World Wide Web
https://doi.org/10.1007/s11280-023-01201-y

A large-scale holistic measurement of crowdsourced edge cloud platform

Yicheng Feng[1] · Shihao Shen[1] · Mengwei Xu[2] · Cheng Zhang[3] · Xin Wang[1] · Xiaofei Wang[1] · Wenyu Wang[4] · Victor C. M. Leung[5,6]

Received: 25 May 2023 / Revised: 12 July 2023 / Accepted: 20 July 2023

Abstract
Edge clouds have become a de-facto paradigm to deliver low and stable networks to delay-critical applications such as Web services and AR/VR. A unique form of edge clouds is those crowdsourced from third parties, e.g., idle PCs or workstations. Such crowdsourced edge platforms can better sink computations closer to users, reduce the purchase cost, and eliminates the carbon generated during manufacturing. Yet, they also face the challenge of out-of-control hardware, e.g., a server dropping in/out anytime. In this paper, we perform the first-of-its-kind measurement of Quality of Service (QoS) for a large-scale crowdsourced edge platform, which covers over 10,000 edge servers, 100,000 users and 10,000,000 user requests. The measurement takes a holistic QoS view: First, we look at how much hardware resources are provided by **edge servers**, how much time they are available for service deployment, how geographic distance affects network performance, and what are the major abnormal behaviors. Second, we analyze the factors affecting service stability and quantify the resource utilization pattern of **containerized services** hosted on those edge servers. Third, we investigate the spatial and temporal features of **user requests** handled by the platform. Many useful and somehow surprising findings are obtained through the above measurements. We also derive insightful implications that could help edge platforms and edge applications to better deliver their services to users.

Keywords Edge computing · Performance analysis · Network measurement

1 Introduction

Contents delivery, AR/VR, and so on are increasingly adopting edge computing paradigm [1–3], as a critical extension to centralized datacenters. By providing in-proximity hardware resources to end users, edge clouds not only effectively alleviate the network bandwidth pressure on the backbone Internet, but also reduce the network delay and thus improve the

✉ Xiaofei Wang
xiaofeiwang@tju.edu.cn

Extended author information available on the last page of the article

Published online: 05 August 2023

service quality [4, 5]. According to Gartner, around 75% of enterprise-generated data will be processed at the edge by 2025 [6].

In the realm of edge deployment, a diverse array of strategies exist to meet the unique demands of modern applications. The industry's leading cloud resource providers have taken steps to construct their own edge infrastructure at a state or city level, as evidenced by Azure Edge Zone [7] and AWS Local Zones [8]. Meanwhile, others are exploring the option of sinking edge servers into buildings or base stations [9, 10]. Regardless of the approach taken, it is clear that the hardware and software that comprise these edge sites are maintained by the service providers themselves, who retain full control over their operation. This approach undoubtedly results in a higher Quality of Service (QoS) for users.

Crowdsourced edge cloud platform is a unique form of edge deployment: the edge servers are recruited from any third parties (namely Edge Hardware Provider or EHP) through a business incentive model. EHP can hand over their unused or idle machines to ESP (Edge computing Service Provider) at anytime and anywhere. The latter takes in and tests the machine, sets it up as an edge site, and exposes its hardware capacity to edge app developers through a unified interface. EHP makes profits from ESP according to how much and how long it provides hardware resources to ESP. Meanwhile, EHP can drop out their machines anytime as well, i.e., an "earn-as-you-go" model.

Therefore, such crowdsourced edge cloud platform[1] has the following advantages over traditional ones. (*i*) **Decentralized-by-nature**. Being geographically distributed and closer to users is vital to the success of edge computing. Edge servers recruited through crowdsourcing are naturally decentralized and operating in close to users. (*ii*) **Cost-efficient**. With crowdsourced hardware, ESP has zero expenditure in purchasing the hardware (i.e., no cold-start fee). Instead, ESP only pays for the exact hardware quota it gets. It makes the large-scale deployment of edge sites much more financially scalable. EHP, on the other hand, gets a flexible way to make profits from their idle machines. (*iii*) **Carbon-friendly**. The manufacturing of electronic devices is an energy-intensive process that usually dominates their lifetime carbon footprint [11]. By leveraging the unused hardware already manufactured, the crowdsourced edge cloud platform does not need new hardware from the manufacturer and therefore can reduce the carbon footprint significantly.

Seemingly attractive, but such platforms face tougher challenges in QoS. This attributes to the inherent uniqueness of the crowdsourced edge platform: all its infrastructure is built upon hardware out of the control of the platform. Those servers could connect/disconnect at any time or get into failure more frequently than a datacenter-level machine. Furthermore, the hardware capacity could vary severely across time. According to our best knowledge, there has been no study on how such platforms have been operating in the wild.

To demystify the status quo of the crowdsourced edge platform, we perform the first-of-its-kind measurement study on a large-scale in-the-wild **C**rowdsourced **E**dge computing **S**ervice **P**latform, namely C-ESP[2], that have been built and operated for nearly four years. C-ESP has deployed over 10,000 edge servers in total that are located across over 1,000 regions. To facilitate the service deployment, C-ESP requires services to be deployed in containerized manner. In total, C-ESP hosts over 100,000 containers.

As shown in Figure 1, our measurement takes a holistic QoS view from top to bottom: server (hardware), container (service), and user (request). Specifically, we collected the detailed usage traces of C-ESP, including the server available time sessions, container

[1] Abbreviated as edge platform in the following.

[2] PPIO Edge Cloud, Paiou Cloud Computing (Shanghai) Co., Ltd., https://www.ppio.cn.

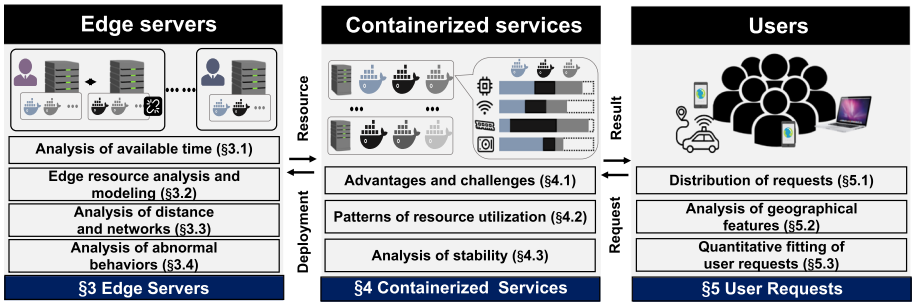

Figure 1 Article overview

resource usage, user requests distribution, etc. Through measurement and analysis, we seek to provide a holistic view of the metrics that impact QoS through the following key questions:

- What are the *quality* and *quantity* characteristics of the hardware resources provided by C-ESP's **edge servers**?
- What are the *stability* and *utilization* characteristics of the **containerized services** hosted on C-ESP?
- What are the *spatial* and *temporal* distribution characteristics of **user requests** handled by C-ESP?

