electronics

Article

Optimizing Network Service Continuity with Quality-Driven Resource Migration

Chaofan Chen [1,2,†], Yubo Song [1,2,*,†], Yu Jiang [1,2,†] and Mingming Zhang [3,†]

1 School of Cyber Science and Engineering, Southeast University, Nanjing 211151, China; 220215466@seu.edu.cn (C.C.); jiangyu@seu.edu.cn (Y.J.)
2 Purple Mountain Laboratories, Nanjing 211189, China
3 State Grid Jiangsu Electric Power Co., Ltd., Information & Telecommunication Branch, Nanjing 210024, China; zhangmms@hotmail.com
* Correspondence: songyubo@seu.edu.cn
† These authors contributed equally to this work.

Abstract: Despite advances in security technology, it is impractical to entirely prevent intrusion threats. Consequently, developing effective service migration strategies is crucial to maintaining the continuity of network services. Current service migration strategies initiate the migration process only upon detecting a loss of service functionality in the nodes, which increases the risk of service interruptions. Moreover, the migration decision-making process has not adequately accounted for the alignment between tasks and node resources, thereby amplifying the risk of system overload. To address these shortcomings, we introduce a Quality-Driven Resource Migration Strategy (QD-RMS). Specifically, QD-RMS initiates the migration process at an opportune moment, determined through an analysis of service quality. Subsequently, it employs a method combining Pareto optimality and the simulated annealing algorithm to identify the node most suitable for migration. This approach not only guarantees seamless service continuity but also ensures optimal resource distribution and load balancing. The experiments demonstrate that, in comparison with conventional migration strategies, QD-RMS achieves superior service quality and an approximate 20% increase in maximum task capacity. This substantiates the strategic superiority and technological advancement of the proposed strategy.

Keywords: service continuity; resource migration; Pareto optimality; load balancing

check for updates

Citation: Chen, C.; Song, Y.; Jiang, Y.; Zhang, M. Optimizing Network Service Continuity with Quality-Driven Resource Migration. *Electronics* **2024**, *13*, 1666. https://doi.org/10.3390/electronics13091666

Academic Editor: Christos J. Bouras

Received: 11 April 2024
Revised: 20 April 2024
Accepted: 22 April 2024
Published: 25 April 2024

1. Introduction

As networking technologies rapidly advance, significant enhancements in network security have strengthened the defenses of network infrastructure against malicious intrusions. Nonetheless, the dynamic nature of cyber threats makes the complete evasion of intrusion risks impractical [1]. Even the most robust security measures may fall short in fully deterring hackers who employ increasingly complex tactics. The "WannaCry" ransomware incident serves as a pertinent example; it exploited system vulnerabilities, affecting vital services globally like hospitals and banks, and resulted in extensive service disruptions and data encryption despite stringent security. Hence, the advent of intrusion tolerance technology is pivotal, designed not merely to counter attacks but, more crucially, to ensure the ongoing operation of essential services during cyber incidents. This is vital for users, as the continuity and reliability of services profoundly impact their perception of the network system's performance [2,3]. In the realm of cybersecurity challenges, maintaining service continuity is central to users' assessment of network service quality.

Thus, the development of effective service migration strategies has become a subject of significant interest within the intrusion tolerance domain [4], garnering substantial attention and investigation from scholars worldwide. Wang et al. [5] proposed a polymorphic heterogeneous security architecture, which is grounded in an intrusion tolerance mechanism. This architecture permits the migration of services from one executor to another in the event of a system attack. Such a migration mechanism is instrumental in preserving system continuity and reliability, thereby bolstering the system's capabilities to tolerate network attacks. The efficacy of this method in delivering a high recovery rate has been validated through experiments, which also highlighted its advantages of quicker execution and reduced costs in link mapping. Kang et al. [6] introduced a primary-backup Virtual Network Function (VNF) placement model, predicated on an availability schedule. This model treats the service migration challenge as an integer linear programming problem, aiming primarily to optimize the least count of continuous available time slots across all service function chains. This objective inherently seeks to mitigate service disruptions attributable to node unavailability. Zhao et al. [7] introduced a Fault-Tolerant Redundant Path Service Migration (FRSM) approach, utilizing a sliding window-based model for pinpointing potential service migration paths. This method integrates re-submission and replication mechanisms to enhance fault tolerance. By opting for several redundant paths, it devises an edge service migration strategy, significantly bolstering the migration's reliability and efficiency. Ma et al. [8] introduced a Predictive, Reliability-Assured Service Migration Path Selection Method (PTSM), which integrates user movement trajectory prediction with the reliability needs of service migration to optimize migration paths. This integration aims to guarantee uninterrupted service and optimal performance. The PTSM method is particularly effective in minimizing service disruptions and upholding service quality.

In the domain of research and implementation concerning service migration strategies, two main obstacles are encountered. First, the majority of existing strategies are only triggered in the event of a node losing its service provision capability or when a specific metric surpasses a preset static threshold. This approach fails to adequately predict or rectify potential decreases in service quality, resulting in prolonged service interruptions and a notable degradation in user experience. Second, current research predominantly focuses on evaluating the volume of task loads on nodes to inform migration decisions, often neglecting the compatibility between tasks and node resources during the migration process. As a consequence, services may be moved to nodes that are close to or have exceeded their capacity, thereby increasing the likelihood of system overloads.

This article presents the Quality-Driven Resource Migration Strategy (QD-RMS), which focuses on a migration timing decision approach influenced by service quality and a node selection algorithm that prioritizes load balancing. QD-RMS utilizes a comprehensive Service Migration Index (SMI) for the accurate assessment of service migration requirements. By employing SMI for evaluation, QD-RMS strategically initiates migration to ensure uninterrupted service continuity. In terms of node selection after determining migration timing, QD-RMS combines Pareto optimality theory with a simulated annealing algorithm to select the most suitable node for migration. This methodology not only facilitates effective service migration but also optimizes resource allocation and achieves load balance, resulting in improved task support after migration and enhanced overall system performance. The contributions of this paper are as follows.

1. Quality-guided migration timing decision: QD-RMS introduces a Service Migration Index (SMI) for the holistic evaluation of node service quality, aiding in the prediction of migration propensities. This index amalgamates crucial indicators like the average distance between users and base stations, service throughput, and node resource utilization. Continual SMI monitoring allows QD-RMS to proactively discern the necessity for service migration, facilitating an immediate assessment of service status. Upon the SMI hitting or surpassing a dynamic threshold, the system interprets this as an indicator of deteriorating service quality, thereby initiating the migration process and adaptively recalibrating the threshold to align with current node conditions. If the SMI remains below this threshold, the service shifts into a dormant state, pending subsequent evaluation. This approach preemptively triggers migration to

avert significant service quality decline, effectively stabilizing service quality levels and curtailing both service downtime and the duration required for migration.

2. Optimize the load through Pareto optimality theory: QD-RMS starts by merging node resources and task loads to evaluate the load status of potential migration nodes, leading to the development of a multi-objective optimization model. This model, utilizing Pareto optimality theory, gauges the load quality of these nodes and is efficiently resolved through the simulated annealing algorithm, facilitating the identification of superior candidate nodes promptly. Distinct from conventional load balancing methods, QD-RMS accentuates the compatibility of resources between tasks and nodes, thereby circumventing the selection of nodes at the brink of resource saturation, minimizing overload risks, and fostering a more sophisticated load-balancing regime.
3. Effective experimental validation: Comprehensive empirical tests were carried out on QD-RMS to assess its effectiveness in real-world applications, particularly in terms of service quality and maximum task capacity at nodes. By simulating four different service requests on a virtual machine platform, we compared the performance of QD-RMS with that of conventional reference strategies in service migration, illustrating their service quality in comparative graphs. The results highlighted QD-RMS's significant advantage in maintaining service quality, evidenced by more stable service performance and smaller fluctuation ranges. Additionally, the load-balancing performance of QD-RMS was evaluated, with experimental data confirming that, under similar hardware configurations, QD-RMS improved the maximum task capacity by approximately 20% compared to traditional load-balancing algorithms.

2. Related Work

Service migration, driven by user mobility and the limited nature of server resources, is an unavoidable component within the realm of mobile intrusion tolerance. The selection of appropriate timing for service migration is crucial, as it directly influences the duration of service interruptions. Notably, the length of these interruptions significantly affects the user's experience of the service.

