PAPER • **OPEN ACCESS**

Hybrid Elastic Scaling Strategy for Container Cloud based on Load Prediction and Reinforcement Learning

To cite this article: Peng Liu *et al* 2024 *J. Phys.: Conf. Ser.* **2732** 012014

Hybrid Elastic Scaling Strategy for Container Cloud based on Load Prediction and Reinforcement Learning

Peng Liu, Weisen Zhao*, Baoliang Zhang and Jing Wang

China Electric Power Research Institute, Beijing 100192, China

***Email:** weisen_2022@qq.com

Abstract. To harness the advantages of both proactive and responsive scaling, adapting to various workload scenarios, this paper introduces a container hybrid scaling strategy called HyPredRL, rooted in load prediction and reinforcement learning. Within the proactive scaling module RL-PM, a load prediction model, MSC-LSTM, predict workloads and, in conjunction with current workload states, leverages reinforcement learning agents for intelligent scaling decisions. The responsive scaling strategy, SLA-HPA, enhances Kubernetes' native scaling strategy, which primarily considers resource utilization, by incorporating response time metrics. Ultimately, a hybrid scaling controller is designed, applying the principles of "rapid scaling out" and "balanced conflicts" to coordinate proactive and responsive scaling. Experimental results demonstrate that HyPredRL outperforms existing methods in SLA violation rate, resource utilization, and request response time, effectively improving application performance and scalability.

Keywords. Container cloud; Elastic scaling; Kubernetes; Load prediction; Reinforcement learning.

1. Introduction

In recent years, cloud computing has attracted extensive attention and research in industry and academia because of its flexible resource management, high scalability and economy[1]. As one of the key features of cloud computing[2], elastic scaling strategy can dynamically adjust the resource quota of the application according to the actual load situation, which plays an important role in ensuring the high availability and stability of the application [3].

According to the scaling pattern, elastic scaling strategies can be divided into reactive scaling and proactive scaling. Among them, reactive scaling regularly detects the load state of the application through the internal scaling controller, and dynamically adjusts the resource allocation according to the pre-set threshold rules of resource indicators[4], which can accurately reflect the current service state and is widely used in cloud platforms and container orchestration platforms. At present, the research related to reactive scaling mainly focuses on threshing-based methods, which mainly include static thresholds[5], adaptive thresholds [6] and hybrid thresholds[7]. However, there are some problems in threshold-based scaling strategies, such as difficult threshold setting and delayed elastic response, which affect the quality of service.

Compared with the reactive scaling strategy, the proactive scaling strategy uses the historical load data to establish a load prediction model to predict the future required resources, so as to adjust the resource supply in advance, which can effectively avoid violating the Service Level Agreement (SLA) and improve the resource utilization rate. Among them, the load prediction model is the basis of the proactive scaling strategy. The current research directions include traditional statistical methods[8], machine learning methods[9], deep

learning methods[10] and hybrid models[11][12]. However, in extreme bursty traffic scenarios, relying solely on prediction or modeling results can easily lead to the loss of effectiveness of the policy, and there is a certain risk of resilience failure.

Therefore, in order to balance the advantages of proactive scaling decisions and extreme traffic conditions to achieve more accurate and timely resource adjustment, This paper proposes HyPredRL (Hybrid Elastic Scaling Strategy based on Load Prediction and Reinforcement learning[13] Learning). In this strategy, a hybrid scaling controller is designed to coordinate the proactive and reactive scaling strategies, and the final scaling decision is made based on the idea of "fast scaling slow scaling" and "balancing conflicts".

2. Hybrid Elastic Scaling Strategy

Aiming at minimizing SLA violation rate and maximizing resource utilization, this paper proposes a hybrid scaling strategy based on load prediction and reinforcement learning. Figure 1 shows the overall framework of the strategy, which includes a load prediction module, an active scaling module, a responsive scaling module, and a hybrid scaling controller.