Our in-depth measurements on those questions lead us to insightful observations and implications as follows.

Dynamic Nature of Edge Servers (Section 3.1) Edge servers exhibit frequent connections and disconnections, with approximately 8% of registered edge servers recording connect/disconnect events daily. Moreover, more than half of the online sessions have a duration of less than one hour. This poses challenges for the efficient utilization of ephemeral server time, especially considering the non-trivial deployment time of services. To address this, we propose two approaches: (i) deploying services with high deployment costs on long-running servers based on historical traces, and (ii) preemptively migrating services by predicting host server disconnection events.

Geographical Distribution and Edge Resources (Section 3.2) The availability of edge resources in a particular geographical location is correlated with the population and GDP of that area. Higher population or GDP corresponds to a greater number of edge servers and resources. Based on these findings, we develop a simple modeling generator to estimate the distribution of edge resources in any region. This tool can assist edge researchers and developers in evaluating their systems and algorithms in diverse and realistic geographical settings.

Impact of Geographical Distance on Edge Network Performance (Section 3.3) The performance of edge networks is intricately linked to geographical distance. As the distance between nodes increases, network latency exhibits a positive correlation. Conversely, network bandwidth shows a positive correlation with the reciprocal of geographical distance. While geographical distance does influence packet loss rate (PLR), it is not the primary factor. These observations shed light on the fundamental principles of edge computing, emphasiz-

ing the importance of reducing geographical distance for optimal network performance and higher QoS.

Service-Level Agreement (SLA) Violations and Causes (Section 4.1) The network is the primary cause of SLA violations, contributing to nearly 66% of the 34,091 recorded violations. On the other hand, devices lead to the most fines, as they often have a more significant impact on service quality. To mitigate costs associated with SLA violations, prioritizing device failure handling in operation and maintenance is crucial.

Heterogeneous Resource Usage of Containerized Services (Sections 4.2 and 4.3) Containerized services exhibit highly heterogeneous resource usage, which also varies significantly over time. This observation emphasizes the importance of deploying containerized services with different resource usage characteristics in a non-conflicting combination on servers. We identify fixed patterns that align well with different services, such as high-demand resources, peak periods, and rapid changes. By leveraging these patterns as tags for each service, we can easily cluster services into different types and better consolidate them on servers.

Mismatch between Request Generation and Available Edge Resources (Sections 5.1 and 5.2) We observe cases where certain areas generate a large number of requests but have limited available resources. Simply scheduling requests to nearby edge services can lead to unbalanced resource usage. Therefore, a globally resource-aware request scheduler is necessary to address this issue effectively.

Temporal Features of User Request Generation (Section 5.3) The number of user requests generated over time follows a Poisson distribution pattern. Based on our analyzed data, we have developed a statistical model that can simulate real-world user request generation. This model can assist edge researchers in evaluating their algorithms and systems in a more realistic simulation environment.

By uncovering these key insights, our research contributes to a deeper understanding of edge computing and provides practical implications for system design and optimization. In summary, this work makes the following contributions:

- We collected large-scale QoS data from a representative crowdsourced edge platform. According to our knowledge, this is the first study that investigates the status quo of such unique form of edge platforms.
- We perform a holistic, in-depth QoS analysis of the platform, which gains insightful results and implications for edge platforms, practitioners, and researchers.
- We implement a set of modeling generators to help readers understand and promote more solutions about the problems mentioned in this paper. They can be obtained from the following link: https://github.com/76481786/Flexible-Measurement-based-Modeling-Generators.

2 C-ESP

In cloud computing platforms, the role of cloud service providers has evolved in the business model, such as Google Cloud Platform (GCP) [12], Amazon Web Services (AWS) [13], and others. These providers construct centralized large-scale cloud computing infrastructures that offer services across a wide geographic area. However, C-ESP, as an edge comput-

ing platform, differs from them. It adopts a decentralized architecture to deploy computing infrastructure at the network edge, aiming to reduce the geographic distance between users and services.

By leveraging the concept of crowdsourcing, C-ESP integrates third-party dispersed resources at the network edge and establishes its own edge servers to ensure high-quality computing power. Additionally, C-ESP has built an edge computing network that covers nearly all areas of China. With over 10,000 edge servers deployed in more than 1,000 regions, C-ESP provides resources to Application Service Providers (ASPs). To the best of our knowledge, C-ESP is one of the largest enterprises applying crowdsourced edge computing in real-world business scenarios.

Moreover, C-ESP generates revenue by providing edge computing power to ASPs. It currently hosts services from numerous ASPs with tens of millions of users. To enable rapid deployment, resource isolation, and service compatibility, all these services adopt containerization technology. Typically, ASPs provide service containers along with their resource requirements, and C-ESP facilitates the deployment and operation of these service containers while ensuring compliance with SLAs.

In summary, C-ESP interacts with three key roles: (i) **EHPs:** C-ESP rents idle servers from EHPs and compensates them based on the resources and server connection time. C-ESP also establishes reward and penalty rules with EHPs to adjust the platform's resource supply and demand. (ii) **ASPs:** C-ESP generates revenue by deploying ASPs' services on edge servers. Additionally, C-ESP establishes SLAs with each ASP individually, defining service indicator thresholds and corresponding fines for violations. (iii) **Users:** Users access ASPs' services deployed on C-ESP through their end devices. While users do not have a direct monetary relationship with C-ESP, their request fluctuations can cause dynamic changes in resource requirements.

