ISBN 978-981-11-7861-0
Proceedings of 2018 the 8th International Workshop on Computer Science and Engineering (WCSE 2018)
Bangkok, 28-30 June, 2018, pp. 634-641
doi: 10.18178/wcse.2018.06.106

Optimizing the Efficiency and Openness of the Kubernetes Based Code Evaluation Platform

Zhixiong Zhang[+], Gang Yin, Tao Wang, Wei Wang

National Laboratory for Parallel and Distributed Processing, National University of Defense Technology, 410073, Changsha, China

Abstract. With the popularity of programming learning, there are more and more online programming platforms and learners can learn programming skills online. That's good news for users who was bothered by configuring a development environment or some other annoying little things. We have built a multi-purpose code evaluation platform which provides many different kinds of coding training courses. This code evaluation platform is based on Kubernetes and provides unique containers for every submission of users' code so that it's not a problem that many different types of code are running at the same time. Furthermore, it allows experts share their programming skills or other computer technical abilities by creating training courses on the platform, and they can even customize the evaluation methods and the evaluation process of their own training courses by producing new evaluation script. To obtain better user experience for both learners and training contributors, some useful auxiliary systems, such as the evaluating queuing system and the script management system have been introduced.

Keywords: code evaluation platform, high concurrency, kubernetes, resource management

1. Introduction

To help users learning programming skills and coding online, we built a coding training platform (educoder.net). Unlike the LeetCode.com or other popular online judge website, this platform provides not only basic programming exercise, but also some complex training, such as Android and Hadoop training courses.

This multi-purpose platform bases on a Kubernetes cluster, and every code evaluation task runs in a specific Kubernetes pod so that different kinds of evaluation tasks won't interfere with each other. For each user of the platform, their code will be executed in its exclusive environment.

And to make the platform more open and gain a sustainable growth opportunity, it's recommended that experienced developers contribute IT training courses to the platform and share their own valuable skills and knowledge. One provides a new type of IT training course should also consider how to evaluate the submission for every task in this course and then create a corresponding evaluation script, so that users reaching to this course can get their submission evaluated and tell whether they master the lesson or not.

There are two biggest problems in the process of building this code evaluation platform. For learners, they care about the stability and timeliness of the evaluation process. But once users gush in, the high concurrency may damage the stability of the platform, because every evaluation task will request for a unique container to run and it will consume resources. If there aren't enough resources to support the operation of all containers, the platform won't guarantee a good user experience for learners.

The other problem is for course contributors. To share new courses in the platform, contributors should not only care about the content of their courses, but also design suitable evaluation methods for courses. The evaluation method for one specific training course will be performed as a shell script. Considering that creating

[+] Corresponding author.
E-mail address zhangzhixiong16@nudt.edu.cn

a suitable shell script may difficult for some contributors, it should be designed properly so that contributors can get an unobstructed contributing experience.

To handle these two problems correctly, some auxiliary systems and other targeted measures have been proposed. For example, the evaluating queuing system is focused on the resource problem in a high concurrency situation and the script management system will help contributors share their knowledge and skills more easily.

2. Related Work

A lot of research work has been conducted on the problem of code evaluation platform designing and resource-related difficulties solving.

PKU ACM Judge Online (poj.org) can check the source code submitted by its users automatically. It provides a good choice for programming learners to practice algorithm.

Programming grid [1] is originally built by Yingwei Luo et al. It is aimed at computer-aided education for programming courses. It designed many resources to support programming courses and forms a virtual platform with teaching assistance and teacher-student interaction for programming courses.

Shiyanlou (shiyanlou.com) is a very famous online training platform, which can provide an isolated environment for every user. It provides a wide variety of programming training courses and users can do some training tasks by following their guide.

In fact, most OJ platforms only focus on basic programming languages' teaching [2, 3] and don't provide comprehensive developing skills at all, it's not enough if learners want to apply their knowledge to developing works. On the other hand, Shiyanlou and some other similar platforms do provide a lot of useful courses to improve users' practical ability, but it lacks an effective feedback mechanism for learners' learning process. Our work provides a solution for the designing of the comprehensive online programming training platform, by deploying the Kubernetes cluster, introducing the evaluation mechanism [4] and attracting course contributors [5]. Some problems arisen in this process have been considered and solved finally [6, 7, 8].