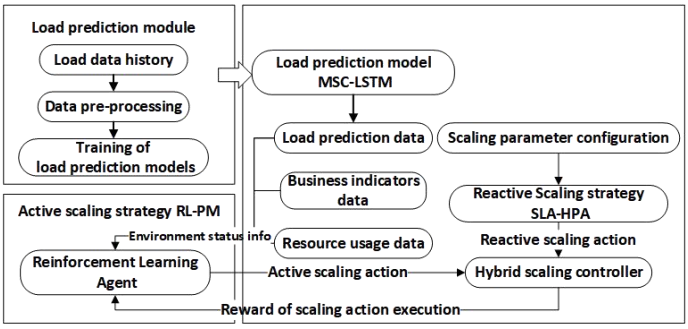

Figure 1. Frame diagram of hybrid elastic scaling strategy

2.1. Active Scaling Strategy RL-PM based on Load Prediction and Reinforcement Learning

2.1.1. Load Prediction Model MSC-LSTM. In order to accurately capture short-term fluctuations and continuous and periodic load changes on multiple scales, the accuracy of load prediction is improved. In this paper, a Spatio-Temporal Load Prediction Model using Multi-Scale Convolutions and LSTM (MSC-LSTM) is designed. Figure 2 illustrates the structural diagram of the MSC-LSTM model. The model uses parallel and stacked atrous convolutional residual networks to extract multi-scale features, including time series features and periodic features, and then models long-term time dependence through LSTM, and finally outputs the load prediction sequence through a fully connected neural network. It is worth noting that the model designs two channels to deal with different inputs, one learns the basic load characteristics, and the other learns the periodic characteristics to better learn the variation law of the load.

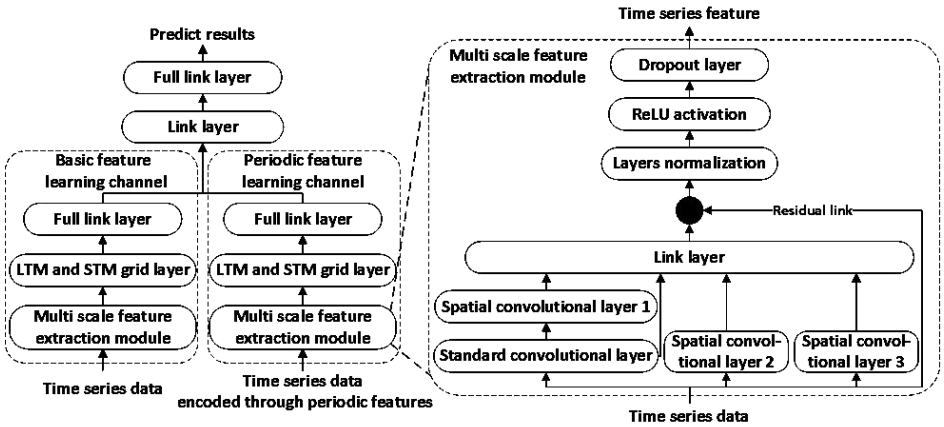

Figure 2. Structure diagram of MSC-LSTM model

In order to evaluate the prediction effect of MD-LSTM, this paper designed relevant experiments, and the experimental results are shown in table 1. The results show that the proposed model is superior to other

comparison models in MSE, RMSE and MAE. Compared with the existing best model CNN-LSTM, the overall MSE index achieved 8.97% improvement on the Google Cluster dataset, 12.06% improvement on the NASA dataset, and 8.91% improvement on the WorldCup dataset.

