Container Performance Prediction: Challenges and Solutions

Jiwei Wang[1(✉)], Yuegang Li[1], Congfeng Jiang[1], Chao Ma[2], Linlin Tang[2], and Shuangshuang Guo[2]

[1] School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, China
{wangjiwei,lyg,cjiang}@hdu.edu.cn
[2] Information & Telecommunications Company, State Grid Shandong Electric Power Company, Jinan 250000, China
machao@sd.sgcc.com.cn, 6335875@qq.com, 416054882@qq.com

Abstract. With popularity of cloud computing services, more and more tasks and services are deployed on large-scale clusters. As an emerging technology in cloud computing field, containers make virtualization extremely lightweight. However, lack of prediction causes scheduling decisions lag behind the dynamics of clouds. Thus, how to carry out performance prediction before container scaling has become an urgent problem to be resolved. Here we emphasized the necessity of container performance prediction and summarized the current research progress and effort of container performance modeling. Finally, we compared pros and cons of numerical analysis and machine learning in terms of practice.

Keywords: Virtualization · Container · Containerized cloud · Performance prediction · Machine learning

1 Introduction

With rise of social network and popularity of IoT (Internet of Things) devices, amounts of data generated around the world is increasing exponentially. However, most of the existing cloud systems are unable to satisfy such capacity of demands effectively. Compared to direct way of running services on separate physical machines, virtualization alleviates the suffering of inflexible problems and make the whole system more cost-efficient. However, full virtualization introduces non-negligible overhead when running services, as VMs (virtual machines) simulate not only the whole operating system, but also hardware architecture and network environments. As result of this, full virtualization is not suitable for compute-intensive and request-intensive scenarios.

Recently, with demand of resource effectiveness, a light-weight virtualization technology called container is actively promoting the development of the next generation of cloud services [17]. Different from full virtualization technologies

H. Gao et al. (Eds.): ChinaCom 2020, LNICST 352, pp. 392–402, 2021.
https://doi.org/10.1007/978-3-030-67720-6_27

(represented by VMs), several containers share the same kernel of the operating system, by which much performance loss resulted from hardware virtualization can be eliminated drastically. In early practice of containers, scheduling strategies are simply transferred from existing works on VMs. However, with containers widely deployed in production practice, many evidences have proved that scheduling containers is a more open question compares to VMs. As container provides flexibility to cloud environment, it brings new challenges like low-isolation, scheduling complexity and compatibility with existing systems.

Above challenges are promising to be tackled by sufficient study of characteristics of containers and novel performance prediction methods.

The remainder of this paper is organized as follows. Section 2 illustrates challenges existing in each modeling methods, while Sect. 3 contrasts numerical analysis with machine learning methods in detail and Sect. 4 concludes the paper and comes up with new potential optimization direction of predictions.

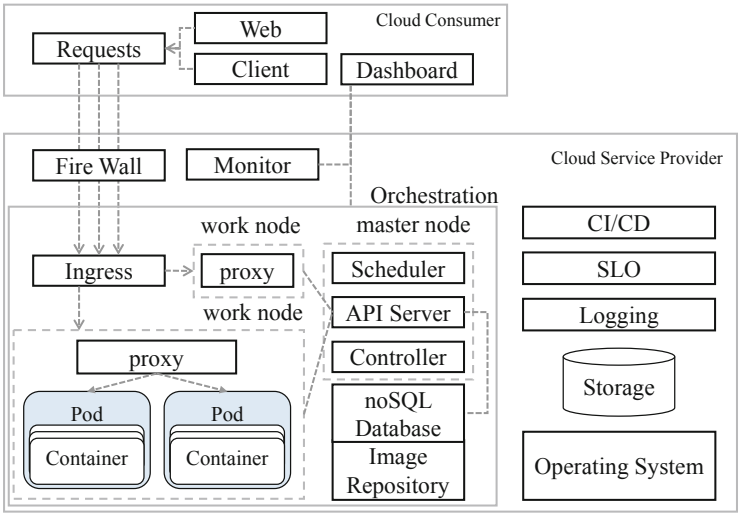

Fig. 1. Paradigm of containerized cloud.

