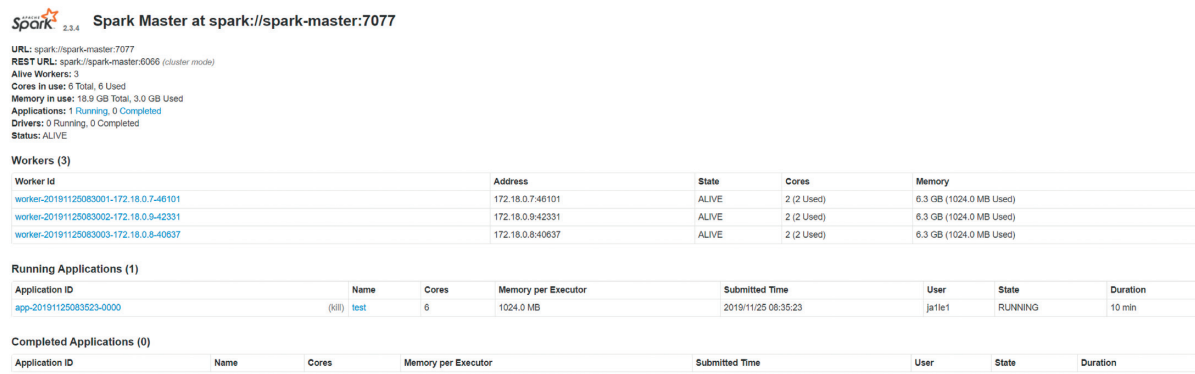


Figure 7: The Spark job on website back-end.

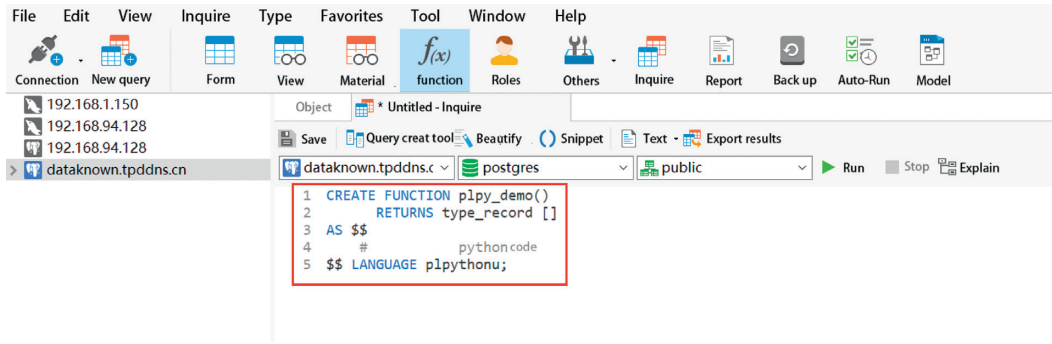


Figure 8: The transformation of algorithms.

administrator rights, because only super administrator users can use it to register with the database using PL/python.

The visualized modelling application supports integrating the configured pl/python algorithms with Qunxian. In the following part, we introduce a deployment way of new functions, shown by the HDP algorithm in the visualized modelling application.

First, we should add the HDP algorithm in the management of algorithms, which is part of data management. Second, we should apply it into the tree of algorithm to become a component of the existing algorithms. Last, we can make full use of it when building the visualized model. The processing part is shown by sequence in Figures 9–11.

5.6. Web Service Stress Test. In order to adapt the microservice characteristics of the adaptive system and the convenience of testing, we still use Docker-based methods to deploy JMeter-based [38] test cases. Select Alpine as the base of the Docker image and configure related environment variables, such as JMETER VERSION and JMETER HOME, which is convenient for decoupling test units and operation and maintenance units. Because JMeter is developed based

