Scholar Publishing Group
International Journal of Big Data Intelligent Technology
https://doi.org/10.38007/IJBDIT.2026.070120
ISSN 2790-0932 Vol. 7, Issue 1: 170-178

Elastic Scaling and Stability Assurance Mechanisms for Distributed Systems under High-Throughput Scenarios

Jiahe Chen

Department of Computer Science, Columbia University, New York City 10027, United States

Keywords: High-throughput scenario; distributed system; elastic scaling; stability assurance; PAHP A multi-dimensional collaborative mechanism

Abstract: The core challenges of high-throughput distributed systems include scaling delay, weak prediction adaptability, and accuracy quality imbalance. This study proposes a dual mechanism collaborative approach of single dimensional prediction (SLMD LightGBM) and multi-dimensional collaboration (PAHP A): single dimensional prediction achieves accuracy improvements of 24.51% and 11.19% in time series prediction tasks by integrating local mean decomposition, LightGBM, and self updating mechanisms, effectively compensating for response delays; The multidimensional collaboration mechanism integrates predicted values with real-time indicators such as load, service quality, and queuing theory models, and avoids resource mismatches through dual allocation verification to ensure decision-making rationality. Experimental verification shows that the optimized system's violation rate (VR) has decreased to 16.3%, which is 8.5% lower than traditional HPA; The 99th percentile delay is strictly controlled within 880 milliseconds; Reduce resource expenditure by 20%, total cost by 25%, and meet performance threshold under high load. Future optimization will focus on three dimensions: refining the smooth adjustment strategy for Pod contraction stage, and suppressing performance fluctuations through dynamic thresholding and gradient algorithms; Validate the effectiveness of load balancing in complex microservice architectures and explore Pareto optimality for multi service resource allocation; By combining distributed computing with improved predictive model algorithms, we aim to break through the performance bottleneck of large-scale clusters and build a high-throughput scenario stability guarantee system that integrates prediction, decision-making, and execution. This will promote the development of elastic scaling technology towards a more intelligent and robust direction.

1. Introduction

Cloud computing[1], as one of the most transformative technological paradigms of the 21st century, achieves centralized management and elastic configuration from local to remote data centers through cloud migration of computing, storage, and network resources. This change has given rise to a new IT service model - developers are able to break free from local hardware

constraints and directly access on-demand computing capabilities from cloud service providers such as Alibaba Cloud and AWS. Enterprises are shifting towards a pay as you go model to reduce IT costs, gain high availability, flexible scalability, and global deployment advantages, and enhance business agility; Cloud native achieves environment consistency, rapid iteration, and efficient collaboration through technologies such as containerization, microservices, CI/CD, and Kubernetes, optimizing the entire development, deployment, and operation process. In cloud native practice, Kubernetes, as the mainstream container orchestration platform, has a built-in HPA reactive auto scaling function that can dynamically adjust container size based on CPU/memory metrics to avoid resource waste and ensure basic performance. However, in high-throughput scenarios, it faces pain points such as temporary resource mismatch and performance fluctuations caused by scaling delays. To address the challenges of elastic scaling and stability assurance in distributed systems in high-throughput scenarios, we propose a dual innovation path: firstly, the SLMD LightGBM load prediction model based on one-dimensional prediction extracts multi-scale features of load data through local mean decomposition, combines the LightGBM gradient enhancement framework to construct a time series prediction model, and introduces a self updating mechanism to continuously track the evolution of load characteristics, effectively solving the problems of response delay and adaptation to dynamic load characteristics. In the experiment, the prediction accuracy was improved by 24.51%/11.19%; Secondly, a PAHP multidimensional collaborative scaling method is proposed, which integrates the advantages of predictive and reactive scaling. By integrating real-time data from multiple dimensions such as network traffic, response time, and custom indicators, and combining queuing theory, a service quality evaluation system is constructed to achieve dual verification of resource allocation optimization and system stability enhancement. The content architecture follows a logical loop of "problem status method verification": first, analyze the active prediction requirements caused by passive automatic scaling delay and dynamic load characteristic changes; Subsequently, the existing achievements in the fields of passive automatic scaling, prediction models, and scaling indicators both domestically and internationally will be systematically reviewed; Subsequently, the design principle of the SLMD LightGBM model and the experimental verification of the one-dimensional prediction driven scaling mechanism will be described in detail; Finally, the strategy design and controller implementation of the PAHP multidimensional collaborative scaling method will be elaborated. The research results will provide theoretical support and practical guidance for the elastic scaling and stability guarantee of distributed systems in high-throughput scenarios, and promote the evolution of cloud computing technology towards a more intelligent and robust direction.

