Design and Implementation of a Cloud-Native Platform for Financial Big Data Processing Course

Ping Yu(✉), Ye Tao, Jie Zhang, and Ying Jin

Department of Computer Science and Technology, Nanjing University, Nanjing, China
{yuping,taoye,zhangj,jinying}@nju.edu.cn

Abstract. With the advent of big data, financial technology steps into a more intelligent era. The demand for big data-related financial application development and operation continues to expand. Hadoop and Spark are popular parallel computing frameworks that are adopted as big data processing platforms, but the time cost of their environment deployment is a significant expense for beginners. BDKit, a big data application development and operation platform based on container cloud, is proposed in this paper, aiming to provide customized and scalable cloud service solutions for teaching practice. It is based on Kubernetes ecosystem and relies on the powerful self-repair and load balancing capability provided by Kubernetes to ensure the stable operation of users' virtual clusters. At the same time, BDKit encapsulates a variety of Docker images to achieve the construction of big data application development and operation environment according to the components required by different users. The services provided by BDKit can greatly reduce the pressure of preparing big data processing environment and help to achieve the purpose of teaching students according to their aptitude.

Keywords: Big Data · Fintech · Kubernetes · Docker · Cloud-Native

1 Introduction

In recent years, artificial intelligence, big data, cloud computing and other computer technologies have been widely applied in the financial industry, making financial technology to step into a more intelligent and digital era. At the same time, a large number of universities began to train interdisciplinary talents in the field of computer science and financial engineering. Interdisciplinary major of computer science and financial engineering focuses on the integration of economics, finance, computer technology and other disciplines. It mainly aims to cultivate students to master fintech and data science, with knowledge in data processing, data mining, programming, software engineering and artificial intelligence.

As one of the major compulsory courses for junior students, financial big data processing course focuses on big data technologies (Hadoop [1], Spark [2], HBase [3], Hive [4], etc.) and their applications on large volume of financial data. It requires students to have mastered the fundamentals of computer science, such as programming languages

W. Hong and Y. Weng (Eds.): ICCSE 2022, CCIS 1813, pp. 180–193, 2023.
https://doi.org/10.1007/978-981-99-2449-3_17

(C++, Java or Python), data structure and algorithm, data base and distributed computing. Different from any course offered by financial engineering majors or other basic computer courses, it is a comprehensive course with an emphasis on practice. It aims to systematically cultivate students' capability of big data processing program design, development, and parameter tuning through a series of experiments. It challenges students because most of them lack training on large-scale computer systems. Due to the particularity of big data processing technology requiring multiple machine nodes, it is very difficult to set up a parallel computing cluster both technically and in terms of manpower and material resources. It costs students and teaching assistants much time on the first step of installing and configuring big data processing platforms, such as Hadoop and HBase, even in pseudo distributed mode. Based on three years of teaching experience, we find it is necessary to provide an online platform for students to rapidly access the big data processing cluster environment.

Cloud computing provides utilities for using computing resources without installing and maintaining them on-premises [5]. Most of the existing large cloud service providers only provide basic system level services, but users still need to build and configure their own environment. Even large cloud service providers such as Ali Cloud and Tencent Cloud, which provide parallel computing environment images such as Hadoop for direct deployment, are mainly oriented to high-performance computing needs of enterprises and large-granularity requirements of direct executing applications rather than teaching and learning purposes, so the granularity of services provided is too large. It is not suitable for individual learners in terms of performance and price. At present, some scholars and teaching staff have carried out exploration in the construction of big data experimental platform [6–8]. However, these big data development and training platforms for teaching are uneven in technology, and generally have the following problems: (1) Some platforms are oriented to web services with high level abstraction, which is often just the executor of Hadoop and Spark programs, so that users lose the opportunity to access the underlying cluster. It is difficult for users to debug and tune their programs. (2) Some platforms are based on VM images and the resource usage is high and the performance is low. The number of cluster nodes and software components is usually fixed, and the degree of customization and extensibility is low. For example, it is difficult to efficiently process streaming financial data on the platform that lacks stream computing components such as Apache Storm [9] or Flink [10].

