Hindawi
Journal of Advanced Transportation
Volume 2022, Article ID 4390923, 16 pages
https://doi.org/10.1155/2022/4390923

Research Article

Automatic Scaling Mechanism of Intermodal EDI System under Green Cloud Computing

Qiang Huang,[1] **Lin Sun**,[1] **Furong Jia**,[1] **Jiaxin Yuan**,[1] **Yao Wu**,[1] **and Jinshan Pan**[2]

[1]*College of Information Engineering, Sichuan Agricultural University, Ya'an City, Sichuan Province 625014, China*
[2]*Southwest Jiaotong University, Chengdu City, Sichuan Province 610031, China*

Correspondence should be addressed to Jinshan Pan; jshpan@swjtu.cn

Received 5 April 2021; Accepted 3 April 2022; Published 30 May 2022

Academic Editor: Chi-Hua Chen

EDI is a hot topic in the research of multimodal transportation informatization, which determines the exchange level of intermodal transportation information. However, its high cost, large system coupling degree and low performance threshold cannot adapt to mass data exchange in high concurrent environment. Therefore, a decentralized, scalable, distributed and efficient data exchange system is formed. It plays a key role in realizing the comprehensive sharing of interdepartmental intermodal information in the cloud environment. In order to solve the problem of mismatching between application load and computing resource capacity and realize on-demand resource allocation and low carbon emission, this paper proposes to build an Extensible EDI system (XEDI) based on MSA and studies the scaling mechanism in container environment. Based on the resource scheduling characteristics of container cloud and considering the distribution and heterogeneity of intermodal cloud computing platform from the perspective of resource allocation, the automatic scaling mechanism of XEDI is established, the scaling model is established, and the automatic scaling algorithm is proposed. For Dominant Resource Fairness for XEDI (XDRF) resource allocation algorithm and Dominant Resource Fairness for XEDI (CXDRF) based on carbon considering energy consumption, the CXDRF algorithm is proved by quantitative experiments to achieve system performance optimization on the basis of ensuring system reliability and effectively reducing energy consumption. XEDI can not only meet the demand of dynamic load and improve service quality, but also reduce resource occupation and save energy by releasing virtual resources when resource utilization rate is low. It has great research significance and practical value for mass data application under low energy consumption conditions.

1. Introduction

Multimodal transportation is recognized as the most efficient mode of transportation service in the world, which is conducive to improving logistics efficiency and reducing logistics costs [1]. Among them, information sharing, as the key technology of the information of molten iron and intermodal transportation, not only determines the development level of the information of intermodal transportation, but also provides information guarantee for the realization of one-stop intermodal transportation service. China has established an intermodal transportation information sharing platform centered on some large ports, which has preliminarily realized data exchange between ports and railway departments, reduced cargo storage time and transportation cost, and effectively improved the efficiency of collaborative operation. Information sharing is the core of intermodal transportation informatization. Because the current information sharing technologies are all "chimney" architectures, they can only solve the problem of information integration in a local range and cannot meet the demand of on-demand information sharing in the cloud environment.

At present, the mainstream research is still the traditional EDI technology. Based on a research project conducted at the Institute of Logistics and Warehousing, Debicki and Kolinskianalyzed the impact of EDI methods on the complexity of information flow in global supply chains [2]. However, the traditional EDI technology has some problems, such as high cost, backward technology, and large coupling degree of the system, and the author does not provide corresponding solutions. Betz et al. applied ICT and

introduced the current application technology, connection type, message standard, and its impact on multimodal transport supply chain based on the international research results of Hamburg Port and Logistics Institute [3]. However, this technology is mainly customized for different users, and it is difficult to adapt to large-scale intermodal transportation systems. Ding explores the functions and operating conditions of relatively independent information systems for railways and ports, combined with traditional information exchange modes, and establishes an electronic platform suitable for information interconnection and intermodal station interoperability [4]. However, the traditional information exchange mode is still adopted, which cannot adapt to the massive data exchange in a high-concurrency environment.

At present, information sharing-related technologies adopted by core intermodal transportation organizations such as ports and railways mainly include the following:

1.1. Electronic Data Interchange. It refers to the formation of structured document data in accordance with relevant standards and the completion of end-to-end electronic data transmission methods. EDI standardizes and formats exchanged information in accordance with agreed protocols (such as EDIFACT and SOAP). It exchanges data between the computer network systems of trading partners through data transmission systems such as mail servers, FTP, and Message Queue (MQ), which can effectively solve the inefficiency of paper-based information transmission.

1.2. Service Oriented Architecture. It is defined as a functional paradigm for integrating dispersed businesses within an enterprise, and its essence is Enterprise Application Integration (EAI) technology that realizes information exchange between heterogeneous systems. The SOA component model realizes business information interaction between heterogeneous systems by defining standardized interfaces between different services and has the characteristics of loose coupling, coarse-grained, and transparency. As a technical realization of SOA, WebService has better openness and decoupling than traditional EDI.

1.3. Enterprise Service Bus. It is a bus-based enterprise-level SOA architecture with features such as interoperability, independence, modularity, and loose coupling. ESB takes services as the basic unit, and services are coordinated through messages to complete related business collaboration, and service consumers do not need to know the technical details of the service provider. ESB can not only reduce the workload of development and maintenance, save costs, and improve the scalability of the system, but also better deal with the heterogeneity of different technologies and protocols.

According to the research of relevant literature, the current application scope, advantages, and disadvantages of information integration technology in intermodal transportation informatization are shown in Table 1.

This paper constructs a self-scaling mechanism for the K8S-based XEDI (Extensible EDI) closed-loop control system, establishes an expansion model, and proposes an automatic expansion algorithm, a resource allocation algorithm, and a resource allocation optimization algorithm considering energy consumption to achieve flexible data sharing and on-demand resource allocation in a cloud environment. The performance of EDI system limits the energy consumption. Finally, the scalability test verifies that the proposed algorithm has good scalability effect and high scalability efficiency under heterogeneous cluster conditions, which can not only ensure the reliability of the system and realize the performance optimization of the system, but also effectively reduce energy consumption. It can not only meet the demand of dynamic load and improve the quality of service, but also reduce resource occupation and save energy by releasing virtual resources when resource utilization is low. It can solve the problems of high cost, system scalability, and insufficient data processing capacity of existing EDI system solutions.

This article is divided into eight parts:

(1) Introduction. This part generally introduces the methods used in this paper, also compares other methods, and discusses their advantages and limitations.

(2) The Introduction of XEDI. This section gives a detailed introduction to XEDI, including its advantages over EDI and the architecture of XEDI.

(3) Scaling models of XEDI. This part introduces the single-index scaling model of XEDI.

(4) Multi-index scaling model. This part introduces the multi-index scaling model of XEDI.

(5) Algorithms. In this part, the scaling algorithm based on the scaling model is introduced.

(6) Evaluation of the algorithm and example. This section tests XEDI's scalability performance.

(7) Conclusions. This part is a summary of the full text, and the performance of the algorithm in the article is summarized.

(8) Prospect. This section introduces the prospect of the algorithm and other application scenarios.

2. The Introduction of XEDI

Cloud computing has been developed as one of the creative platforms that give dependable, virtualized, and adaptable cloud resources over the Internet. Intermodal transportation refers to the "carriage of goods by two or more modes of transport." Traditional system framework of the intermodal transportation is rigid and lacks information sharing [5]. However, cloud computing helps provide a new direction to solve these bottlenecks and realize the informatization of the intermodal transport.

Electronic Data Interchange (EDI) refers to a standard for exchanging business documents, such as invoices and

Table 1: Comparison of different information integration technologies.

Information integration technology	Scope of application	Advantage	Disadvantage
EDI	Widely used in port informatization for data exchange between institutions	EDI message standards are perfect, which can better meet business needs	The cost is high, the technology is backward, the system coupling is large, the performance of the remote call method is low, the performance threshold is low, and it cannot adapt to the massive data exchange in the high concurrency environment
WebService	Used for business system integration of some electronic ports and ports	Mature technology, low coupling, low cost, and easy implementation of SOAP data standards	Its essence is a Web-RPC system, and the SOAP-based remote call method has a low performance threshold, and the supported message types are limited
ESB	There are relatively few applications in the interoperability industry, which is more suitable for the complex internal environment	Complete system, with standard adapters and extensible interfaces, low development, maintenance, and management costs, and strong compatibility with heterogeneity issues	The structure is cumbersome, the scalability is poor, and the software and hardware requirements are high. If different protocols are uniformly converted into SOAP messages through the adaptor and then XML parsing, there will be more unnecessary format conversions, especially the processing efficiency of large data packets

purchase orders, in a standard form between computers through the use of electronic networks like the Internet. It is widely used in the information sharing mechanism of intermodal transport. However, as time goes by, there appear more and more defects of EDI, such as high powerful consumption and low performance threshold, which make it hard to adapt to the mass data exchange under the cloud computing environment [6]. In order to realize the elasticity of information sharing, which expands when faced with high concurrent information processing and vice versa, we have to build a lighter and more flexible EDI system, named XEDI system in our paper.