2. Correlation theory

2.1 Cloud-native architecture serves as the core engine for enterprise digital transformation

Against the backdrop of deepening digital transformation, the structural limitations of traditional IT architecture have become increasingly apparent—under the physical server deployment model, resource utilization has long been constrained within the range of 15% – 20%, and hardware procurement cycles have become more like shackles restricting business expansion agility; Although virtualization technology has increased utilization to 40% – 60% through resource pooling, the high resource overhead and minute level startup delay of virtual machines (VMs) still hinder the improvement of deployment efficiency. Cloud native architecture, with its advantages of lightweight containers, secondary booting, and cross environment consistency, has become the core paradigm of modern application development. Its evolutionary logic began with the resource fragmentation dilemma in the era of physical servers, where early single container technology broke

through secondary booting but there was a gap in cross host network/storage management. In 2013, Docker container technology [2] achieved application and infrastructure decoupling through namespace isolation and federated file system innovation, laying the technical foundation. With the exponential growth of application scale, Kubernetes achieves automated scheduling of container clusters through declarative APIs, automatic fault recovery, and elastic scaling mechanisms, promoting cluster management upgrades. In 2015, CNCF[3] was established to initiate the process of ecological standardization, integrating technologies such as containers, microservices, and service grids to form a collaborative innovation ecosystem of "technology community industry". The core features of cloud native are reflected in the definition of IaC environment, microservice decoupling, and automated operation and maintenance, supplemented by design principles such as immutable infrastructure, service grid communication decoupling, and chaos engineering fault pre verification. Promote CI/CD to become a standard, strengthen environment configuration traceability and system observability. From an industry perspective, cloud-native is reshaping enterprise IT construction models—internet enterprises, as pioneers, fully leverage its agility to support business innovation; traditional industries such as finance and manufacturing are also gradually adopting cloud-native technology to accelerate digital transformation. However, new challenges are faced during implementation, including increased system complexity and technology stack selection issues, requiring enterprises to continuously optimize and adjust in practice.

2.2 Containerization technology, cloud-native lightweight architecture, and distributed system support

Containerization technology [4], as the core foundation of cloud native architecture, utilizes operating system level virtualization mechanisms to encapsulate applications and their dependencies into standardized and lightweight running units, ensuring environmental consistency throughout the entire process from development to production, and completely solving the classic engineering dilemma of "it works on my machine". Its core features include: standardized packaging format to achieve cross platform compatibility of "build once, run anywhere"; Lightweight design enables containers to share the host operating system kernel, compressing boot time to seconds or even milliseconds, significantly improving hardware resource utilization by 30% -50%, and reducing resource overhead compared to traditional virtualization technology; By automating the image management process, we promote the full automation of container image construction, distribution, and deployment, facilitating the smooth implementation of Continuous Integration and Continuous Delivery (CI/CD), allowing development teams to focus on business logic innovation and avoid deployment risks caused by environmental differences. This technology has also spawned a complete cloud native toolchain ecosystem, including container orchestration systems (such as Kubernetes), service meshes[5] (such as Istio), monitoring tools (such as Prometheus), etc., forming a "orchestration communication observation" technology loop. On the path of technological evolution, from early physical server deployment to virtualization technology (full virtualization, semi virtualization), and then to container technology, each generation has achieved quantitative optimization in resource utilization, performance, and other aspects, forming a complementary coexistence pattern. Despite facing challenges such as blurred security boundaries, complex network configurations, and difficulty in storage management, these issues are gradually being alleviated with the maturity of technology practices such as encrypted communication in service grids and automated mounting of storage orchestration tools. In the dimension of application value, container technology shows strong adaptability in Internet enterprise microservice architecture, financial core system transformation, industrial edge computing and other scenarios, and its characteristics of "lightweight, high flexibility, fast iteration" have become the key technical