Traditionally, the decision to migrate services is triggered by monitoring the utilization rate of a system's static resources, initiating migration when this rate meets a predefined threshold [9]. Kulshrestha et al. [9] have refined this approach by introducing a three-parameter Exponential Weighted Moving Average (EWMA3) model, aimed at detecting overload situations in hosts within cloud data centers. Conversely, Yang et al. [10] developed a resource transaction model for determining the optimal timing for migration, albeit with the limitation of considering only bandwidth as the measure for migration thresholds, thus narrowing its applicability. These studies primarily focused on the static utilization rate of server resources. However, in distributed systems with constrained resources and stringent Quality of Service (QoS) demands, such as Mobile Edge Computing (MEC), relying solely on static thresholds for service migration is deemed inappropriate [11].

Beyond server load considerations, the dynamic nature of user mobility and request patterns also plays a pivotal role in service migration strategies. X. Chen et al. [12] emphasized the real-time monitoring of network conditions and user demands to dynamically initiate service migration, aiming to minimize latency and enhance service continuity. E. Bozkaya et al. [13] leveraged digital twin models to simulate user mobility and the MEC environment, facilitating dynamic predictions of service migration needs and timing. H. Wang et al. [14] adopted deep reinforcement learning to optimize the timing of service migration with a focus on reducing user access delays and network costs. This approach dynamically adapts to user mobility patterns and network status changes. Z. Liang [15] et al. explored user mobility support within the mobile edge computing context, employing an optimization-based method to dynamically determine the optimal timing for service migration based on user mobility trajectories and edge network resource status. MoDEMS [16] predicts user future locations and mobility patterns by analyzing mobility data, enabling the system to make proactive migration decisions to ensure service continuity and quality as users move to new geographical locations.

This body of research on service migration timing highlights the challenges posed by the high heterogeneity of mobile edge networks and the specificity of applications, making rapid service migration without compromising user experience a challenging endeavor.

In the domain of intrusion tolerance, service migration trust and security garner significant attention. The proximity of edge nodes to end-users in mobile intrusion-tolerant networks, coupled with the limited computational and storage resources available at edge servers, introduces higher security risks compared to traditional cloud computing. Edge nodes are vulnerable to network attacks, and the control of a service node by malicious attackers can lead to severe security threats such as data theft and privacy breaches. Yi Xu et al. [17] employed three-way decision theory in service migration strategies, enhancing decision-making flexibility and predictability by considering the necessity and timing of migration, thereby avoiding unreliable migration nodes. Le et al. [18] developed a trust score-based algorithm for migration node selection, combining EigenTrust and RLIoT reputation systems with dynamic distance algorithms to provide credible choices. Chunlin Li et al. [19] demonstrated how SDN architecture supports granular network management and security policy enforcement, offering real-time insights into network traffic and node behavior to indirectly improve trust management in migration nodes. An Du et al. [20] applied reinforcement learning to dynamically adjust service deployment in response to user mobility, balancing the trade-off between service migration response times and quality maintenance. PreGAN [21] aims to predict edge nodes likely to fail, facilitating preemptive service migration to minimize security vulnerabilities and data loss risks.

In addition to the "trust and security" concerns at the terminals and server levels, ensuring the "capability security" within edge networks emerges as a critical aspect. The inherently uneven distribution of tasks across the MEC system, driven by user mobility and task randomness, underscores the necessity for load balancing on edge network servers. This balancing act not only safeguards the servers' "capability security" but also forms a foundational element of service migration's overall security framework. Through the dispersion of task processing across devices, X. Dai et al. [22] have effectively alleviated the burden on edge servers, facilitating a more equitable allocation of resources. Moon et al. [23] introduced a novel approach by advocating for the subdivision of tasks into multiple subtasks for parallel processing across several servers. This strategy, aimed at load equilibrium, entails reallocating subtasks from overloaded to under- or adequately-loaded MECs, based on prevailing load disparities. Addressing the challenge of load balancing within the MEC from a novel angle, Zhao et al. [24] conceptualized it as a dual objective of minimizing energy consumption and reducing queue redundancy, subsequently applying the Lyapunov optimization technique as a resolution mechanism. Similarly, Liu et al. [25] ventured into the domain of game theory for load-balancing solutions, initially framing the issue as a population game model before introducing two evolutionary dynamics and protocol modification-based algorithms for load equilibrium. Song S and colleagues [26] refined the task offloading process to diverse edge nodes, ensuring cost-efficiency while averting the risk of overloading individual nodes, thereby promoting a harmonious distribution of workload.

The realm of intrusion-tolerant service migration research is rapidly expanding, yet it is not devoid of limitations in existing studies. Predominantly, the exploration into the timing of service migration has been marked by a narrow focus on singular factors, such as service node overload scenarios or the proximity between users and service nodes, neglecting the multifaceted nature of service quality indicators. Furthermore, the discourse on the security of service nodes within intrusion-tolerant services remains comparatively underexplored, especially against the backdrop of terminal security, with the applicability and scope of trust models in evaluating service nodes presenting certain constraints. Drawing from these observations, the discourse on intrusion-tolerant service migration has garnered extensive attention and undergone significant evolution in recent years. Nonetheless, there remains a pressing need for deeper inquiries into aspects of migration timing and node security. Anchored in the objectives of enhancing security and efficiency within intrusion-tolerant service migration, this paper aspires to align with and contribute to the vanguard of ongoing research trajectories in this field.

3. Preliminaries

3.1. Service Migration

Service migration refers to the process of transferring services from the original server to a new server within a different geographical region when users move beyond the service coverage or when server resources are insufficient to support the services. The process is shown in Figure 1. The goal of service migration is to ensure the uninterrupted operation of computing services for users during their movement. To achieve this, the following points must be addressed:

1. Service continuity: If a user moves to another location, the services associated with that user must remain available.
2. Service mobility: Services must be capable of being migrated to new servers when needed.
3. User state mobility: The data associated with user services must be moved along with the services.

The transfer of user context information is crucial during the service migration process, as it determines whether users can seamlessly continue using the current services. Context can be categorized into the following four types:

1. System context: This includes any information related to computing and communication systems.
2. User context: This refers to any context information associated with user characteristics.
3. Environmental context: This encompasses any context information related to the physical environment, excluding system and user context.
4. Temporal context: This defines any context information related to time.

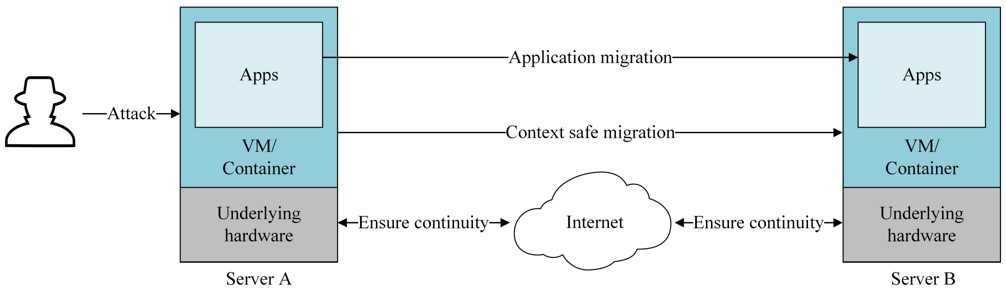

Figure 1. Schematic diagram of data transfer in service migration.

Furthermore, service migration presents security-related challenges, as different enterprises operate servers. If data are sent from a trusted source server to a destination server that is in a compromised state, it may potentially give rise to security issues. Hence, the security of the servers should also be taken into consideration during the service migration process.

3.2. Load Balancing

Load balancing is a cost-effective mechanism for expanding network equipment bandwidth, enhancing data throughput and processing capabilities, and bolstering network robustness, all without the need for additional hardware investment. Its central principle involves evenly distributing tasks across all servers to ensure the efficient operation of all devices. As a key apparatus for achieving this goal, load balancers disperse inbound requests across multiple servers, maintaining balanced loads and preventing network bottlenecks and single points of failure. Additionally, load balancing facilitates automatic failover, fault tolerance, and redundancy backup to ensure continuous system operation in emergencies. Common load balancing algorithms include the following:

1. Random Algorithm: Random algorithms distribute incoming requests by selecting servers at random. These methods do not account for the server's load or response metrics, relying solely on random selection.
2. Round Robin Algorithm: Requests are distributed to servers in a sequential, rotating order. Despite its simplicity, it does not account for performance differences between servers, which can lead to inefficient resource allocation.
3. Least Connection Algorithm: Requests are assigned based on the current number of connections to each server, aiming for balanced task distribution. However, this can still result in unbalanced resource utilization when server performance varies significantly.

3.3. Pareto Optimality

Pareto optimality, introduced by Italian economist Vilfredo Pareto, is a seminal concept in welfare economics and is predominantly applied in the context of multi-objective optimization problems. The core of Pareto optimality is to find a solution that facilitates the optimal trade-off among various conflicting objectives, such as cost, efficiency, and environmental impact. The set of solutions that achieves this optimality forms the "Pareto frontier" or the "Pareto boundary". At any point along this frontier, enhancing one objective necessitates compromising another.