3. Methods

Overview. To build an open code evaluation platform and protect the user experience of both learners and contributors, the evaluating queuing system and the script management system have been put into use. Both of them are aimed to improve the efficiency and the openness of the platform and make it a better place to learn basic and advanced IT skills.

The basic architecture of our platform and the interaction of the two auxiliary systems can be illustrated by Figure 1.

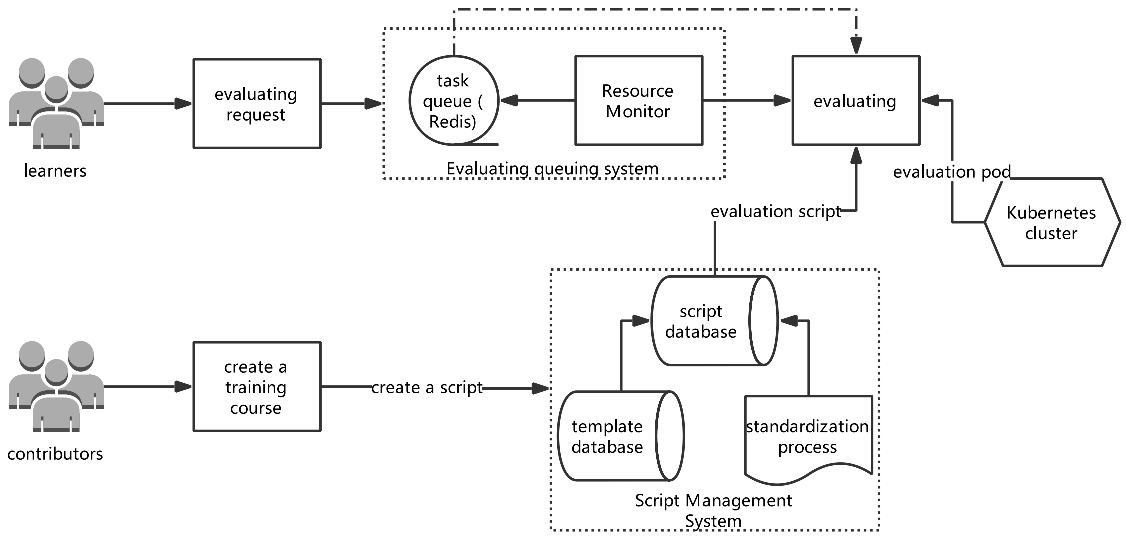

Fig. 1: Basic work flow of the platform.

Evaluating queuing system is primarily designed to optimize the learners' user experience while the script management system is contributor-oriented. Both of them are key subsystems and they work well together to support the operation of the platform.

It has been proved that the platform can hold a concurrent volume brought by 600 users' evaluation requests, and it will tend to lose control gradually if there are more and more users gush in. As shown in figure 2, every code evaluation request will run in a specific pod and a part of resources, including CPU and memory, will be consumed at the same time. If the usage of resources is unlimited, the evaluation process may gradually become very slow or even wrong [9]. To solve this problem, the evaluating queuing system brings a better resource control solution.