3 Exploring edge servers

C-ESP operates a diverse array of edge servers from various sources, including (i) its own internally built and deployed computing servers, (ii) large-scale idle computing servers leased from external organizations, and (iii) smaller servers rented from individual users. In this section, we delve into the characteristics of edge servers, beginning with an examination of their available time. Subsequently, we gather data from multiple dimensions to comprehensively quantify the resources provided by edge servers. Lastly, we explore any anomalous behaviors observed in these servers.

3.1 Analysis of available time

C-ESP utilizes a crowdsourcing model to leverage idle servers from third parties. However, the inclusion of these servers, which are not fully controlled, introduces additional uncertainty to the platform. Thus, we place particular emphasis on examining the available time of edge servers operating within the crowdsourcing model. Through collaboration with C-ESP, we have collected server connection and disconnection records spanning 418 days, comprising a total of 185,698 records. The collected data includes server IDs, timestamps, connection/disconnection events, and server resource information.

Figure 2(a) illustrates the daily number of server connections and disconnections within C-ESP, revealing a relatively balanced distribution and ensuring a stable server count on

the platform. On average, 8.53% of the total number of servers have connection records per day, while 8.30% of the total number of servers have disconnection records. However, it is important to note that **frequent changes in server status may necessitate frequent adjustments in service deployment**, resulting in increased costs compared to traditional platforms. To maintain QoS stability, C-ESP employs a bonus-based reward and penalty mechanism for maintenance. As service resource requirements fluctuate regularly (as demonstrated in Section 4.2), C-ESP applies higher bonuses/penalties for servers connecting/disconnecting during periods of high resource demand, such as afternoon and evening peaks. This helps minimize server status changes and subsequently adjusts service deployment during idle periods.

Furthermore, we analyze the distribution of online time among the servers in our collected data, depicted in Figure 2(b). Notably, over half of the recorded online time instances are less than one hour, with an average online time of 337,782 seconds (approximately four days) across all servers. Considering the time required for service deployment, **a significant number of short-term connections can lead to inefficient service deployment**, whereby services are deployed but become unavailable due to subsequent server disconnections. In response, C-ESP is planning to develop a mechanism for predicting whether connected servers will soon disconnect and whether disconnected servers will soon reconnect. These prediction results can enhance service deployment and migration policies, ultimately improving QoS.

Finally, we examine the relationship between server resources and online time, as illustrated in Figure 2(c). The resource amount of each server is calculated by summing the four metrics of CPU, disk, memory, and bandwidth after normalization using the min-max method. The analysis reveals a trend where servers with higher resources tend to have longer online durations. This can be attributed to low-resource servers often originating from small organizations or individuals who may lack long-term planning in server management.

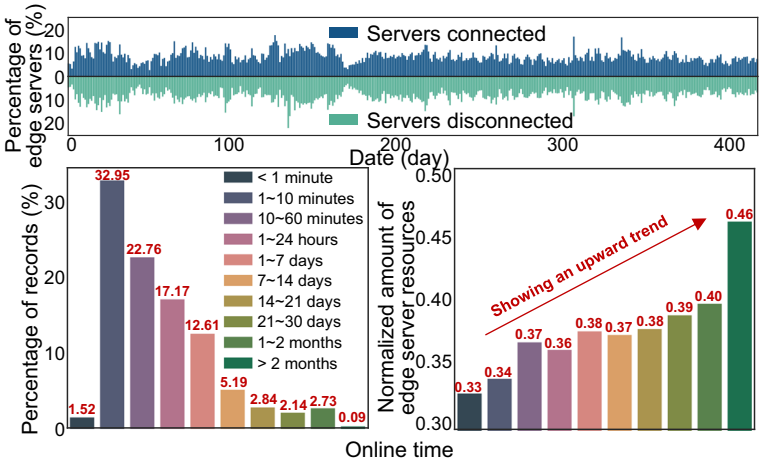

Figure 2 The analysis of (a) daily connect/disconnect distribution (top), (b) online times distribution (left) and (c) average resource distribution (right)

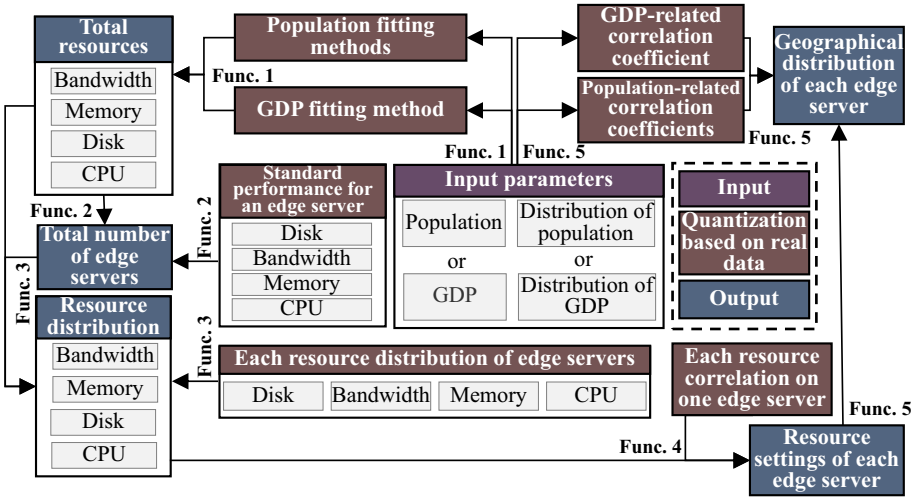

Figure 3 Edge Server Model Generation (*ESMG*) architecture

3.2 Edge resources analysis and modeling

In the domain of analyzing edge resources for practical applications or modeling purposes, two significant barriers have impeded progress. The first barrier concerns the scarcity of edge resource datasets with reference value. Commercial constraints often hinder companies from sharing comprehensive datasets, and even when such datasets are made available, they may not be sufficiently flexible to support relevant research and practical applications. The second major challenge in effectively modeling and quantitatively analyzing edge resource datasets stems from the high degree of heterogeneity in terms of hardware resources and geographical distribution among edge servers. This heterogeneity poses substantial challenges for researchers aiming to uncover meaningful patterns in the data.

Fortunately, `C-ESP` provides a research-worthy edge resource dataset that overcomes these challenges. Leveraging this dataset, which encompasses 13,036 edge servers and includes key parameters such as CPU, bandwidth, memory, disk, latitude, and longitude, we are able to explore the impact of heterogeneity in real-world business scenarios and discover potential patterns within the `C-ESP` architecture. Through an in-depth analysis, we identify a set of quantifiable patterns that underlie the `C-ESP` ecosystem. These patterns are synthesized into a coherent architecture known as the Edge Server Model Generation (*ESMG*).