This concept is grounded in the principle of Pareto efficiency. This principle posits that in a system with two objectives, labeled A and B, and under conditions of fixed total resources without the possibility of additional investments, enhancing one objective invariably leads to the diminution of the other. Pareto efficiency is the state in which no further improvements can be made to any objective without diminishing the others. In practical terms, Pareto optimality is achieved by formulating and adhering to specific constraints. Decision-makers can delineate the objective function and constraints to explore all viable solutions within the solution space, thereby identifying the set of Pareto optimal solutions.

4. Methodology

4.1. Active Migration Timing Based on Service Quality

In recent years, within the field of intrusion tolerance, studies have proposed the concept of "active migration" in service migration, as opposed to the earlier default "passive migration" (where migration occurs only when a service node is unable to continue providing service due to certain factors or when a user completely exits the coverage area of the service node). The comparative analysis of active versus passive migration is presented in Table 1. Active migration ensures a better service experience. Active migration refers to the process during which, to safeguard user service experience, decisions on service migration are made based on operational factors such as the load on service nodes, the distance between users and service nodes, service quality, etc. This occurs while the service node remains in normal operation; thus, it is termed as active migration.

Table 1. Comparison between active migration and passive migration.

	Active Migration	**Passive Migration**
Service node status during migration	Running	Suspended
Factors affecting the timing of migration	Quality of users and status of service node	Availability of services
Purpose of migration	Ensure high-quality services	Ensure continuous services

This paper introduces a Quality-Driven Resource Migration Strategy, rooted in the concept of active migration, which significantly enhances the intrusion tolerance mechanism by enabling migration through anticipatory resource management and optimizing the load on network resources. This strategy not only minimizes service disruptions caused by migration but also ensures service continuity and security through rapid resource reorganization in response to network security threats, thereby strengthening the system's self-protective capabilities following an intrusion.

Following a review of existing research, it is evident that current studies on active migration predominantly determine migration timing based on the resource utilization of service nodes or user locations [9], often overlooking the service quality of these nodes. However, significantly compromised service quality, resulting from various factors, can severely impact the user experience. Therefore, QD-RMS recognizes service quality as a critical metric for determining the timing of active migration and initially introduces a method for this determination in this section.

4.1.1. Service Migration Index

In order to quantify the migration trend of services, the Service Migration Index (SMI) is introduced, as shown in Equation (1):

$$SMI = \frac{D \cdot Q}{f(R)} \tag{1}$$

where D is the average distance between all session users and the service node, Q is the total throughput of all session instances, and R is the resource utilization of the service node. Since resource utilization rates do not necessarily have an inverse proportional relationship with the SMI, Equation (1) employs a function $f(R)$ in the denominator. The function $f(R)$ illustrates that the impact of resource utilization rates on the Service Migration Index decreases first and then increases, with a peak value at 70%. This implies that when the resource utilization rate reaches 70%, the service node is in an optimal state in terms of resource utilization for the Service Migration Index. We can efficiently calculate D, Q, R, and $f(R)$ as follows.

$$D = \frac{\sum_{i=1}^{n} d_i}{n} \tag{2}$$

$$Q = \sum_{i=1}^{n} q_i \tag{3}$$

$$R = \frac{W_1 \cdot r_1^{norm} + W_2 \cdot r_2^{norm} + W_3 \cdot r_3^{norm} + W_4 \cdot r_4^{norm}}{\sum_{j=1}^{4} W_j} \tag{4}$$

$$f(R) = -50R^2 + 50R \tag{5}$$

where d_i is the distance between user i and the service node; q_i is the throughput of a particular session instance i; n is the total number of users; and W_i and r_i^{norm} represent the weight and the normalized utilization rate of a specific resource i, respectively.

Due to varying degrees of resource requirements by different applications running on service nodes, it is necessary to assign different weights to various resources based on the type of application. For instance, in Internet of Things (IoT) networks, an intelligent metering application aggregating utility meter readings may have a higher weight assigned to disk storage resources, whereas a high-definition video streaming application may prioritize network resources for enhanced defense mechanisms. Furthermore, in environments with multiple types of resources, each resource may exhibit distinct scales and units. Standardization ensures that when resources are allocated, weighting factors are applied appropriately according to application requirements, thereby reducing scale differences among resources. Consequently, the efficiency of resource utilization is evaluated through standardized and normalized metrics. For instance, CPU utilization can be described in millions of instructions per second (MIPS), memory utilization in megabytes (MB), network bandwidth utilization in megabits per second (Mbps), and disk utilization in gigabytes (GB). Resource utilization is standardized so that all metrics, such as CPU, memory, disk, and network bandwidth utilization rates, are represented on a uniform scale ranging from 0 to 100%. The r_i^{norm} of a specific resources i can be determined by employing Equation (6).

$$r_i^{norm} = \frac{r_i - r_{min}}{r_{max} - r_{min}} \tag{6}$$

where r_i denotes the utilization rate of a specific resource i, while r_{max} and r_{min} represent the maximum and minimum utilization rates observed across all resources, respectively.

4.1.2. Service Monitoring

The concept of the Service Migration Index was introduced in the preceding text. To calculate the Service Migration Index, the monitoring of service programs is necessary, including measuring metrics such as throughput and resource utilization rates. The combination of Prometheus and exporters is considered for monitoring service programs. The metrics and methods for data collection are encapsulated in Table 2.

Table 2. The metrics and methods for data collection.

Data	Collection Method	Collection Cycle
Throughput (queries_per_second)	Exposing exporter	1 (s)
CPU Utilization (node_cpu_seconds_total)	Exposing exporter	1 (s)
Memory Utilization (node_memory_MemTotal_bytes)	Exposing exporter	1 (s)
Storage Utilization (node_disk_io_time_second)	Exposing exporter	1 (s)
Network Activity (node_netstat_Tcp_(Active | Passive)Opens)	Exposing exporter	1 (s)
Average user session distance	Provided by base station	Determined by base station

Prometheus is a service monitoring software that encompasses four common metric types:

1. Counter: An incrementing counter used for tracking request counts, task completion counts, and similar metrics.
2. Gauge: A metric that can take on arbitrary values, suitable for metrics like CPU utilization, memory usage, and other numerical data.
3. Histogram: Categorizes a set of observations into multiple bins and exposes the number of bins and the respective observation value ranges.
4. Summary: An extension of Histogram that aggregates data (sums, counts) and provides metrics like mean and sample standard deviation.

Exporter is a tool designed to export metric data from existing applications or systems. Exporter is often an essential component in Prometheus monitoring systems as it feeds monitoring data to Prometheus. Common exporters include the following:

1. Node Exporter: For Linux/Unix systems, providing data on CPU, disk, network, memory, and more.
2. Blackbox Exporter: For probing network ports, HTTP, DNS, ICMP, and other network services to assess service statuses.
3. Redis Exporter: For monitoring the performance of Redis databases and exporting metric data to Prometheus.

The operational principle of Prometheus and exporters involves each exporter exposing an HTTP interface to Prometheus. Periodically, Prometheus calls the metrics data interface provided by the exporter to retrieve data. The exporter gathers data from the target system and transforms it into a format supported by Prometheus. Due to its independent process nature, the exporter can seamlessly integrate with any application or system. Hence, it is feasible to deploy the required exporter for monitoring service programs on individual service nodes via Kubernetes, deploy Prometheus on a management platform, and establish connections with the exporters on the service nodes, thus enabling the monitoring of service programs on each node, as illustrated in Figure 2.