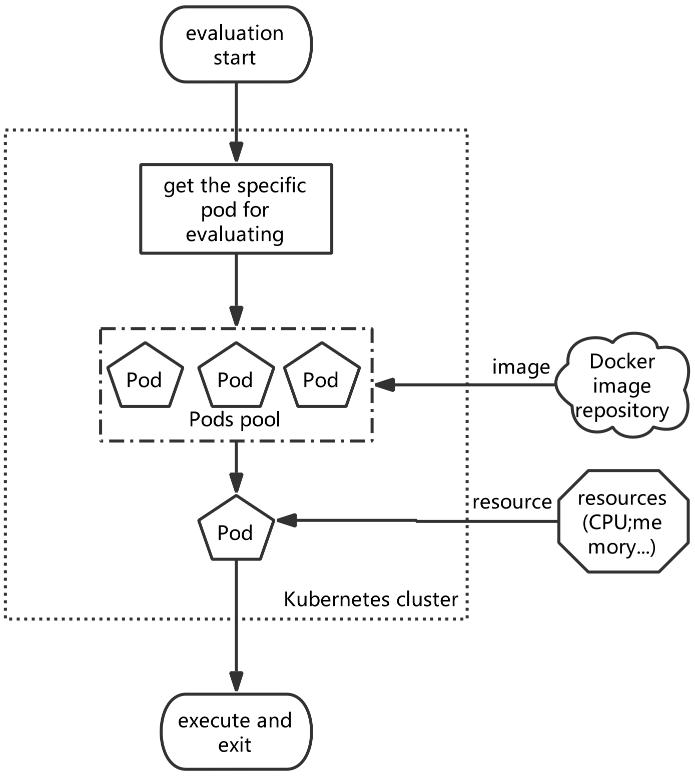

Fig. 2: Basic process of evaluation.

And the script management system is another creation to ensure that any IT expert in a field can share their skills or knowledge easier. This auxiliary system standardizes the basic process of code evaluation and tries to reuse existed evaluation scripts. The threshold of customizing a new IT training course will be reduced.

Evaluating queuing system. As is mentioned before, the code evaluation platform usually needs to work in a high concurrency environment [10]. To handle these evaluation requests, there are lots of Kubernetes pods asking to be built and run, which is very resource-intensive. In fact, when too many evaluation requests gush in and the number of pods exceeds the capacity of servers, the evaluation system will not reliable any more. So, it must be ensured that the platform works with enough remaining resources [11]. The most intuitive way to achieve this is denying the execution of evaluation tasks when the usage of resources is beyond the limit, and if these tasks can be deposited and waited to execute until the resource crisis is lifted, the stability of the code evaluation system will be improved a lot.

Evaluating queuing system is designed to make evaluation tasks line up and wait opportunities to execute when resources are scarce (as shown in figure 3). When an evaluation request is sent to the code evaluation platform, it will check if there are too many pods in the system and calculate resource usage at this moment. Specifically, the evaluating queuing system will push tasks into a sorted set built of Redis when there are more than 200 pods existing or the usage of memory has exceeded 80% of the whole system. What's more, users will be informed of their evaluation requests' positions in the queue so that they can decide whether to wait for evaluation results or leave to next challenge, and no matter how they choose, their waiting tasks will be executed and the results will be persistent finally.

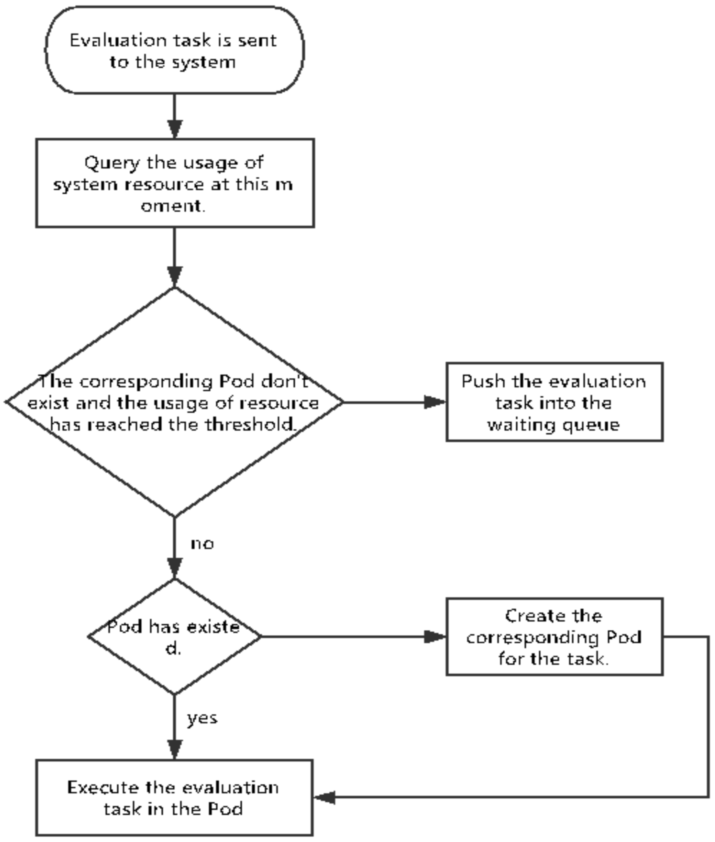
Evaluation task is sent to the system
Query the usage of system resource at this m oment.
The corresponding Pod don't exist and the usage of resource has reached the threshold.
Push the evaluation task into the waiting queue
no
Pod has existe d.
Create the corresponding Pod for the task.
yes
Execute the evaluation task in the Pod

