A VIRTUAL EXPERIMENT DESIGN APPROACH FOR BIG DATA BASED ON CONTAINERS AND PYTHON LANGUAGE

Dazhou Li[1*] and Wei Gao[2]

[1]Dr. Dazhou Li, Shenyang University of Chemical Technology, China, lidazhou@syuct.edu.cn
[2] Prof. Dr. Wei Gao, Shenyang University of Chemical Technology, China, gaowei_syuct@foxmail.com
*Corresponding author

Abstract

Given the inconsistency between the experimental environment and the production environment, the high hardware cost of the big data production environment which is difficult for ordinary universities to bear, and the complicated installation of software related to the big data field which is not easy to reproduce and other practical teaching problems in the current stage of big data experimental teaching, the advantages and disadvantages of the existing solutions are analyzed and a virtualization method is proposed to virtualize hundreds of thousands of virtual The design method of Big Data virtual experiments based on containers and Python language is proposed to meet the practical teaching needs of undergraduate Big Data courses by virtualizing hundreds of virtual servers on several physical servers to construct a private cloud of Big Data experiment servers within the university. By optimizing the design for a small-scale server cluster environment in universities, the redundant modules are streamlined, and only teaching experiment-related modules are retained, geared towards teaching simulation, saving the limited funds of ordinary universities and revitalizing state-owned assets; at the same time, open-source software is used to avoid intellectual property costs in future teaching sessions.

Keywords: Container technology, Python language, big data technology, virtual simulation experiments.

1. INTRODUCTION

In recent years, many universities have opened big data majors. At this stage, the experimental teaching process of big data is mainly based on the skilled use of local stand-alone big data software, with students familiarising themselves with stand-alone operations on one of their local computers. Such practical exercises are far from the actual big data application scenarios. In the actual scenario, big data processing exists in the form of server clusters. In Big Data practical teaching, it is not possible to provide a Big Data server cluster for each student, considering various factors such as the construction conditions of the school laboratory. This results in students not being able to truly experience the complete process of big data processing and receive realistic big data practical training, making it difficult to achieve the pre-determined objectives of the course.

In addition, in practical teaching, the experimental environment for teaching Big Data applications in practice generally requires running on Linux and multiple machines are needed to build a server cluster. The installation process involves the installation of Linux systems, virtual machine software, JDK, Hadoop, Spark, Hbase, Hive, and other open-source software, which makes the installation process tedious and the

configuration workload extremely heavy. Sometimes the same operation can be successful on one computer but fail on another.

2. DESIGN METHODOLOGY FOR VIRTUAL EXPERIMENTS

In this paper, through virtualization, hundreds of virtual servers are virtualized on a few physical servers to construct a private cloud of big data experiment servers within the school, which can then meet the practical teaching needs of the big data courses of data science and big data technology majors and computer science and technology majors. The mainstream container technology can be used to form a virtual private cloud of servers at this stage. Based on container technology, each virtual server is not only identical to the real server during use, but the external performance and operation and maintenance methods of the virtual server are also identical to the actual production environment. This enables the creation of a virtual big data server cluster for students with limited university experimental teaching resources at this stage.

The platform is developed using OpenStack and container hybrid cloud technology (Beloglazov et al., 2015). The system is eventually deployed within the university, using a private cloud B/S architecture. During the Big Data practice teaching period, current undergraduate students access the virtual simulation experiment platform through the college labs with remote login, and the resources provide strong support for undergraduate students' Big Data practice capability training and teachers' practice teaching guidance.

At this stage, the platform is oriented towards high-end Internet enterprises, and very few of them are oriented towards the education sector and local universities. Similar platforms are mainly for large commercial Internet enterprises like Ali, Baidu, Jingdong, Meituan, and Tencent, providing high-performance processors, huge memory, and massive storage space, so the prices of similar products are unaffordable for ordinary universities. The demand for dozens of servers in ordinary universities is overlooked in the commercialized category because of the small profits generated and the lack of interest from vendors to offer products targeted at the education sector. In addition, at this stage, similar products are mainly sold as software and hardware bundles, which can lead to the dilemma of having to buy a new batch of hardware servers before the hardware servers that universities already have are fully used, thus resulting in a waste of state-owned assets of universities.