Therefore, we hope to propose a cloud-native solution named as BDKit for financial big data processing course. It can provide services at appropriate levels of abstraction, so that on the one hand, it can build a customized virtual cluster according to the needs of students to provide the parallel computing environment, and on the other hand, it can ensure students to have fully independent operation of the cluster environment. Besides big data practice, students also get the opportunity of learning about cloud computing technology if they wish.

This paper provides the following contributions:

- We propose a big data teaching platform based on container cloud. It supports the creation of Hadoop/Spark clusters and the online development, running, and monitoring of big data applications such as MapReduce, Spark, and Hive. With the help of BDKit, students can put more efforts on big data program design and optimization.
- We design a series of basic experiments and some comprehensive projects for students to apply big data technology in financial big data processing. Students are more confident with the technology needs of the fintech after course training.
- The cloud-native solution has stimulated students' interest in big data and cloud computing. Based on students' feedback, it proves to be effective for interdisciplinary major of computer science and financial engineering.

2 Platform Design

2.1 Requirements

With the maturity of open-source big data frameworks like Hadoop and Spark, and the ecosystem of big data components built around them, it's getting easier to develop applications that can handle massive amounts of data. However, the hardware acquisition, environment construction and state management of the cluster are still quite tedious and difficult. Even if a user has enough hardware to build a cluster, setting up a cluster and configuring the environment can be a huge and complex task, especially for beginners, if the user is not familiar with Linux commands or other related tools.

Based on three years of teaching practice, we found that the junior students have common difficulties in preparing big data clusters. They were puzzled by various system-level problems and had to put much time on solving them. The frustration diminished their interest in learning big data technology even though they were good at programming and algorithms. After investigation, the most frequent problems are summarized in Table 1.

In fact, solving these problems is not the key point of this course. In order to simplify the construction and deployment of the big data framework and provide users with a real distributed environment with convenient access and rich computing components, it is particularly important to build an integrated online platform for teaching and practicing.

With specific consideration on financial scenarios, financial data generally has the characteristics of "stream data" and needs to be processed quickly in a short time. It is necessary to include streaming computing components in the supporting platform. However, there are different frameworks available in the big data community, including Spark streaming, Storm and Flink. The platform is thus expected to be open for different frameworks. In addition to this, operability of the platform is also important. As big data technology is strongly connected with cloud computing, a cloud-native platform will be user-friendly to students to understand the combination of these two mainstream technologies.

Table 1. Common problems

No.	Problem	Possible Cause
1	Name Node disappears after restarting computer	The fold of *hadoop.tmp.dir* disappears after restart
2	Name Node disappears after starting HDFS	The configuration of *hdfs-site.xml* is wrong
3	Resource Manager disappears when executing MapReduce task	VM or docker does not allocate enough memory or computer has not enough memory
4	MapReduce or spark task stucks	Computing resources are insufficient
5	No hbase master found when shutting down HBase	The *pid* file is deleted by operating system
6	HMaster can not start	HBase environment setting has error

2.2 Platform Overview

BDKit is based on Kubernetes ecology [11, 12], which can automatically reschedule, restart and replication, thus has strong self-repair capability. At the same time, it relies on the Docker [13–15] virtualization technology and abstracts each node as a pod. A virtual Hadoop/Spark cluster consists of multiple nodes. Each user can create more than one node according to different application scenarios. On the one hand, it greatly saves system resources compared to the virtual machine method. It provides a more fine-grained computing needs of cloud services to users. On the other hand, users can control the virtual cluster that belongs to oneself completely through a web-based terminal with root authority. In this way, students can get more in-depth exposure to the cluster, and customize the cluster at a higher level. With different images prepared in advance, we can provide instances of different environments according to users' requirements, so that users can select Hadoop and Spark ecological components on demand. In addition, BDKit also provides a shared data set, which can be mounted by disk to facilitate all users of the platform to directly access the data set prepared by the platform administrator, generally the teacher or teaching assistants, for experiments according to their teaching needs. Finally, it also integrates popular web-based interactive code editor services (e.g., VsCode code-server [16] and Jupyter notebook [17]) and UI components of Hadoop and Spark monitor for online development and monitoring. To facilitate code management, BDKit adopts Git [18] service to enable pulling code from remote repositories and pushing modified code. Even if one deletes his/her cluster, all application codes are still available in the his/her remote repository.