Fig. 3: Work flow of evaluating queuing system.

Once an evaluation pod is ready to destroy and release its resource, the evaluating queuing system will check if there are waiting tasks in the queue. And the first one in that queue will get the chance to run.

Pods' Delayed-deleting Policy. As pointed out, an evaluation task which is waiting in the queue will execute only if another task has finished and its evaluation pod ready to be destroyed. To shorten the waiting time for a task in queue, it's necessary to delete an idle pod as soon as possible. At the very beginning of designing the code evaluation platform, a pod will keep alive for 30 minutes once it was built, and that is too long to adapt the evaluating queuing system. Essentially, the evaluating queuing system is a compromise solution to the problem of insufficient resources, and the waiting time of an evaluation task is a potential threat for the user experience. So, a better pod deletion policy should be employed.

Delayed-deleting policy is utilized to optimize the effect of the evaluating queuing system. Every time when an evaluation task arrives and its specific pod is created, there will be a 5 minutes countdown (active period). This 5 minutes countdown will be restarted every time when the corresponding pod is triggered by a new evaluation request, and if there isn't any new evaluation task during the countdown, the pod will be forced to delete finally. What's more, when an evaluation task has passed the challenge, the delayed-deleting policy will check if the pod is idle and safe to be deleted, and if no any other tasks in this pod, it will be destroyed immediately.

Script Management System. Our code evaluation platform is aimed to open to both learners and contributors who share their IT skills or knowledge. Attracting more contributors is one of the most important thing to the development of the platform [12]. But in fact, the evaluation mechanism of our platform is always the biggest obstacle, because contributors should not only care about the content of their courses, but also the creation of evaluation scripts. The evaluation process is defined by a corresponding shell script. And if contributors want to share new types of IT training courses, the platform may ask them to provide the suitable evaluation script at the same time if there are not suitable and reusable scripts. The evaluation script will run in the Kubernetes pod and check users' submission.

Providing evaluation script is not a simple work for lots of contributors. Shell programming skills are essential and contributors should find out suitable evaluation process for their courses. And that undoubtedly raised the threshold for the contributors.

Script management system is designed to reduce difficulty for contributors to create evaluation scripts. And the basic work flow of this auxiliary system can be illustrated by figure 4. In most instances, the type of a new training course may already exist in the platform so that the contributor can reuse the corresponding evaluation script directly. What's more, if one wants to create a new type of training course and there isn't any reusable evaluation script, the script management system will help the contributor create a new script quickly and easily. Script management system standardizes the basic process of evaluation so that contributors can focus on the personalized part of the whole process. In general, the platform will only ask contributors to provide the compilation methods and the execution methods of the language type their courses belong to, and then a complete evaluation script will be automatically generated. And of course, contributors can continue to modify the generated scripts if needed.