When and how does the system stretch specifically? Though Kubernetes (K8s), a mature open-source system for automating deployment, scaling, and management of containerized applications, provides an ideal platform for hosting various workloads, automated scaling of the cluster itself is not currently offered, and thus, it is necessary to rebuild an automated scaling model based on that [7, 8].

Definition 1. XEDI system is the lighter and more flexible EDI system that we build, which provides open messages all intermodal participating organizations through the cloud.

Definition 2. Dominant Resource Fairness for XEDI (XDRF) is an algorithm designed to allocate the resources of Pods more fairly and perform better in calculating.

Contributions: this work has the following key contributions:

(1) It built the XEDI system and stretching model of that.

(2) It presented the algorithm to realize the scaling progressing.

(3) It provided a comparative analysis between our algorithm and others.

(4) It evaluated the energy consumption of the cloud system.

Compared with traditional EDI technology, XEDI has the following advantages:

(1) Low cost and high concurrency. Adopt micro-service unit encapsulation. The message processing module is encapsulated through microservices, which can be flexibly scheduled in the container cloud environment and simplify the construction of the scaling mechanism to achieve high concurrent message processing with variable loads with minimal computing resource consumption.

(2) Support remote call. Use asynchronous message mechanism. The asynchronous message protocol adapter (Takia) is used to realize message reception and forwarding, and the high-performance distributed queue system (Kafka) is used to replace the inefficient remote call and folder delivery polling mechanism of the traditional EDI system.

(3) Good scalability. Adopt an extensible message processing module. Message processing should be modularized, by encapsulating different message type processing procedures into micro-service units, and configured and extended according to message types and access protocols.

Different from the traditional EDI system, XEDI does not deploy to each port but manages it in a unified manner under a resource support system, renting functions such as

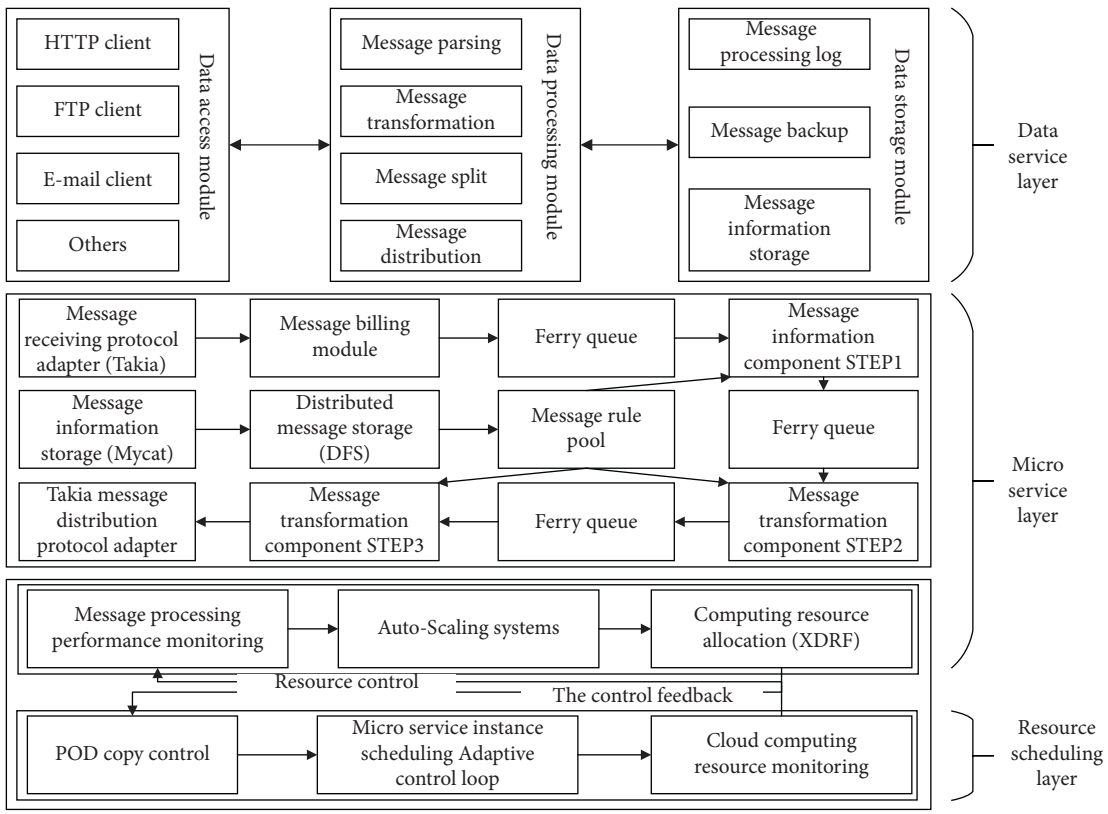

Figure 1: Hierarchical architecture of XEDI.

message exchange and distribution to participating institutions and users in the form of EDIaaS to reduce overall costs. However, the system design is mainly aimed at large-scale intermodal information platforms and has poor adaptability to the differentiated needs of individual users.

The construction of XEDI's architecture makes it clear about how the messages interact from different EDI systems under cloud. XEDI system are composed of Data Service Layer, Micro-Service Layer, and Resource Scheduling Layer from top to bottom. The logic structure of Data Service Layer is similar with the traditional EDI system. Considering business operation compatibility, Data Service Layer consists of three models: Data Access Modal, Data Processing Modal, and Data Storage Modal to make messages received and sent, parsed, and transformed and make messages stored [9]. To adapt containers scheduling, Data Processing Modal rebuilds the decentralized modal using micro-service. The last layer takes charge of component scheduling and the feedback of performance monitoring. The architecture is shown in Figure 1:

3. Scaling Models of XEDI

Most of the current scaling models and algorithms are designed based on IaaS VMS and can be divided into vertical and horizontal scaling modes [10]. However, it takes a long time to configure and start and stop virtual machine instances, so the scaling response is poor in real-time. Unlike IaaS, lightweight container clouds can scale applications in real time in a larger cluster environment. Because the container is an immutable carrier, only supporting horizontal scaling model, although the current container arrangement system has set up a simple response telescopic mechanism (for example K8S HPA [11]), but because only to copy an application based on memory and CPU load control, application scope is limited and has yet to have related research for complex component system scaling problem. Because XEDI's micro-service components are interconnected, there is no general scaling control by abstracting services into independent nodes [12]. In this paper, a closed-loop control system based on XEDI is proposed to build a self-scaling mechanism to achieve elastic data sharing and on-demand resource allocation in the cloud environment.

The scaling tactic is a function whose input is the indicator vector obtained from the XEDI monitoring module. Each dimension of the vector represents a monitoring indicator. In addition, the monitoring module ensures a long enough historical record. The record matrix P corresponding to the index data collection is shown as formula:

$$P = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1(m-1)} & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2(m-1)} & x_{2m} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{(n-1)1} & x_{(n-1)2} & \cdots & x_{(n-1)(m-1)} & x_{(n-1)m} \\ x_{n1} & x_{n2} & \cdots & x_{n(m-1)} & x_{nm} \end{bmatrix}. \quad (1)$$

N specifically represents the length of historical records, and m specifically represents the number of monitoring indicators.

The output of the scaling strategy is scaling index I, that is, $I = f_i(P)$, f_i is the scaling strategy function. If I is large, it can be interpreted as urgent to expand capacity and vice versa. In order to realize the quantization of the scaling decision, the system usually sets the expansion threshold I_{up} and the shrinkage threshold I_{down} in advance. If $I > I_{up}$ appears, the expansion process will be carried out at this time; if $I < I_{\text{down}}$, the shrinkage process will be carried out at this time.

According to the choice of P, f_i can be divided into single index strategy and multi-index strategy. When $m = 1$, f_i is a single-index scaling strategy; when $m > 1$, f_i is a multi-index scaling strategy. At the same time, according to the choice of f_i, the scaling strategy can be divided into response strategy and prediction strategy.

Although the single index algorithm is simple, it is prone to the miscalculation of scaling. In terms of scaling strategy, compared with a responsive scaling strategy, a predictive scaling strategy can make a prediction based on historical load and make scaling decisions earlier, which has a better scaling effect [13]. We propose a multi-index scaling model of XEDI based on a single index predictive scaling strategy.

In the single index algorithm, the input matrix P can be simplified as the historical window vector of load indicators, defined as follows:

3.1. Responsive Scaling Strategy. The nonprediction model is generally based on the historical window $H(x, n)$ to make a weighted average of the index x as the response value $V_r(H(x, n))$, where x is the index, and n is the window size. The following formula (2) is used for calculation:

$$V_r(H(x,n)) = \sum_{i=1}^{n} q_i x_i, x_i \in H(x,n), \tag{2}$$

where q_i is the weight coefficient. When $q_i = 1/n$, $V_r(H(x, n)) = V_{avg}(H(x, n))$ is the average value of indicator window $H(x, n)$. According to formula (3), the response scaling index I can be obtained:

$$I = f_i(P) = V_r(H(x,n)). \tag{3}$$

In particular, when $n = 1$, $I = x_n$, that is, scaling according to the current load, which is currently the industry commonly used scaling strategy.