2.3 Layered Architecture

As shown in Fig. 1, BDKit consists of four logical layers, bottom-up namely virtualization layer, business layer, API interface layer and view layer.

(1) **Virtualization Layer:** BDKit runs on top of a Kubernetes cluster, relying on Kubernetes for cluster management. It exposes all ports through the abstraction of services. To meet the differentiated needs of users, BDKit provides various docker images pre-installing a variety of big data development components. Users can directly choose the appropriate image when creating a cluster, or compose different components on demand to get a customized cluster. For example, a user can get a cluster of the last version of Hadoop and Spark frameworks together with a specific version of Anaconda [19] and Jupyter notebook.

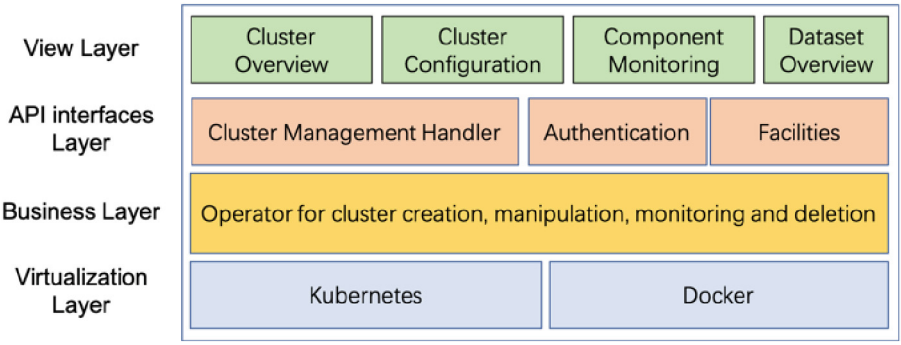

Fig. 1. Layered architecture of BDKit

(2) **Business Layer:** It is the kernel part of BDKit. We design and implement a big data processing cluster operator based on Operator SDK [20]. The operator defines the controller of Hadoop/Spark cluster and all relevant cluster resources. It is in charge of creating, manipulating and deleting clusters.

- *Cluster Architecture*: The architecture of Hadoop/Spark cluster created by BDKit refers to the master-slave mode of Hadoop, that is, it contains one master node and multiple slave nodes. The master node, including Name Node and Resource Manager, manages metadata of the HDFS and scheduling tasks. Slave nodes include Data nodes and Node Managers. They store actual data and manage the single node. Based on the master-slave mode, the master and slave nodes are deployed to run in independent pods. On the Kubernetes platform, the communication between pods is handled by a service. A headless service of the same name is deployed for each pod in the cluster. In addition, the cluster also sets up NodePort service for the master pod to expose the pod interior for Web UI viewing, and Hadoop/Spark cluster management. Figure 2 shows the logical architecture of the cluster.
- *Cluster Operator*: Kubernetes allows developers to extend the Kubernetes API with CustomResourceDefinition (CRD). Each custom resource requires a controller to perform specific operations. Operator SDK is a toolkit to manage Kubernetes native applications, called Operators, in an effective, automated, and scalable way [20]. Based on the operator framework, BDKit defines big data