The foundation of *ESMG* lies in six distinct direct mapping functions, developed through quantitative analysis of the information contained within each dimension of the dataset. By utilizing these quantitative models, *ESMG* is capable of flexibly representing the various attributes of edge servers and geographic information within a specific target region. To demonstrate the quantifiable patterns and their effects in *ESMG*, we focus on Figure 3 ("Func." is an abbreviation for "Function").[3]

Function 1: Edge resources mapping In *ESMG*, Function 1 utilizes the total GDP or population of a region as input to estimate the total amount of various resource types available

[3] *ESMG* focuses on providing a modelling approach that approximates real business application scenarios rather than an optimal deployment solution for edge servers.

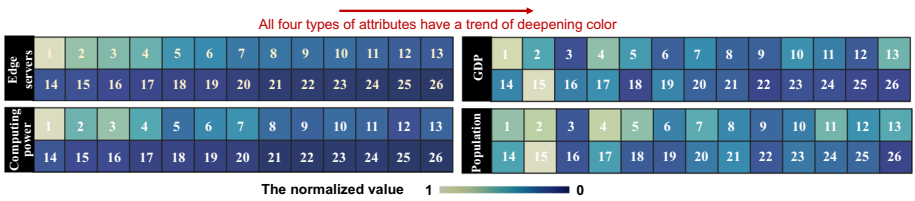

Figure 4 Distribution of edge servers, computing power resources, GDP, and population for 26 provinces under the same order

in that region. Figure 4 illustrates the distribution of computing resources and the number of edge devices across the 26 provinces in China. Notably, we observe a close positive correlation between the resource distribution and the population/GDP of each province. Hence, population and GDP serve as the entry point for *ESMG*, and we have quantified this correlation as shown in Figure 5. The circular and square markers in the data represent special regions. Circular markers denote areas with low resources but high population and GDP characteristics, while square markers denote areas with high resources but low population and GDP. These regions were not included in the fitting process as they represent exceptional cases. For example, square regions often provide computing power at a lower cost, while circular regions represent the opposite. Apart from these special regions, resource amounts in general regions are linearly related to population or GDP. To capture the overall trend, a linear fit is performed after removing these two outliers. Finally, we obtain the linear fit equations, which are presented in Table 1. The parameters in Table 1 include GDP (billion dollars), population (10^8 people), CPU (cores), bandwidth (Bytes/s), disk (Bytes), and memory (Bytes). Additionally, we use r in Table 1 to represent the Pearson correlation coefficient, which quantifies the goodness-of-fit.

Function 2: Edge server number mapping Function 2 processes the total amount of each resource type in a region to estimate the number of edge servers in that region. The process begins by calculating the mean values based on the data from 13,036 edge servers, resulting in an average edge server performance of 11.510 cores for CPU, 782.759 MB/s for bandwidth, 17.519 GB for memory, and 2456.536 GB for disk. Subsequently, four estimates of the number of edge servers are obtained by dividing the total amount of each resource type in the region. Finally, Function 2 outputs the mean value of these four estimates.

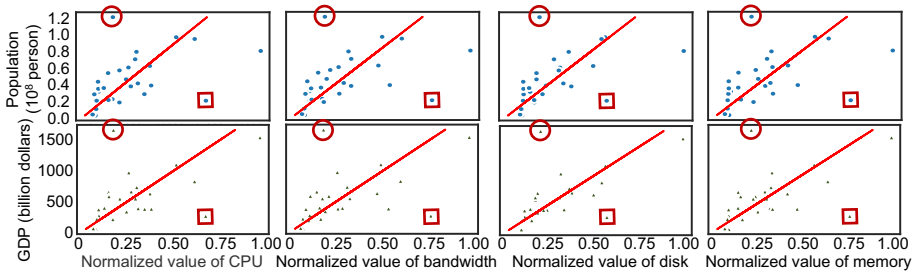

Figure 5 The correlation of various types of resources with population and GDP. In addition to the circular and square markers representing special regions, the resource quantities in other areas are linearly associated with population or GDP

Table 1 Linear fitting equations for population or GDP with each resource

x	y	Linear fitting equation
Population	CPU	$y = 1.324 \times 10^{-4}x - 1.595 \times 10^{3}, r = 0.73$
	Disk	$y = 3.576 \times 10^{7}x - 6.052 \times 10^{14}, r = 0.76$
	Memory	$y = 2.104 \times 10^{5}x - 2.401 \times 10^{12}, r = 0.74$
	Bandwidth	$y = 9.145 \times 10^{3}x - 1.035 \times 10^{11}, r = 0.72$
GDP	CPU	$y = 1.206 \times 10^{1}x - 1.341 \times 10^{3}, r = 0.84$
	Disk	$y = 3.316 \times 10^{12}x - 5.657 \times 10^{14}, r = 0.89$
	Memory	$y = 1.906 \times 10^{10}x - 1.947 \times 10^{12}, r = 0.84$
	Bandwidth	$y = 8.086 \times 10^{8}x - 7.387 \times 10^{10}, r = 0.80$

Function 3: Edge resource distribution mapping Function 3 in *ESMG* takes the total number of each resource type and the number of edge servers in a region as input to estimate the distribution of each resource type in that region. To accurately model the heterogeneity of edge servers, the resource distribution must consider the heterogeneity distribution of edge servers for each resource type. Function 3 quantifies this distribution, as depicted in Figure 6, enabling the determination of resource distributions based on the quantified resource heterogeneity of real edge servers.

