Refined Balanced Resource Allocation Based on Kubernetes

Qizheng Huo [1], Chengyang Li[2], Shaonan Li[1], Yongqiang Xie [1, *] and Zhongbo Li [1,*]

[1] *Institution of Systems Engineering, Academy of Military Sciences, Beijing, 100141, China*
[2] *School of Electronics Engineering and Computer Science, Peking University, Beijing, 100871, China*

Abstract

Cloud video conferencing and cloud video command play an important role in various fields. The balanced utilization of the underlying resources of cloud video conferencing systems is an important research direction for cloud video. At present, most of the cloud video server cluster resources are native scheduling algorithms, which are simple and weak in scene specificity. The native algorithm can meet the basic needs, but cannot well refine and balance scheduling resources. The previous research mostly carried out scheduling by formulating rescheduling strategies. The configuration is complex, and the amount of engineering is large. In this paper, the Refined Balanced Resource Allocation algorithm is proposed by introducing the coefficient of variation. Compared with the default algorithm, it is easy to implement in engineering, and can better achieve balanced scheduling of resources, avoid resource fragmentation, and distribute pods on the cluster more dispersed. After simulation experiments, the algorithm proposed in this paper improves Spread Score, and the balance of large cluster resources has increased by 14.28%. The algorithm proposed in this paper is quite effective and feasible.

Keywords

Balanced scheduling, container cloud, Kubernetes, resource scheduling

1. Introduction

Cloud video services play an important role in communication in our contemporary life. Since the COVID-19 pandemic, people's offline travel and communication have been greatly restricted. Due to a large number of users, the resource consumption of the server cluster is relatively large. In certain scenarios, it needs to face the problems of a sudden increase in access services and excessive server pressure. The emergence of cloud computing and container[1] technology can solve such problems, so the research on cloud-based server resource scheduling becomes very necessary. It can achieve better QoS through reasonable resource scheduling under limited hardware resources.

Since 2013, with the rise of container technology [2, 3], many PaaS public cloud vendors such as Google and IBM have begun to use container technology. At present, many container cloud platforms provide application service running platforms through technologies such as Docker[4, 5] containers and Kubernetes [6]. Kubernetes is an open source project of Google, which comes from the internal project Borg [7, 8], and has a market share of 80%. Burns, Grant, and Oppenheimer[8] compared the container orchestration tools Borg, Omega and Kubernetes in detail. Although Kubernetes is superior to its predecessors, the configuration files are complex and require complete configuration semantics and corresponding debugging tools. This shows that the current cloud-native scheduling strategy is somewhat simple, the scenario is single, and the adaptability to specific scenarios is weak.

Based on the above research, the survey found that the current cloud video has three challenges. First, the default scheduling strategy is too simple to meet the scheduling requirements of specific scenarios. Second, the default algorithm avoids fragmentation of resources on nodes during balanced scheduling, but high CPU and memory loads on specific nodes will occur, resulting in unbalanced use of cluster resources. Third, the unbalanced distribution of pods increases the node failure rate.

ICBASE2022@3rd International Conference on Big Data & Artificial Intelligence & Software Engineering, October 21-23, 2022, Guangzhou, China
*Corresponding author e-mail: fitz2020@foxmail.com (Yongqiang Xie)

Our main contribution is three-fold.

1. This paper proposes a Refined Balanced Resource Allocation (RBRA)algorithm by introducing the coefficient of variation. While the resources in the nodes are used in a balanced manner, the pods on the cluster is more dispersed.
2. Compared with the default algorithm, the RBRA algorithm is easier to implement in engineering, and can better achieve balanced scheduling of resources, avoid resource fragmentation.
3. The refined balanced allocation of cluster resources decreases the node failure rate.

After simulation experiments, the algorithm proposed in this paper improves the spread score compared with the default algorithm, and the balance of large cluster resources has increased by 14.28%.

2. Related work

A Kubernetes cluster usually consists of a master and multiple nodes[9]. The master node is responsible for controlling the entire cluster. When a new demand Pod is submitted, Kubernetes will filter the nodes according to the default algorithm, score the nodes according to the default strategy, and select the optimal node. The default schedule is divided into two parts that can be customized:

Predicates: It filters out nodes that do not meet the conditions from all nodes in the current cluster according to the scheduling policy.