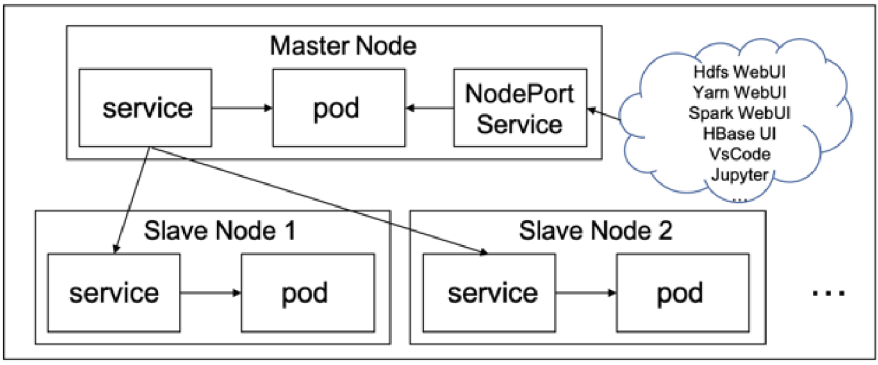

Fig. 2. Logical architecture of Hadoop/Spark cluster

CRD for Hadoop/Spark cluster resources, and constructs a controller to operate Hadoop/Spark cluster resources at business layer, e.g., automatic deploying and monitoring of cluster resources. Figure 3 shows the workflow of the cluster controller to construct a Hadoop/Spark cluster. When the cluster instance is available, it creates PVC and Git container as needed and create a master pod with relevant services. It checks whether the number of exiting slave pods (foundNum) equals the expected slave node number (expectNum) defined by the user. If not equal, the controller will create or delete slave pods to ensure foundNum is the same as expectNum.

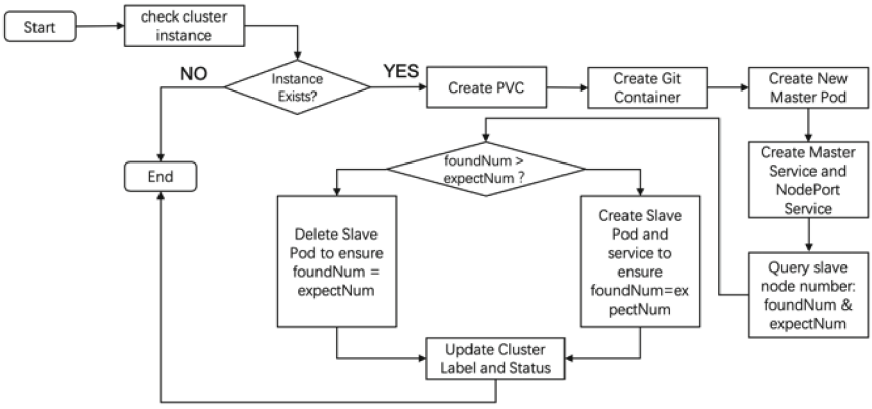

Fig. 3. Cluster controller workflow

(3) **API Interface Layer:** To separate concerns of front-end and back-end, BDKit designs an API interface layer to define APIs to listen to GET/POST requests from front-end clients, and invoke back-end API handlers to process the requests including cluster creation, deletion, etc. Its functions include:

- *Authentication*: The module is responsible for user login authentication and login status detection. BDKit implements permission authentication based on LDAP user authentication mechanism. It makes use of the middleware mechanism to judge the route type after receiving the request. If the route is private, the token is obtained from the request parameters and its validity is verified. Only the request that passes the token check is processed.
- *Cluster Management*: BDKit APIs provide comprehensive cluster management interfaces for the front end, including cluster creation, deletion, query and other functions. The above functions are based on the interfaces provided by the operator in business layer.
- *Facilities*: In addition to cluster management interfaces, the layer also provides interfaces for obtaining image list, data set list, current user information, and cluster container status, etc.

(4) **View Layer:** As a cloud service, BDKit provides web UI which is platform independent. Users can access it anytime from any device through a web browser. The view layer hides complex cluster deployment details for users, and provides concise and efficient operation and intuitive monitoring of big data related tasks, thus improves users' development efficiency. It is based on React development framework and Ant Design Pro UI framework to design web interfaces and implement the acquisition and display of back-end data. Its modules are as follows:

- *Cluster Configuration*: If there is no cluster instance, the homepage displays a list of available images for the user to select. It also allows users to customize their own image by composing different components. Figure 4 shows the page of image self-definition. When the image is chosen or composed, the user can customize the number of slave nodes in the cluster, the address of the Git repository, and the public data set provided by the platform. Big data computing components are also composable and customizable. Figure 5 is the cluster resource configuration page.
- *Cluster Overview*: When a cluster is created, the homepage displays the overall cluster status and provides shortcuts for accessing VsCode code-server, HDFS, etc.
- *Component Monitoring*: In addition to providing ports to access management interfaces such as HDFS and Yarn, BDKit integrates web UIs of Hadoop and Spark on the monitoring page to check the status of big data computing components.
- *Dataset Overview*: This module lists the public dataset configured by the platform administrators (e.g., the teacher or teaching assists). They have the privilege to upload experimental dataset to the platform. Students can only browse and mount the dataset to their cluster.