3.2. Predictive Scaling Strategy. Compared with a responsive scaling strategy, a predictive scaling strategy can predict the historical load and make scaling decisions earlier, which has a better scaling effect [14]. In this paper, the autoregressive model is adopted to design a predictive scaling strategy, which is generally used in the stage of statistics and signal processing. As a random process, it is mostly used for modeling and forecasting various natural phenomena. Although XEDI message load changes are not the case. AR(p) specifically represents the p-order autoregressive model in this study. The definition of AR(p) model is as follows:

$$X_t = c + \sum_{i=1}^{p} \varphi_i X_t - i + \varepsilon_t. \tag{4}$$

The X_t is model variables, φ_i is the model of the regression coefficient, c is a constant (usually zero), ε_t is a random error, and p is the order number.

In the process of AR (1), a sliding window composed of multiple cycle monitoring indicators is used to predict the load value of future cycles, which are called adaptive Windows. According to $H(x, n)$ of the history window with length n, let the length of the adaptive window be w and iteratively predict the value of a new period based on n recent historical records. AR (1) can predict the index x_i of w future periods in the adaptive window, where $n < I < = n + w$, x_i is calculated iteratively by formula:

$$x_i = x_{avg} + \rho(1)\left(x_{i-1} - x_{avg}\right) + e_i, \tag{5}$$

where x_{avg} represents the mean value of x_i in the history window, e_i represents noise (generally 0), and $\rho(1)$ represents the relation function when the delay step number is *1*. $\rho(1)$ is calculated by the following formula:

$$\rho(l) = \frac{1}{n + w - l} * \sum_{i=1}^{n-l} \frac{\left(x_i - x_{avg}\right)\left(x_{i+l} - x_{avg}\right)}{\sigma_n^2}. \tag{6}$$

where σ_n^2 represents the standard deviation of the historical window.

Then, the predicted peak value can be obtained from the w window of indicator x_i. It is reasonable that when indicator x is the load rate, it can be calculated by the following formula:

$$V_p(H(x,n)) = \max\{x_i | i \in (n, n + w]\}, \tag{7}$$

The same as formula (3), the expansion index can be predicted $I = f_i(P) = V_p(H(x, n))$.

Although AR (1) algorithm solves the problem of index prediction in the time window, it can only realize the prediction of a single index load. Literature [15, 16] has proved through experiments that when the selection of indicators does not match with the type of load, the real load of the application cannot be shown, and even if the algorithm is rigorous, it will fail. The multi-index algorithm can rely on the comprehensive analysis of multiple load indicators to correctly judge the scaling time and effectively avoid the situation that the load is too large, the application scale cannot be adjusted correctly, and the request cannot be responded to normally.

4. Multi-Index Scaling Model

The basic idea of implementing the multi-index scaling strategy is to transform the multi-index load into a single index set. According to the above analysis, the input P of the scaling strategy is an $n * m$ matrix, and the output is the scaling index I. The multi-index scaling strategy is shown in the following formula:

$$I = f_i(P), x_{mn} \in P, m > 1, n \geq 1. \tag{8}$$

The calculation steps of I are as follows:

4.1. Convert Multiple Indicators into Single Indicators. The weighted average is carried out for each index in each row of P, and the transformation formula in the k row is as follows:

$$x_k = \sum_{i=1}^{m} x_{ki} * \frac{x_{ki}}{\sum_{j=1}^{m} x_{kj}}. \tag{9}$$

You take all the rows, you transform the input matrix P, and you get a vector that has dimension n, which is $H(x, n)$.

4.2. A Single Index Is Used to Calculate the Load. After converting the multi-index matrix into a single index vector, the single index scaling model can be used to calculate the scaling index I and carry out the scaling decision-making process.

At the same time, in terms of index selection, XEDI adopts the multi-index comprehensive trigger strategy that can reflect the performance most directly, so as to avoid the failure of prediction algorithm due to the failure of CPU, memory, and other indirect indexes to reflect the real status of message processing. The multi-index predictive scaling algorithm can select the trigger point of scaling more effectively and effectively prevent excessive scaling operation when combined with the cooling time of scaling.

Based on [17] Dominant Resource Fairness (DRF) algorithm and Dominant Resource Fairness for XEDI (XDRF), which is an extension of the two above-mentioned algorithms, it is designed to allocate the resources of PODs more fairly and perform better in calculating.

Assuming that there are n available computing nodes in the current cluster operating environment of XEDI, each computing node has m resources in total, Q_k represents the performance evaluation score of node k, η_k represents the ratio of the performance evaluation score of node k to the average score, and T_k represents the resource type characteristics of node k, and the encoding is consistent with Definition 3:

Definition 3. (XEDI performance context). Parameter *XEDI.C* = {XEDI performance index set ∪ XEDI resource I resource status index set} is the performance context of the current XEDI system.

$\partial_{i,k}$ represents the adaptation factor of *POD(i)* on machine k, $D_{i,j}$ represents the demand of a copy of *POD(i)* for resources of type j, with $D_i = D_{i,1}, D_{i,2}, D_{i,3}, \ldots, D_{i,m}$, S_i represents the dominant share of *POD(i)*, R_j^k represents the total amount of resources of type j on node k, and $Ru_{i,j}^k$ represents the number of resources of type j that *POD(i)* has been allocated on node k, $Rc_{k,j}$ represents the number of resources of type j on node k that can be allocated, and W_i represents the weight of calculating *POD*(i). The calculation process is as follows:

(1) W_i of each POD weight requiring capacity expansion in the POP set is

$$W_i = \frac{V(H(POD.MEMinsR, n)) + V(H(POD.CPUinsR, n))}{2}. \tag{10}$$

(2) The ratio of the performance evaluation score of node k to the average score is

$$\eta_k = \frac{nQ_k}{\sum_{i=1}^{n} Q_i}. \tag{11}$$

(3) The adaptive factor and dominant share S_i of *POD(i)* on node k were calculated, and k was

$$\begin{cases} S_{i,k} = \dfrac{\max_{j=1}^{m}\left\{Ru_{i,j}^k / R_j^k\right\} * \partial_{i,k}}{W_i}, \\ \partial_{i,k} = a * \eta_k + (1-a) * \dfrac{POD(i).RTP\&T_k}{POD(i).RTP}; s.t. 0 \leq a \leq 1. \end{cases} \tag{12}$$

(4) Calculate the leading share S_i (DS value) of *POD(i)* as the sum of its leading shares on each node:

$$S_i = \sum_{k=1}^{n} S_{i,k}. \tag{13}$$

(5) The resource Predicates set of computing *POD(i)* is

$$N_{pre}(i) = \left\{k | k \in \{1, 2, \ldots, n\}; \forall j \in \{1, 2, \ldots, m\}; D_{i,j} \leq Rc_{k,j}\right\}. \tag{14}$$

(6) The JTH resource allocation to *POD(i)* is determined by the following Priority:

When j is odd, the copy of *POD(i)* is allocated with suitable high-quality resources, as shown in the following formula:

$$s.t. k \in N_{pre}(i); \frac{\partial_{i,k} - a\eta_k}{1-a} > 0. \tag{15}$$

When j is even, the copy of *POD(i)* is allocated with suitable inferior resources, as shown in the following formula:

$$s.t. k \in N_{pre}(i); \frac{\partial_{i,k} - a\eta_k}{1-a} > 0. \tag{16}$$

5. Algorithms

The automatic scaling algorithm of XEDI is designed based on the scaling model in Section 2, which mainly solves the problem of when the message processing module scales in the container cloud environment. According to the threshold of the scaling index, the scaling process is divided into two algorithms: Algorithm 1 is for expansion, and Algorithm 2 is for shrinkage. The scaling algorithm firstly obtains the monitoring data, and under the condition that

the performance is not abnormal, calculate the load index set and calculate the XEDI message workload [18]. If the message's expansion index exceeds the expansion threshold, it traverses all the packet processing packets in sequence and calculates the data load of the corresponding POD. If the expansion index of the data packet exceeds the expansion threshold, the POD replica set is expanded to improve data throughput. On the contrary, if the expansion index is lower than the reduction threshold, the POD replica set is scaled down to release resources, and under the premise of ensuring the concurrent processing performance of the message, the resource occupation is minimized (Algorithm1).