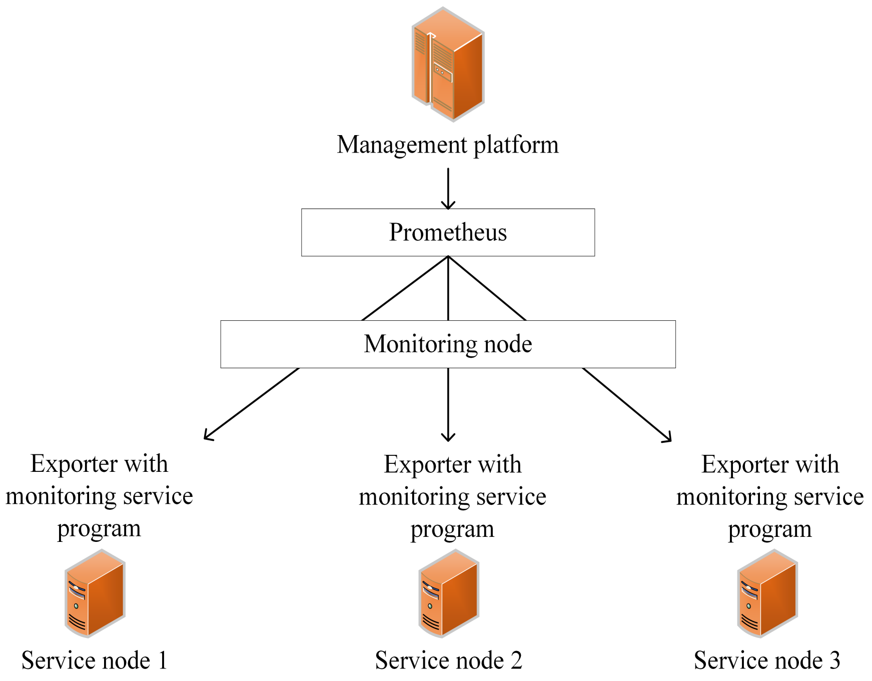
Management platform
Prometheus
Monitoring node
Exporter with monitoring service program
Exporter with monitoring service program
Exporter with monitoring service program
Service node 1
Service node 2
Service node 3

Figure 2. Service monitoring method.

4.1.3. Active Migration Process

Initiating service migration prematurely may result in the underutilization of the host's resources. Conversely, delayed migration can adversely affect user experience. Thus, optimal timing for service migration hinges on setting an appropriate migration threshold; migration should be triggered when the Service Migration Index hits this threshold. This section introduces a proactive migration strategy guided by the Service Migration Index and migration threshold, determining the optimal juncture for transferring service operations. This strategy encompasses three phases: index monitoring, service migration, and threshold updating. The detailed procedure is depicted in Figure 3.

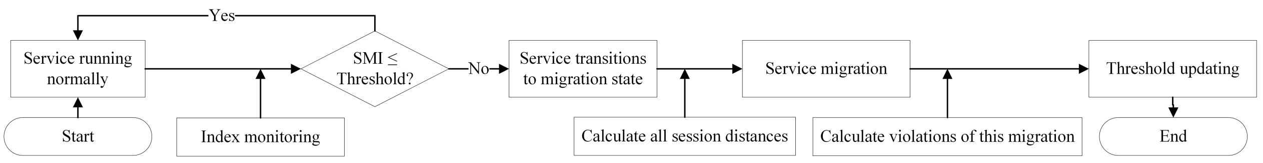
Yes
Service running normally
SMI ≤ Threshold?
No
Service transitions to migration state
Service migration
Threshold updating
Start
Index monitoring
Calculate all session distances
Calculate violations of this migration
End

Figure 3. Active migration process.

1. Index Monitoring Module: This module monitors the status of applications on service nodes using monitoring software. It calculates the Service Migration Index SMI in real-time based on relevant parameters and compares it with the migration threshold to determine the migration trend for the service. The primary objective of the monitoring program is to monitor and compute parameters. As shown in Algorithm 1, this module calculates the real-time SMI by continuously collecting data on the current service's resource utilization, the average distance of all session users, and the total throughput of all session instances Q. If the SMI is less than the set threshold (SMI-T) for the current round, the system will sleep for a period before resuming the process. If the SMI is greater than or equal to the SMI-T threshold, the service status is changed to migration mode, the loop is terminated, and the service migration module is invoked.

Algorithm 1 Index monitoring

```
Input: Application $APP$, Resource weight $W$, Threshold $SMI\text{-}T$
Output: Application state $app\_status$
  $app\_status \leftarrow normal$
  while $app\_status == normal$ do
    $f(R) \leftarrow calculate\_f(R)(APP, W)$
    $D \leftarrow calculate\_D(APP)$
    $Q \leftarrow calculate\_Q(APP)$
    $SMI \leftarrow calculate\_SMI(f(R), D, Q)$
    if $SMI < SMI\text{-}T$ then
      Wait some time
    else
      $app\_status \leftarrow migration\_trend$          /*Application enters migration state*/
    end if
  end while
  run Service migration
  return $app\_status$
```

2. Service Migration Module: Initially, all session instances within the service are scanned and sorted based on the distance of each user from the nodes, and users are sequentially moved to a migration queue for migrating instances starting with users farthest from the nodes. This process continues until the *SMI* drops below the migration threshold, at which point migration stops. During this process, violations by migrating users are monitored and recorded. The service migration program primarily selects session instances for migration based on the *SMI*. It iterates through all current session instances, calculates the distance of each user from the service nodes, adds this information to a distance queue, and then proceeds to migrate users in order of their distance from the service nodes until the *SMI* falls below the set threshold (*SMI-T*), at which point the loop stops, and the threshold update module is called. Algorithm 2 demonstrates the procedure of a service migration.

Algorithm 2 Service migration

```
Input: Application $APP$, Service migration index $SMI$
Output: otal number of service violations $sla\_violations\_count$
  while $SMI \geq SMI\text{-}T$ do
    for all $instance$ do
      $Distance[] \leftarrow [d[instance_1] \ldots d[instance_n]]$
      Sort $Distance[]$
      $MI \leftarrow \arg\max(d[instance_i])$
      migration $MI$
      $max.sla\_violations \leftarrow calculate\_sla(MI)$       /* Calculate the maximum number of
      violation incidents about $MI$*/
      if $max.sla\_violations$ then
        $sla\_violations\_count += max.sla\_violations$
      end if
    end for
  end while
  run Threshold updating
  return $sla\_violations\_count$
```

3. Threshold Updating Module: This module compares the product of distance and average waiting delay with a predefined base value to evaluate the reasonableness of the threshold. A violation value of 0 indicates an appropriately set threshold for migration in the current context, and a slight random increase in the threshold can enhance the utilization of service nodes. If the violation value is greater than 0, it

signifies that the threshold set is too high, causing multiple violations by the time the SMI reaches the threshold. In such cases, a small random decrease in the threshold is necessary. The threshold update program first calculates the current migration round for the service. If the migration rounds are relatively low, such as equal to or less than 5, the SMI-T is randomly adjusted based on violations, with the adjustment value correlated to the violation severity. For higher migration rounds, the module takes the SMI-T and average waiting delay into account, using the method of least squares to fit a function. The computation determines the maximum value of the independent variable for which the fitted function equals 0, providing a new SMI-T value. The threshold update strategy is shown in Algorithm 3.

Algorithm 3 Threshold updating

```
Input: Application $APP$, Distance $D$, Average waiting delay $delay$, Migration round
  $migration\_round$, Total number of service violations $sla\_violations\_count$
Output: updated threshold $SMI$-$T$
  if $migration\_round \leq 5$ then
    if $sla\_violations\_count == 0$ then
      $SMI$-$T$ increase small random number
    else
      $SMI$-$T$ descend small random number
    end if
  else
    $f(x) \leftarrow polyfit(D, delay)$ /*Using least squares method for function fitting*/
  end if
  $X \leftarrow \arg\max_{f(x)==0} x$ /*Calculating the maximum value of the independent variable
  when the fitted function f equals 0*/
  $SMI$-$T \leftarrow X$
  return $SMI$-$T$
```

4.2. Node Selection Algorithm Based on Load Balancing

Following the determination of the need for migration as outlined in Section 4.1, the ensuing critical task is to strategically select the optimal target node to manage the migration. In this section, we present a migration node selection algorithm designed to enhance migration decisions through load balancing, aiming to improve system performance and maintain stability across the network. The essence of load balancing is to evenly distribute various tasks across each server in the network as much as possible, allowing these servers to share the workload and enabling all network devices to operate efficiently. Neglecting server loads during the migration process may result in some servers bearing a heavy burden of tasks while others remain idle. This imbalance can lead to overloaded servers or even server crashes, along with the wastage of resources on many idle servers. Therefore, when allocating tasks to servers, it is essential to strive for a balanced distribution, ensuring that the load on each server is close to equilibrium, thus maximizing the utilization of the computational and storage resources provided by the servers.

Figure 4 illustrates an uneven load distribution within a service area. From the depicted load states in the figure, it is evident that the load distribution across the servers is uneven. For instance, servers S_2, S_4, and S_6 are already experiencing high levels of load, whereas S_1 and S_5 remain at relatively low load levels. As analyzed earlier, an uneven load distribution significantly decreases the operational efficiency of servers and results in wastage of computational and storage resources.