support for building modern distributed systems.

3. Research method

3.1 Kubernetes cluster architecture supports elastic scaling and resource scheduling for distributed systems

As the cornerstone of cloud native architecture, Kubernetes cluster [6] can be regarded as a model of distributed system resource scheduling with its "master-slave collaboration" architecture design—the master node uses API servers as the portal, the scheduler as the decision center, and the controller as the execution engine, combined with etcd to build a "digital twin" global state management to form a closed-loop control loop. This design far exceeds the traditional script based operation and maintenance "passive response" mode, actively converging the cluster state to the expected value, like implanting a "self-healing" gene; The controller achieves state consistency through a three-stage cycle of "observation comparison execution". The replicated controller maintains the number of Pod replicas with military discipline, and the deployed controller supports rolling updates and rollbacks with version control thinking, demonstrating the "programmable operations and maintenance" capability that traditional operations and maintenance cannot match. HPA[7]breaks through static configuration limitations and shifts towards intelligent scheduling mechanisms—it tracks CPU and memory metrics in real-time, dynamically increasing or decreasing the number of Pod replicas. In peak traffic scenarios, this dynamic adjustment reduces cross regional communication latency by 20% - 30%, becoming the core guarantee for stability in high-throughput scenarios. Through diversified service types such as ClusterIP, NodePort, Loadbalance, and topology aware routing strategies, system resource utilization is pushed to a new level. This dialectical design of "isolation sharing" maximizes resource efficiency while ensuring security boundaries, becoming a sophisticated paradigm for resource management in cloud native architecture.

3.2 Kubernetes elastic scaling and stability optimization

In cloud native application scenarios, business workloads often exhibit significant fluctuations, which puts higher demands on dynamic adjustment of resource allocation. Kubernetes builds a dual dimensional automatic scaling mechanism through horizontal Pod Autoscaler (HPA) and vertical Pod Autoscaler (VPA), which improves resource utilization while ensuring service quality, laying the foundation for its ability as a core container orchestration platform. HPA is suitable for stateless service expansion scenarios, such as web server clusters. By monitoring Pod resource usage (such as CPU utilization and memory usage), combined with data collected by measurement servers, the expected number of replicas is calculated using a formula:

$$R_{e,d} = \left\lceil R_{ec} \times \frac{m_c}{m_d} \right\rceil \tag{1}$$

among which,$R_{e,d}$Represents the expected number of replicas,R_{ec}For the current number of replicas,$m_c$for the current measurement value,$m_d$for the target metric value,$\lceil \cdot \rceil$ to round up the function. When the calculation result does not match the current number of replicas, HPA will trigger a scaling operation to adjust the number of Pods to optimize resource allocation. VPA is more suitable for scenarios with significant changes in state applications or resource requirements, by dynamically adjusting the resource quotas of individual Pods (such as CPU and memory limits) to meet business needs and avoid performance degradation caused by insufficient resources. This "horizontal vertical" dual dimensional expansion mechanism not only reflects Kubernetes' academic

rigor in the field of resource scheduling, but also highlights its value extension in engineering practice. Figure 1 shows the process of the HPA automatic scaling framework, including the interaction logic between the metric server, controller, and Pod cluster.