Table 1. Comparison of CPU usage load results (unit:10^{-2})

Model	Forecast Step-size	Google Cluster dataset			NASA dataset			WorldCup dataset		
		MSE	RMSE	MAE	MSE	RMSE	MAE	MSE	RMSE	MAE
LSTM	12	0.453	0.673	5.139	0.291	0.539	3.299	0.439	0.663	4.804
LSTM-ED	12	0.441	0.664	5.170	0.282	0.531	3.303	0.408	0.639	4.477
TCN	12	0.426	0.653	4.990	0.266	0.516	3.112	0.408	0.638	4.503
Transformer	12	0.389	0.624	4.711	0.258	0.508	3.018	0.401	0.633	4.651
pCNN-LSTM	12	0.368	0.606	4.187	0.257	0.506	2.920	0.393	0.627	4.405
MTD-LSTM	12	**0.335**	**0.579**	**3.751**	**0.226**	**0.475**	**2.682**	**0.358**	**0.598**	**4.076**

2.1.2. Active scaling policy RL-PM. Although the scaling strategy based on load prediction can allocate resources in advance to adapt to future load changes, it usually uses rules to determine the number of instances as a supplement to the threshold control method, which is difficult to adapt to the needs of complex scenarios and limits the flexibility and accuracy of the scaling strategy. Therefore, this paper designs a Proactive Mode based on Reinforcement learning (RL-PM). Figure 3 shows the framework of RL-PM. In this strategy, the reinforcement learning agent makes reasonable elastic scaling decisions, which can adjust the number of Pod service instances more accurately and intelligently.

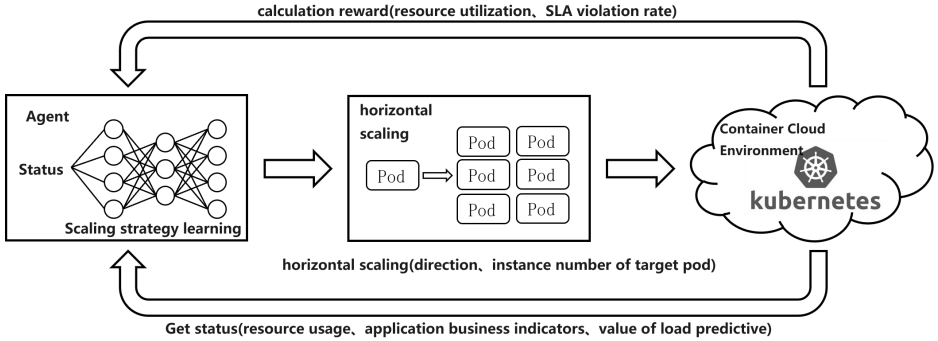

Figure 3. RL-PM frame diagram

First, the agent gets the state information from the container cloud environment. Then the agent chooses the elastic scaling action through the policy and acts it on the container cloud environment. After that, as shown in equations (1) and (2), the reward function calculates the reward value based on resource utilization $Util$ and service quality $Perf$ and feeds it back to the agent. Finally, through interactive learning with the environment, the agent continuously ADAPTS to the changes of the environment state, the expansion action and the reward value to find the optimal expansion strategy.

$$penalty_{RT} = \begin{cases} w_{RT} * \dfrac{RT_{obs}}{RT_{sla}}, RT_{obs} > RT_{sla} \\ 0, otherwise \end{cases} \tag{1}$$

$$r_t = Util * Perf = \frac{Resource_{consumed}}{Resource_{deployed}} * \left(\frac{RT_{sla}}{RT_{obs}} - penalty_{RT}\right) \tag{2}$$

Where, $Resource_{consumed}$ and $Resource_{deployed}$ represent the number of resources used and allocated respectively, RT_{obs} is the maximum request response time delay stipulated by SLA, $penalty_{RT}$ is the average response time delay of all instances in the current cycle, represents the penalty for violating SLA when the average response time delay is higher than the maximum request response time delay, w_{RT} and is an adjustable penalty factor.

2.2. Reactive Scaling policy SLA-HPA

Kubernetes' native Horizon Pod Autoscaling (HPA) strategy is a reactive scaling strategy based on resource utilization without taking into account application SLAs. Therefore, combined with SLA indicators, this paper designs a reactive scaling strategy SLA-HPA (Horizon Pod Autoscaling based on Service Level Agreement). This policy will set different SLA maximum response delay according to the application characteristics, so that each application can meet its SLA criteria. Firstly, the monitoring module will obtain the request response time of the current application. If it exceeds the maximum response delay set by the user, SLA-HPA will calculate the number of Pod instances of the scaled application according to the request load. In addition, the policy will scale up or down based on the current resource utilization to stay within the specified resource utilization upper and lower bounds.