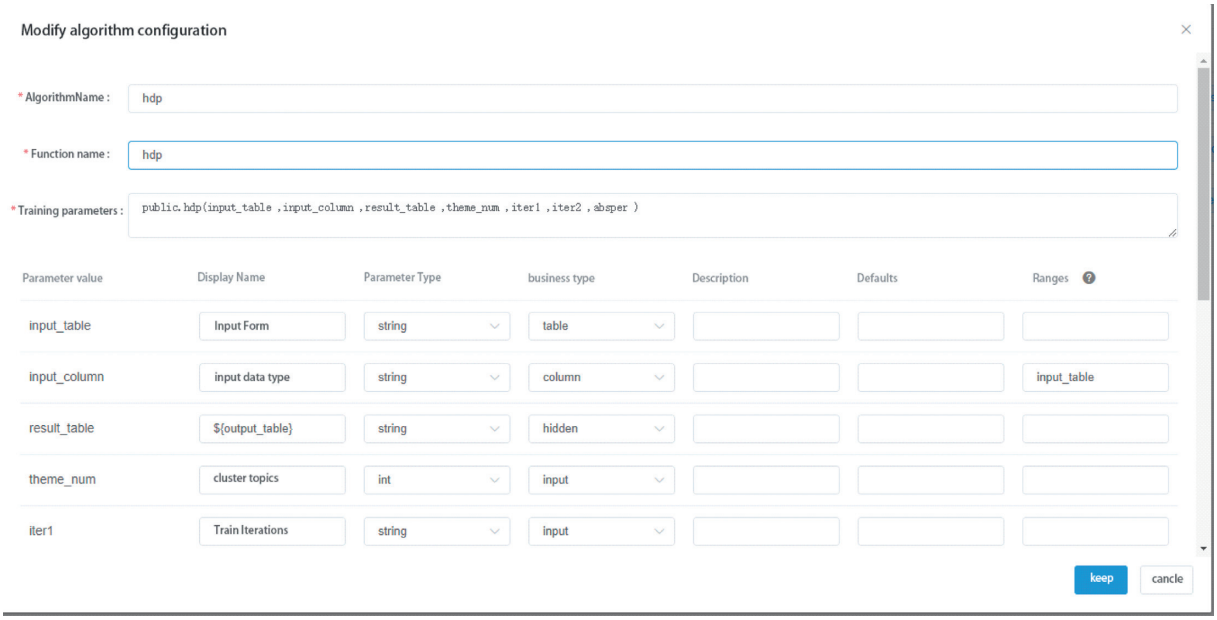


Figure 9: Step1 of processing of the HDP algorithm.