2 Challenges in Performance Modeling

As paradiam of containerized cloud illustrated in Fig. 1, orchestration plays the core role inside a containerized cloud. In essence, container-orchestration systems is the manager of container life-circle, and their strategies determine the QoS (Quality of Service) and resource efficiency of the long-run services. In this section, we studied existing container modeling solutions. To our knowledge, container performance modeling methods can be divided into two main parts: traffic modeling methods that study outer factors and resource modeling methods that

capture inner characteristics from containers to whole containerized cloud. Their benefits and challenges are concluded in Table 1.

Table 1. Modeling Solutions for containerized cloud

Category	Solutions	Benefits	Challenges
Incoming	Requests;	User behaviors characterized;	Lack of difference between tasks;
	Task structure;	Inner relationship of tasks captured;	Dependencies are hard to be obtained;
	Scientific computing;	Accurate processing time prediction;	Scenarios are specified and highly limited;
Runtime	Physical constraints identification;	Clarified boundaries;	Heterogeneous problems;
	Dynamic resource consumption;	Fine-grained model;	Absence of external factors;

2.1 Incoming Modeling

Request. Service calls, usually raised by requests event-driven, are strongly related to workload. However, every single container has limited request process capacity. The study of requests characteristics provided lots of inspiration for how to rule the performance of containers based on the real-time requests situation [15].

Task Structure/Topology. As dependencies existing between tasks, order of tasks processing is always strictly defined in advance. Bao et al. [1] modeled every application as DAGs (Directed Acyclic Graphs), in which services inside of applications are defined as verticals and volume of data delivered between each service is defined as weight. Yang et al. [27] transformed DAG-formed tasks to queue models, thus high complexity is avoided. However, dependencies aren't always explicitly declared, so such modeling methods' application scenario is highly restricted.

Scientific Computing. Nowadays, researchers in physics, chemistry, and biology start to upload their datasets to the scientific computing cloud for faster processing speed and lower monetary cost. Baughman et al. [3] found that algorithm configuration, datasets and the load level almost totally determined the performance of applications. However, in most of the general cloud systems, incoming data is organized as streams. So, modeling for datasets is limited to a narrow scenario.

2.2 Resource Modeling

As different services reveal diverse resource consumption patterns [7], one of the most tricky challenges is how to properly place containers with different resource patterns so that resource utilization and communication latency will be well balanced.

To better explain above issues, we assume that there are three services, which are named as A1S1, A1S2 and A1S3 separately and construct one single application together. Two manners of placement strategy are showed on Fig. 2 and distinguished by gray dashed line. The first represents the anti-affinity manner. Services are distributed to different containers, which will maximize the resource balancing (services with different patterns are co-located on the same machine) and disaster recovery (a few of failed nodes no longer matters). However, the extreme anti-affinity manner introduces significant communication overhead. On the right of the dashed line is the affinity manner, which represents that all services of one application are deployed into one single node. Affinity manner minimizes the communication overhead, but it worsens the resource efficiency, as some physical resources will be idle or saturate (memory and I/O in the example respectively). To achieve a better trade-off between resource efficiency and QoS violation, sufficient resource modeling study should be carried out.

Physical Constraints Identification. Profiling physical resource is challenging in practice, as it's difficult to decide which level of abstraction can best satisfy demand. At the same time, as legacy servers still deployed inside the clusters, heterogeneous problems make it tougher to handle resource modeling. In order to deal with the relationship between resource demand and resource availability, physical resource boundaries should be firstly clarified [12]. Moreover, heterogeneous resource like GPU (Graphic Processing Unit) hasn't been well supported by virtualization. To accommodate this, reasonable methods of allocating GPU resources were proposed to avoid contentions [6,26].

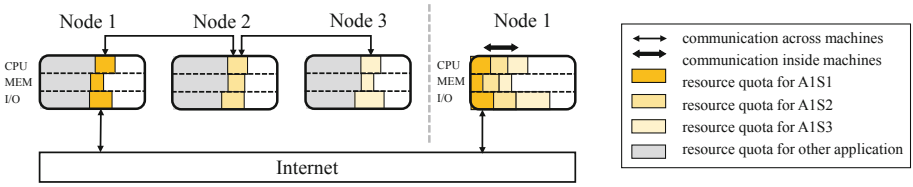

Fig. 2. Different manners of placement

Dynamic Resource Consumption. Dynamic features of consumption capture the status of the resource (e.g. CPU, memory, disk I/O, network bandwidth,

etc.) allocated to containers or left for idle. Rahman et al. [22] modeled end-to-end latency for micro-service by utilizing collected CPI (Clock per Instruction) of the VMs which host containerized micro-services. CPI describes the contention on physical resources caused by different containers or applications. However, most resource modeling methods make predictions without attention to incoming traffics. Lack of concern about external factors will limit the range of observation and thus hard to make accurate predictions in advance.