6. Fairness Analysis of XDRF and CXDRF Algorithms

In the process of cloud resources sharing, the efficiency and fairness of the allocation of resources are the most important properties, widely considering the encourage sharing, cheat blocking, no jealous, and Pareto efficiency as an important index of judging allocation mechanism, and the following XDRF algorithm in POD expansion process is used to further discuss the equality of the allocation of resources:

Theorem 1. *XDRF is incentive sharing*

Proof. if there are k PODs to expand, for any *POD (i)*, *POD (j)*, satisfying $i \neq j, i \leq k, j \leq k$, $POD(i), POD(j) \in$ collection (POP), if satisfies $S_i < S_j < \ldots < S_k$, then the allocation of *POD(i)* results in the amount of resources D_i, and the total amount of resources decreases as $R = R - D_i$. According to formulas (3)–(6), the increase of the used resource Ru_j^k will cause the DS value S_i of *POD(i)* to increase. While $S_i > S_j$, *POD(i)* stops allocating and allocates resources to POD(j) to minimize the DS values alternation of different POD. When the load falls back, each POD will call Algorithm 3 to release the excess resources to ensure the resource-share of other POD in order to guarantee the resource-share of the next expansion, and the proof is completed. □

Theorem 2. *XDRF prevents strategic operations*

Proof. suppose that there are two resources r_1 and r_2, and the total resources are R_1 and R_2 respectively; there are two computing tasks i and j, and their resource demand vectors are $D_i = d_{i,r1}, d_{i,r2}$ and $D_j = d_{j,r1}, d_{j,r2}$. If the following relationship exists, $(d_{i,r1}/R_1) > (d_{i,r2}/R_2)$, $(d_{j,r1}/R_1) < (d_{j,r2}/R_2)$, then the dominant resource of computing task i is r_1, and the dominant resource of computing task j is r_2. If x_i and x_j are the number of subtasks of calculation tasks i and j respectively, the x_i and x_j are calculated by the following formula:

$$\begin{cases} d_{r,r1} * x_i + d_{j,r1} * x_j \leq R_1, \\ d_{r,r2} * x_i + d_{j,r2} * x_j \leq R_2, \\ \left(d_{i,r1}/R_1\right) * x_i = \left(d_{i,r2}/R_2\right) * x_j. \end{cases} \quad (17)$$

Assume that POD(i) increases its dominant resource demand from D_i to D_i' in order to obtain more shares, and the dominant resource of POD(i) is r, and then $d_{i,r} < d_{i,r}'$; if the dominant resource of POD(j) is p, according to formula (17), it can be known that when capacity expansion is completed, $(d_{i,r}/R_r) * x_i = (d_{j,p}/R_p) * x_j$, and $(d_{i,r}'/R_r) * x_i = (d_{j,p}/R_p) * x_j$, because $(d_{j,p}/R_p) * x_j$ keeps the same , so $(d_{i,r}/R_r) * x_i = (d_{i,r}'/R_r) * x_i$ comes out, with the contradiction of $d_{i,r} < d_{i,r}'$, and the proof is completed.

It indicates that POD cannot increase its allocation share by falsely reporting resource demand and cannot be deceptive in meeting resource demand. □

Theorem 3. *XDRF is free of jealousy*

Proof. assume that POD(i) is jealous of POD(j)'s resource quota; that is to say, POD(j)'s resource quota is larger than POD(i), and these resources are also needed by POD(i). If these resources are $r \in r_1, r_2, \ldots, r_m$, the following two situations should be considered:

(1) If r is the dominant resource of POD(i) and POD(j), then r can only be the same resource. According to the hypothesis $d_{i,k} < d_{j,k}$ and according to formula (17) $(d_{i,r}/R_r) * x_i = (d_{j,r}/R_r) * x_j$, then $x_{i>}x_j$, that is, by allocating more copies for POD(i) to balance its dominant resource, so the resource allocation of i will not be affected.

(2) If r is not the dominant resource of i but is relatively important for POD(i), POD(j) occupies more quotas, and if the dominant resource distribution of POD(i) and POD(j) is q and p, then there is the following relationship:

$(d_{j,p}/R_p) * x_j = (d_{i,q}/R_q) * x_i > (d_{j,r}/R_r) * x_j > (d_{i,r}/R_r) * x_i$, consider the following two scenarios simultaneously:

(a) if $(d_{j,p}/R_p) > (d_{i,q}/R_q)$, then $x_i > x_j$, and in order to satisfy the above relationship, the demand of POD(i) on r is far less than that of POD(j), that is, $d_{j,r} ? d_{i,r}$, and r is not an important resource of POD(i), which contradicts the hypothesis.

(b) if $(d_{j,p}/R_p) < (d_{i,q}/R_q)$, then $x_i < x_j$, and it can be obtained from the above relationship, $d_{j,r} \geq d_{i,r}$, which is the same as the case a), so the demand of POD(i) on r is less than or equal to that of POD(j), which is inconsistent with the hypothesis, and the proof is completed. □

Theorem 4. *XDRF is satisfy Pareto efficiency*

Proof. according to the definition of Pareto efficiency, it is assumed that POD can increase its quota without affecting the quota of other pods. According to the hypothesis, for POD(i), Pareto improvement exists to make it increase the resource share of POD(i) without affecting the share of other nodes. According to lemma (8) in literature [19], there is at least one saturated resource in POD using DRF. Suppose that the share

```
name: autoScalingUp
input: none
output: none
Define variable I: Expansion index; Define variable scalingStrategy: Capacity expansion policy: Non-predictive capacity expansion if
the value is 0, predictive capacity expansion if the value is not 0; Define a collection<POD>: POD collections that need to be
expanded; Define a collection<EXPOD>: A collection of pods with poor performance; Define a collection<POP>: POD optimization
solution set;
  Main-loop {
    retrieve XEDI context from CAT as XEDI.C; //Get the XEDI context from CAT
    //Calculate the average message processing time and average throughput of XEDI
    XEDI .T_avg(n) = V_avg(H(XDE I.T_ins,n)); XEDI .V_avg(n) = V_avg(H(XDE I.V_ins,n))
    if (XEDI .T_avg(n) > XEDI .T_max and XEDI .V_avg(n) < XEDI .V_max) {
      //POD nodes with normal throughput but abnormal message processing time
      for (step i from 1 to 3) {
        for (each XEDI pod in step i from K8S) {
          retrieve pod. C;
          // Calculate the average data processing time and average throughput of POD
          POD.T_avg(n) = V_avg(H(POD.T_ins,n)); POD.V_avg(n) = V_avg(H(POD.V_ins,n))
          if (POD.T_avg(n) > POD.T_max and POD.V_avg(n) < POD.V_max) {
            add this unhealthy pod to collection<EXPOD>;
          }}}
        report collection<EXPOD> to CAT as performance exception; //Report exception triggers to CAT
        enter next loop;
      }
      If (scalingStrategy = = 0) {
        //According to formulas (2) and (3), q_i = 1, the responsive capacity expansion index based
        //on mixed load rate is calculated
        I = V_avg(H(XEDI .TVinsR,n));
      } else {
        //According to formulas (5)–(7), the predictive expansion index was calculated
          I = V_p(H(XEDI .TVinsR,n));
      }
      if (I > I_max) {
        //Enter the expansion process and get the POD set of XEDI
        for (each XEDI pod in K8S) {
          retrieve pod. C;
          //Avoid frequent POD scaling by cooling-off time
          if (currentTime-pod. lastScalingTime < pod. C. T_c) {
            enter next loop;
          }
          //Calculate POD expansion index
          If (scalingStrategy = = 0) {
            //According to formulas (2) and (3), q_i = 1, the response expansion index based on
            //mixed load rate was calculated
            I_p = V_avg(H(POD.TVinsR,n));
          } else {
            //According to formulas (5)–(7), the predictive telescopic index is calculated
            I_p = V_p(H(POD.TVinsR,n));
          }
          //Make scaling decisions
          if (I_p > I_max) {//Calculate and update POD expansion metrics
            If (scalingStrategy = = 0) {
              //In response mode, q_i = 1 is calculated according to formulas (2) and (3) to calculate //data grouping processing
time and throughput,
              //Queue wait time and response metrics for CPU and memory utilization
              update pod context (
V_avg(H(POD.Tins,n)),V_avg(H(POD.Vins,n)),V_avg(H(POD.Qins,n)),V_avg(H(POD.MEMinsR,n)),V_avg(H(POD.CPUinsR,n)));
            } else {
              //Predictive mode, according to formulas (4)–(6), calculate data grouping //processing time, throughput, Queue
wait times and predictors of CPU and //memory utilization
              update pod context (
```

Algorithm 1: Continued.

```
V_{p(H(POD.Tins,n))}, V_{p(H(POD.Vins,n))}, V_{p(H(POD.Qins,n))}, V_{p(H(POD.MEMinsR,n))}, V_{p(H(POD.CPUinsR,n))},);
        }
        add this pod to collection<POD>;
    }}
    //Calculate configuration optimizations for POD collections that need to be scaled up
    for (each pod in collection<POD>) {
        //POD optimization scheme is calculated by queuing theory system
        compute PodOptimizationPlan(pop) for pod by queue system;
        add this pop to collection<POP>;
    }
    //Confirm whether K8S resources meet the expansion conditions
    if (R not adequate for collection<POP> scaling-up) {
    //When the available resources are used up, try to apply for resources from the container
    //cloud and preempt dynamically when the resources are insufficient try to apply resource
    //increment as R_{c1};
        if (R_{c1} == 0) {
            //Performance is abnormal and additional computing resources cannot be
            //requested from the container cloud report resource exhausted exception to
            //CAdvisor;
                enter next loop;
        }
    R_c = R_{c1}; R = R + R_c; //Update total resources
        }
        //Allocate resources for POD according to Algorithm2
        call XDRF algorithm for collection<POP> with Rc By XTuning. Scheduler;
    }
    If (I < I_min) {
        //Normal performance without expansion, according to the Algorithm 3 asynchronous
        //trigger shrinkage process
        asyn_invoke auto scaling-down with XEDI.C;
    }}
```

Algorithm 1: XEDI's automatic expansion algorithm.

of POD(I) in resource r is increased from $s_{i,r}$ to $s'_{i,r}$, where $s_{i,r} = d_{i,r} * x_i$. According to Theorem 2, POD(I) cannot increase $s_{i,r}$ by increasing $d_{i,r}$; therefore, POD(I) can only increase the quota of resource r by increasing x_i. It can be obtained from lemma (8) that I has at least one saturated resource w. Therefore, increasing xi cannot increase the share of w. Therefore, it contradicts the hypothesis that Pareto improvement does not exist, and the proof is completed.