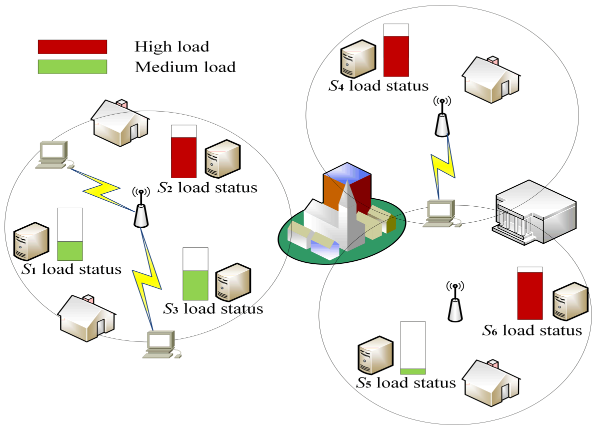

Figure 4. Imbalanced load distribution.

Figure 5 depicts a schematic diagram illustrating the load status at each server after load-balancing adjustments. By appropriately distributing tasks to balance the workload among servers, load balancing aims to enhance the operational efficiency among servers.

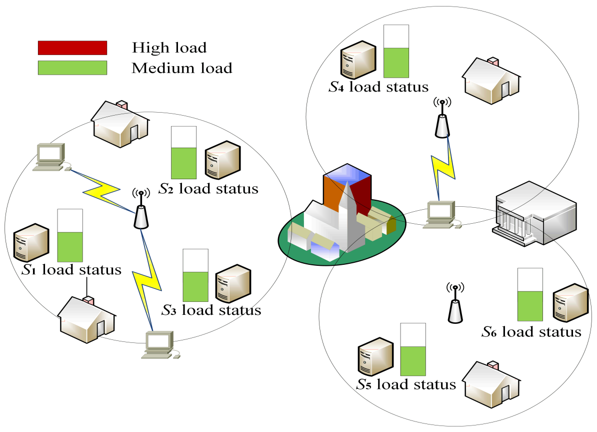

Figure 5. Balanced load distribution.

Currently, there has been extensive research in the field of load balancing. Classic load-balancing approaches are implemented using random algorithm, round-robin scheduling, and least-connection algorithm. Additionally, there are new explorations into load-balancing issues. For example, P. Zhao et al. [24] initially framed load balancing in MEC (Multi-access Edge Computing) as a problem of minimizing energy consumption and queue redundancy. Then, they applied Lyapunov algorithms to solve such optimization problems. L. Liu et al. [27] considered forcing an overloaded server to randomly select two servers from a group of adjacent servers and then unload to the one with the smallest load. However, this approach leads to redirecting a large number of tasks to originally lightly loaded servers, causing them to become overloaded, thus failing to address the issue of load imbalance. Abedin et al. [28] used game theory to solve load-balancing problems by formulating the problem of minimizing costs as a transportation problem and employing the Vogel approximation method to compute the optimal solution. Similarly, D. Liu et al. [25] considered introducing game theory to solve load-balancing problems. They initially formulated the load-balancing problem as a population game model and proposed two load-balancing algorithms based on evolutionary dynamics and modified protocols.

However, the aforementioned studies mainly focus on balancing the number of tasks among servers, neglecting a consideration of the sizes of tasks themselves. Assuming that the computational resource requirements for Task A are several times greater than Task B, treating Tasks A and B as equivalent tasks and allocating server resources based solely on task quantity may not fully utilize server resources. Therefore, this section further advances the existing load-balancing algorithms by incorporating considerations for matching task resource demands with server resource capacities, proposing a migration node selection algorithm based on load balancing to more effectively utilize the computational resources of service nodes.

4.2.1. Resource-Based Load Balancing

Considering the existence of a cluster of n servers $S = (S_1, S_2, S_3, \ldots, S_n)$, for a server S_i, its total resource content is a resource vector R_i^{init}, as shown in Equation (7):

$$R_i^{init} = (C, M, H, B) \tag{7}$$

where C, M, H, and B represent the CPU, memory, disk, and network bandwidth resource, respectively. Additionally, the vector of remaining available resources at the current moment t is the following:

$$R_i^t = (C_t, M_t, H_t, B_t) \tag{8}$$

where C_t, M_t, H_t, and B_t represent the remaining available resources of the CPU, memory, disk, and network bandwidth resource at the current moment t, respectively.

A closer ratio of R_i^{init} to R_i^t to ω $(0 < \omega < 1)$ suggests a higher level of resource utilization equality. This indicates that the resources are being used more efficiently and effectively. When the two resource vectors are not proportional, this indicates a deviation from the optimal utilization of resources. τ represents the cosine similarity between two resource vectors, serving to quantify the degree of deviation in resource utilization.

$$\tau = \cos\left(R_i^t, R_i^{init}\right) \tag{9}$$

If the allocated tasks can correct the deviation in resource utilization rate, it can be considered as beneficial for the load balancing of server resources.

Let the resource vector required for the user's task at time t be denoted as D_i^t. The remaining resource vector after accepting the task $\dot{R}_i^t$ can be calculated using Equation (10). We can then determine the deviation in resource utilization following the acceptance of task $\dot{\tau}$ using Equation (11).

$$\dot{R}_i^t = R_i^t - D_i^t \tag{10}$$

$$\dot{\tau} = \cos\left(\dot{R}_i^t, R_i^{init}\right) \tag{11}$$

If $\sigma = \dot{\tau} - \tau > 0$, it indicates that the task can contribute to improving the balance of resource utilization in the server.

4.2.2. Task-Based Load Balancing

For the server S_i, the vector of total resource content and remaining available resources at the current moment t is denoted by Equations (7) and (8). The load level on each resource can be calculated using the following equations:

$$LB_C = \frac{C_t}{C} \tag{12}$$

$$LB_M = \frac{M_t}{M} \tag{13}$$

$$LB_H = \frac{H_t}{H} \tag{14}$$

$$LB_B = \frac{B_t}{B} \tag{15}$$

where LB_C, LB_M, LB_H, and LB_B represent the load level on CPU, memory, disk, and network bandwidth resource, respectively.

Furthermore, ω is the load level of the server S_i, which can be calculated using Equation (16). The average load on the server cluster $\bar{\omega}$ is given by Equation (17).

$$\omega = \frac{LB_C + LB_M + LB_H + LB_B}{4} \tag{16}$$

$$\bar{\omega} = \frac{\sum_{i=1}^{n} \omega_i}{n} \tag{17}$$

Finally, the variance value of the average load on the server cluster $\dot{\omega}$ can be calculated by Equation (18), which reflects the load balancing of tasks undertaken by individual servers in the server cluster.

$$\dot{\omega} = \sum_{i=1}^{n} (\omega_i - \bar{\omega})^2 \tag{18}$$

4.2.3. Migration Node Selection Algorithm Based on Task-Resource Matching

The preceding two subsections analyzed the balance of server resources and the load balancing of task quantities across servers. Therefore, the consideration of load balancing on nodes in the candidate server pool during task migration can be transformed into optimization problems concerning the aforementioned two metrics. This issue fundamentally constitutes a multi-objective optimization problem. Due to the heterogeneity of user devices and the diversity of user tasks, optimizing one metric inevitably impacts another. Introducing Pareto optimality theory measures superior load balancing solution nodes. The concept of Pareto optimality refers to a state of resource allocation where modifying the allocation status does not compromise any objective function, while benefiting at least one objective function. If no further allocation changes can be made in this manner, the solution is considered one of the Pareto optimal solutions. Thus, the consideration of load balancing aims to achieve Pareto optimal status for the two aforementioned metrics. However, given the need for service continuity in the data center, excessive time spent seeking Pareto optimal solutions may lead to a misplaced emphasis, thereby compromising the original objective of efficient and high-quality migrations. Hence, in this scenario, considerations should be made regarding the magnitude of service nodes, delineating different scenarios. When the number of candidate nodes does not exceed a specific threshold (subject to real-world conditions), comprehensive screening is conducted, wherein each node undergoes Pareto adjustments to find the optimal solution.

Considering that the search latency of Algorithm 4 would become unmanageable with a large number of nodes, the adoption of Algorithm 5 with a simulated annealing approach is contemplated for expedited exploration, aiming to identify a set of superior solutions within an acceptable timeframe.