Table 2. Related Works in Performance Modeling and Prediction

Reference	Observed metrics*								Predicted objects	Constraints**	Base algorithm	Evaluation***
	CPU	MEM	NET	TOP	INC	LAT	BEH	ETC				
Meng et al. [18]	•								Container	\	ARIMA	Simulated
Zhao et al. [28]	•								Pod	\	ARIMA	Simulated
Kalim et al. [15]	•			•					Distributed systems	\	Prophet	Simulated
Callara et al. [5]							•		Application	Sampling rate	Probabilistic	Real-world
Bauer et al. [2]	•	•	•		•				Micro-service	\	Hybrid decomposition	Simulated
Hu et al. [12]	•	•	•		•				Container	Multi-resource	MCFP***	Simulated
Bao et al. [1]	•	•			•		•	•	Micro-service	Budget	Regression	Simulated
Baughman et al. [3]	•	•			•				Scientific workflow	\	Regression	Real-world
Thinakaran et al. [26]								•	Container	Resource	Regression	Simulated
Rahman et al. [22]	•							•	Micro-service	SLO	Regression/LSTM	Simulated
Jindal et al. [14]	•				•	•		•	Micro-service	SLO	Regression	Simulated
Scheuner et al. [23]	•	•	•	•				•	VM	\	Regression	Real-world
Podolskiy et al. [21]	•				•	•			Container	SLO	Regression	Real-world
Yang et al. [27]				•			•	•	Micro-service	Consumer number	LSTM	Real-world
Tang et al. [24]	•	•	•						Container	\	LSTM	Real-world

* "MEM" stands for "memory", "NET" for "network", "TOP" for "throughput", "INC" for "Incoming traffic/ request", "LAT" for "latency", "BEH" for "behavior", "ETC" for "other metrics".
** "\" means "not defined or quantified explicitly in the paper".
*** "MCFP" stands for "minimum-cost flow problem".

3 Solution Comparation: Numerical Analysis or Machine Learning?

Some studies supposed that workloads of cloud environment are changing irregularly, so its performance is unable to be predicted [4]. Under such assumptions, most works make scheduling decision base on performance-aware mechanism. However, their methods inevitably lag behind current demands of containers. To pre-empt the delays in scheduling and further improve the resource-efficiency, performance prediction is necessary.

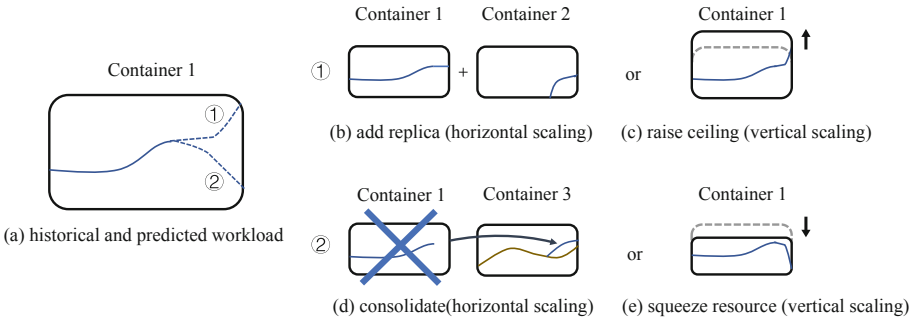

Fig. 3. Why prediction?

Benefits of container performance prediction can be illustrated in Fig. 3. In (a), historical workload is represented in solid line, while predicted workload in dashed line. Let's think about two predicted workload trend: increase and decrease. As light-weight nature of containers, horizontal and vertical scaling decision can be done in advance to avoid significant performance fluctuation. If coming spike can be foreseed by schedulers, replica of a container can be added to balance the additional workload, as shown in (b). Although action of activating new container replicas is transparent to cloud users, activating itself is still a CPU-intensive work. Therefore, as illustrated by in (c), raising resource ceilings for containers is an easier way to meet the surging demand. Above actions aim to maintain high-available services, in contrast, if workload will go down in the future, fractional idle resources can be re-collected for higher resource efficiency. In horizontal way demonstrated in (d), containers with low workload can be consolidated into one. Also, shown in (e), resource can be squeezed for other containers by vertical scaling way.