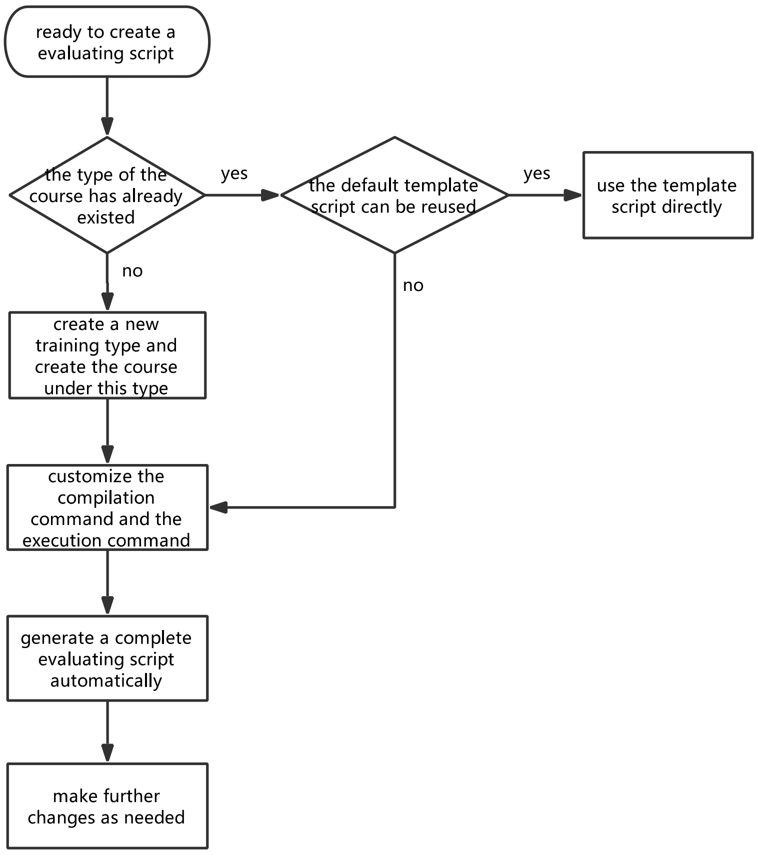
ready to create a evaluating script
the type of the course has already existed
yes
the default template script can be reused
yes
use the template script directly
no
no
create a new training type and create the course under this type
customize the compilation command and the execution command
generate a complete evaluating script automatically
make further changes as needed

Fig. 4: Work flow of script management system.

Once a new type of evaluation script puts into use, the script management system will pick this script as the default template of this programming language type and recommend follow-up training courses to reuse this script.

4. Results and Discussion

Results. High concurrency was the biggest hurdle for the stability of evaluation process [13,14]. Learners may send lots of evaluation requests for their code and some of their code is complicated and time-consuming. Fortunately, the evaluating queuing system helps the platform to avoid catastrophic consequences. There won't be too many evaluation pods in the platform at a moment so that resources of servers are exempt from exhausting. As is shown in the figure below, after the evaluating queuing system has been adopted, evaluation time stops losing control finally.

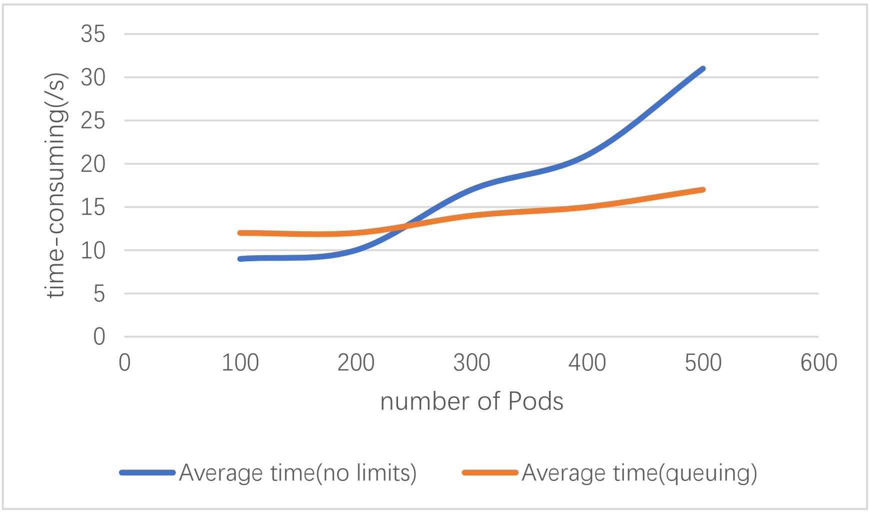

Fig. 5: Average time-consuming trends of the two strategies.