In response to the above two problems, the virtual experiment simulation platform is redesigned and optimized for the small-scale server cluster environment of universities at the early stage of development, streamlining a large number of redundant modules and retaining the functional modules related to teaching experiments, to mainly focus on teaching simulation and adapt to the small-scale server cluster characteristics of ordinary universities at this stage. In addition, the virtual experiment simulation platform should not be bundled with hardware, which increases the difficulty of developing the system software, but the hardware resources of the platform can use the existing computing servers of the university, saving the waste of money brought by re-purchasing servers, and playing a good role in promoting the full use of state-owned assets for the university. The relevant part of the software of the virtual experiment simulation platform should use open source software, thus further ensuring that the platform will not add additional intellectual property costs in the future.

3. TECHNICAL MEANS OF VIRTUAL EXPERIMENTS

The biggest problem in the professional practice teaching of big data in colleges and universities is that there is no supporting simulation experimental environment for the deployment of the big data environment. To reduce the cost of laboratory construction and the learning difficulty of big data application practice, the virtual simulation experiment is based on virtualized container technology, which builds a lightweight big data experiment platform. The virtual simulation experiment of comprehensive application of big data technology based on virtualization platform aims to meet the needs of universities to offer big data practice courses. It mainly contains practical content of Big Data applications, involving processes such as data pre-processing, data storage and management, data analysis, and data visualization, covering the installation and usage of systems and Big Data software such as Linux, Kafka, Hadoop, Spark, server cluster operation and maintenance. The platform enables the virtual creation of Big Data clusters with limited computing resources, provides a variety of lab environments that match the curriculum, focuses on the technology itself, and dramatically improves students' efficiency in learning Big Data server cluster usage and operation and maintenance techniques.

With the continued emergence of cloud technologies, the paradigm for developing and deploying applications on the cloud has changed, not only because of provisions such as scalability and reliability but also because the cloud supports continuous integration and delivery with as little downtime as possible

(Cegielski et al., 2012). The main architecture used to develop cloud-based applications is the microservices architecture. The foundation behind this architecture is a combination of loosely coupled services that implement business capabilities. It focuses on constructing single-function components with well-defined interfaces. One of the main reasons for its popularity is that businesses have adopted an agile approach that can configure items and update modules in real-time without affecting the application. Microservice architectures also address the issue of application scaling, as each microservice is bundled in its codebase, as opposed to a holistic approach where the entire application is bundled in a single code base. One of the key factors in implementing a microservices architecture is virtualization. Virtualization can be implemented using virtual machines and containerization.

Virtualization technology, represented by containers, has the characteristics of being lighter and easier to deploy quickly than traditional virtual machine technology, which can significantly reduce the total amount of resources used in a cloud platform and improve the efficiency of application distribution and deployment. As a result, container-based cloud computing platforms are rapidly becoming a research hotspot in both academic and industrial circles for big data technologies. Containers consist of a complete runtime environment that includes an application along with the dependencies, libraries, and other binaries needed to run the application (Anderson, & Charles, 2015). All of these are bundled in a single package. By containerizing the application, unexpected situations between the application and the infrastructure are resolved.

The architecture of a container differs from that of a virtual machine. Containers share an operating system in which each running process is isolated in userspace so that containers end up consuming less space than virtual machines. These containers out-perform virtual machines in different behaviors, such as better start-up times, resource allocation, and less redundancy. To make applications platform-independent, packaged build images of container-generated code are used [9]. As applications grow, the number of microservices increases, which makes it difficult to manage and deploy services manually. As a result, a "continuous integration" and "continuous delivery" approach is used to automate deployment whenever new updates are made to the code base.