The essence of XDRF meeting Pareto efficiency is the constraint on resource occupation by P. When resource allocation reaches saturation, POD cannot increase its share anymore, unless it occupies resources of other pods, whose behavior will be forbidden by XDRF.

The XEDI system is deployed with the help of InterThings, a virtual cluster environment of containers, and it is tested under the following two aspects: scaling effect and scaling velocity [20]. The former one refers to comparing frequencies of message processing using different automatic scaling algorithms, while the latter one refers to testing whether PODS can be effectively adjusted with the change of load [1]. □

7. Scalability Test of the Algorithm

7.1. Scaling Effect Test. The throughput limit and resource allocation algorithm efficiency of XEDI under different POD copy sizes were tested, among which the Takia adapter was configured into SYN mode; that is, the request-response was not conducted until the message conversion of the three steps was completed. The POD copy quota of the three steps tested was configured as <0.2c, 128M>, <0.4c, 128M>, <0.3c, 256M>, and the resource vectors were <1, 0, 0>, <0, 1, 1>, and <1, 0, 1>. In order to compare the capacity expansion effect, XEDI configured the three STEP copies of the test Topic into even capacity expansion mode (from 2 to 16 to expand capacity on 4 heterogeneous computing nodes), where Dell-R710 and Dell-R620, respectively, correspond to CPU and memory storage computing resources and used Mesos DRF and XDRF as XEDI POD allocation algorithms, respectively [21].

The VUser of LoadRunner adopts the trapezoid incremental graph until the HTTP-503 error appears in the response result. Thus, the response frequency of server requests, data throughput frequency, and maximum concurrent request number of XEDI under different replica configurations can be obtained, as shown in Table 2:

And the relationship of the data in Table 2 can be shown in Figures 1–4.

According to Figure 4, through XMON's monitoring of POD's comprehensive load rate, the overall load rate of XDRF is higher than that of Mesos' DRF algorithm during the POD distribution process, which indicates that the resources are

```
name: autoScalingDown
input: C: XEDI performance context;
output: none
  //If the XEDI resource occupancy rate is low, it will not shrink, reducing the number of
  //unnecessary shrinkages
  if (R_a < R * θ_max) { terminate scaling-down;}
  //Get all the POD sets for XEDI
  retrieve all pods of XEDI from K8S as collection<POD>;
  for (POD pod: collection<POD>) {
    retrieve pod. C;
    //Avoid frequent POD scaling by cooling-off time
    if (currentTime-pod. lastScalingTime < pod. C. T_c) {
      enter next loop;
    }
    /*Calculate POD expansion index*/
    If (scalingStrategy == 0) {
      //According to formulas (2) and (3), q_i = 1, the response capacity expansion index based
      //on mixed load rate was calculated
      I = V_avg(H(PO D.TVinsR,n));
    } else {
      //According to formulas (5)–(7), the predictive expansion index was calculated
      I = V_p(H(PO D.TVinsR,n));
    }
    //Adjust pods with lower load rates
    if ((I < I_min) {
      //Reduce the number of copies of POD according to the flex index
      pod. replicas = *I;
      //Refresh the POD's copy number configuration so that the POD's shrinkage takes effect
      refresh pod replication for K8S with XTuning. Scheduler;
    }
    //Stop shrinking when the resource utilization rate falls below the resource load rate
    if (R_a < R * θ_min) { terminate scaling-down;}
}
```

Algorithm 2: XEDI's automatic shrinkage algorithm.

better utilized overall. Combined with Figure 5 it can be also seen that XDRF algorithm and dynamic weighting and resource types match, and the more the urgent priority allocation, the more the reasonable resources, as well as the equilibrium between different node performances, so the two resources allocation performance is better than the default resource allocation algorithm as a whole.

7.2. Scaling Velocity Test

7.2.1. Comparison of Scaling Effects of Different Scaling Strategies. In the cloud deployment response scale and scale forecasting strategy, respectively, two cluster instances, Takia ferry mode is configured to ASYN enough throughput to ensure that the front end POD configuration is the same as the first step in the test, the initial replications to 1, and in the test phase of the load scenario, LoadRunner VUser adopts arch random graph, the two cluster instances at the same time to request access to 16 min to test the system's response to the load, including the expansion of the trigger and execution and time efficiency to solve this problem. According to the interface of the capacity enlargement algorithm, the capacity expansion threshold was set as 75% and the capacity reduction threshold as 45% [22]. The capacity expansion index adopted the time-throughput composite load rate, and the cooling time of capacity expansion was 2 min (note: in production environment, to avoid frequent capacity expansion caused by load fluctuation near the threshold, the value was generally more than 10 minutes). The test results are shown in Figure 6.

As can be seen from Figures 1 and 6, in the initial stage, the load rate is lower than 40%, and the total number of POD copies is specifically 3. At 3 min, the load increases sharply, and the server load rate rises rapidly to nearly 80%, higher than the capacity expansion threshold. With capacity expansion triggered, the number of POD copies increased to 8 at 4 min. After that, the load was reduced to 40%, less than the shrinkage threshold. Since it was in the "cooling-off time" stage, the shrinkage operation was not triggered, and the two expansion and shrinkage operations within a short period of time in this stage were prevented. At 5 min, the load returned to the rising trend, reached 75% at 7 min, and triggered the second capacity expansion operation. At 8 min and 9 min, respectively, the number of copies of the two expansion strategies increased to 12. At 11 min, the load was reduced, and the volume reduction operation was triggered when it was lower than the volume reduction threshold. The copies of the two capacity

Algorithm name: XDRFforPOD
/*The number of nodes is n, and the resource dimension is m^*/
Input: $R = <R_1 = <R_{1,1}$ to $R_{1,m}>, \ldots, R_k = <R_{k,1}$ to $R_{k,m}>, \ldots, R_n = <R_{n,1}$ to $R_{n,m}>>$: total resource collection; collection<POP>: POD optimization scheme collection;
Output: none
Define variable z = collection<POP>.size: the number of PODs to be calculated; Define variable $R_{1u} = Ru^1_{1,1}, \ldots, Ru^k_{i,j}, \ldots, Ru^n_{z,m}$: the set of allocated resources, $Ru^k_{i,j}$ represents the number of resources of type j that has been allocated by POD(i) on node k; Define variable $R_c = Rc_{1,1}, \ldots, Rc_{k,j}, \ldots, Rc_{n,m}$: unallocated resource set, $Rc_{k,j}$ represents the number of resources of type j on node k that can be allocated; Define variable $W = W_1$toW_z: The weight set of POD to be optimized;

```
for (i from 1 to z) {
    Calculate the weight of the POD in the collection<POP> according to formula (10) and fill the collection W;
}
for (k from 1 to n) {
    Calculate the cluster nodes η_k according to formula (10) and arrange them in ascending order;
}
do {
    for (i from 1 to z) {
    For R and Ru sets, calculate the dominant share S_i of POD(i) according to formulas (11) and (12), and update the collection<POP> collection, sorted by S_i in ascending order;
}
//Get the POD with the smallest dominant share (i)
picking POP(i), the first element of collection<POP>;
POD(i) = POP(i).POD;
//Get POD(i) resource requirements such as CPU and memory
calculate resource demand of POD(i) as D_i;
Calculate the resource Predicates set Npre(i) of POD(i) according to formula (14);
if (Ru + D_j ≤ R) {
    According to formulas (13) or (14), a copy resource r is allocated to POD(i), where r = = D_i;
    //Load and run the copy instance
    let replication as result of loading and running POD(i).replicationConfig with r;
    //Register the copy, monitor the data queue and participate in data processing services
    register this replication as consumer to kafka with POP(i).topic;
    //Update resource usage
    Ru+ = D_j; Rc− = D_i;
    //Refresh the dominant share of POD(i) according to formulas (12) and (13)
    refresh dominant share for POD(i);
    if (POP(i).dPR-- == 0) {
        //The POD has been expanded and deleted from collection<POP>, and no longer enters //the subsequent allocation process
        POD(i) scaling-up done;
    }}
//The cluster node resources are exhausted, record the POD information that has not been
//allocated and exit DRF
else {
    get unsatisfied POPs as collection<UPOP> from collection<POP>;
    report collection<UPOP> to CAdvisor;
    terminate XDRF;
}
/ When collection<POP> is empty, all POD allocation is completed
If (allocation done for all pod in collection<POP>) {
    report to XTuning allocation done with R and collection<POP>;
    terminate XDRF;
}} while (true)
```

Algorithm 3: XDRF algorithm.

expansion strategies were reduced to 6 and 8, respectively, and the volume expansion was not carried out in the following 2 min cooldown. At 16 min, the volume reduction operation was triggered by load drop, and the number of copies was reduced to 3, which verified the effectiveness of XEDI dynamic capacity expansion. It can also be seen from the figure above that, compared with the response capacity expansion strategy, the predictive capacity expansion strategy is more active than the response strategy, because it can predict the load status of the subsequent time series in advance. Therefore, the capacity expansion preparation can be carried out before the load expansion, so as to obtain better system processing performance and throughput [23].