In fact, evaluating queuing system makes sure that the usage of resources is limited to a reasonable value, which leads to a lower failure rate of evaluation. As we all know, usability is the most important aspect in the designing process, and the queuing system has resolved the biggest problem which may disrupt the usability of the platform. Another benefit brought by the evaluating queuing system is that now we can monitor the system status better. The usage of resources of the servers is easy to be gotten and can be designed as a switch of the queuing system. To adopt the strategy of queuing evaluation requests, the platform can run stably and error-free.

And of course, to reduce the negative impact of evaluating queuing system, the adoption of delayed-deleting policy is also indispensable. As is mentioned in the method section, after using the evaluating queuing system, some evaluation requests may be in waiting status for a while if the platform lacks of resources. The occasion of deleting one evaluation pod has been adjusted by delayed-deleting policy so that a waiting evaluation request will be executed as soon as possible.

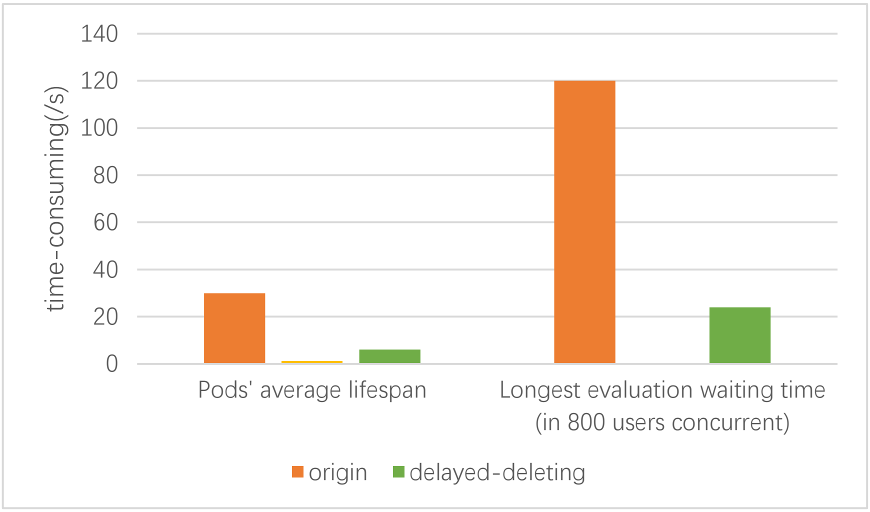

Fig. 6: Comparison between the two pod-deleting policies.

Script management system has helped more than 30 contributors to create more than 50 training courses in the platform, and some of them are complex and large-scale. The number and the quality of training courses are steadily improving. Contributors can share their courses easier and they can more focus on the content of their sharing now.

Characteristic Analysis. The two auxiliary systems, including the evaluating queuing system and the script management system, have made the code evaluation platform robust, efficient and user friendly.

Robust: By adopting the evaluating queuing system, the load of the platform won't be too heavy to exceed the limit. In a high concurrency environment, evaluating queuing system ensure the stability of the platform and make it run regularly all the time.

Efficient: Even it lacks of resources and the evaluating queuing system is triggered, the delayed-deleting policy will take both stability and effectiveness into account. On the one hand, the delayed-deleting policy guarantees that the code evaluation platform can handle a large number of tasks efficiently. On the other hand, when a user is coding and trying to pass the evaluation, the platform will keep his Kubernetes pod alive during the active period and protect the coding environment in the pod, which will lead to a better user experience.

User friendly: For learners, the stability of the platform can ensure their good experience when they try to master courses, and many measures have been adopted to ensure the efficient of the evaluation process. And for contributors, the platform ensures that they can do as little as possible to create an evaluation script and pay their attention to the course content by adopting script management system, which makes the platform easier to use.

All of these characteristics make our platform a perfect choice for learning basic program languages or advanced software engineering skills. What's more, it opens to all experts who want to share their knowledge and try to make this process more attractive.