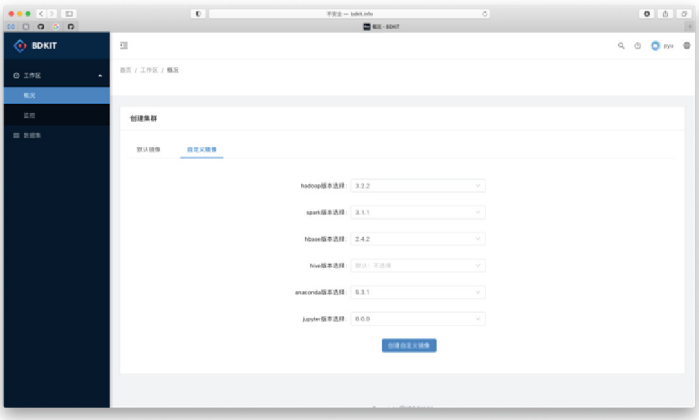

Fig. 4. Image self-definition page

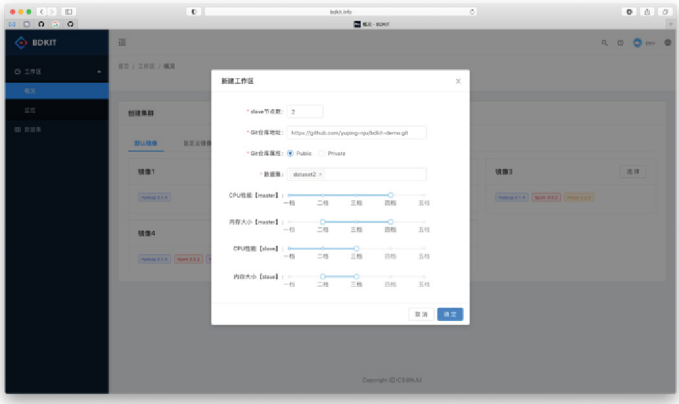

Fig. 5. User cluster resource configuration page

3 Experiments

The teaching objectives of financial big data processing course includes letting students: (1) be more in-depth understanding of the basic principles of big data processing; (2) be more familiar about emerging technologies in the field of big data; (3) be more confident with finance technology needs.

To achieve the objectives in a semester with 72 class hours in total, we arrange the teaching contents into five stages, accompanied with four experiments and one final project.

3.1 Experiments Design

Table 2 shows the class teaching contents and corresponding experiments in the last semester. The last column defines the experimental environment options.

Table 2. Experiments design

Stage	Teaching Contents	Experiments	Environments
1	Introduction to Hadoop and HDFS	Installing Hadoop in a standalone mode and learning HDFS operations	Standalone
2	Basic and advanced MapReduce Programming	Massive text data processing with inverted index and word co-occurrence in MapReduce	Distributed Cluster or BDKit
3	Introduction to NoSQL database, HBase and Hive	Learning HBase/Hive operation and programming	Distributed Cluster or BDKit
4	Introduction to Spark ecosystem and stream computing	Massive data aggregation and classification in Spark	Distributed Cluster or BDKit
5	Introduction to financial big data processing	Individual loan default forecast (final project)	Distributed Cluster or BDKit