Priorities: Score these nodes; finally select the node with the highest priority and bind it to the Pod.

The default balanced resource allocation algorithm, which belongs to priorities, selects nodes with the most balanced resource usage. Usually, when the number of resources is two, the algorithm first calculates the CPU and memory usage on the node. The difference is calculated between CPU and memory usage. 10 minus 10 times the difference between CPU and memory usage. The final score is obtained.

$$R_{CPU} = \frac{\text{CPUuse}}{\text{CPUtotal}}, \tag{1}$$

$$R_{Mem} = \frac{\text{Memuse}}{\text{Memtotal}}, \tag{2}$$

$$\text{Score}_{\text{DefualtBalance}} = 10 - 10|\, R_{CPU} - R_{Mem}| \tag{3}$$

When the number of resources is three or more, the algorithm calculates the ratio of requests to allocable resources. Then mean of the ratio is calculated followed by calculating the standard deviation. After that, the Score represents 1 minus standard deviation then multiply by 100.

Ratio of CPU

$$R_{CPU} = \frac{\text{CPUrequest}}{\text{CPUallocable}}, \tag{4}$$

Ratio of Memory

$$R_{Mem} = \frac{\text{Memrequest}}{\text{Memallocable}}, \tag{5}$$

Ratio of Storage

$$R_{Storage} = \frac{\text{Storequest}}{\text{Stoallocable}}, \tag{6}$$

mean of ration

$$mean = R_{CPU} + R_{Mem} + R_{Storage}, \tag{7}$$

then the standard deviation

$$Std = [(R_{CPU} - mean)^2 + (R_{Mem} - mean)^2 + (R_{Storage} - mean)^2]^{\frac{1}{2}}/3, \tag{8}$$

the final Score

$$\text{Score}_{\text{DefualtBalance}} = (1 - Std) * 100 \tag{9}$$

Scholars have also made improvements to the default algorithm. The existing research mostly adopts the combined scheduling strategy to schedule resources. Du Jun[10] expanded the default algorithm library by designing a new preselection function and introducing a balance factor, but the balance of cluster resources has not yet been achieved. The preemptive scheduling scheme proposed by Song Lin [11] optimizes the original grading basis. By setting a new sub-priority to schedule Pods, a more prioritized scheduling strategy is realized, but the tasks of preemptive eviction cannot be saved. Xu [12]et al. proposed a priority scheduling strategy based on network load, which uses network load status as an indicator to prioritize virtual machines for final scheduling, and its environment does not match cloud computing.

Ghofrane [13]et al. proposed the KubCG algorithm to study CPU and GPU resource scheduling. KubCG reduces the time of cluster resource scheduling by 36%, and the resource utilization is more balanced. Xu[14] et al. proposed a Swift load balancing method that uses CPU, memory and I/O utilization as resource dynamic scheduling on OpenStack-based platforms, but it can only be used for OpenStack cloud platforms. Awad [15] proposed a mathematical model using load balancing mutation (balancing), a cloud computing scheduling and allocation based on particle swarm optimization (LBMPSO), which greatly reduces the time for resource scheduling. However, these studies focus on changing the weight and priority to achieve a better scheduling strategy but ignore the balance of resources within the cluster. These scheduling strategies generally involve multiple components[16, 17], rather than simply designing a scheduling algorithm, so there is still no specific design scheme that can solve the imbalance of cluster resources. The default algorithm avoids fragmentation of resources on nodes during balanced scheduling, but high CPU and memory loads on specific nodes still occur, resulting in unbalanced use of cluster resources.