Function 4: Edge server resource configuration mapping Function 4 in *ESMG* takes the resource distribution of each resource type in a region as input to estimate the resource configuration for each edge server. While the distribution of each resource type is quantified in Function 3, the correlations between different resource types remain unclear. For instance, an edge server with high CPU resources is likely to have abundant memory resources as well. To capture these correlations, *ESMG* employs Spearman correlation coefficients to quantify the relationships between various resource types within an edge server, as shown in Table 2. The coefficients reveal that the correlation between CPU, bandwidth, and memory is higher compared to the correlation between disk and other resources. Thus, it is crucial to closely

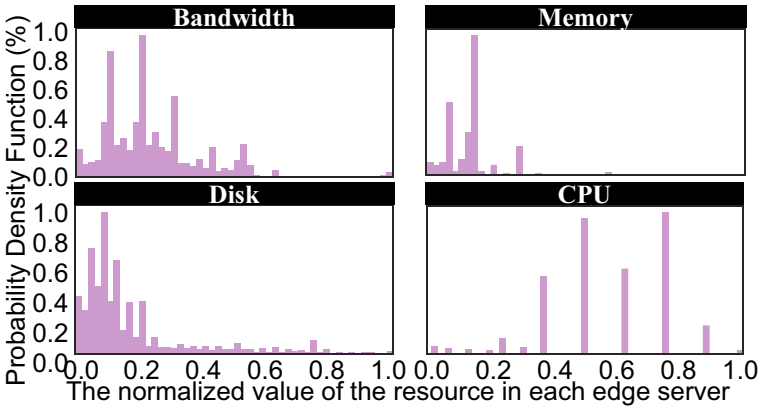

Figure 6 Probability density distributions (normalized) of four types of resources (bandwidth, memory, disk, CPU) on each edge server

Table 2 Spearman correlation coefficient between different resources of edge servers

	CPU	Disk	Memory	Bandwidth
CPU	1	0.15	0.42	0.42
Disk	0.15	1	0.29	0.36
Memory	0.42	0.29	1	0.44
Bandwidth	0.42	0.36	0.44	1

approximate the resource correlation constraints presented in Table 2 when determining the resource configuration.

Function 5: Edge server geographic distribution mapping. Function 5 in *ESMG* takes the geographic distribution of GDP or population and the edge server resource allocation in a region as input to estimate the geographic distribution of edge servers within that region. Based on the findings in Table 1, we observe a relationship between the distribution of edge servers and the distribution of GDP or population, although the distributions are not entirely identical. Therefore, in addition to using the distribution of GDP or population to map the distribution of edge servers, Function 5 incorporates random adjustments to refine the heterogeneity. These random adjustments must adhere to the constraints outlined in Table 1.

It is worth noting that the functionality of *ESMG* extends beyond the aforementioned direct mapping relationships. For example, users can utilize *ESMG* to investigate edge server heterogeneity in business scenarios by simply inputting the GDP or population of a region. By sequentially invoking Functions 1 to 5, a series of inputs and outputs can generate edge server configurations for regions of any size. We provide the necessary supporting data and the implementation code of *ESMG* in our **open-source project**, as mentioned in Section 1.

3.3 Analysis of geographical distance and networks

Compared to the cloud computing paradigm, one of the fundamental aspects of edge computing is the deployment of edge servers in geographically distributed locations, enabling performance optimization through reduced geographic distance between services and users. Consequently, we place special emphasis on understanding the impact of geographical distance to quantitatively elucidate the performance changes brought about by adopting the edge computing paradigm. As geographical distance primarily affects communication transmission while exerting little direct influence on computation, caching, and storage, our analysis focuses on the communication aspect of data collection. Specifically, we analyze the dataset from `C-ESP`, which encompasses edge servers from various Internet Service Providers (ISPs) and provinces, resulting in a total of 4896 data entries. The dataset includes network speed (upstream bandwidth), delay, packet loss rate (PLR), longitude, latitude, and ISP information. Among the ISPs, the major providers are ISP-A, ISP-B, ISP-C, and ISP-D, with the first three accounting for the majority of the market share.

We partition the dataset into four groups based on ISPs, namely ISP-A (1832 items), ISP-B (1349 items), ISP-C (1676 items), and ISP-D (39 items).[4] Figure 7 illustrates that the network performance of each ISP exhibits a similar variation pattern, maintaining consistent characteristics. Moreover, network delay and speed display a regular trend of variation with

[4] Due to the limited amount of data for ISP-D, the credibility of its performance analysis is low.

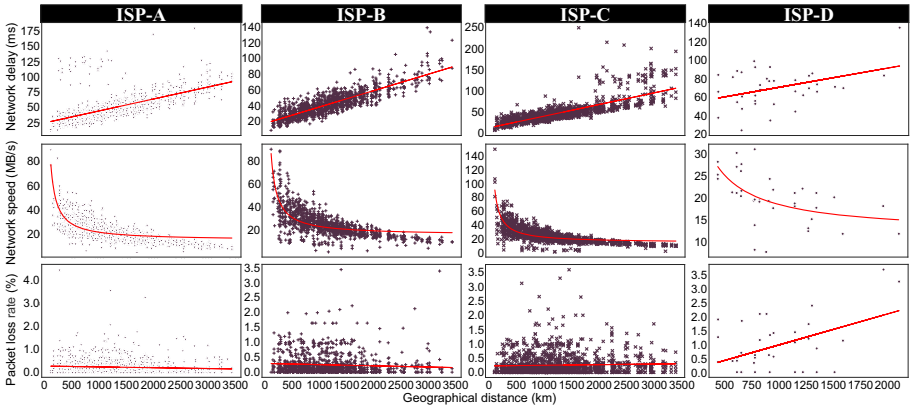

Figure 7 Analysis of edge server network performance and geographical distance

respect to geographical distance, while PLR is relatively unaffected. However, it is worth noting that PLR is not entirely independent of geographical distance. Although the wired transmission distance does exert some influence, its impact is considerably smaller compared to factors such as congestion and buffer overflow [14]. To provide further quantitative insights, we derive fitting equations that capture the relationship between each network performance parameter and geographical location, as presented in Table 3. In the fitting equations, x represents the geographical distance, y represents the network performance, and r denotes the Pearson correlation coefficient. These fitting equations reaffirm the association between each network performance parameter and geographical location.[5]

In summary, our findings indicate that (i) network delay exhibits a positive correlation with geographical distance; (ii) network speed (bandwidth) displays a positive correlation with the reciprocal of geographical distance, and (iii) geographical distance is not the primary factor influencing PLR. Our observations provide support for the core concept of edge computing, where the deployment of geographically distributed edge servers can optimize network performance and enhance QoS by minimizing geographical distance. As hardware devices approach the upper limit of their development, this trend will continue to drive improvements in network performance across the edge landscape.