"Continuous integration" is part of the practices involved in the principles of software programming (AlexanderPoth et al., 2018). Git is one such common tool that is widely used to maintain versions of code. Each developer works on their branch, which is cut off from the master branch. Whenever a developer implements a feature or fixes a bug, a merge request is sent from the current branch to the master branch. A script to execute the test cases is triggered, the script runs all the unit tests written and if the code passes all the test cases the merge is complete, otherwise the merge is rejected. When merging code into the master branch, each developer has a very responsible role. This increases the developer's responsibility and therefore needs to ensure that his changes do not affect the build and do not interfere with the work of other developers. To avoid merging conflicts, short-lived branches with small features are preferred to long-lived branches with large features.

"Continuous delivery" is defined as the ability to deploy new features and bug fixes to a live server when needed. The CD part is executed after all updated code is located in the successful configuration entries of the master branch. Run a script that picks code from the master branch, prepares it for building, and deploys it to the test environment/production environment. This allows developers to test the code outside of unit tests for updates across multiple parameters, such as user interface tests, integration tests, etc. Tests can be performed to identify problems in advance. For continuous delivery there is also a term called continuous deployment, the difference between the two is that when following continuous deployment there is no need for manual intervention/confirmation, meaning that with each code check-in the new build will be deployed to the specified server. The goal of continuous delivery is to build software in a way that can be released to a production environment at any time. It deals with a series of automated processes. These processes are designed to be deployed to production safely and quickly. Each change is delivered to a production-like environment, called a segmented environment. Rigorous automated testing ensures that business applications and services work as expected. Each change is tested during the delivery phase so that the application can be safely deployed into the production environment.

In the Virtual Emulation Experiment, the main files are in the Github directory, which contains the Github CI/CD files implementing continuous integration and continuous delivery. the docs directory holds files for documentation purposes. the node_modules directory contains all the library packages used during the development of the Virtual Emulation Experiment. the scripts directory contains all of the Virtual Emulation Experiment project's script files, such as bash scripts, python scripts, etc. The src directory contains all the files for the main system. The terraform directory contains all the terraform files for deployment. test directory

consists of test files. The files .eslintignore and .eslint.RC is the server files used for the eslint library. yml is used for container writing local development. the Dockerfile file is used for building container images. dockerfiles describes how to create an image and Docker can generate the image and save it to a repository. docker Engine is responsible for managing containers (starting, stopping, etc.), while Docker Compose is responsible for container configuration on a single host system. docker compose is used for development and testing environments. It is possible to define which services are needed and what their configuration is in the container. the Docker Compose file can be used for this purpose.

4. ACKNOWLEDGEMENT

We would like to take this opportunity to acknowledge the hard work of all our editors and also the efforts of all the anonymous reviewers who have provided very valuable and helpful comments. We thank them all for their invaluable contributions, and for helping with our research. Project supported by Liaoning Higher Education Society 13th Five-Year Plan General Topics (GHYB160163); Ministry of Education, Department of Higher Education, Collaborative Education Project with University-Industry Cooperation (201801128005, 201902233001); Shenyang University of Chemical Technology Education and Training Project (No. 35).

REFERENCE LIST

AlexanderPoth, MarkWerner, XinyanLei, AlexanderPoth, MarkWerner, & XinyanLei, et al. (2018). How to deliver faster with ci/cd integrated testing services?. Springer, Cham.

Anderson, & Charles. (2015). Docker [software engineering]. IEEE Software, 32(3), 102-c3.

Beloglazov, A. , & Buyya, R. . (2015). Openstack neat: a framework for dynamic and energy-efficient consolidation of virtual machines in openstack clouds. Concurrency & Computation Practice & Experience, 27(5), 1310-1333.

Cegielski, C. G. , Jones-Farmer, L. A. , Yun, W. , & Hazen, B. T. . (2012). Adoption of cloud computing technologies in supply chains. The International Journal of Logistics Management, volume 23(2), 184-211(28).