Table 2: Comparison of pressure test results before and after replica expansion.

Number of copies	Index					
	Request response frequency (fetches/sec)		Data throughput frequency (bytes/sec)		Maximum number of concurrent requests	
	DRF	XDRF	DRF	XDRF	DRF	XDRF
2	0.1624	0.1645	32.5	33.4	60	64
4	0.3056	0.3742	65.4	66.8	113	137
6	0.4475	0.5238	85.2	87.1	152	174
8	0.5834	0.6925	112.4	114.5	184	206
10	0.7423	0.8292	121.7	124.3	216	268
12	0.7893	0.9482	137.6	139.7	262	338
14	0.8621	1.0179	142.8	145.2	287	377
16	1.0016	1.1382	146.9	150.4	321	406

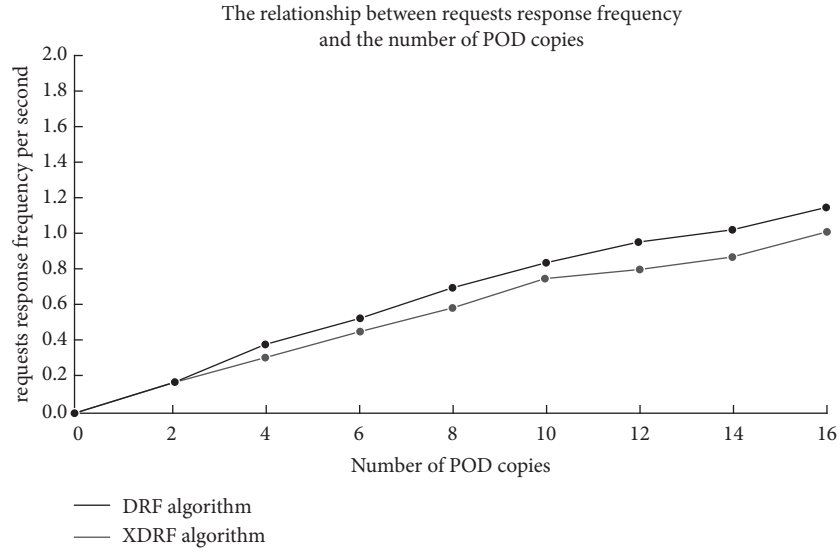

Figure 2: The relationship between XEDI request-response frequency and the number of POD copies.

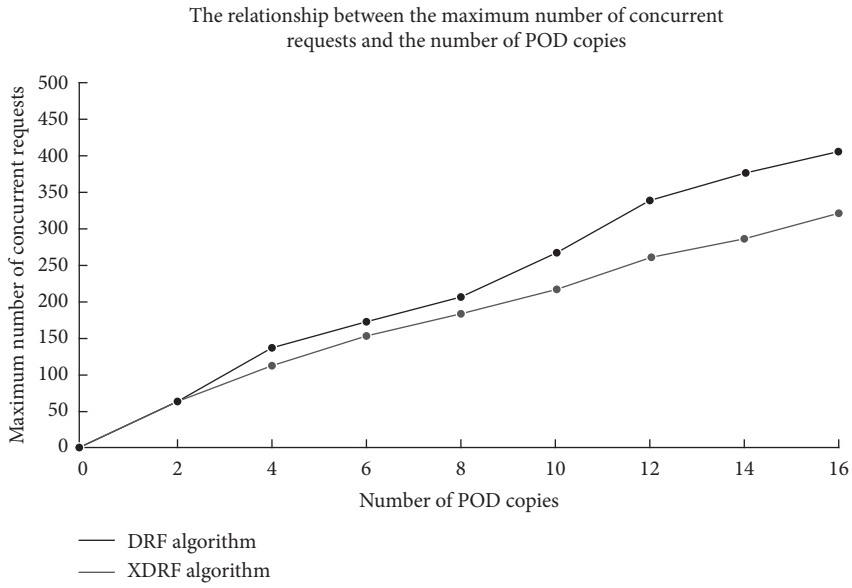

Figure 3: The relationship between the maximum number of XEDI concurrent requests and the number of POD copies.

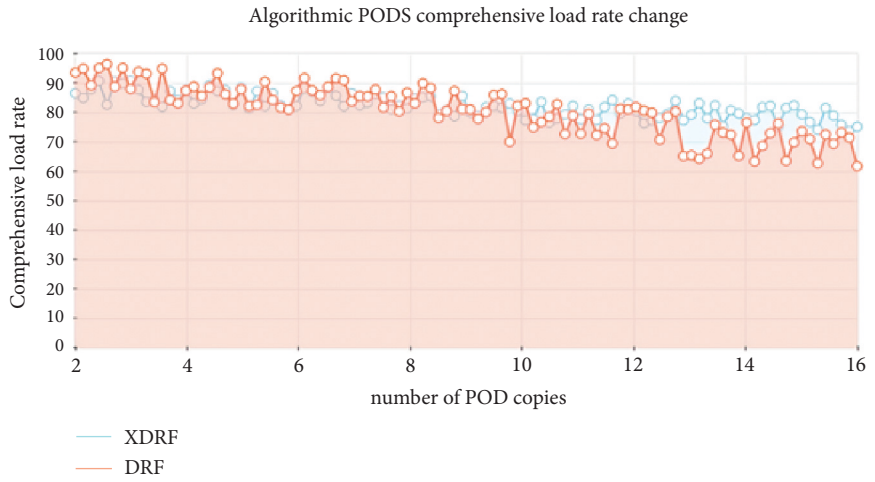

Figure 4: XEDI-PODS CPU-RAM comprehensive load rate change trend.

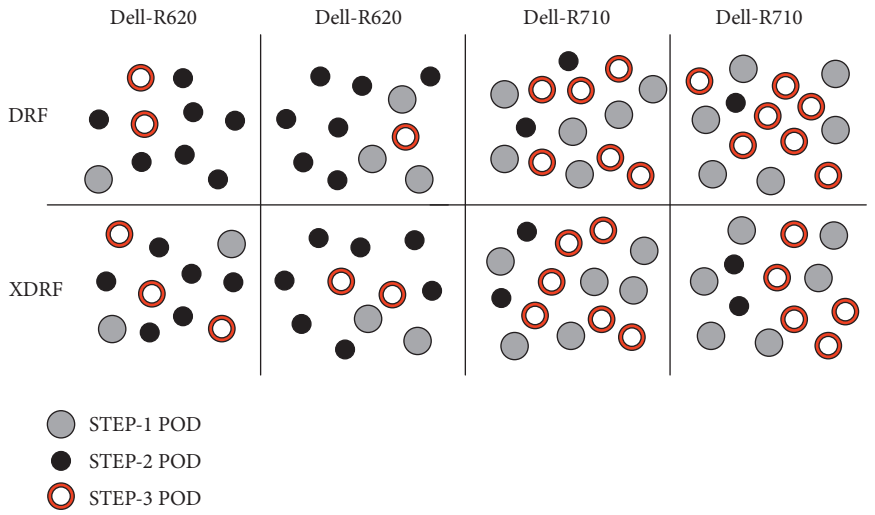

Figure 5: Resource allocation results of different resource allocation algorithms under the same load.