2.3. Hybrid scaling controller HyPredRL

Active scaling and reactive scaling have some limitations due to the characteristics of their respective strategies. It is necessary to combine the advantages of the two strategies to enhance the flexibility of automatic scaling. However, the decision logic of these two strategies to obtain elastic scaling actions is inconsistent, which may produce different elastic scaling actions and lead to decision conflicts. In order to coordinate the two strategies and give full play to their different advantages, this paper proposes a proactive and reactive hybrid elastic scaling strategy HyPredRL, which uses the idea of "fast scaling slow scaling" and "balancing conflicts" to jointly control the execution of the proactive and reactive scaling strategies.

Referring to Kubernetes native HPA policies, HyPredRL periodically schedules reactive and proactive scaling policies to make decisions. The reactive scaling module monitors business metrics and resource usage data, and makes scaling action decisions according to preset rules to cope with extreme traffic peaks in time. According to the predicted load data and other environmental state information, the reinforcement learning agent is used to make reasonable active scaling action decisions. The final scaling action information includes the type of policy (reactive or proactive), the number of application Pod instances to scale or shrink, and so on.

In addition, in order to improve the consistency and accuracy of the configuration, this paper designs a hybrid scaling control rule based on the ideas of "fast scaling and slow scaling" and "balancing conflicts", and applies it to the hybrid scaling controller, the specific rules are shown in table 2. According to the rules, when the scaling direction of the two strategies is the same or one of the strategies does not scale, the "fast scaling and slow scaling" strategy is adopted. When the expansion direction of reactive scaling and proactive scaling is opposite, the idea of "balance conflict" is used to solve the conflict, that is, the final action and instance number are determined by judging the positive or negative difference between the expansion action and the contraction action.

Table 2. Hybrid scaling control rules

Active / Reactive	dilatation	Volume of contraction（VOC）	Not scaling
dilatation	dilatation$Max(N_P, N_R)$	$\begin{cases}\text{dilatation}, N_R - N_{\mathrm{P}} > 0 \\ VOC, N_R - N_{\mathrm{P}} < 0\end{cases}$ number：$\lvert N_R - N_{\mathrm{P}}\rvert$	dilatation N_{R}
Volume of contraction （VOC）	$\begin{cases}\text{dilatation}, N_{\mathrm{P}} - N_R > 0 \\ \text{VOC}, N_{\mathrm{P}} - N_R < 0\end{cases}$ number：$\lvert N_{\mathrm{P}} - N_R\rvert$	VOC $Min(N_P, N_R)$	Not scaling
Not scaling	dilatationN_{P}	Not scaling	Not scaling

3. Comparative Experiment and Analysis of Elastic Expansion

3.1. Experimental Environment

In order to verify the scalability performance of the hybrid elastic scaling strategy under different load scenarios, this paper designs a comparison test. The experiments were deployed on a physical machine configured with an Intel Core i7-9700 and 32GB memory. Among them, the software version of Kubernetes is

1.22, and the cluster includes a master node and two worker nodes, each of which is a CentOS7 virtual machine with 4 cores and 8GB memory. Each application Pod instance has 0.5 core CPU and 512MB memory resources, and the CPU utilization threshold is 60%. The application in the experiment uses SpringBoot to develop the test application and exposes metrics through Prometheus Exporter.

3.2. Control Methods

In the experiment, the following three container elastic scaling strategies are selected to compare with the hybrid scaling strategy of this paper.

(1) Native HPA (Horizon Pod Autoscaling)[14]: Kubernetes' native horizontal elastic scaling strategy, which is a reactive scaling strategy that dynamically increases or decreases the number of Pod instances of the application based on a threshold to adjust resource provisioning.

(2) Bi-LSTM-HPA[15]: a scaling strategy based on load prediction proposed by Dang-Quang et al. This strategy uses the Bi-LSTM model to predict the future request load, and adjusts the number of instances according to the predicted value to cope with changes.