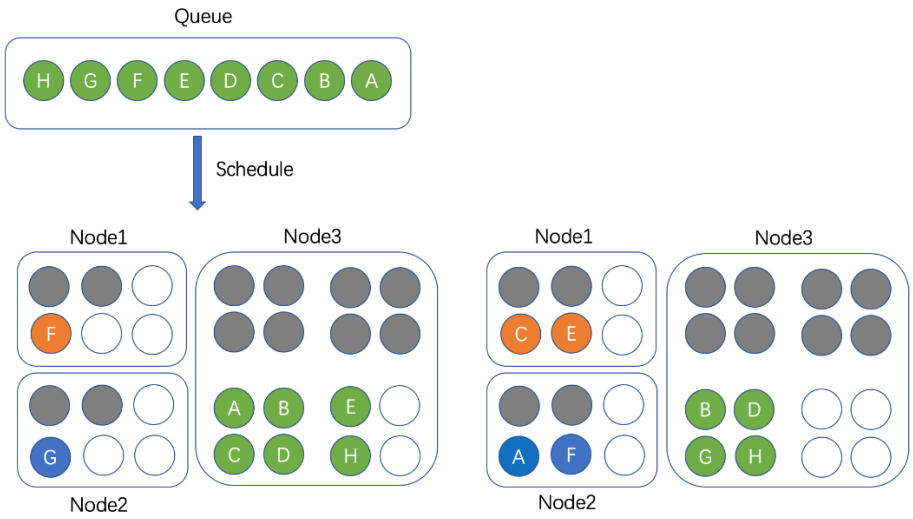

Figure 1: Result of Default algorithm and one of the possible desired results. When a list of pod requirements is submitted, according to the default algorithm, pods will be preferentially scheduled to the nodes with the largest remaining node resources, as shown in (a); (b) represents the scheduling result after introducing the coefficient of variation to eliminate the influence of the order of magnitude of nodes in the cluster.

3. Algorithm

Based on the possible business requirements on the server, a Refined Balanced Resource Allocation (RBRA)algorithm is proposed. After deconstructing the resource scheduling algorithm[18], it is possible to qualitatively analyze the demanded resources, maximize the utilization of resources, allocate node resources in a balanced manner, and avoid continuously assigning pods to nodes with a large number of remaining resources. Decentralized configuration of underlying resources reduces the impact of single points of failure.

3.1. Refined Balanced Resource Allocation Algorithm

To eliminate the influence of the level of data values and the different measurement units on the measurement value of dispersion degree, it is necessary to calculate the coefficient of variation[19]. Therefore in the process of balanced scheduling, this algorithm introduces the coefficient of variation to eliminate the influence of the order of magnitude of node resources.

3.2. Definition of Coefficient of Variation

For resource utilization data, set as $x_1, x_2, \ldots, x_n$, n represents the number of resources, x_n represents the ratio of the nth resource, and $\bar{x}$ represents the average number of $x_1, x_2, \ldots, x_n$. Then we calculate the degree of dispersion of the set of data.

Then the variance is

$$S^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}, \tag{10}$$

The standard deviation is

$$S = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}}, \tag{11}$$

where

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}, \tag{12}$$

V stands for the coefficient of variation

$$V = \frac{\sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}}}{\frac{\sum_{i=1}^{n} x_i}{n}} = \frac{S}{\bar{x}}. \tag{13}$$

Considering the influence of the order of magnitude of resources in the cluster in scoring, the coefficient of variation is introduced into the scheduling algorithm to form a multi-index scheduling algorithm such as CPU, memory and storage.

The algorithm flow is as follows:

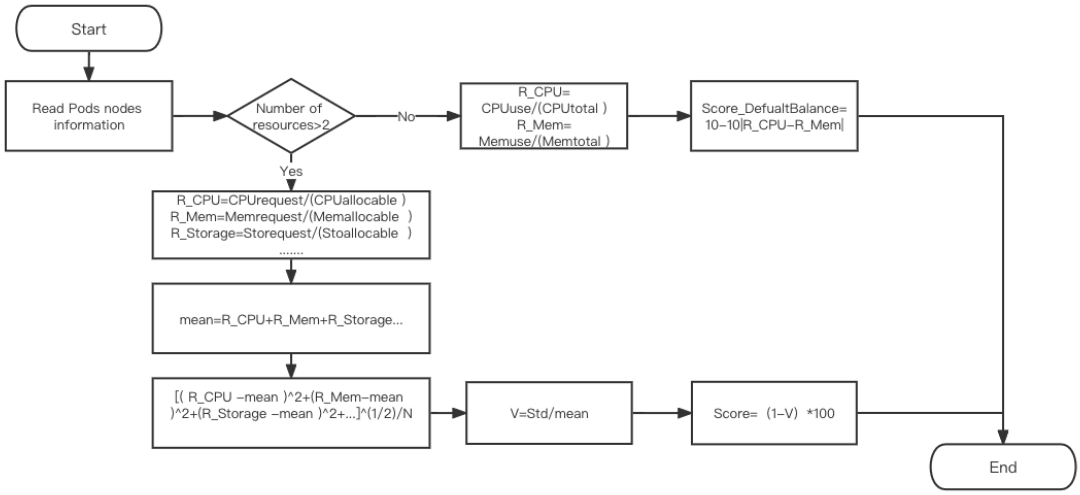

Figure 2: The algorithm flow.