Algorithm 4 Comprehensive screening of Pareto optimal nodes

```
Input: Server nodes ID, User demand resource vector
Output: Set of servers that achieves Pareto optimality $\mu$
  for $i \in n$ do
    Calculate the resource balance degree $\dot{\tau} - \tau$ after server $i$ accepts the task
    Calculate the load balancing degree $\dot{\omega}$ after server $i$ accepts the task
  end for
  for $i \in n$ do
    for $y \in n$ do
      if ($\dot{\tau}_y \geq \dot{\tau}_i$ and $\dot{\omega}_y \geq \dot{\omega}_i$) and ($\dot{\tau}_y \geq \dot{\tau}_i$ or $\dot{\omega}_y \geq \dot{\omega}_i$) then
        Replace node $i$ with node $y$
        Record the serial number of node $i$
      end if
    end for
    $\sigma \leftarrow \sigma \cup y$
    Delete all recorded nodes from $n$
  end for
```

Algorithm 5 Simulated annealing searches for Pareto optimal nodes

```
Input: Serialized server nodes ID, User demand resource vector
Output: Set of servers that achieves Pareto optimality $\sigma$
  for $i \in n$ do
    Calculate the resource balance degree $\dot{\tau} - \tau$ after server $i$ accepts the task
    Calculate the load balancing degree $\dot{\omega}$ after server $i$ accepts the task
    Perform range perturbation $Rand(i \pm \epsilon)$ to generate new solutions
    Calculate the degree of equilibrium
    if $new > old$ then
      Accept and record new optimal
    else
      Accept with a certain probability according to metropolis guidelines
    end if
    Iterate for multiple times
    Terminate gradually according to annealing temperature
  end for
```

5. Evaluation

5.1. Active Migration Based on Service Quality

In our research, we evaluated passive and active migration strategies across four service program types. The outcomes for the passive migration algorithm are depicted in Figure 6, with the experiment's duration plotted on the x-axis in seconds. The left y-axis indicates the number of request sessions, represented by blue bars in the graph. The right y-axis calculates the product of average distance D and average request waiting time AWT, in milliseconds, offering a composite measure of spatial and temporal efficiency in service delivery. This product, $D \times AWT$, acts as a service quality index SQ, where the distance D reflects the spatial component related to transmission delay.

The red line in the figure denotes the benchmark for desirable service quality, preset in this experiment. In contrast, the green curve shows the actual performance of the service program. The threshold represented by the red line is adaptable, allowing for customization based on various operational scenarios and requirements. For this experiment, the threshold was set at 50 ms, indicating a higher standard of service quality. Any segment of the green curve surpassing the red line signals a period when the service quality dips below the established threshold.

Passive migration is initiated when the resources for a service program are depleted or when users exit the service area, rendering continued service impossible. Consequently, services may remain active but suffer from diminished quality. As demonstrated in Figure 6, for matrix calculation, video service, and website service, the green curve, indicative of service demand, escalates swiftly with increasing service requests. When session numbers escalate to a certain threshold, the latency, represented by the green curve, substantially exceeds the established benchmark for superior service quality, subjecting users to excessive delays and a substandard service experience. Notably, in game service, approaching 150 session requests triggers passive migration due to resource depletion at the node. This leads to the migration of about half the session processes from the node, causing a decline in the green curve to below the threshold, which corresponds to an enhancement in service quality. The analysis of passive migration experiments illustrates the challenge in selecting the optimal timing for passive migration, crucial for maintaining service quality at edge nodes across various scenarios.

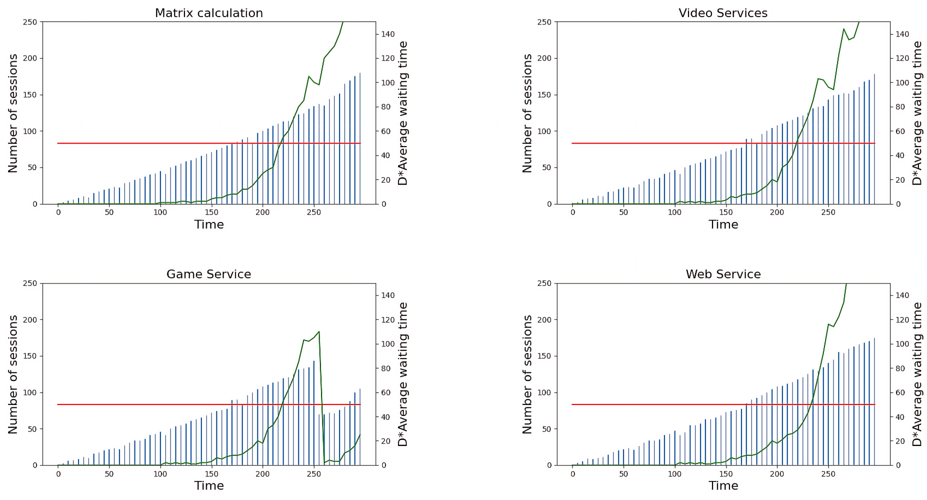

Figure 6. Results of passive transfer experiment.

Figure 7 presents the experimental results of proactive migration based on static resource utilization rates. The decision to migrate is contingent upon reaching a predetermined threshold of resource utilization at the service node, meaning that migration occurs when the node's resource utilization hits this predefined limit.

The experimental findings indicate that, compared to passive migration, proactive migration based on static resource utilization can, to some extent, ensure a higher quality of service. For instance, in the experiment involving matrix calculation, migration was initiated once the session count hit 100, effectively reducing latency to below the baseline. However, the performance in video and web service applications was less than ideal. In the video service scenario, despite the latency significantly exceeding the baseline, migration was not executed because the resource utilization had not met the established threshold. A similar situation occurred with the web service application, where migration was delayed until the session count neared 150, reaching the resource utilization ceiling, by which point the latency had already significantly exceeded the baseline, indicating severely compromised service quality. Conversely, in the game service, migration was triggered even though the actual latency had not yet reached the baseline, leading to the underutilization of the service node's resources.

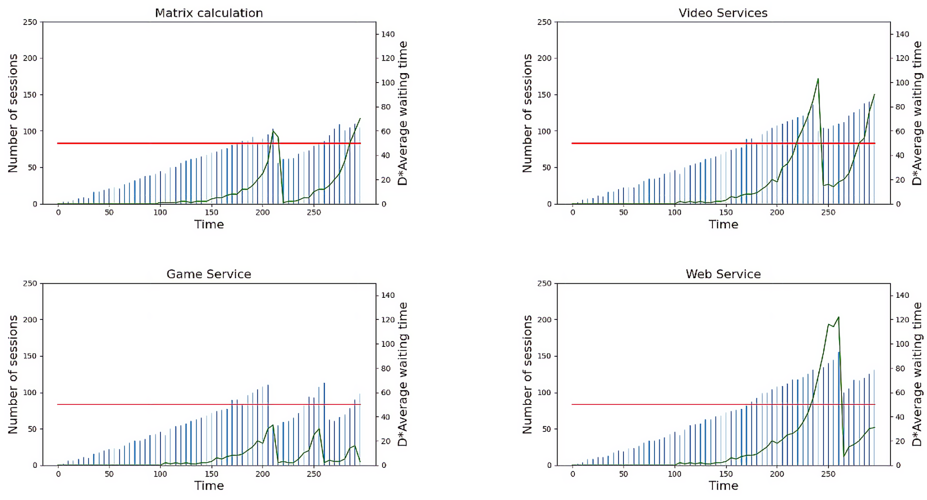

Figure 7. Results of active migration based on static resource utilization.

Further expanding on this, we present the findings based on the service quality of proactive migration. As depicted in Figure 8, across the four service programs tested, migration is instantaneously triggered when the green curve (representing real-time service quality metrics) surpasses the red line (the set threshold), subsequently maintaining a position almost equivalent to the red line. This effectiveness stems from the algorithm's dynamic threshold-updating mechanism, where the Service Migration Index Threshold (SMI-T) is adjusted according to the deviation from the standard Service Quality (SQ) value. This adjustment enables the green curve to precisely gauge the SMI's upper boundary in the experiments. In comparison to passive migration and proactive migration based on static resource utilization, the service quality-based proactive migration strategy proposed herein optimizes the timing of migration, thereby maximizing the utilization of service node resources without compromising on service quality.

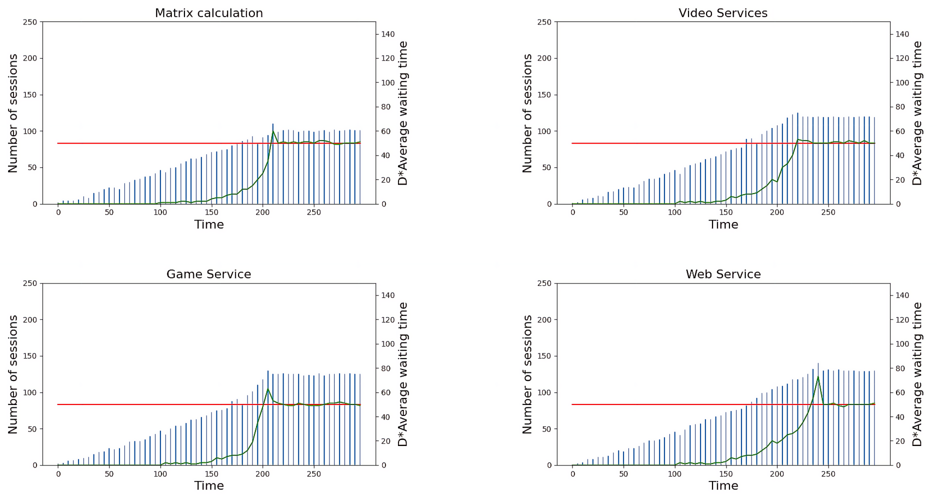

Figure 8. Results of active migration based on quality of service.