3.4 Analysis of abnormal behavior

Given the distributed deployment of edge servers, their operation and maintenance become more challenging, which, in turn, negatively impacts QoS. Consequently, we place significant emphasis on identifying abnormal behaviors exhibited by edge servers. To achieve this objective, we collected 428,160 operation and maintenance data points from `C-ESP` spanning 139 days. Each data point includes the server ID, the number of abnormal behaviors observed on a given edge server in a day, and the corresponding category (as shown in Table 4).

Since detection occurs every five minutes in `C-ESP`, each abnormal behavior can be considered to last for a duration of five minutes. As indicated in Table 4, we observe that the average value for *Machine line drop times* is notably high. This phenomenon arises from `C-ESP`'s use of link aggregation, which enables a server to have multiple network connec-

[5] Since our dataset does not contain the actual routing data of the transmission, we calculate the straight-line distance by computing the geographic coordinates as an approximate substitute.

Table 3 Quantify the relationship between geographical distance and networks

Network attribute	ISP	Fitting equation
Network delay	ISP-A	$y = 0.01959x + 23.89529, r = 0.67$
	ISP-B	$y = 0.02091x + 17.21486, r = 0.85$
	ISP-C	$y = 0.02725x + 12.20337, r = 0.78$
	ISP-D	$y = 0.02069x + 50.28237, r = 0.40$
Network speed	ISP-A	$y = 6831/x + 14.60886, r = 0.73$
	ISP-B	$y = 7629/x + 15.45950, r = 0.68$
	ISP-C	$y = 8258/x + 14.39965, r = 0.73$
	ISP-D	$y = 6653/x + 11.71027, r = 0.56$
Network PLR	ISP-A	$y = -0.00004x + 0.26561, r = 0.08$
	ISP-B	$y = -0.00004x + 0.29397, r = 0.07$
	ISP-C	$y = 0.00002x + 0.22678, r = 0.04$
	ISP-D	$y = 0.00111x - 0.12556, r = 0.47$

tions. Consequently, if a portion of the network lines become unavailable, it contributes to the count of *Machine line drop times*, while complete server unavailability is recorded as *Offline times*. Moreover, 7.85% of *Machine line drop times* is recorded as 288, representing an entire day. Therefore, although 76.41% of the data reflects a value of 0, the average value reaches 33.92. Additionally, the occurrence of *High I/O load times* is worth highlighting. Despite the availability of sufficient disk resources for `C-ESP` (as demonstrated in Section 4.2), the average value for *High I/O load times* remains high. This observation suggests that, apart from disk storage space, due consideration needs to be given to disk I/O when optimizing system performance.

Subsequently, we employ correlation coefficients to quantify the relationships between different abnormal behaviors, with the aim of identifying their underlying causes. As illustrated in Figure 8, most abnormal behaviors exhibit little correlation with one another, indicating the challenge in predicting the occurrence of abnormal behaviors through correlation alone. However, we identify two pairs of abnormal behaviors that display a strong correlation: (*i*) *Abnormal IP change times* correlates with *Machine line drop times*, suggesting the importance of verifying the normalcy of IP addresses after disconnection and reconnection events. (*ii*) *Offline times* correlates with *Unavailable time*, highlighting the frequency of server offline instances as a primary cause of unavailability.

Table 4 The distribution of abnormal behavior

Abnormal behavior	Average	Percentage of zero
High CPU load times	0.532	95.57 %
High I/O load times	6.464	80.20 %
High latency times	4.084	77.18 %
Offline times	0.150	93.48 %
Machine line drop times	33.92	76.41 %
Abnormal IP change times	1.727	91.52 %
Unavailable time (in seconds)	1464	93.52 %

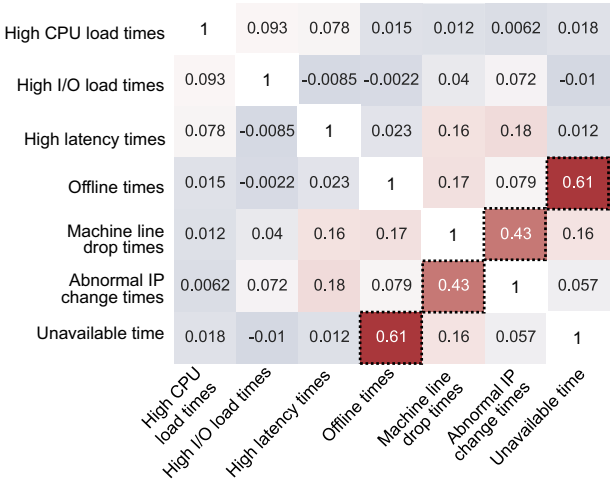

Figure 8 Spearman correlation coefficient of edge servers among deferent abnormal behaviors

4 Exploring containerized services

In terms of containerized services, C-ESP hosts services from partner ASPs. To improve the hardware compatibility and rapid deployment of services, C-ESP extensively uses containerization technology, and the container images of services are usually made by ASPs. Based on the above, we collected data about containerized services in containers and servers. Relying on these data, this section provides a detailed analysis of service stability and service resource utilization.

When it comes to containerized services, C-ESP hosts services provided by partner ASPs. To enhance hardware compatibility and facilitate rapid deployment, C-ESP extensively employs containerization technology, with the container images typically being created by the ASPs. Leveraging the collected data on containerized services within containers and servers, this section presents a comprehensive analysis of service stability and resource utilization.

4.1 Analysis of stability

In contrast to the centralized operation and maintenance approach of cloud computing, the distributed deployment characteristic of edge computing poses challenges to service reliability. However, given that C-ESP's core business revolves around providing high-quality infrastructure resources to ASPs, ensuring service reliability becomes paramount. To this end, we collected data over a period of approximately two months, with the assistance of C-ESP. Since C-ESP offers various Service-Level Agreement (SLA) guarantees to ASPs, where different types of SLA violations result in varying fines imposed by C-ESP, our reliability measurements primarily focus on two key metrics: (*i*) the number of failures attributed to each factor, aimed at identifying the factors most likely to cause failures, and (*ii*) the number of fines resulting from failures attributed to each factor, intended to explore the factors posing the greatest threat to stability.