1) Read Pods nodes information
2) The number of resources ≤2, $R_{CPU} = \frac{\text{CPUuse}}{\text{CPUtotal}}$, $R_{Mem} = \frac{\text{Memuse}}{\text{Memtotal}}$. The calculation of the Score is the same as the default algorithm. $\text{Score} = 10 - 10|\, R_{CPU} - R_{Mem}|$.

3) The number of resources >2. According to the requirements of N resources of a pod, such as N=3, the CPU usage request, memory usage request, and storage usage request are counted as, CPUrequest, Memrequest and Storequest respectively.
4) According to the submitted service request, obtain the resource usage of candidate nodes, mark N resources of each node, and obtain the available resources of the node, for example, N=3, CPU, Memory, and Storage are counted as CPUallocable, Memallocable and Stoallocable respectively.
5) Obtain the mean value of each resource's ratio - the ratio of each resource, for example, CPU, Memory, and Storage are recorded as "cpuFraction", "memoryFraction", and "storageFraction" respectively.
6) Find the average mean after adding the ratios of each resource in 5).
7) Use the standard deviation to measure the dispersion of the ratio of resources, and find the standard deviation of the ratio of resources Std.

$$Std = [(\ cpuFraction - mean\ )^2 + (\ memoryFraction\ -\ mean\ )^2 + (\ storageFraction\ -\ mean\ )^2]^{1/2}/3$$

8) Calculate the coefficient of variation V.

$$V = \frac{Std}{mean}$$

9) The score is

$$Score = (1 - V) * 100$$

Select the node with the highest score, when there are multiple nodes with the same highest score, randomly select the node as the deployment node.

4. Experiments

This section introduces the experimental results of the proposed RBRA algorithm based on simulation. First, the basic configuration of the experimental environment is introduced, and then the parameter settings of the experiment are described. In Section 4.1, we introduce the parameter configuration and scheduling results of the RBRA algorithm when the node resource difference is an order of magnitude 10. In Section 4.2, we show the scores of the RBRA algorithm under different cluster sizes and the imbalance of the cluster.

For a cluster, the cluster standard deviation reflects the balance of cluster resources. It can be obtained by calculating the number of Pods on the cluster which is the same as the way used by Lin [20]. Standard deviation is also widely used in other fields like analysis of radar data[21]. Some scholars have also made improvements to the calculation dispersion degree according to the actual situation[22].

Define the cluster resource imbalance as $Cluster_{std}$, J represents the number of nodes, $P(j)$means the number of pods on node$_j$ $(j \in J)$.

$$Cluster_{std} = \frac{\sqrt{\sum_{j \in J}(P(j) - P_{AVE})^2}}{J}, \tag{14}$$

Then the average of each node is P_{AVE}.

$$P_{AVE} = \frac{\sum_{j \in J} P(j)}{J}. \qquad R_j = \frac{\sum P(j)}{N}. \tag{15}$$

The smaller the $Cluster_{std}$ is, the more balanced the distribution of Pods in the cluster, and the more dispersed the resource distribution of the cluster is.

4.1. Parameter Configuration

Through the Kube-scheduler-simulator and local Kubernetes cluster environment, the algorithm proposed in this paper is well verified. The experimental machine is Lenovo Blade 7000, processor

Intel Core I7-9700K, memory 32G, hard disk capacity 1T, operating system Windows10, CentOS7.9, software VmWare.