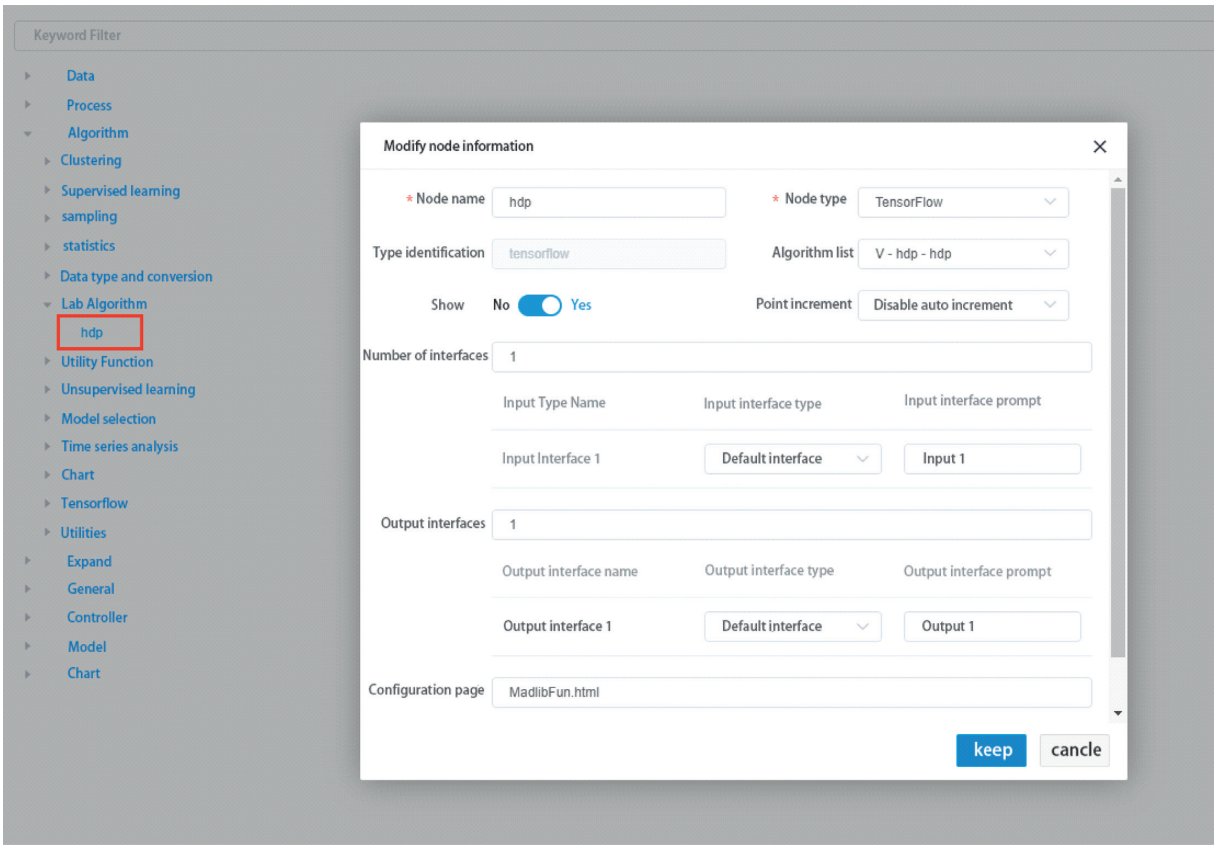


Figure 10: Step2 of processing of the HDP algorithm.

on the Java language, the service environment also needs to be configured with the Java SE Runtime Environment (JRE) [39]. In specific test cases, two scripts can be prepared: one is a test script, and the other is a run script. Test scripts are used to test the writing of rules for specific use cases, such as the hard coding of the system web port. The run script mainly

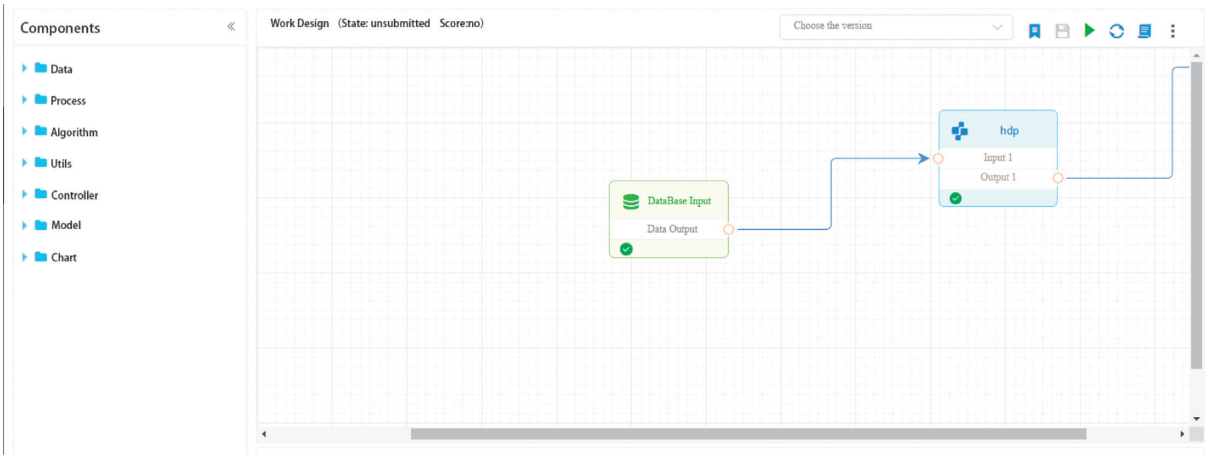
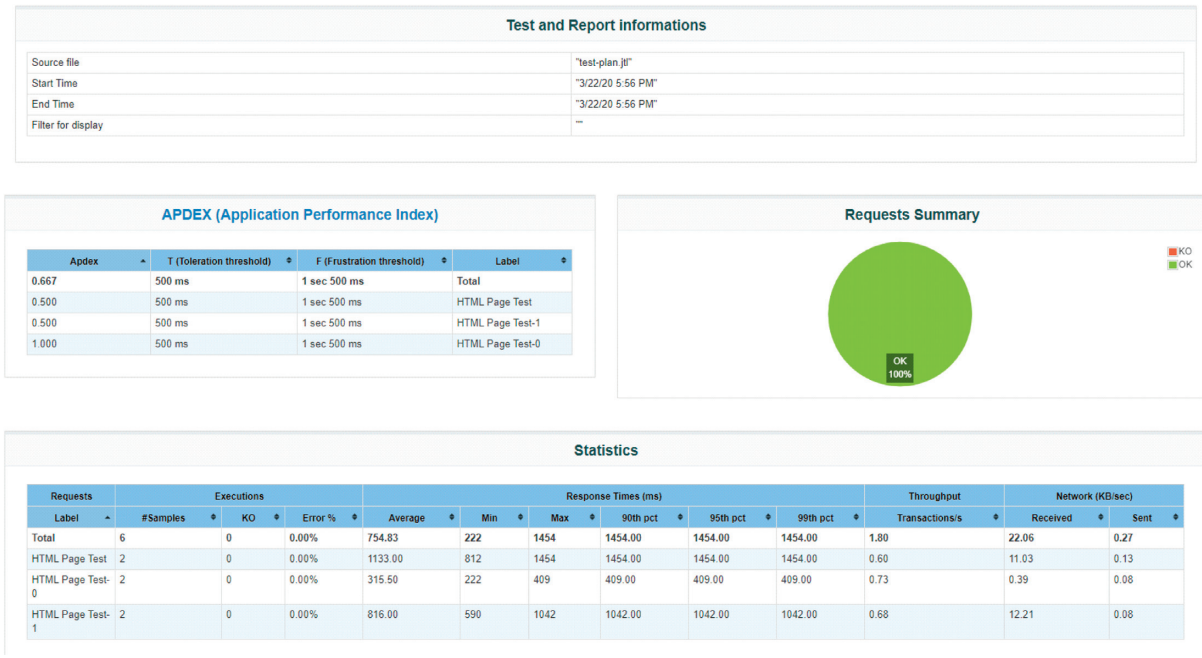