Due to the confidentiality of SLAs as proprietary business information, we are unable to provide full details. Hence, we generalize the data into four categories: (*i*) **Network**, which

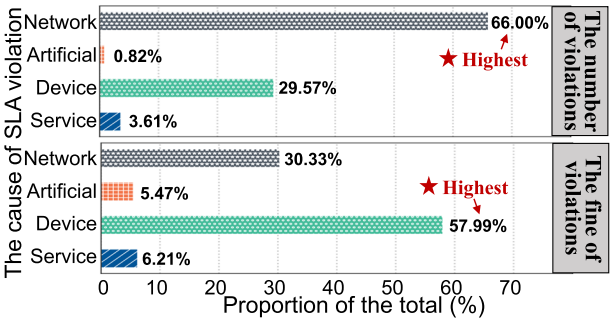

Figure 9 Analysis of (a) the number of violations of service SLA guarantees (top) and (b) the corresponding amount of fines incurred (bottom)

encompasses factors such as network connection and communication; (*ii*) **Artificial**, which includes factors like adjustments and testing conducted by engineers; (*iii*) **Device**, which entails factors such as server disconnections by the owner and hardware failures; and (*iv*) **Service**, which covers factors related to the configuration and operation of the container.

As depicted in Figure 9, the causes of SLA violations are categorized into four groups. Among these, **the network is the primary factor responsible for SLA violations**, accounting for 66% of the overall violations. This phenomenon can be attributed to several reasons: (*i*) C-ESP's edge servers are deployed in remote areas with unstable networks, making them more prone to SLA violations; (*ii*) C-ESP utilizes link aggregation on certain edge servers to aggregate bandwidth resources, resulting in a larger number of network links compared to the number of servers, thereby increasing the likelihood of SLA violations.

Interestingly, **device-related issues contribute to the most significant fines resulting from SLA violations**. Although the network is responsible for the majority of SLA violations (30.33% of fines), the device factor accounts for 29.57% of the total SLA violations, yet results in 57.99% of the fines. Upon closer investigation, we find that partial line disconnections are the most common cause of SLA violations in the network category. Such failures lead to the unavailability of only a portion of the bandwidth resources of edge servers. In contrast, most device failures render the entire server's resources unavailable, resulting in more severe SLA violations and consequently larger fines. Therefore, it is crucial to prioritize addressing device failures, as they pose the greatest threat to ensuring SLA guarantees.

Consequently, proactive measures need to be taken to address device failures. On one hand, optimizing server operation and maintenance practices is advised to prevent SLA violations. On the other hand, it is recommended to establish a recovery mechanism within the edge cluster to minimize the impact of failures.

4.2 Overall resource utilization

Container technology is a lightweight virtualization approach that enables developers to package applications and their dependencies, facilitating easy migration and deployment across servers. Containers offer several advantages, including resource isolation between services, rapid migration and deployment, and improved compatibility. These benefits make containers well-suited for edge computing, which has led to increased attention in this domain. Given the multi-tenant nature of C-ESP, frequent service deployment adjustments, and the heterogeneous nature of edge servers, containerized services provide a natural solution to

address these challenges. To this end, C-ESP utilizes Docker [15] as its container runtime and has developed its own container management system, drawing inspiration from the architecture of Kubernetes [16].

Considering that most containerized services deployed in C-ESP serve user requests, user behavior tends to exhibit periodic patterns over time. These changing patterns directly influence the resource requirements of the services, potentially leading to time-dependent resource requirement patterns for containerized services. To capture the actual performance of containerized services, we collected monitoring data from over 10,000 edge servers and the service containers running on them, with the assistance of C-ESP.

The collected data comprises monitoring logs from over 10,000 edge servers and their corresponding service containers over a single day. Each edge server and container generate logs every 5 minutes, resulting in a total of 288 logs per day. The data includes four resources: memory, bandwidth, disk, and CPU. From this dataset, we extract resource requirements and real-time utilization changes for each resource, as depicted in Figures 10, 11, and 12. To enhance the data's persuasiveness, we overlay the data from all edge servers and containers. The solid line in the figures represents the mean, while the shading indicates the variance.

First, Figure 10 illustrates the resource utilization of containerized services on the edge servers. It can be observed that memory and disk utilization remains relatively stable throughout the day, suggesting that these resources require minimal reservation to handle potential application peaks. In contrast, CPU and bandwidth requirements exhibit significant variations over the course of a day, with bandwidth utilization demonstrating the greatest variability. Moreover, a strong correlation between changes in CPU and bandwidth utilization can be observed. There are two prominent peaks in CPU and bandwidth utilization within a day: the afternoon peak from 11:00 to 14:00 and the evening peak from 19:00 to 23:00. The evening peak exhibits the highest utilization (1.74 times and 2.43 times the daily average for CPU and bandwidth, respectively), with both resources significantly higher than during nighttime.

The findings highlight that although CPU and bandwidth exhibit volatility over time, discernible patterns exist. In practical scenarios, particular attention should be given to CPU and bandwidth resource utilization during the afternoon and evening peak hours, as these periods are more prone to resource shortages. Additionally, since CPU and bandwidth utilization fluctuate significantly, improving their utilization poses greater challenges compared to disk and memory utilization. There is a trade-off between efficiency and quality for CPU and bandwidth resources: (*i*) Reserving a large number of resources to handle peak loads guarantees service quality but results in resource waste. (*ii*) Utilizing idle resources and releasing them before peak periods improves overall resource utilization. However, the uncertainty sur-

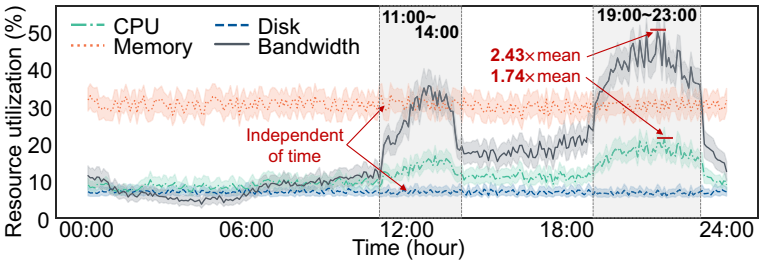

Figure 10 Average resource utilization (including CPU, disk, memory and bandwidth) of all edge servers in a day