When nodes with orders of magnitude difference appear in the cluster, they can be randomly scheduled to achieve balanced scheduling of resources by RBRA. First, we set up a master with two nodes. The allocable resource of the Node0 is CPU 4000m (m is one-thousandth of a core), Memory 32GB, Storage 400GB, and the allocable resource of the Node1is CPU 40000m, Memory 320GB, Storage 4000 GB. Then the resource requested by Pod is CPU 300 m Memory 1GB Storage 10GB. The default algorithm continues to deploy Pods to Node1, and our algorithm can implement scheduling to Node0 or Node1 after introducing the coefficient of variation, which can better disperse Pods and make cluster resources more balanced.

We set clusters with different numbers of nodes and conduct scheduling experiments. The results are displayed in 4.2

4.2. Multi-node scheduling experiment

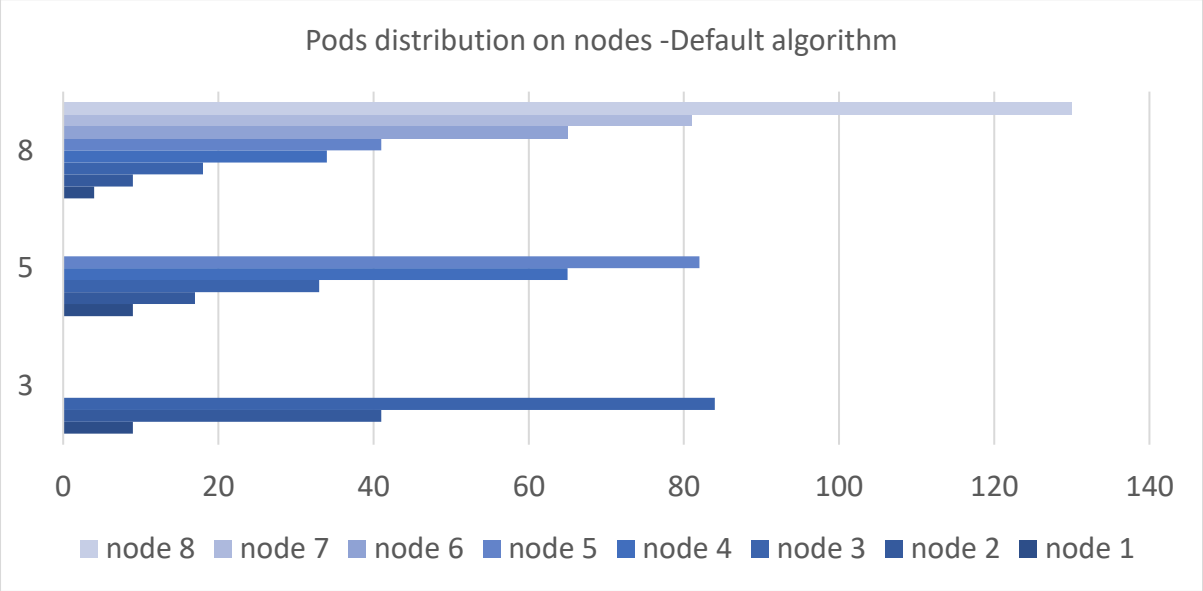

Figure 3: Pods distribution on nodes with default algorithm.