5.2. Migration Node Selection for Task-Resource Matching

To assess the effectiveness of our migration node selection algorithm based on task-resource matching, we conducted several tests across diverse scenarios and compared the results of our algorithm with those of the random algorithm, round robin algorithm, and least connection algorithm in the same situation. First, we simulated a user who sends 30 task requests to the load balancer. Figure 9 shows the tasks assigned to each server where the x-axis is the server serial number, and the y-axis is the number of tasks assigned. Figure 10 shows the resource deviation of service system nodes after load balancer allocation. It can be seen from the results of the Figures 9 and 10 that the four algorithms have little difference in the number of tasks assigned, but there is a significant difference in the degree of resource deviation. Since the traditional random algorithm, round robin algorithm, and least connection algorithm do not consider the problem of resource matching between tasks and servers, it can be seen from the experimental results that the resource deviation of the service node is large. This also means that the resource utilization of service nodes is not balanced, and the full capacity of the server is not fully utilized. The task resource matching algorithm proposed in this paper can ensure lower resource deviation and allocate different types of tasks to appropriate service nodes more reasonably.

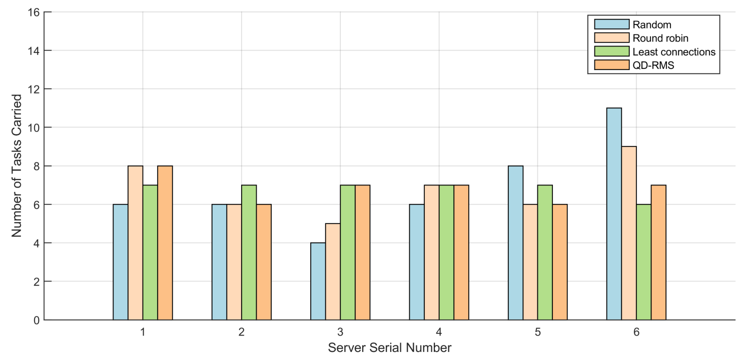

Figure 9. Results of 30 assignment requests.

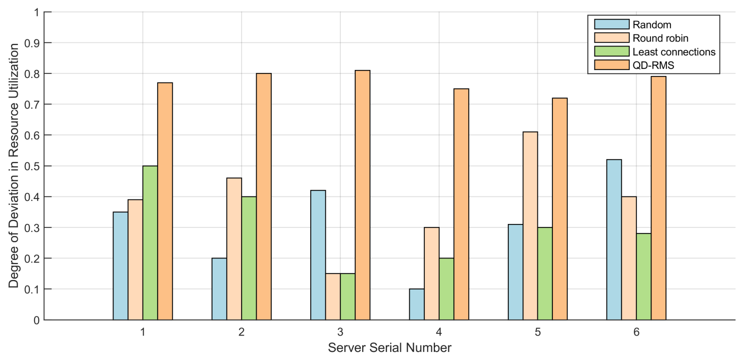

Figure 10. Resource deviation degree of a service node after 30 tasks are assigned.

Second, we simulated users who apply for tasks from the load balancer indefinitely until a resource on the server is exhausted. Figure 11 shows the tasks assigned to each server and Figure 12 shows the degree of deviation in resource utilization τ (as the value approaches 1, resource utilization becomes more balanced; see Equation (9)) of service nodes after an unlimited number of tasks are assigned. It can be seen from the results that under the infinite number of task requests, the QD-RMS proposed in this paper has a significant improvement in the number of tasks carried out and still maintains a good performance in the resource deviation.

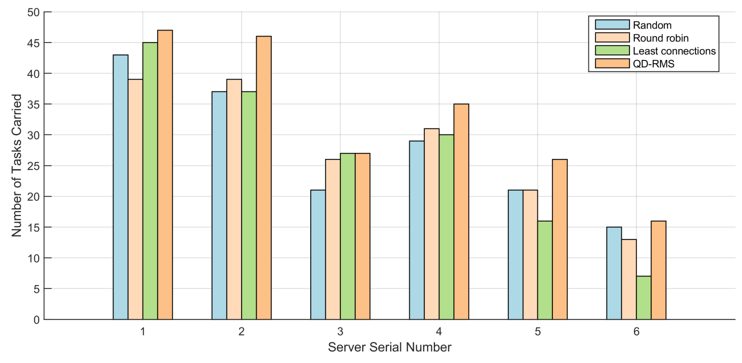

Figure 11. Results of an unlimited number of task requests.

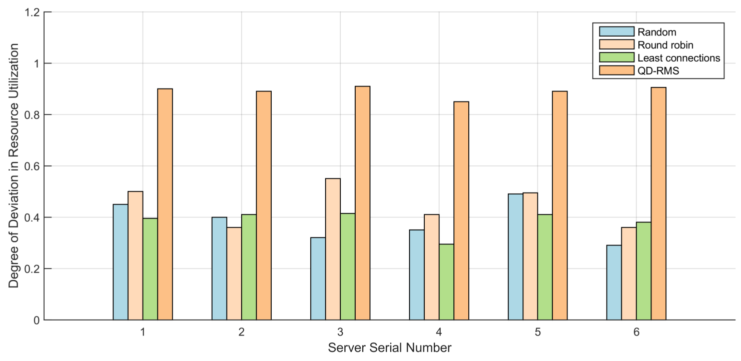

Figure 12. Resource deviation degree of a service node after an infinite number of tasks are assigned.

Lastly, we compare the maximum task carrying capacity of various algorithms, that is, the total number of tasks carried out by all service nodes. As shown in Figure 13, the algorithm proposed in this paper can make full use of the resources of service nodes because it takes into account the matching degree between the resources required by different task types and the server resources. Under the same equipment conditions, the maximum task load is increased by about 20%.

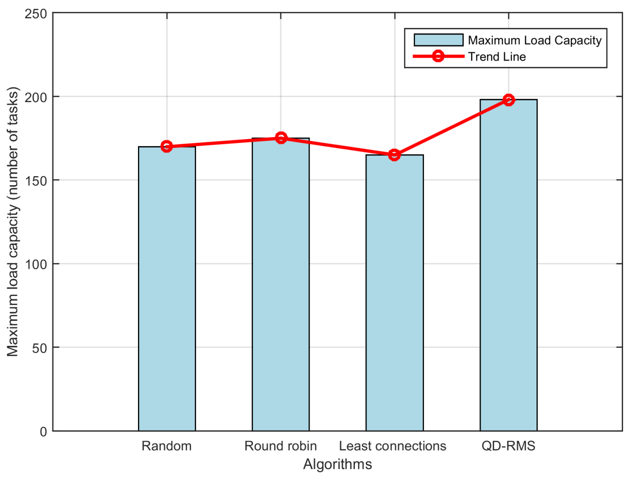

Figure 13. Comparison of Maximum Task Capacity

6. Conclusions

In conclusion, this research introduces the Quality-Driven Resource Migration Strategy (QD-RMS), a policy focused on optimizing resource allocation and enhancing decision making in service migration, guided by service quality metrics. QD-RMS proactively triggers resource migration prior to any decline in service quality through a service migration index and a dynamic threshold mechanism, thereby ensuring uninterrupted migration and significantly mitigating fluctuations in service quality. Moreover, leveraging Pareto optimization and the simulated annealing algorithm, this strategy efficiently selects the optimal node for migration within a feasible computational timeframe, optimizing resource utilization and load balancing.

QD-RMS effectively mitigates the delayed response of conventional migration strategies to potential service quality declines. Through the continuous monitoring of the Service Migration Index (SMI), it facilitates the real-time assessment of service status, allowing for the early activation of migration processes before significant degradation in service quality, thus diminishing variability in service quality and minimizing service downtime. Additionally, QD-RMS employs a multi-objective optimization model that amalgamates node resources and task load, enhancing the load-balancing process and preventing service migration to nodes at the risk of resource overutilization, consequently reducing system overload risks.

To ascertain QD-RMS's efficacy, a series of experiments were conducted, simulating varied service requests on a virtual machine platform to benchmark QD-RMS against conventional resource migration strategies. The experimental results indicate that QD-RMS not only maintains service quality but also reduces service fluctuation and increases the maximum task capacity by approximately 20%, thereby validating its superior performance and advanced load-balancing capabilities in real-world settings.