In this section, we will take an insight of state-of-art solutions for container performance prediction. As two main methods of implements, numerical analysis and machine learning work with totally different manner and have their own advantages and drawbacks (summarized in Table 3).

Table 3. Predicting solutions for containerized cloud

Category*	Solution	Advantages	Disadvantages
NA	ARIMA	Low complexity; Fast process time;	Sensitive to outliers; Lack of internal dynamics;
	Prophet	Support for seasonality & holiday effect;	Lack of internal dynamics;
ML	Regression	Training rapidly; Interpretable model;	Sensitive to outliers;
	LSTM	Well support for multi-variables and multi-objectives problems;	Large training datasets are necessary; Fail in some corner cases;

* "NA" stands for Numerical Analysis, while "ML" represents for Machine Learning.

3.1 Numerical Analysis

On behalf of a wide range of methods to search for mathematical laws underlying raw data, numerical analysis plays an important role in the design of the prediction mechanism. Although many researchers blamed traditional numerical analysis methods for their inflexibility and suboptimal solutions, short calculation time and problem-specific design make numerical analysis a competitive candidate for solving performance prediction in containerized cloud.

ARIMA (Autoregressive Integrated Moving Average)-Based Methods. ARIMA is commonly adopted in stationary time series scenario. Recently, efforts in adopting ARIMA in cloud environment harvested the good results by forecasting coming workload spikes [18,28]. However, as can be seen from Table 2, above ARIMA-based methods only took CPU as the indicator. To our knowledge, it is hard to capture internal dynamics of the system by observing a few features. Also, data source fed in ARIMA-based methods should be regularly spaced. That means, high-quality monitoring and elaborately preprocess are necessary. Although time series analysis methods' fatal flaws hampered their future development, their low complexity makes it a light-weight way to predict rapidly without too much computing/energy consumption.

Prophet. To avoid drawbacks (e.g. observed data should be spaced equally and outliers should be removed in advance) existing in ARIMA, Facebook released a temporal sequence prediction framework called Prophet [9]. Here, we briefly explain the mechanism of Prophet.

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t \tag{1}$$

$g(t)$ represents the trend model, $s(t)$ stands for seasonality (e.g. fluctuate daily, weekly or yearly), $h(t)$ is the holiday effect (e.g. festival or big sale), and ϵ_t is a normally distributed error item. Prophet generates future time series based on the yearly, weekly, and daily seasonality of the historical data, so it outperformed most of other similar frameworks in scale and flexibility [25]. By integrating Prophet framework as traffic forecast methods, Kalim et al. [15] established a model to obtain the throughput rate between each pair of components. However, Prophet framework is still lack of concerning of internal dynamics, and makes prediction from the surface.

As indicated above, the goal of numerical analysis is not to solve problems in far-ranging scenarios. In contrast, they are designed to fit in a highly specific area. However, elaborate adjustments of numerical analysis methods will gain certain accuracy in prediction and thus release positive effects on system performance.

Table 4. Works in prediction optimization

Reference	Target	FS*	AD*	EL*	Evaluation	Baseline
Liu et al. [16]	Adaptability	×**	•	×	Real-world	Basic version model (err)
Iqbal et al. [13]	Adaptability	×	•	•	Real-world	Liu et al. [16] (err)
Grohmann et al. [10]	Manual overhead	•	•	•	Simulated	Manually labelling (err)
Eismann et al. [8]	Manual overhead	•	•	×	Simulated	Basic version model (speed)
Rahman et al. [22]	Manual overhead	•	\	×	Simulated	SLO target

* "FS" stands for feature selection, while "AD" for adaptive design, "EL" for ensemble learning.
** "•" stands for fully support, while "×" for not support, "\" for not clearly claimed.

3.2 Machine Learning

Cloud services iterate rapidly, approaches without adaptability will soon become obsolete. Every update of the numerical analysis costs much domain expert efforts. To avoid such defect, machine learning methods, which don't need too much prior knowledge, are becoming increasingly popular [11,20].