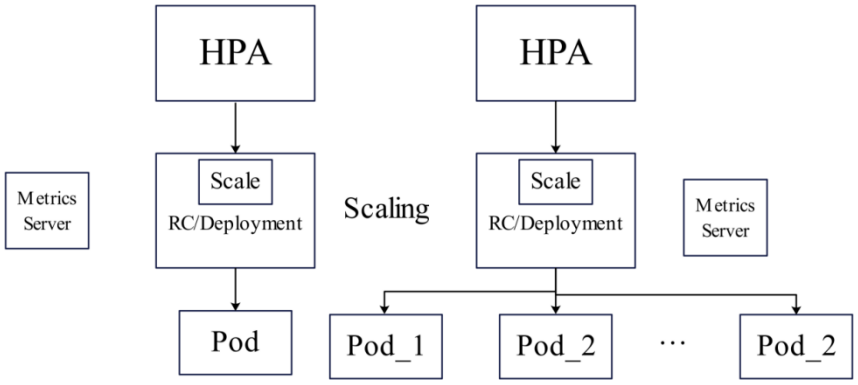

Figure 1 Structure diagram of Kubernetes HPA component collaborative scaling mechanism

VPA intelligently predicts future resource requirements and provides optimization configuration suggestions by continuously monitoring and analyzing Pod resource consumption data (such as CPU and memory usage, peak values, and changing trends), while retaining an appropriate safety margin to cope with sudden loads. Its adjustment strategy adopts a progressive approach to avoid forcing Pod restarts, ensuring the consistency and continuity of critical stateful service data. VPA and HPA collaborate to build an elastic solution: VPA optimizes individual Pod resource quotas and dynamically adjusts CPU/memory limits, while HPA regulates the number of Pod replicas based on overall load. In scenarios where load fluctuations are significant but horizontal scaling is not applicable, the two form a complementary mechanism of "vertical optimization horizontal scaling", which improves resource utilization by 15% -20% compared to traditional static allocation. The automatic scaling performance relies on multidimensional parameter tuning - monitoring indicators need to take into account both system level (such as CPU/memory utilization) and business indicators. The strategy configuration needs to optimize the trigger threshold, stable window, and scaling step size. Reasonably setting the stable window can reduce ineffective scaling by 20%; Resource quotas and node capacity constraints ensure system stability through dynamic scheduling.

3.3 SLMD-LightGBM Dynamic Prediction

The default reactive scaling mechanism of Kubernetes (such as HPA) is based on threshold triggering, which exposes significant latency defects in high dynamic scenarios - when traffic surges, it needs to go through three stages: monitoring cycle collection, demand calculation, and Pod startup, during which insufficient resources lead to performance avalanche; When traffic drops sharply, resource reduction lags behind, leading to the waste of "idle costs". Taking the example of a dual Pod handling a 40RPS encounter with a 120RPS burst, when HPA expands to 3Pod, traffic may have already fallen back to 60RPS, forming a vicious cycle of "response delay resource surplus", threatening the stability and cost-effectiveness balance of high concurrency scenarios. The SLMD LightGBM load prediction model[8]constructs a high-precision prediction engine through gradient boosting decision tree (GBDT)[9], and its core algorithm follows the mathematical logic of "initializing the prediction function to minimize the loss function":

$$f_0(x) = \operatorname{argmin}_c \sum_i L(y_i, C) \tag{2}$$

Estimation through negative gradient residuals:

$$r_{mi} = -\left[\frac{\vartheta L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=f_{m-1}(x)} \quad (3)$$

Fit regression tree to learn residuals and optimize leaf node values:

$$C_{mi} = \text{argmin}_c \sum_{x_i \in R_{mi}} L(y_i, C) \quad (4)$$

The final prediction function iteration is:

$$f_m(x) = f_{m-k}(x) + \sum_j C_{m,j} I\left(X \epsilon R_{mj}\right) \quad (5)$$

the final prediction function is

$$F(x) = f_0(x) + \sum_{m=1}^{M} \sum_{j=1}^{J} C_{m,j}\, I\left(X \epsilon R_{mj}\right) \quad (6)$$

The SLMD framework and LightGBM algorithm are integrated into the model, and a self updating mechanism is embedded to dynamically adapt to the evolution of load characteristics. Intelligent blade strategy selects the maximum gain node, resulting in smaller losses under the same number of blades, improving accuracy and efficiency; Compared to traditional passive response, active prediction reduces decision latency by over 40%, optimizing resource utilization and system stability in high concurrency scenarios.