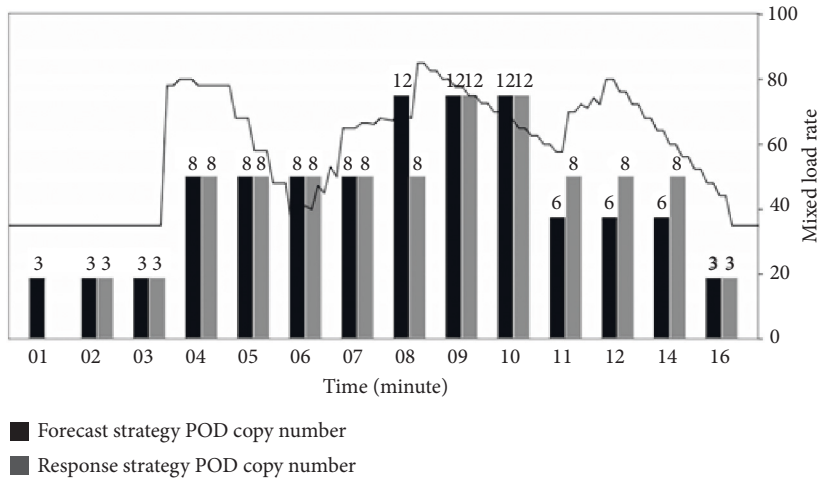

Figure 6: Comparison diagram of automatic capacity expansion effect of XEDI under different strategies.

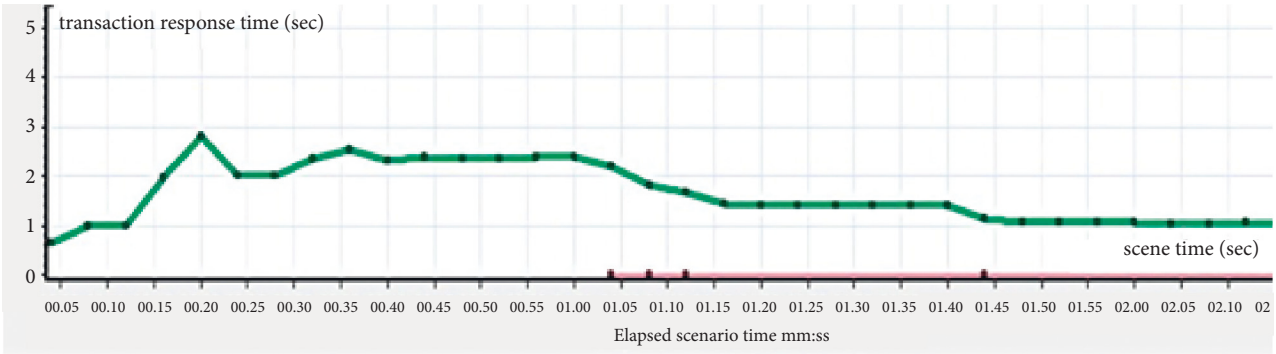

Figure 7: Response spectrum of automatic expansion system.

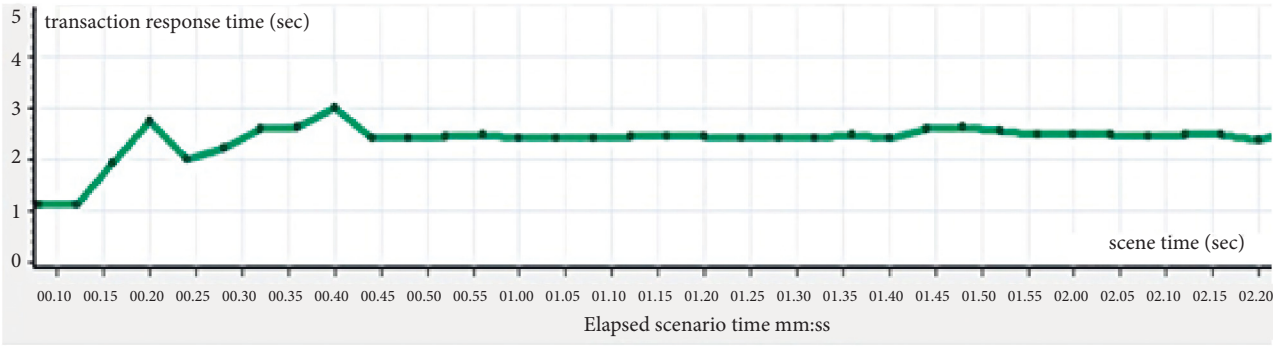

Figure 8: Response spectrum of nonexpandable system.

7.2.2. Performance Comparison between Closed and Open Scaling Strategy. Based on the above test scenarios, and further comparison does not have scale characteristics of the traditional "stovepipe" through information sharing system with elastic performance difference between unit ITIU information sharing, namely, validation expansion module performance improvement effect of information sharing service, we will have the response type expansion cluster instance XTuning closed, as well as the expansion and test instance and the expansion of the client's response performance. Using the same server configuration, set the front module to the SYN mode, and at the same time, set the LoadRunner VUser map of 300 concurrent users trapezoidal map; the threshold arrival time is 50 sec, cycle for 3 min, and do not test points recording two-cluster-instance transaction response time, and the test results are shown in Figures 7 and 8; the *X*-axis is time, the vertical axis for the transaction response time.

By comparing the two figures, it can be found that the response time of the two cluster instances is basically the same in the early stage, and the system throughput of the server load reaches the threshold at about 50 sec. In Figure 7, as the capacity expansion scenario starts to expand, the message response time decreases to about 1 sec after the capacity expansion. In Figure 8, as the cluster instance shuts down the capacity expansion component, the response time of the system after stabilization remains around 2.5 sec. It can be seen that the automatic capacity expansion system can effectively maintain the service performance of the client when the system load increases.

In order to compare the performance difference of XEDI components in the container environment and the virtual machine environment of the current mainstream cloud platform, two XEDI cluster instances that respond to the scaling mode are deployed in the container and virtual machine environments [24, 25]. The Takia adapter is configured in SYN mode, the POD configuration is the same as the first test, the initial number of copies is 1, the node in the virtual machine mode also uses the same configuration, and the initial number of nodes is also 1. LoadRunner's VUser map is a ladder map of 200 concurrent users, with a period of 2 minutes. Record the transaction response time of the two test cluster instances separately to evaluate the response level of the container and virtual machine scaling to the load under the same configuration. The test results are shown in Figures 9 and 10.

As can be seen from the above figure, the average response time of the container environment is about 1 sec, which is significantly better than the virtual machine environment. In addition, because the container is a lightweight process-level service, the refresh time of the POD copy only takes about 5 sec, so Figure 9 can quickly complete the expansion operation in the early stage of the load and reduce the transaction response time to less than 1 sec. The virtual machine startup and deployment time is an operating system level operation. As can be seen from Figure 10, the transaction response time in the virtual machine mode increases as the load reaches about 90 sec before completing the first expansion operation. It can be seen that, in terms of

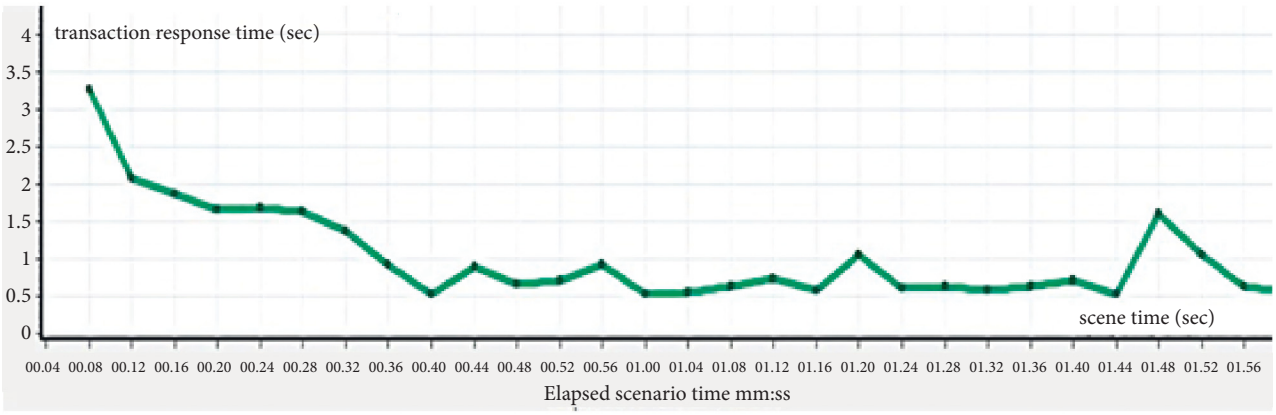

Figure 9: Container mode expansion deployment response time.