- **Experiment 1:** After introduction of Hadoop and HDFS, we assign an experiment of Hadoop installation and HDFS operation. As a first step to big data technology ecosystem, we require students to install Hadoop in standalone mode in their own computers. Linux is recommended as the default operating system. After that, we suggest them to install Hadoop and HDFS in pseudo distributed mode. For the students who are interested in parallel computing, we suggest them to construct a distributed cluster by means of virtual machine or docker.
- **Experiment 2:** The most important part of this course is training big data processing skills. We put many efforts on teaching how to design MapReduce programs with a series of examples, such as wordcount, sorting, word co-occurrence, inverted indexing, page rank, etc. After basic programming training, we introduce how to design parallel machine learning algorithms for big data analysis with MapReduce framework. In the second experiment, we ask students to process massive text data (e.g., Shakespeare's collected work) to accomplish the tasks of inverted indexing and word co-occurrence. Since Hadoop is implemented in Java, Java is the recommended programming language. Python is also allowed if the students are not familiar with Java.
- **Experiment 3:** NoSQL is a new scheme of database design that enables the storage and querying of data outside the traditional structures in RDBMS [21]. In the third stage, we clarify the principles of NoSQL database and compare NoSQL with SQL, taking an example of HBase. HBase is built on top of HDFS that provides a way of storing sparse data sets, which is commonly used in many big data applications. We also introduce Hive which is a data warehouse in the Hadoop ecosystem. Hive facilitates reading, writing, and managing large datasets residing in distributed storage by using SQL. To let students better understand NoSQL database and data warehouse,

we assign the tasks of creating big tables, storing data in HBase and Hive, manipulating and querying data with a series of operations.

- **Experiment 4:** Spark is a unified engine for executing data engineering, data science, and machine learning on single-node machines or clusters [2]. It is now the most widely-used computing engine for big data. In this stage, we introduce the fundamental principles of Spark and its important libraries, including Spark SQL, Spark ML, GraphX and Spark Streaming. We ask students to do massive data aggregation and classification by use of Spark ML. By collecting streaming data with a data sever such as Kafka [22], students are able to complete the stream computing task with Spark Streaming.
- **Final Project:** Financial innovation upgrades the need of making full use of big data technology and practicing big data thinking. In the end, we discuss big data applications in financial field with some real-world cases, such as credit risk forecasting, insurance fraud identification, bank customer portrait. We design a final project which derives from a big data competition. It takes personal credit in financial risk control as the background, and predicts the possibility of default according to the information of the loan applicant, so as to judge whether the loan is approved or not. The training data selected in this project is the default record data of a network credit loan product. The project consists of four tasks which involve data statistics (MapReduce and Spark programming), data online analytical processing (Hive or Spark SQL) and machine learning (Spark ML). In the past semesters, we have chosen different topics for the final project, including sentiment analysis of securities news, prediction of e-commerce returned customers, etc.

Except the first experiment, all other work should be done in a distributed cluster. BDKit provides convenient computing environment for the students who have difficulty in setting up a cluster with sufficient resources.

3.2 Experiments Feedback

In the end of last semester, we made a survey among the students about the time spent in each experiment and got some feedback about BDKit. In the class, 12 students (Group 1) finished all work by using their own computers. 20 students (Group 2) learned to use BDKit after a short tutorial given by the teaching assistant. They did Exp. 2–4 and the final project (Exp. No. 5 in Table 3) based on BDKit. Table 3 shows the average time costing on each experiment by the two groups. It shows that BDKit can help students to save much time and effort, especially on cluster setting up and framework configuration. Figure 6 is a snapshot of Kubernetes pods creating by BDKit. Each student was assigned at least three pods to execute big data processing tasks. Among Group 2, only two students selected Jupyter notebook as the code editor, all others chose VsCode. In the terminal of VsCode, they could control their Hadoop/Spark cluster with command line tools.

Table 3. Time cost (hour)

Exp. No	Expected Time	Average Time without BDKit (Group 1)	Average Time with BDKit (Group 2)
1	2	6.8	/
2	6	16.3	7.4
3	4	8.1	3.2
4	6	8.9	5.1
5	12	19.3	10.3