Beyond ensuring continuous network services, QD-RMS extends its utility to optimizing resource scheduling within networks. It dynamically reallocates resources by harnessing service quality metrics and load data, thereby adapting to ever-evolving network demands and traffic flows. Nevertheless, its centralized decision-making framework is not without drawbacks. It can lead to transmission bottlenecks as the scale of resources expands. To mitigate this, forthcoming enhancements will pivot toward a distributed framework, aiming to distribute the responsibilities of resource management and migration decisions across multiple units. This shift will alleviate the load on central management nodes and enhance overall efficiency.

Author Contributions: Y.S. and Y.J. primarily handled conceptualization and methodology; C.C. carried out software development and implementation; Y.J. conducted the experimental validation; C.C. was responsible for data collection and management; Y.S. and C.C. prepared the initial draft of the manuscript; Y.S., Y.J. and M.Z. performed the review and editing of the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Research on Intrusion Tolerance and Protection Technology of Power Data Center (Grand No. 5700-202218185A-1-1-ZN).

Data Availability Statement: The data presented in this study are available on request from the corresponding author due to privacy and ethical considerations.

Conflicts of Interest: Author Mingming Zhang was employed by the company "State Grid Corporation of China". The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest. The authors declare that this study received funding from the project "Research on Intrusion Tolerance and Protection Technology of Power Data Center" (Grand No. 5700-202218185A-1-1-ZN). The funder was not involved in the study design; the collection, analysis, or interpretation of the data; the writing of this article; or the decision to submit it for publication.

References

1. Wu, J. Cyberspace Endogenous Safety and Security. *Engineering* **2022**, *15*, 179–185. [CrossRef]
2. Tan, K.H.; Lim, H.S.; Diong, K.S. Modelling and Predicting Quality-of-Experience of Online Gaming Users in 5G Networks. *Int. J. Technol.* **2022**, *13*, 1035–1044. [CrossRef]
3. Lu, H.; Li, M.; Zhang, Y. Research on optimization method of computer network service quality based on feature matching algorithm. *Proc. J. Phys. Conf. Ser.* **2021**, *1982*, 012005. [CrossRef]
4. Flora, J. Improving the security of microservice systems by detecting and tolerating intrusions. In Proceedings of the 2020 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), Coimbra, Portugal, 12–15 October 2020; pp. 131–134.
5. Wang, Z.; Jiang, D.; Wang, F.; Lv, Z.; Nowak, R. A polymorphic heterogeneous security architecture for edge-enabled smart grids. *Sustain. Cities Soc.* **2021**, *67*, 102661. [CrossRef]
6. Kang, R.; He, F.; Oki, E. Virtual network function allocation in service function chains using backups with availability schedule. *IEEE Trans. Netw. Serv. Manag.* **2021**, *18*, 4294–4310. [CrossRef]
7. Zhao, J.; Dai, M.; Xia, Y.; Ma, Y.; He, M.; Peng, K.; Li, J.; Li, F.; Fu, X. FRSM: A Novel Fault-Tolerant Approach for Redundant-Path-Enabled Service Migration in Mobile Edge Computing. In *Proceedings of the International Conference on Web Services*; Springer: Cham, Switzerland, 2022; pp. 1–12.
8. Ma, Y.; Dai, M.; Shao, S.; Xia, Y.; Li, F.; Shen, Y.; Li, J.; Li, Y.; Peng, H. A Performance and Reliability-Guaranteed Predictive Approach to Service Migration Path Selection in Mobile Computing. *IEEE Internet Things J.* **2023**, *10*, 17977–17987. [CrossRef]
9. Kulshrestha, S.; Patel, S. An efficient host overload detection algorithm for cloud data center based on exponential weighted moving average. *Int. J. Commun. Syst.* **2021**, *34*, e4708. [CrossRef]

10. Yang, L.; Yang, D.; Cao, J.; Sahni, Y.; Xu, X. QoS Guaranteed Resource Allocation for Live Virtual Machine Migration in Edge Clouds. *IEEE Access* **2020**, *8*, 78441–78451. [CrossRef]
11. Velrajan, S.; Sharmila, V. QoS Aware Service Migration in Multi access Edge Compute Using Closed Loop Adaptive Particle Swarm Optimization Algorithm. *J. Netw. Syst. Manag.* **2023**, *31*, 17. [CrossRef]
12. Chen, X.; Bi, Y.; Chen, X.; Zhao, H.; Cheng, N.; Li, F.; Cheng, W. Dynamic Service Migration and Request Routing for Microservice in Multicell Mobile-Edge Computing. *IEEE Internet Things J.* **2022**, *9*, 13126–13143. [CrossRef]
13. Bozkaya, E. Digital twin-assisted and mobility-aware service migration in Mobile Edge Computing. *Comput. Netw.* **2023**, *231*, 109798. [CrossRef]
14. Wang, H.; Li, Y.; Zhou, A.; Guo, Y.; Wang, S. Service migration in mobile edge computing: A deep reinforcement learning approach. *Int. J. Commun. Syst.* **2023**, *36*, e4413. [CrossRef]
15. Liang, Z.; Liu, Y.; Lok, T.M.; Huang, K. Multi-Cell Mobile Edge Computing: Joint Service Migration and Resource Allocation. *IEEE Trans. Wirel. Commun.* **2021**, *20*, 5898–5912. [CrossRef]
16. Kim, T.; Chen, S.; Im, U.; Zhang, X.; Ha, S.; Joe-Wong, C. MoDEMS: Optimizing Edge Computing Migrations for User Mobility. *IEEE J. Sel. Areas Commun.* **2023**, *41*, 675–689. [CrossRef]
17. Xu, Y.; Zheng, Z.; Liu, X.; Yao, A.; Li, X. Three-way decisions based service migration strategy in mobile edge computing. *Inf. Sci.* **2022**, *609*, 533–547. [CrossRef]
18. Le, V.T.; Ioini, E.; Barzegar, H.R.; Pahl, C. Trust Management For Service Migration Multi-access Edge Computing environments. *Comput. Commun.* **2022**, *194*, 167–179. [CrossRef]
19. Li, C.; Zhu, L.; Li, W.; Luo, Y. Joint edge caching and dynamic service migration in SDN based mobile edge computing. *J. Netw. Comput. Appl.* **2021**, *177*, 102966. [CrossRef]
20. Du, A.; Jia, J.; Chen, J.; Guo, L.; Wang, X. Online two-timescale service placement for time-sensitive applications in MEC-assisted network: A TMAGRL approach. *Comput. Netw.* **2024**, *244*, 110339. [CrossRef]
21. Tuli, S.; Casale, G.; Jennings, N.R. PreGAN: Preemptive Migration Prediction Network for Proactive Fault-Tolerant Edge Computing. In Proceedings of the IEEE INFOCOM 2022—IEEE Conference on Computer Communications, London, UK, 2–5 May 2022; pp. 670–679.
22. Dai, X.; Xiao, Z.; Jiang, H.; Alazab, M.; Lui, J.C.S.; Dustdar, S.; Liu, J. Task Co-Offloading for D2D-Assisted Mobile Edge Computing in Industrial Internet of Things. *IEEE Trans. Ind. Inform.* **2023**, *19*, 480–490. [CrossRef]
23. Moon, S.; Lim, Y. Task Migration with Partitioning for Load Balancing in Collaborative Edge Computing. *Appl. Sci.* **2022**, *12*, 1168. [CrossRef]
24. Zhao, P.; Tao, J.; Rauf, A.; Jia, F.; Xu, L. Load balancing for energy-harvesting mobile edge computing. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **2021**, *104*, 336–342. [CrossRef]
25. Liu, D.; Hafid, A.; Khoukhi, L. Workload Balancing in Mobile Edge Computing for Internet of Things: A Population Game Approach. *IEEE Trans. Netw. Sci. Eng.* **2022**, *9*, 1726–1739. [CrossRef]
26. Song, S.; Ma, S.; Zhao, J.; Yang, F.; Zhai, L. Cost-efficient multi-service task offloading scheduling for mobile edge computing. *Appl. Intell.* **2022**, *52*, 4028–4040. [CrossRef]
27. Liu, L.; Chan, S.; Han, G.; Guizani, M.; Bandai, M. Performance modeling of representative load sharing schemes for clustered servers in multiaccess edge computing. *IEEE Internet Things J.* **2018**, *6*, 4880–4888. [CrossRef]
28. Abedin, S.F.; Bairagi, A.K.; Munir, M.S.; Tran, N.H.; Hong, C.S. Fog load balancing for massive machine type communications: A game and transport theoretic approach. *IEEE Access* **2018**, *7*, 4204–4218. [CrossRef]