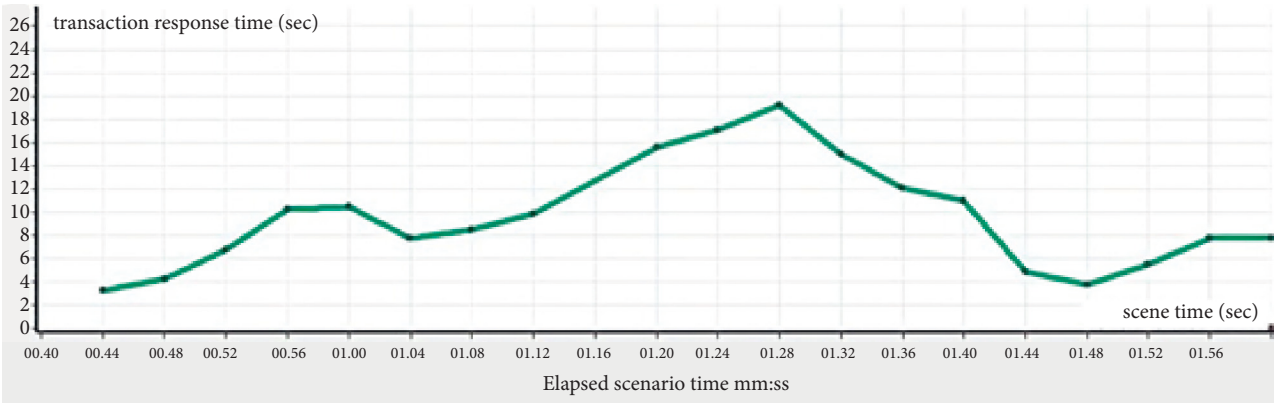

Figure 10: Response time of expansion deployment in virtual machine mode.

scalability and agility, container clusters have obvious advantages over virtual machines.

8. Conclusions

In this paper, we have proposed the autoscheduling algorithm XDRF in the cloud environment. This paper incorporates a detailed evaluation of the XEDI stretching model toward the workloads of CPU and RAM. Through quantitative experiments, it was verified that the XDRF algorithm could achieve the system performance optimization on the basis of guaranteeing system reliability and reduce energy consumption effectively [26]. The work in this paper also has clarified that the model can meet the demand of dynamic load and improve the service quality according to the two tests.

9. Prospect

9.1. Standardization of Cloud Platform for Combined Iron and Water Transport. Cloud computing is an effective way to optimize the existing intermodal information layout and application management model, and it also brings new challenges to intermodal business and data standards under the cloud environment. Although the intermodal cloud platform adopts a centralized management model, it is difficult to integrate a large number of heterogeneous intermodal applications on a unified cloud platform without a unified intermodal information standard. Although simple migration can achieve unified management of applications, it cannot effectively use virtual resources to optimize cloud service models. Therefore, researching the intermodal information standards that adapt to the cloud environment is crucial to the landing application of intermodal cloud platforms.

9.2. Construction of Intermodal Blockchain. Combined transportation of iron and water is a multiparty collaborative business process, and the security and traceability of information sharing are extremely important. Blockchain is the latest information sharing and storage technology. It can not only effectively simplify the intermodal business process, but also effectively protect the security of shared data. How to combine blockchain with intermodal information technology, build intermodal blockchain, and realize intermodal smart contracts and data traceability is also of great significance and requires a lot of follow-up research work.

Data Availability

Data used to support the finding of this study are available within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by Sichuan Agricultural University education reform project (X2013039 and X2014025) "Agricultural Information Engineering" Sichuan key laboratory of higher education; the National Key Research and Development Program fund (2017 yfb1200702 and 2016 yfc0802208); the National Nature Foundation Project (61703351); the Science and Technology Research Project of China Railway Corporation (P2018T001); and the Science and Technology Plan Project of Sichuan Province (2018RZ0078 and 2019JDR0211).

References

[1] Q. Lei, W. Liao, Y. Jiang, M. Yang, and H. Li, "Performance and scalability testing strategy based on kubemark," in *Proceedings of the IEEE 4th International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*, pp. 511–516, IEEE, Chengdu, China, 2019, April.

[2] C. C. Lo, K. M. Chao, H. Y. Kung, C. H. Chen, and M. Chang, "Information management and applications of intelligent transportation system," *Information Management and Applications of Intelligent Transportation System*, vol. 2015, Article ID 613940, 2 pages, 2015.

[3] T. Debicki and A. Kolinski, "Influence of EDI approach for complexity of information flow in global supply chains," *Business Logistics in Modern Management*, vol. 18, 2018.

[4] J. Betz, E. Jaskolska, M. Foltynski, and T. Debicki, "The impact of communication platforms and information exchange technologies on the integration of the intermodal supply chain," in *Integration of Information Flow for Greening Supply Chain Management*, pp. 131–141, Springer, Berlin, Germany, 2020.

[5] L. Ding, "Multimodal transport information sharing platform with mixed time window constraints based on big data," *Journal of Cloud Computing*, vol. 9, no. 1, pp. 1–11, 2020.

[6] S. Liu, C. Yin, D. Chen, H. Lv, and Q. Zhang, "Cascading failure in multiple critical infrastructure interdependent networks of syncretic railway system," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[7] T. Yang, X. Peng, D. Chen, F. Yang, and M. Muneeb Abid, "Research on trans-region integrated traffic emergency dispatching technology based on multi-agent," *Journal of Intelligent and Fuzzy Systems*, vol. 38, no. 5, pp. 5763–5774, 2020.

[8] S. Taherizadeh and V. Stankovski, "Dynamic multi-level autoscaling rules for containerized applications," *The Computer Journal*, vol. 62, no. 2, pp. 174–197, 2019.

[9] L. Li, "Energy consumption management of virtual cloud computing platform," in *Proceedings of the IOP Conference Series: Earth and Environmental Science*, vol. 94, no. 1, Article ID 012193, 2017.

[10] D. Chen, S. Ni, C. A. Xu, and X. Jiang, "Optimizing the draft passenger train timetable based on node importance in a railway network," *Transportation Letters*, vol. 11, no. 1, pp. 20–32, 2019.

[11] A. Sun, T. Ji, Q. Yue, and F. Xiong, "IaaS public cloud computing platform scheduling model and optimization analysis," *International Journal of Communications, Network and System Sciences*, vol. 4, no. 12, pp. 803–811, 2011.

[12] T.-T. Nguyen, Y.-J. Yeom, T. Kim, D.-H. Park, and S. Kim, "Horizontal pod autoscaling in Kubernetes for elastic container orchestration," *Sensors*, vol. 20, no. 16, p. 4621, 2020.

[13] Z. Cao, *Research on Dynamic Scaling of Web Application Deployment in Cloud Platform*, Doctoral dissertation, Fudan University, Shanghai, China, 2012.

[14] A. Chandra, W. Gong, and P. Shenoy, "Dynamic resource allocation for shared data centers using online measurements," in *International Workshop on Quality of Service*, Springer, Berlin, Germany, 2003.

[15] J. Bi, Z. Zhu, R. Tian, and Q. Wang, "Dynamic provisioning modeling for virtualized multi-tier applications in cloud data center," in *Proceedings of the IEEE 3rd International Conference on Cloud Computing*, pp. 370–377, IEEE, Miami, FL, USA, 2010, July.

[16] W. Ou and Q. Hu, "Modeling and simulation of CIMS logistics dispatching system," *Computer Integrated Manufacturing Systems*, vol. 10, no. 9, pp. 1067–1072, 2004.

[17] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant resource fairness: fair allocation of multiple resource types," in *Proceedings of the 8th USENIX conference on Networked systems design and implementation*, vol. 11, p. 24, Massachusetts, MA, USA, 2011, March.

[18] H. W. Wang, C. H. Chen, D. Y. Cheng, C. H. Lin, and C. C. Lo, "A real-time pothole detection approach for intelligent transportation system," *Mathematical Problems in Engineering*, vol. 2015, Article ID 869627, 7 pages, 2015.

[19] W. Pan, Q. L. Zhong, X. D. Fu, and Z. Y. Yu, "Design and implement of workflow engine based on spring," *Journal of Northeast Normal University (Natural Science Edition)*, vol. 3, 2007.

[20] Q. Zhang, L. Liu, C. Pu, Q. Dou, L. Wu, and W. Zhou, "A comparative study of containers and virtual machines in big data environment," in *Proceedings of the IEEE 11th International Conference on Cloud Computing (CLOUD)*, pp. 178–185, IEEE, San Francisco, CA, USA, 2018, July.

[21] Y. Al-Dhuraibi, F. Paraiso, N. Djarallah, and P. Merle, "Elasticity in cloud computing: state of the art and research challenges," *IEEE Transactions on Services Computing*, vol. 11, no. 2, pp. 430–447, 2017.

[22] J. Zhao, X. Zhu, and L. Wang, "Study on scheme of outbound railway container organization in rail-water intermodal transportation," *Sustainability*, vol. 12, no. 4, 2020.

[23] K. A. Kuzmicz and E. Pesch, "Approaches to empty container repositioning problems in the context of Eurasian intermodal transportation," *Omega*, vol. 85, pp. 194–213, 2019.

[24] L. Knapcíková and P. Kaščák, "Sustainable multimodal and combined transport in the European Union," *Acta Logistica*, vol. 6, no. 4, pp. 165–170, 2019.

[25] J. Ližbetin, "Methodology for determining the location of intermodal transport terminals for the development of sustainable transport systems: a case study from Slovakia," *Sustainability*, vol. 11, no. 5, 2019.

[26] J.-S. Pan, N. Liu, S.-C. Chu, and T. Lai, "An efficient surrogate-assisted hybrid optimization algorithm for expensive optimization problems," *Information Sciences*, vol. 561, pp. 304–325, 2021.