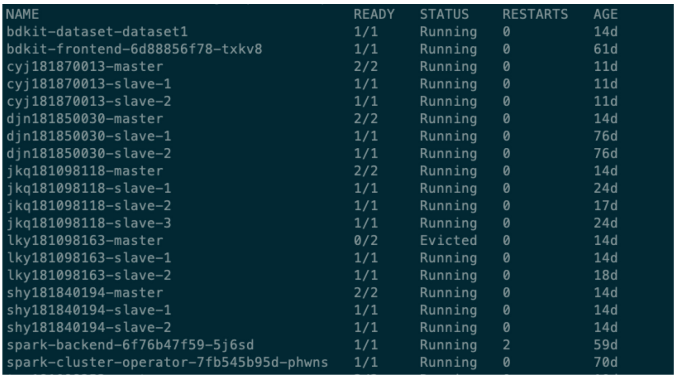

Fig. 6. Snapshot of pods created by BDKit

4 Evaluation

BDKit is deployed in cs-cloud, which is a cloud supercomputing platform of Nanjing University. BDKit runs in a Kubernetes cluster with 1 master node and 2 worker nodes. The master node is allocated 48 core CPU and 128 GB memory. Each worker node has 8 core CPU and 16 GB memory. The cluster is scalable by easily adding or removing worker nodes. Due to the access restriction, it can only be visited from campus network now. User accounts are registered by teaching assistants in advance.

4.1 Performance Evaluation

To verify that BDKit works as we expect, we use BDKit for the most basic WordCount MapReduce task to test its performance under different volumes of inputs. The experimental design is as follows: The user first created a Hadoop cluster containing one master and two slaves on BDKit, and then processed the text files of different sizes as input for WordCount. The time command in Linux was used to measure the actual experience time of the user. The size of the text files varied from 1MB to 800 MB. When the input file size is 800MB, the actual execution time on BDKit is about 100s. Moreover, it can be seen from Fig. 7 that the processing time increases linearly with the input size,

indicating that the BDKit platform can meet the performance requirement of teaching and learning.

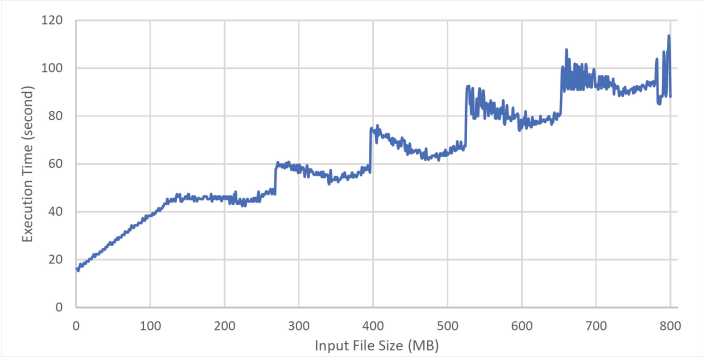

Fig. 7. WordCount execution time on BDKit

4.2 Comparison

There are mainly two types of big data platforms on the market: commercial cloud platform and educational training platform.

Compared with commercial cloud platform such as Amazon EC2 and Ali Cloud, BDKit provides students with customizable experiment environment. They are free to compose different components on demand. All datasets used in the experiments can be uploaded in advance. cStor is a popular educational big data platform [23]. Its technology stack largely overlaps with BDKit, but differs in the actual experience. cStor only provides SSH remote login interface and VIM for users. It does not support specific component and resource customization. BDKit provides VsCode and integrates with complete compilation packaging environment and code version control system. Jupyter Notebook is also available for Python developers. BDKit is more friendly to students. As a platform based on a new technology stack, it is also open for research objectives.

4.3 Teaching Effect

Financial big data processing course was initiated in 2017. It is a compulsory course of computer and finance interdisciplinary experimental class. We started to develop BDKit in late 2019 based on the requirements collected from three classes of students (2017–2019). It was put into use in September 2020. During two semesters of trial, we got some bug reports and improvement suggestions from students. Most students gave their approval to BDKit in their final semester report. Two students from business school and department of mathematics took this course as an elective one. They said that they wouldn't have finished the experiments without BDKit. It is worth mentioning that three students from the experimental class took part in the development and maintenance of BDKit. They had more interest in software technology than financial engineering. They

learned to develop and test cloud-native applications in their spare time. BDKit helps to achieve the purpose of teaching students according to their aptitude.

Based on the graduation report from two classes of students (2020 and 2021), at least one third of the students chose computer science for further study or employment. This course was awarded "a good course in my mind" by graduates in 2019.