Figure 11: Step3 of processing of the HDP algorithm.



Test and Report informations

Source file	"test-plan.jtl"
Start Time	"3/22/20 5:56 PM"
End Time	"3/22/20 5:56 PM"
Filter for display	""

APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
0.667	500 ms	1 sec 500 ms	Total
0.500	500 ms	1 sec 500 ms	HTML Page Test
0.500	500 ms	1 sec 500 ms	HTML Page Test-1
1.000	500 ms	1 sec 500 ms	HTML Page Test-0

Requests Summary

Statistics

Requests	Executions			Response Times (ms)						Throughput	Network (KB/sec)	
Label	#Samples	KO	Error %	Average	Min	Max	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	6	0	0.00%	754.83	222	1454	1454.00	1454.00	1454.00	1.80	22.06	0.27
HTML Page Test	2	0	0.00%	1133.00	812	1454	1454.00	1454.00	1454.00	0.60	11.03	0.13
HTML Page Test-0	2	0	0.00%	315.50	222	409	409.00	409.00	409.00	0.73	0.39	0.08
HTML Page Test-1	2	0	0.00%	816.00	590	1042	1042.00	1042.00	1042.00	0.68	12.21	0.08

Figure 12: The result of web service stress test.

writes common run routines such as the startup and destruction of microservices.

Figure 12 shows the result of the web service stress test with the help of JMeter. In requests summary, all the test requests passed as required. In the statistics form, all the response times are within the controllable range.

6. Conclusion

We have presented Qunxian, a new microservice-based big data analysis platform, which is deployed on Google Cloud Engine running across distributed computing resources. And we construct the big data ecosystem environment based on Docker. On the foundation of platform infrastructure, we introduce two web-based applications. For students who want to get started with a big data ecosystem environment without any barriers on educational purposes, we introduced JupyterHub, with which every user can program online on their environment and keep their data persistence on private data volume. For researchers who want to corporate efficiently while sharing their models based on their specific format of data, we introduce the other web-based application, Visualized Modelling tool, in which data science research people can share their ideas without the cold data and code only. We do our experiment on Google Cloud Platform and check its feasibility.