Regression. Many works utilize regression models to explore quantified relationship between monitored metrics and container service performance [1,3,14,21–23,26]. With benefits of simplicity, training of most regression can be done rapidly and models are interpretable enough and easy to be generalized to more areas. However, regression methods are sensitive to outliers. As load of containers is highly dynamic, high latency or crash down of communication between containers may cause logging system to collect certain ratio of abnormal data. Regression model is not good at handling such exceptions, so preprocess of outliers is necessary if regression is applied in the containerized scenario.

LSTM (Long Short-Term Memory). Traditional time series methods can't deal with multi-variables and multi-objectives problems well. To tackle this problem, many researchers start to employ LSTM into containerized cloud. LSTM works well in the field of time series and thus suitable in prediction. Tang et al. [24] captured features between metrics by bidirectional LSTM, which outperformed basic LSTM in the ability of context-awareness. In the scenario of Web service and database, the accuracy of prediction improved over 50% compared to traditional ARIMA and LSTM-based methods. Miao et al. [19] utilized LSTM to make traffic prediction with the concern of heterogeneous characteristics of edge nodes. Although LSTM offers an excellent way to solve the multi-variables

and multi-objectives problems, as back-propagation process will fail when the layers are too deep, LSTM can't deal with periods exhibited in days or longer. Moreover, effective LSTM models are based on large training sets, and even highly manually adjusted machine learning methods may fail in some corner cases. Therefore, before we apply a new prediction solution, sufficient test and comparison between numerical analysis and machine learning methods should be done in several scenarios. Also, as listed in Table 4, manually tuning can be replaced by automatic methods gradually. Automatic methods not only save time and efforts, but also make predicting models adaptive enough to new environment even after certain modification in configurations.

4 Conclusion

In this paper, we clarified challenges in container performance modeling and emphasized a lot in the state-of-art achievements in terms of profiling and predicting containers. We illustrated pros and cons in solutions proposed by researchers and provided suggestions according to different scenarios. For future work, practical experiments will be carried out to compare each solution quantitatively. Moreover, based on sufficient insights, we plan to propose our own design of container orchestration platform.

Acknowledgments. This work is supported by the National Natural Science Foundation of China (No. 61972118).

References

1. Bao, L., Wu, C., Bu, X., Ren, N., Shen, M.: Performance modeling and workflow scheduling of microservice-based applications in clouds. IEEE Trans. Parallel Distrib. Syst. **30**(9), 2114–2129 (2019)
2. Bauer, A., Lesch, V., Versluis, L., Ilyushkin, A., Herbst, N., Kounev, S.: Chamulteon: coordinated auto-scaling of micro-services. In: 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), pp. 2015–2025. IEEE (2019)
3. Baughman, M., Chard, R., Ward, L.T., Pitt, J., Chard, K., Foster, I.T.: Profiling and predicting application performance on the cloud. In: UCC, pp. 21–30 (2018)
4. Boza, E.F., Abad, C.L., Narayanan, S.P., Balasubramanian, B., Jang, M.: A case for performance-aware deployment of containers. In: Proceedings of the 5th International Workshop on Container Technologies and Container Clouds, pp. 25–30 (2019)
5. Callara, M., Wira, P.: A probabilistic learning approach for predicting application launches in cloud computing architectures. In: 2019 IEEE/SICE International Symposium on System Integration (SII), pp. 584–589. IEEE (2019)
6. Chen, Q., Oh, J., Kim, S., Kim, Y.: Design of an adaptive GPU sharing and scheduling scheme in container-based cluster. Cluster Comput. **23**, 1–13 (2019)
7. Chen, W.Y., Ye, K.J., Lu, C.Z., Zhou, D.D., Xu, C.Z.: Interference analysis of colocated container workloads: a perspective from hardware performance counters. J. Comput. Sci. Technol. **35**, 412–417 (2020)