5. Conclusion and Future Work

Conclusion. In this paper, we introduce our code evaluation platform briefly and put forward some problems and defects that the platform faced up. We focus on these weak points and try to find ways to avoid traps or solve problems. For this purpose, we propose several auxiliary systems and strategies. The evaluating queuing system and the pods' delayed-deleting policy works together to make sure that our platform obtains both higher stability and better usability in a high concurrency environment. And the script management system designed to help contributors earn an easier way to build their own IT training courses, which will obtain a sustainable development for the platform. The efficiency and the openness of our platform have been improved a lot since these methods put into use.

Future Work. The evaluating queuing system is the key to maintain the stability of our platform, and there is room for optimization. We prepared for adding more conditions to help the queuing system decide whether or not to start working, such as the real-time status of servers and the concurrent number of evaluation tasks which are in process.

As for the pods' deletion policy, we prepare for building a more flexible policy to handle different types of training tasks in our platform. Some of pods may need longer survival time but others don't, so it will help if the setting is not fixed.

And the Docker image management system will come soon, it will further reduce the difficulty of creating a new type of IT training course. Both Docker images and evaluation scripts are important resources in the platform and these resources should be managed correctly and efficiently.

6. Acknowledgements

This work is supported by the National Science Foundation of China with Grant No. 61502512, No.61432020, No. 61472430 and No.61532004. And lots of thanks should be given to EDUCODER team members, especially to Jingquan Huang, Ao Dai, Zhengdong Chen and Wei Wang for their valuable help.

7. References

[1] Luo Y, Wang X, Zhang Z. Programming grid: a computer-aided education system for programming courses based on online judge[C]//Proceedings of the 1st ACM Summit on Computing Education in China on First ACM Summit on Computing Education in China. ACM, 2008: 10.

[2] Petit J, Giménez O, Roura S. Jutge. org: an educational programming judge[C]//Proceedings of the 43rd ACM

technical symposium on Computer Science Education. ACM, 2012: 445-450.

[3] Wu J, Chen S, Yang R. Development and application of online judge system[C]//Information Technology in Medicine and Education (ITME), 2012 International Symposium on. IEEE, 2012, 1: 83-86.

[4] Kosowski A, Małafiejski M, Noiński T. Application of an online judge & contester system in academic tuition[C]//International Conference on Web-Based Learning. Springer, Berlin, Heidelberg, 2007: 343-354.

[5] Tonin N A, Bez J L. Uri online judge: A new interactive learning approach [J]. Computer Technology and Application, 2013, 4(1).

[6] Joy A M. Performance comparison between linux containers and virtual machines[C]//Computer Engineering and Applications (ICACEA), 2015 International Conference on Advances in. IEEE, 2015: 342-346.

[7] Marmol V, Jnagal R, Hockin T. Networking in containers and container clusters [J]. Proceedings of netdev 0.1, February, 2015.

[8] Saito H, Lee H C C, Hsu K J C. Kubernetes Cookbook [M]. Packt Publishing Ltd, 2016.

[9] Nordstrom M L, Ricard G R, Vriezen J J, et al. High concurrency in use manager: U.S. Patent 5,265,245[P]. 1993-11-23.

[10] Yan C, Chen N, Shuo Z. High-Performance Elastic Management for Cloud Containers Based on Predictive Message Scheduling [J]. Future Internet, 2017, 9(4): 87.

[11] Zhu J, Zhang J, Zhang C, et al. CHRS: Cold Start Recommendation Across Multiple Heterogeneous Information Networks[J]. IEEE Access, 2017, 5: 15283-15299.

[12] Zhu J, Zhang J, Wu Q, et al. Constrained Active Learning for Anchor Link Prediction Across Multiple Heterogeneous Social Networks[J]. Sensors, 2017, 17(8): 1786.

[13] Zhu J, Zhang J, He L, et al. Broad Learning based Multi-Source Collaborative Recommendation[C]// ACM, 2017:1409-1418.

[14] Lobb R, Harlow J. Coderunner: A tool for assessing computer programming skills [J]. ACM Inroads, 2016, 7(1): 47-51.