4. Results and discussion

4.1 Experimental verification and multi index performance advantage analysis of SLMD LightGBM model

The experiment uses the NASA-HTTP request volume dataset and cluster CPU utilization time series to validate the model. NASA-HTTP contains 3461612 HTTP request records. Four week minute level summary data is taken, and an input-output set is constructed through a sliding window. Outliers exist in extreme values above 200, and zero values are replaced with 1 to avoid zero denominators. The CPU dataset is taken from Cluster Trace v2018 machine "m2024" and divided into training and testing sets proportionally. In the preprocessing stage, NASA-HTTP sequences are decomposed into multiple product functions and residual terms through local mean decomposition, capturing high-frequency to low-frequency temporal features; The CPU sequence has small fluctuations and no outlier handling has been done. The evaluation uses multiple indicators, including MAPE, MAE, RMSE, MSE, R^2, and training time. In NASA-HTTP validation, the predicted curve of this model is closer to the overall trend and local fluctuations of the original RPS sequence. The specific indicators are compared in Table 1.

Table 1 Comparison of performance evaluation metrics for load forecasting models

Model	MAPE	MAE	RMSE	MSE	R^2
ARIMA	18.03	6.05	8.23	67.72	0.41
LightGBM	17.35	5.31	7.42	55.03	0.52
LSTM	17.08	5.16	7.10	50.37	0.56
Bi-LSTM	16.15	4.84	6.46	41.80	0.58
SLMD-LightGBM	11.19	3.00	4.43	19.63	0.80

In terms of training time, SLMD LightGBM only takes 32.93 seconds to complete training on the CPU dataset, which meets the demand for fast updates in high dynamic scenarios. LMD

decomposition effectively captures the local fluctuations and non-stationary features of time series. Combining LightGBM's histogram algorithm with intelligent leaf growth strategy, the model reduces computational complexity and memory consumption while ensuring prediction accuracy. This feature enables it to adapt to real-time load forecasting requirements in Kubernetes elastic scaling scenarios, achieving efficient resource scheduling.

4.2 Model experiment

In dynamic load scenarios, although the active adjustment strategy of single dimensional load prediction alleviates the lag problem of local passive scaling, it has significant limitations - the prediction accuracy is easily affected by load bursts and uncertainties, making it difficult to fully reflect the multidimensional system state. To this end, a multidimensional collaborative scaling mechanism is proposed: by integrating prediction results with real-time monitoring indicators, it not only retains the forward-looking advantage of prediction, but also compensates for the shortcomings of a single prediction model, providing a more robust and elastic solution for complex production environments. The real-time measurement correction stage compares the current actual RPS with the predicted value, adjusts the number of Pods based on the error ratio, and ensures the real-time and accuracy of scaling decisions. Multidimensional measurement selection should be combined with application characteristics: network applications focus on request rate, latency, and error rate; AI applications focus on abnormal CPU utilization; In memory databases are sensitive to memory pressure. Taking the typical web application Podinfo as an example, its monitoring module collects the request volume per second, the prediction module accesses Prometheus through the HTTP protocol to obtain the average request volume, uses SLMD LightGBM to predict the next minute's request volume, and combines service latency and real-time RPS to determine the final Pod count; The monitoring module continuously collects cluster status indicators to provide data support for prediction and expansion; The scaling module triggers HPA based on measurement values to achieve automatic scaling of Pods, ensuring a balance between system stability and cost-effectiveness in high dynamic scenarios.