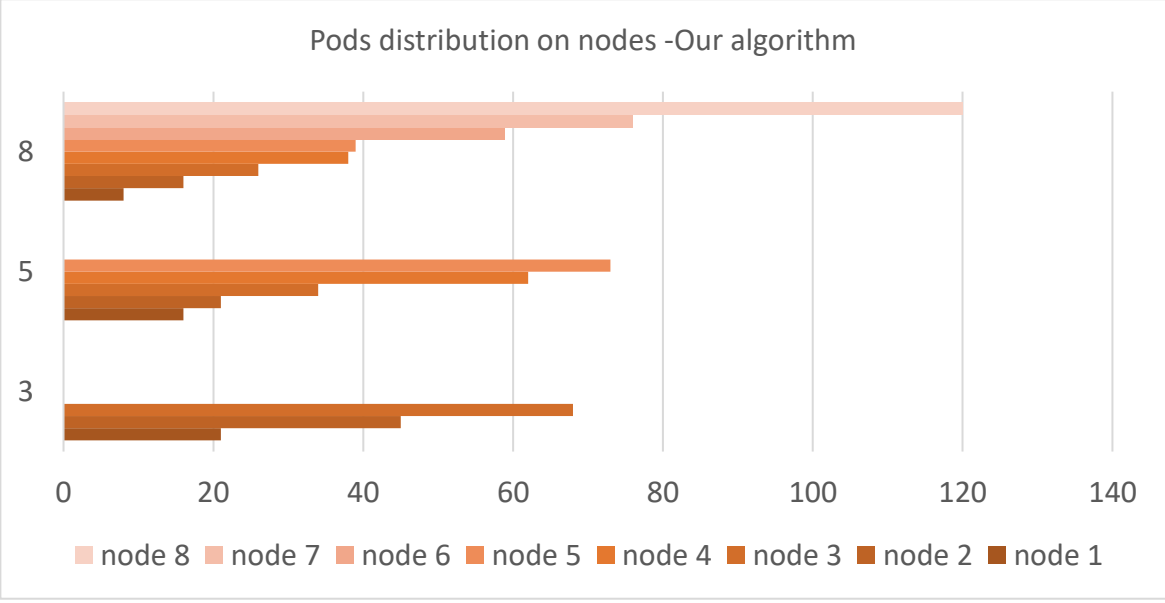

Figure 4: Pods distribution on nodes with our algorithm.

As shown in Figure 3 and Figure 4, the range of the default algorithm scheduling is larger, and its degree of dispersion is larger than that ours. Our algorithm makes the distribution of Pods on the cluster more balanced. We will next set and calculate the score of the cluster.

Experiments, statistics and analysis of the above two algorithms are carried out respectively. Spread Score=P_1* P_2...*P_n n represents the number of nodes, and P_n represents the number of Pods on the nth node. Table 1 shows the Spread Score.

Table 1
Spread Score

Nodes	Pods	*Spread Score*	
		Default Scheduler	*Our Algorithm*
3	134	30996	64260
5	206	26911170	51705024
8	382	618271898400	2653862215680

Compared with the default algorithm, our algorithm avoids fragmentation of cluster resources, and the Spread Score is higher according to table 1.

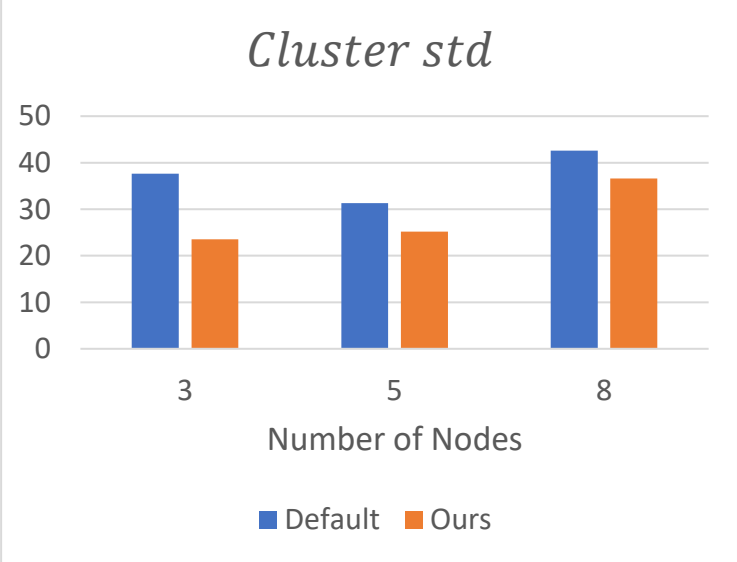

Figure 5: *Cluster std*

Figure 5 represents the imbalance of clusters by $Cluster_{std}$. Moreover, the pod distribution on the cluster is more balanced. When the number of nodes is three, the amount of data is small and not representative. As shown in Figure 5, the balance of large cluster resources has increased by 14.28% when the number of nodes is 8.

5. Conclusion

In this paper, by introducing the coefficient of variation, we propose the RBRA algorithm. When the order of magnitude difference of cluster resources is large, the resources can still be allocated to a node in a balanced manner, while ensuring that the resource utilization within the nodes is also balanced. It solves the problem of single policy mismatch caused by the default algorithm, resulting in a high load on the CPU, memory, etc. on the node, and uneven use of cluster resources, reducing the node failure rate. We conduct simulation experiments based on the local environment. The RBRA algorithm can better achieve balanced scheduling for clusters with large differences in the order of magnitude of resources, avoiding the devastating impact of a single node failure. According to the experimental data, it has a better application effect on the balanced scheduling of pods of different cluster sizes.