(3) DScaler[16]: a scaling strategy based on deep reinforcement learning proposed by Xiao et al., which adaptively learns and adjusts the scaling strategy through the learning of deep Q network to adapt to different load conditions and environmental changes.

3.3. Experimental Results and Analysis

In this experiment, steady fluctuation and abnormal burst request loads were selected from the test set of World Cup load data set for elastic scaling experiments, and SLA violation rate and average CPU utilization were used as evaluation indicators. The results are shown in figure 4.

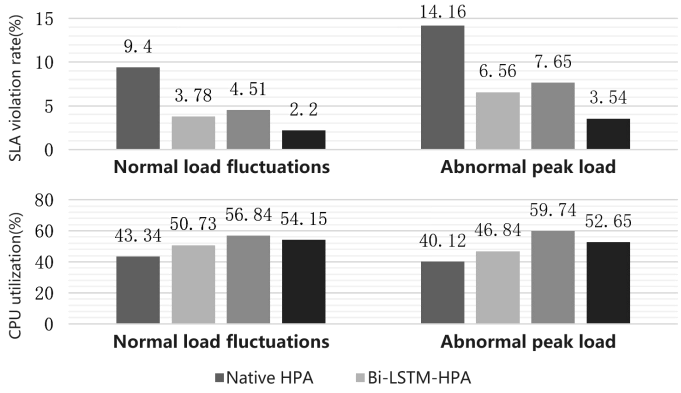

Figure 4. Results of SLA violation rate and average CPU utilization

It can be seen from the figure that HyPredRL has lower SLA violation rate and higher CPU utilization, which is better than the other comparison methods as a whole. In the face of abnormal burst load, the reactive scaling strategy in this paper determines the required number of Pod instances based on the current peak load. At the same time, HyPredRL selects the scaling action with larger expansion among the two strategies, which makes up for the defects of inaccurate load prediction and the small expansion of action space of the reinforcement learning based scaling method. Compared with Bi-LSTM-HPA, Dscaler and the proactive scaling strategy of this paper, the SLA violation rate of HyPredRL is reduced by 3.02%, 4.11% and 1.26%, respectively. In addition, the hybrid scaling rule adopts the idea of "fast scaling and slow scaling", so more Pod instances will be allocated, which leads to a decrease in CPU utilization.

At the same time, the experiment also compares the request response time of different scaling strategies under smooth fluctuation and abnormal burst load, and the specific results are shown in figure 5. Among them, the horizontal line represents the upper limit of the response time threshold stipulated by the SLA, and each point represents the average request response time in a unit time period. It can be seen from the figure that, compared with other comparison models, HyPredRL has fewer SLA violation periods, a deeper distribution of response time, and a lower overall average response time, indicating that the scaling strategy proposed in this paper can not only reduce the SLA violation rate, but also improve the performance of application processing requests.