5 Conclusion

In the era of digital economy and the new international situation, the integration of computer and finance is becoming more and more important. High-level talents in the field of financial engineering must be compound talents with solid foundation of computer technology and finance. Nowadays, a growing number of universities offer double majors in computer science and finance, such as Tsinghua university and Nanjing University. The project has attracted many top students since its inception. It requires to explore a new path of interdisciplinary talent training of the two challenging subjects. Financial big data processing course is an attempt to blend the two disciplines, as a teaching program with fintech features. It focuses on training students to apply big data processing skills in financial data analysis.

To overcome the problems of complex big data computing cluster and high threshold of operation, this paper designed and implemented an online experimental platform named BDKit. It is a container cloud-based platform for developing and running big data applications. It supports the creation of Hadoop/Spark clusters and the online development, running, and monitoring of big data applications. It can realize personalized cluster configuration and minute-level cluster creation, greatly reducing the workload of big data application developers. BDKit needs to pursue higher stability and security while ensuring its functional availability. In the next step, we will upload some typical financial data processing cases to the platform as learning materials. User registration will be open when we deploy it on a public cloud.

Acknowledgements. This work is supported by the National Natural Science Foundation of China (Grant No. 62072225) and the reform of university computer empowerment education in the new era project (Research on the reform of university computer empowerment education in New Liberal Arts) founded by University Computer Course Teaching Steering Committee of the Ministry of Education.

We would also like to thank all the students who participated in the development of BDKit. They are Kai Wang, Manjie Yuan, Yitong Zhao, Maidi Wang, Dongyu Wang and Haogang Wang.

References

1. Hadoop. http://hadoop.apache.org
2. Spark. http://spark.apache.org
3. HBase. http://hbase.apache.org
4. Hive. http://hive.apache.org
5. Cloud Computing. https://www.ibm.com/cloud/learn/cloud-computing

6. Rabkin, A., Reiss, C., Katz, R., et al.: Experiences teaching MapReduce in the cloud. In: 43rd ACM Technical Symposium on Computer Science Education, pp. 601–606. ACM Press, New York (2012). https://doi.org/10.1145/2157136.2157310
7. Eckroth, J.: Teaching big data with a virtual cluster. In: 47th ACM Technical Symposium on Computer Science Education, pp. 175–180. ACM Press, New York (2016). https://doi.org/10.1145/2839509.2844651
8. Ngo, L.B., Duffy, E.B., Apon, A.W.: Teaching HDFS/MapReduce systems concepts to undergraduates. In: 2014 IEEE International Parallel & Distributed Processing Symposium Workshops, pp. 1114–1121. IEEE Press, New York (2014). https://doi.org/10.1109/IPDPSW.2014.124
9. Apache Storm. http://storm.apache.org
10. Apache Flink. http://flink.apache.org
11. Brewer, E.A.: Kubernetes and the path to cloud native. In: 6th ACM Symposium on Cloud Computing, p. 167. ACM Press, New York (2015). https://doi.org/10.1145/2806777.2809955
12. Kubernetes. https://kubernetes.io/docs/concepts/overview/kubernetes-api/
13. Docker. https://www.docker.com
14. Bhimani, J., Yang, Z., Leeser, M., Mi, N.: Accelerating big data applications using lightweight virtualization framework on enterprise cloud. In: 2017 IEEE High Performance Extreme Computing Conference, pp. 1–7. IEEE Press, New York (2017). https://doi.org/10.1109/HPEC.2017.8091086
15. Shah, J., Dubaria, D.: Building modern clouds: using docker, kubernetes & Google cloud platform. In: 9th Annual Computing and Communication Workshop and Conference, pp. 184–189, IEEE Press, New York (2019). https://doi.org/10.1109/CCWC.2019.8666479
16. VS Code in the browser. https://github.com/coder/code-server
17. Jupyter Notebook. https://jupyter.org
18. Git. https://git-scm.com
19. Anaconda. https://www.anaconda.com
20. Operator SDK Integration with Operator Lifecycle Manager. https://sdk.operatorframework.io/docs/olm-integration/
21. NoSQL Databases. https://www.ibm.com/cloud/learn/nosql-databases
22. Apache Kafka. https://kafka.apache.org
23. cStor. http://www.cstor.cn/proTextdetail_13713.html