6. References

[1] Boettiger C. An introduction to Docker for reproducible research, with examples from the R environment [J]. ACM SIGOPS Operating Systems Review, 2014, 49(1): 71-9.

[2] Bozzon A, Cudré-Mauroux P, Pautasso C. [Lecture Notes in Computer Science] Web Engineering Volume 9671 || Using Docker Containers to Improve Reproducibility in Software and Web Engineering Research [J]. 2016, 10.1007/978-3-319-38791-8(Chapter 58): 609-12.

[3] Kek P Q. Docker : build, ship, and run any app, anywhere [J]. 2017.

[4] Merkel D. Docker [J]. Linux Journal, 2014.

[5] Chung M T, Quang-Hung N, Nguyen M T, et al. Using Docker in High Performance Computing Applications; proceedings of the 2016 IEEE Sixth International Conference on Communications and Electronics, F, 2016 [C].

[6] Bernstein, David. Containers and Cloud: From LXC to Docker to Kubernetes [J]. IEEE cloud computing, 2014.

[7] Verma A, Pedrosa L, Korupolu M, et al. Large-scale cluster management at Google with Borg; proceedings of the Tenth European Conference on Computer Systems, F, 2015 [C].

[8] Wilkes, John, Brewer, et al. Borg, Omega, and Kubernetes [J]. Communications of the Acm, 2016.

[9] Brewer E A. Kubernetes and the path to cloud native; proceedings of the the Sixth ACM Symposium, F, 2015 [C].

[10] Du J. Improvement of cloud resource scheduler based on Kubernetes [D]; Zhejiang University, 2016.

[11] Song L. Design and implementation of resource scheduling and monitoring system based on Kubernetes [D]; Beijing University of Posts and Telecommunications, 2019.

[12] Xu Z, Xiao L, Zhan W, et al. An VM Scheduling Strategy Based on Hierarchy and Load for OpenStack; proceedings of the 2016 7th International Conference on Cloud Computing and Big Data (CCBD), F, 2016 [C].

[13] Ahmed G, Gil-Castieira F, Costa-Montenegro E. KubCG: A dynamic Kubernetes scheduler for heterogeneous clusters [J]. Software: Practice and Experience.

[14] Min X, Ming L, Jianzhong Z, et al. Swift Load Balancing Algorithm Based on OpenStack [J]. Computer System Applications, 2018, 027(001): 127-31.

[15] Awad A I, El-Hefnawy N A, Abdel_Kader H M. Enhanced Particle Swarm Optimization for Task Scheduling in Cloud Computing Environments [J]. Procedia Computer Science, 2015, 65: 920-9.

[16] Laboratory Z U S. Docker containers and container clouds [M]. Docker containers and container clouds, 2015.

[17] Ogbuachi M C, Gore C, Reale A, et al. Context-Aware K8S Scheduler for Real Time Distributed 5G Edge Computing Applications; proceedings of the 2019 International Conference on Software, Telecommunications and Computer Networks (SoftCOM), F, 2019 [C].

[18] Zheng G, Fu Y, Wu T. Research on Docker Cluster Scheduling Based on Self-define Kubernetes Scheduler [J]. Journal of Physics: Conference Series, 2021, 1848(1): 012008-.

[19] Ahn C, Fan H, Skinner C S. Effect of imbalance and intracluster correlation coefficient in cluster randomized trials with binary outcomes [J]. Computational Statistics & Data Analysis, 2009, 53(3): 596-602.

[20] Zhu L, Junjiang L I, Liu Z, et al. A multi-resource scheduling scheme of Kubernetes for IIoT [J]. Systems Engineering and Electronic Technology, 2022, 33(3): 10.

[21] Siteng L I, Meilin Y, Lin L I, et al. Evaluation on Data Quality of X-band Dual Polarization Weather Radar Based on Standard Deviation Analysis [J]. Journal of Arid Meteorology, 2019.

[22] Shi J, Luo D, Hong W, et al. How to estimate the sample mean and standard deviation from the five number summary? [J]. 2018.