4.3 Effect analysis

In high dynamic and high concurrency distributed scenarios, traditional reactive scaling mechanisms are difficult to meet the stability and cost-effectiveness requirements of high-throughput scenarios due to latency issues. This article proposes a multidimensional collaborative elastic scaling scheme, which integrates load prediction models and real-time monitoring indicators to achieve precise dynamic adjustment of resource allocation, effectively solve lag problems, and improve system response speed and resource utilization. The core of this mechanism lies in collaborative prediction results and multi-dimensional auxiliary indicators. Distributed elastic expansion breaks through the limitations of traditional HPA by optimizing the LightGBM model to deeply analyze historical request rate sequences, accurately capturing subtle trends in future loads, and achieving a collaborative breakthrough between predictive intelligence and dynamic adjustment. In high dynamic scenarios of distributed systems, traditional HPA is difficult to meet the requirements of high throughput stability and cost efficiency due to its lag. The new mechanism predicts future load by optimizing the LightGBM model to parse historical request rate sequences, dynamically coupling with 1-second delay threshold and real-time request error rate to drive Pod intelligent scaling, and introducing queuing theory models (M/M/1/M/c) and CPU/memory indicators to construct multidimensional constraints, balancing stability and economy. Experiments have shown that optimizing MaxPod to 15 under peak load reduces resource overhead by 20% and total cost by 25%; The violation rate is 16.3%, and the 99th percentile delay is $\leqslant$ 880ms, which is

more than 40% higher than the traditional solution. Extending the Pod activity window can maintain the service quality when the load drops. The architecture forms a "monitoring prediction scaling" closed loop, adapts to sudden/periodic/stable load modes, and solves the problem of HPA lag.

5. Conclusion

In high-throughput distributed systems, traditional reactive scaling falls into a vicious cycle of "delayed response resource waste performance fluctuation" due to lag and adaptability defects. This article proposes a dual engine mechanism of "single dimensional accurate prediction+multi-dimensional collaborative regulation": the SLMD LightGBM model is used to fuse local mean decomposition, gradient enhancement, and self updating mechanisms in the single dimension, achieving a MAPE of 24.51% and 11.19% on two datasets, which is 30%-50% lower than traditional methods, alleviating response delay and providing forward-looking support; But there is a vulnerability to "precision and error", where sudden loads or prediction deviations may cause resource imbalances. Multidimensional PAHP A through the paradigm of "prediction as decision" and constructs a three-dimensional framework of "prediction real-time service". It combines prediction models, real-time monitoring, service quality evaluation, and M/M/c queuing theory dynamic calibration to form a dual adjustment mechanism. It is necessary to critically examine the optimization space of response speed in extreme scenarios and the problem of adaptive deviation of heterogeneous system parameters. In the future, we will focus on three directions: smooth adjustment of Pod contraction, collaborative optimization of microservice load balancing, and complexity reduction of distributed computing. Through the "prediction regulation optimization" closed-loop, we will promote the transition of the system from passive response to active control, and achieve intelligent expansion of resilience and stability in high-throughput scenarios.

References

[1] *Klinke H. Cloud Computing[J]. Springer, Cham, 2025.DOI:10.1007/978-3-031-88130-5_17.*

[2] *Jiang Q, Zhang W, Lin Z, et al. A Technical Overview of Docker Container Security Threats[C]//International Conference on Cyberspace Simulation and Evaluation.Springer, Singapore, 2025.DOI:10.1007/978-981-96-4509-1_27.*

[3] *Liu Y, Herranz A H, Sundin R C. RoboKube: Establishing a New Foundation for the Cloud Native Evolution in Robotics[J].IEEE, 2024.DOI:10.1109/ICARA60736.2024.10552996.*