8. Eismann, S., Grohmann, J., Walter, J., Von Kistowski, J., Kounev, S.: Integrating statistical response time models in architectural performance models. In: 2019 IEEE International Conference on Software Architecture (ICSA), pp. 71–80. IEEE (2019)
9. Facebook Open Source: Prophet: Automatic forecasting procedure. https://github.com/facebook/prophet. Accessed 7 Oct 2020
10. Grohmann, J., Eismann, S., Elflein, S., Kistowski, J.V., Kounev, S., Mazkatli, M.: Detecting parametric dependencies for performance models using feature selection techniques. In: 2019 IEEE 27th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), pp. 309–322. IEEE (2019)
11. He, H., Hu, J., Da Silva, D.: Enhancing datacenter resource management through temporal logic constraints. In: 2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 133–142. IEEE (2017)
12. Hu, Y., Zhou, H., de Laat, C., Zhao, Z.: Concurrent container scheduling on heterogeneous clusters with multi-resource constraints. Future Gener. Comput. Syst. **102**, 562–573 (2020)
13. Iqbal, W., Berral, J.L., Erradi, A., Carrera, D., et al.: Adaptive prediction models for data center resources utilization estimation. IEEE Trans. Netw. Serv. Manag. **16**(4), 1681–1693 (2019)
14. Jindal, A., Podolskiy, V., Gerndt, M.: Performance modeling for cloud microservice applications. In: Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering, pp. 25–32 (2019)
15. Kalim, F., et al.: Caladrius: a performance modelling service for distributed stream processing systems. In: 2019 IEEE 35th International Conference on Data Engineering (ICDE), pp. 1886–1897. IEEE (2019)
16. Liu, C., Liu, C., Shang, Y., Chen, S., Cheng, B., Chen, J.: An adaptive prediction approach based on workload pattern discrimination in the cloud. J. Netw. Comput. Appl. **80**, 35–44 (2017)
17. Maenhaut, P.J., Volckaert, B., Ongenae, V., De Turck, F.: Resource management in a containerized cloud: status and challenges. J. Netw. Syst. Manag. **28**(2), 197–246 (2020)
18. Meng, Y., Rao, R., Zhang, X., Hong, P.: CRUPA: a container resource utilization prediction algorithm for auto-scaling based on time series analysis. In: 2016 International Conference on Progress in Informatics and Computing (PIC), pp. 468–472. IEEE (2016)
19. Miao, Y., Wu, G., Li, M., Ghoneim, A., Al-Rakhami, M., Hossain, M.S.: Intelligent task prediction and computation offloading based on mobile-edge cloud computing. Future Gener. Comput. Syst. **102**, 925–931 (2020)
20. Orhean, A.I., Pop, F., Raicu, I.: New scheduling approach using reinforcement learning for heterogeneous distributed systems. J. Parallel Distrib. Comput. **117**, 292–302 (2018)
21. Podolskiy, V., Mayo, M., Koey, A., Gerndt, M., Patros, P.: Maintaining SLOs of cloud-native applications via self-adaptive resource sharing. In: 2019 IEEE 13th International Conference on Self-Adaptive and Self-Organizing Systems (SASO), pp. 72–81. IEEE (2019)
22. Rahman, J., Lama, P.: Predicting the end-to-end tail latency of containerized microservices in the cloud. In: 2019 IEEE International Conference on Cloud Engineering (IC2E), pp. 200–210. IEEE (2019)

23. Scheuner, J., Leitner, P.: Estimating cloud application performance based on micro-benchmark profiling. In: 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), pp. 90–97. IEEE (2018)
24. Tang, X., Liu, Q., Dong, Y., Han, J., Zhang, Z.: Fisher: an efficient container load prediction model with deep neural network in clouds. In: 2018 IEEE International Conference on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom), pp. 199–206. IEEE (2018)
25. Taylor, S.J., Letham, B.: Forecasting at scale. Am. Stat. **72**(1), 37–45 (2018)
26. Thinakaran, P., Gunasekaran, J.R., Sharma, B., Kandemir, M.T., Das, C.R.: Kube-knots: resource harvesting through dynamic container orchestration in GPU-based datacenters. In: 2019 IEEE International Conference on Cluster Computing (CLUSTER), pp. 1–13. IEEE (2019)
27. Yang, Z., Nguyen, P., Jin, H., Nahrstedt, K.: MIRAS: model-based reinforcement learning for microservice resource allocation over scientific workflows. In: 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), pp. 122–132. IEEE (2019)
28. Zhao, A., Huang, Q., Huang, Y., Zou, L., Chen, Z., Song, J.: Research on resource prediction model based on kubernetes container auto-scaling technology. In: IOP Conference Series: Materials Science and Engineering, vol. 569, p. 052092. IOP Publishing (2019)