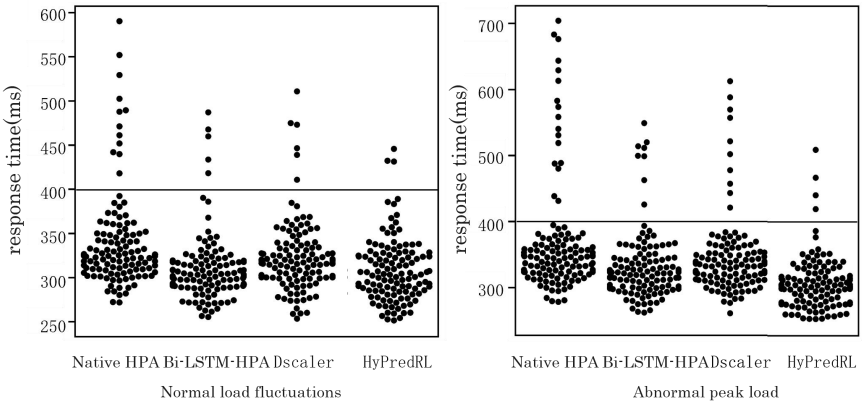

Figure 5. Comparison of response times

4. Conclusion

In this paper, we propose a hybrid scaling strategy called HyPredRL, designed to cope with complex workload scenarios. The strategy fully integrates the advantages of proactive scaling and reactive scaling, and adopts the strategy of "fast scaling slow scaling" and "balance conflict" to control the execution of proactive and reactive scaling strategies accurately and reasonably. The experimental results show that HyPredRL has advantages in SLA violation rate, resource utilization and request response time, improves the processing performance of applications, and can better adapt to different load conditions and service requirements to achieve efficient operation and high-quality service of container cloud platform.

References

[1] Luo J Z, Jin J H, Song A B, et al. Cloud Computing: Architecture and Key Technologies[J]. *Journal of Communications*, 2011, **32**(07): 3-21.
[2] Badger M L, Grance T, Patt-Corner R, et al. Cloud computing synopsis and recommendations[M]. National Institute of Standards & Technology. 2012. 30-32.
[3] Al-Dhuraibi Y, Paraiso F, Djarallah N, et al. Elasticity in cloud computing: state of the art and research challenges[J]. *IEEE Transactions on Services Computing*, 2017, **11**(2): 430-447.
[4] Kumbhare A G, Simmhan Y, Frincu M, et al. Reactive resource provisioning heuristics for dynamic dataflows on cloud infrastructure[J]. *IEEE Transactions on Cloud Computing*, 2015, **3**(2): 105-118.
[5] Hasan M Z, Magana E, Clemm A, et al. Integrated and autonomic cloud resource scaling[C]. 2012 IEEE Network Operations and Management Symposium. 2012. 1327-1334.
[6] Vaquero L M, Morán D, Galán F, et al. Towards runtime reconfiguration of application control policies in the cloud[J]. *Journal of Network and Systems Management*, 2012, **20**: 489-512.
[7] Botrán T L, Alonso J M, Lozano J A. Comparison of auto-scaling techniques for cloud environments[C]. Actas de las XXIV Jornadas de Paralelismo. Limencop. 2013. 187-192.
[8] Tang X, Liao X, Zheng J, et al. Energy efficient job scheduling with workload prediction on cloud data center[J]. *Cluster Computing*, 2018, **21**: 1581-1593.
[9] Liu C, Liu C, Shang Y, et al. An adaptive prediction approach based on workload pattern discrimination in the cloud[J]. *Journal of Network and Computer Applications*, 2017, **80**: 35-44.
[10] Tang X, Liu Q, Dong Y, et al. Fisher: An efficient container load prediction model with deep neural network in clouds[C]. 2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications (ISPA). 2018. 199-206.
[11] Patel E, Kushwaha D S. A hybrid CNN-LSTM model for predicting server load in cloud computing[J]. *The Journal of Supercomputing*, 2022, **78**(8): 1-30.
[12] Xu M, Song C, Wu H, et al. esDNN: deep neural network based multivariate workload prediction in cloud computing environments[J]. *ACM Transactions on Internet Technology (TOIT)*, 2022, 22(3): 1-24.
[13] Botvinick M, Ritter S, Wang J X, et al. Reinforcement learning, fast and slow[J]. *Trends in Cognitive Sciences*, 2019, **23**(5): 408-422.

[14] Rossi F, Cardellini V, Presti F L. Hierarchical scaling of microservices in kubernetes[C]. 2020 IEEE international conference on autonomic computing and self-organizing systems (ACSOS). 2020. 28-37.
[15] KubernetesDang-Quang N M, Yoo M. Deep learning-based autoscaling using bidirectional long short-term memory for Kubernetes[J]. *Applied Sciences*, 2021, **11**(9): 3835.
[16] Xiao Z, Hu S. DScaler: A Horizontal Autoscaler of Microservice Based on Deep Reinforcement Learning[C]. 2022 23rd Asia-Pacific Network Operations and Management Symposium (APNOMS). 2022. 1-6.