[4] *Liu, X., & Yang, D. (2025, March). LLM Data Strategy: Improving Data Availability and Efficiency. In Doctoral Symposium on Computational Intelligence (pp. 425-437). Singapore: Springer Nature Singapore.*

[5] *Zhang, Q. (2025, October). Application of Reinforcement Learning in Dynamic Advertising Content Generation. In 2025 2nd International Conference on Software, Systems and Information Technology (SSITCON) (pp. 1-5). IEEE.*

[6] *Wang Y. Application of Data Completion and Full Lifecycle Cost Optimization Integrating Artificial Intelligence in Supply Chain[J]. 2025.*

[7] *Sun, J. (2025). Research on Business Data-driven Risk Prediction Methods Based on Machine Learning. Advances in Computer and Communication, 6(4).*

[8] *Wu, Y. (2025, October). Multi-Level Belief Rule Base Modeling Architecture and Intelligent Optimization Technology for Decision Support Systems. In 2025 2nd International Conference on Software, Systems and Information Technology (SSITCON) (pp. 1-8). IEEE.*

[9] *Chen, M. (2026). Research on Privacy-Preserving AI Model Training and Validation Methods Based on Federated Learning.*

[10] *Xu, D. (2026). Analysis of the impact of video infrastructure optimization on large-scale content quality improvement.*
[11] *Hou, Y. (2026). Research on Heterogeneous Server Upgrade Strategies and Resource Utilization Efficiency Oriented Toward Green Computing Objectives. Advances in Computer and Communication, 7(1).*
[12] *Huang, J. (2025, September). Performance Evaluation Index System and Engineering Best Practice of Production-Level Time Series Machine Learning System. In 2025 International Conference on Intelligent Communication Networks and Computational Techniques (ICICNCT) (pp. 01-07). IEEE.*
[13] *Pan, H. (2025, March). Research on Efficient Computing Model of Hartree Fock and Density Functional Theory Based on GPU Acceleration. In Doctoral Symposium on Computational Intelligence (pp. 485-496). Singapore: Springer Nature Singapore.*
[14] *Zhang, Q. (2026). Security Improvement and Application of Identity and Access Management in Saas Platform.*
[15] *Yiting Hong. Differentially Private High-Dimensional Business Data Publishing and Analysis Algorithm. International Journal of Business Management and Economics and Trade (2026), Vol. 7, Issue 1: 28-35.*
[16] *Wu, W. (2025, June). Construction and optimization of intelligent gateway software management platform based on jenkins cluster management under cloud edge integration architecture in industrial internet of things. In International Conference on 6G Communications Networking and Signal Processing (pp. 633-645). Singapore: Springer Nature Singapore.*
[17] *Wu Y. Optimization of Generative AI Intelligent Interaction System Based on Adversarial Attack Defense and Content Controllable Generation[J]. 2025.*
[18] *Chen M. Research on Automated Risk Detection Methods in Machine Learning Integrating Privacy Computing[J]. 2025.*
[19] *Yanchun Wang. (2025) Research on Enhancing ERP System Efficiency Through AI in Cross-border Supply Chain Environments. Advances in Computer and Communication, 6(5), 268-273.*
[20] *Wu, Y. (2026). Federated Learning-based Algorithm Design for Privacy Preservation in Cross-domain Data Sharing. Engineering Advances, 6(1).*
[21] *Ding, J. (2025). Exploration of Process Improvement in Automotive Manufacturing Based on Intelligent Production. International Journal of Engineering Advances, 2(2), 17-23.*
[22] *Zhang, C., Han, J., Zou, Y., Dong, K., Li, Y., Ding, J., & Han, X. (2024, April). Detecting the anomalies in LiDAR pointcloud. In WCX SAE World Congress Experience. SAE Technical Paper.*
[23] *Huang, J. (2025, August). Research on Multi-Model Fusion Machine Learning Demand Intelligent Forecasting System in Cloud Computing Environment. In 2025 2nd International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS) (pp. 1-7). IEEE.*