For future work, we will deploy the platform on Kubernetes clusters, which is a portable, extensible, open-source platform for managing containerized workloads and services, if the platform serves more students. In this way, the

automating deployment and scaling of the two containerized applications may help infrastructure get well utilized.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is partly supported by the National Key Research and Development Program of China (Grant no. 2018YFC0830300), Science and Technology Program of Fujian, China (Grant no. 2018H0035), Science and Technology Program of Xiamen, China (Grant no.3502Z20183011), and Fund of XMU-ZhangShu Fintech Joint Lab.

References

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, Cambridge, MA, USA, 2016.

[2] S. Chen, C. Wu, and Y. Yu, "Analysis of plant breeding on hadoop and spark," *Advances in Agriculture*, vol. 2016, Article ID 7081491, 6 pages, 2016.

[3] T. M. Mitchell, *Machine Learning*, 1997.

[4] J. M. Cavanillas, E. Curry, and W. Wahlster, *New Horizons for a Data-Driven Economy: A Roadmap for Usage and Exploitation of Big Data in Europe*, Springer, Berlin, Germany, 2016.

[5] J. Kobusiński, J. Brzeziński, and A. Kobusińska, "On increasing dependability of web services-an approach to design a failure detection service 2018," in *Proceedings of the 16th IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC 2018)*, IEEE, Athens, Greece, pp. 504–511, August 2018.

[6] M. D. Scott, M. A. Foltz, J. Affaki et al., "System for universal remote media control in a multi-user, multi-platform, multi-device environment," U.S. Patent 10-031-647, 2018.

[7] J. M. Perkel, "Why Jupyter is data scientists' computational notebook of choice," *Nature*, vol. 563, no. 7732, pp. 145–147, 2018.

[8] J. Lu, G. Liu, K. Wu et al., "Location-aware web service composition based on the mixture rank of web services and web service requests," *Complexity*, vol. 2019, Article ID 9871971, 16 pages, 2019.

[9] M. Milligan, "Interactive hpc gateways with jupyter and jupyterhub," in *Proceedings of the Practice and Experience in Advanced Research Computing 2017 on Sustainability, Success and Impact*, pp. 1–4, New Orleans, LA, USA, July 2017.

[10] Y. Zhang, M. Wei, C. Cheng et al., "Exploiting delay-aware load balance for scalable 802.11 PSM in crowd event environments," *Wireless Communications and Mobile Computing*, vol. 2017, Article ID 6128437, 11 pages, 2017.

[11] C. Pahl and P. Jamshidi, "Microservices: a systematic mapping study," in *Proceedings of the 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*, pp. 137–146, Macau, China, November 2016.

[12] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux Journal*, vol. 2014, no. 239, p. 2, 2014.

[13] K. Shvachko, H. Kuang, S. Radia et al., "The hadoop distributed file system," in *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, IEEE, Incline Village, NV, USA, pp. 1–10, May 2010.

[14] J. Hellerstein, C. Ré, F. Schoppmann et al., "The MADlib analytics library or MAD skills, the SQL," 2012, http://arxiv.org/abs/1208.4165.

[15] S Butlin, "Method of browser-server communication," U.S. Patent 10-151-11, 2002.

[16] K. Artto, T. Ahola, and V. Vartiainen, "From the front end of projects to the back end of operations: managing projects for value creation throughout the system lifecycle," *International Journal of Project Management*, vol. 34, no. 2, pp. 258–270, 2016.

[17] E. You, *Vue. js*, Diakses Dari Httpsvuejs Org Pada Tanggal, Washington, DC, USA, 2018.

[18] H. Zhang, T. H. Lee, K. Chow et al., "Drag and drop interaction between components of a web application," U.S. Patent 10-048-854, 2018.

[19] M. Ferguson, *Architecting a Big Data Platform for Analytics*, p. 30, A Whitepaper prepared for IBM, Armonk, NY, USA, 2012.

[20] D. Guo, W. Wang, G. Zeng et al., "Microservices architecture based cloudware deployment platform for service computing," in *Proceedings of the 2016 IEEE Symposium on Service-Oriented System Engineering (SOSE)*, IEEE, Oxford, UK, pp. 358–363, March 2016.

[21] S. Kim, J. W. Kim, J. Park et al., "Elice: an online CS education platform to understand how students learn programming," in *Proceedings of the Third (2016) ACM Conference on Learning@Scale*, pp. 225–228, Edinburgh, UK, April 2016.

[22] T. Staubitz, H. Klement, R. Teusner et al., "CodeOcean-A versatile platform for practical programming excercises in online environments," in *Proceedings of the 2016 IEEE Global Engineering Education Conference (EDUCON)*, IEEE, Abu Dhabi, UAE, pp. 314–323, April 2016.

[23] Z. Zou, Y. Zhang, J. Li et al., "EasyHPC: an online programming platform for learning high performance computing," in *Proceedings of the 2017 IEEE 6th International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, IEEE, Hong Kong, China, pp. 432–435, December 2017.

[24] I. Foster, "Service-oriented science," *Science*, vol. 308, no. 5723, pp. 814–817, 2005.

[25] J. Schwieren, G. Vossen, and P. Westerkamp, "Using software testing techniques for efficient handling of programming exercises in an e-learning platform," *Electronic Journal of E-Learning*, vol. 4, no. 1, pp. 87–94, 2006.

[26] M. Keith, M. Schincariol, and M. Nardone, *XML Mapping Files*, pp. 593–654, Apress, New York, NY, USA, 2018.

[27] H. Suryotrisongko, D. P. Jayanto, and A. Tjahyanto, "Design and development of backend application for public complaint systems using microservice spring boot," *Procedia Computer Science*, vol. 124, pp. 736–743, 2017.

[28] F. Gutierrez, *Pro Spring Boot*, Apress, New York, NY, USA, 2016.

[29] M. T. Nygard, "Release it!: design and deploy production-ready software," Pragmatic Bookshelf, Raleigh, NC, USA, 2018.

[30] S. Mahajan, A. Alameer, P. McMinn et al., "Automated repair of internationalization presentation failures in web pages

using style similarity clustering and search-based techniques," in *Proceedings of the 2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST)*, IEEE, Västerås, Sweden, pp. 215–226, April 2018.

[31] A. Fox, R. Griffith, A. Joseph et al., *Above the clouds: A Berkeley View of Cloud Computing*, University of California, p. 2009.

[32] N. Naik, "Building a virtual system of systems using docker Swarm in multiple clouds,," in *Proceedings of the 2016 IEEE international Symposium on Systems Engineering (ISSE)*, IEEE, Edinburgh, UK, pp. 1–3, October 2016.

[33] L. Stanescu, M. Brezovan, and D. D. Burdescu, "Automatic mapping of MySQL databases to NoSQL MongoDB," in *Proceedings of the 2016 Federated Conference on Computer Science and Information Systems (FedCSIS)*, IEEE, Gdansk, Poland, pp. 837–840, September 2016.

[34] D. Omilusik, "Spring boot," U.S. Patent 4-660-299, 1987.

[35] F. Feng and L. Zou, "Design of scratch interactive online learning platform based on webpack and react technology," *Computer Knowledge and Technology*, vol. 2018, no. 20, p. 25, 2018.

[36] D. T. Hoang, O. Chernomor, A. Von Haeseler, B. Q. Minh, and L. S. Vinh, "UFBoot2: improving the ultrafast bootstrap approximation," *Molecular Biology and Evolution*, vol. 35, no. 2, pp. 518–522, 2018.

[37] K. K. Das, R. Raghu, and C. J. Rawles, "Imaging subsurface properties using a parallel processing database system," U.S. Patent 9 720-117, 2017.

[38] E. H. Halili, *Apache JMeter: A Practical Beginner's Guide to Automated Testing and Performance Measurement for Your Websites*, Packt Publishing Ltd, Birmingham, UK, 2008.

[39] A. Kasko, S. Kobylyanskiy, and A. Mironchenko, *OpenJDK Cookbook*, Packt Publishing Ltd, Birmingham, UK, 2015.