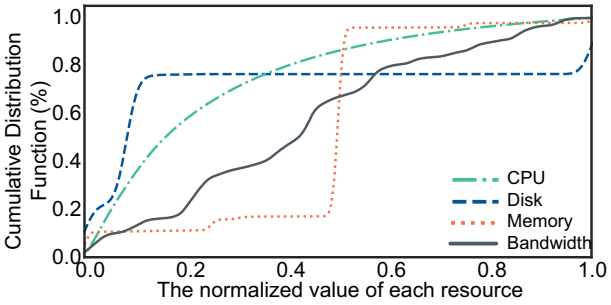

Figure 11 Cumulative distributions (normalized) of four types of resources (CPU, disk, memory, bandwidth) required by containerized applications

rounding peak arrival times increases the likelihood of resource shortages. Hence, **CPU and bandwidth resources face a more pronounced trade-off between efficiency and quality.**

4.3 Utilization patterns of different services

The heterogeneity of resource requirements for containerized applications is evident, as depicted in Figure 11. To further investigate this, we analyze the actual resource utilization variations for containers from 18 different ASPs in Figure 12. Given the multi-tenant nature of C-ESP, the containers are grouped into 18 categories based on their respective ASPs, and the resource utilization of each group is illustrated. The figure clearly demonstrates that the resource requirement distribution patterns for different container types are highly distinct, posing challenges for efficient service container deployment on servers. Specifically, this heterogeneity may result in the depletion of specific resources on a server while other resources remain underutilized. Such a situation can hinder the deployment of new containers due to the depletion of a particular resource, leading to resource wastage.

As illustrated in Figures 10 and 12, the resource isolation capability of containers divides idle resources into two categories, necessitating considerations from two perspectives for optimizing resource utilization. Firstly, the idle resources on the edge servers are considered **direct resources** that can be reallocated, allowing for improved resource utilization through

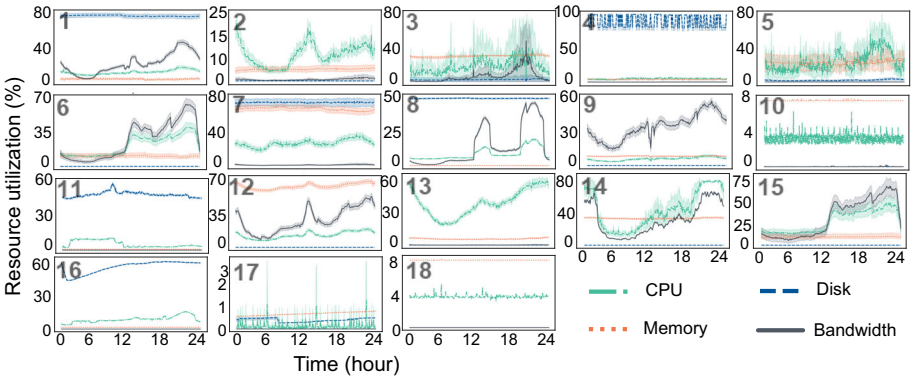

Figure 12 The resource utilization changes of multiple types of service containers in a day

optimized container deployment scheduling. Secondly, the idle resources within edge containers are regarded as **indirect resources** that cannot be directly reallocated. Due to the resource isolation capability of containers (where resource allocation configurations can be set through container engines), these resources need to undergo transformation through container scaling policies before they can be effectively utilized. Alternatively, directly utilizing these resources by increasing the load through request offloading policies is also a feasible approach.

Thus, containerization not only provides resource isolation but also introduces **a new challenge of matching server-container resource requirements**, i.e., how to arrange containers on edge servers to fully utilize available resources. Furthermore, the distribution of different containers for a single resource type is often uneven, leading to **a new challenge of matching container-container resource requirements**. This challenge pertains to deploying containerized services with complementary resource requirements on an edge server to enhance resource utilization.

The problem of server-container resource requirement matching is particularly challenging in the edge computing paradigm due to distributed server deployment, resource limitations, and server heterogeneity. While similar challenges exist in centralized cloud computing paradigms [17–19], `C-ESP` employs a recommendation system to identify mismatched server-container pairs on a daily basis. Engineers leverage their experience and the recommendation system's results to make service deployment adjustments. However, currently, there is no fully automated and reliable solution available for this problem within the `C-ESP` scenario.

To address this issue effectively, it is crucial to determine whether two services complement each other and quantify resource fluctuations. Taking Figure 12 as an example, we propose a classification method that summarizes service resource characteristics across six dimensions:

- **High-demand resources**. It represents the type of resource in high demand, such as disk (1, 4, etc.),[6] memory (10, 12, etc.), CPU (2, 13, etc.), bandwidth (6, 9, etc.);
- **Peak period**. It represents the peak periods of the service, such as the afternoon peak (12), evening peak (1, 14, etc.), or both (6, 15, etc.).
- **Rapid change**. It represents the rate of change in resource requirements, with some changing rapidly (8, 15, etc.) and others changing slowly (1, 9, etc.).
- **Time-dependent**. It indicates whether the resource requirements vary with time, with some being time-dependent (6, 14, etc.) and others not (4, 7, etc.).
- **Predictability**. It indicates whether different containers of the same service exhibit consistent resource utilization, as indicated by the shaded range in Figure 12. Some services show consistency (1, 6, etc.), while others do not (3, 5, etc.).
- **Resource correlation**. It indicates whether different resource changes are correlated, for example, some services exhibit similar CPU and bandwidth changes (6, 14, etc.), while others do not (2, 13, etc.).

To provide further quantification, we offer a set of quantitative methods for these six dimensions as illustrated in Table 5. These dimensions can be used as labels to categorize and analyze various service containers, enabling comprehensive quantification of resource demand patterns and informing modeling and algorithm design.

Moreover, to provide a more realistic representation of container resource fluctuations, we offer a model generator for containerized services as part of our **open source project** (introduced in Section 1). By specifying a list of desired container types, as depicted in

[6] The numbers here and below correspond to the indexes in Figure 12.

Table 5 Quantify each classification dimension

Dimension	Quantitative approach
High-demand resources	The **proportion** of time in a day as the highest utilized resource type.
Peak period	The **ratio** of the average resource utilization for each hour to the daily average.
Rapid change	The **absolute average slope** of the line formed by each sample point and the previous sample point.
Time-dependent	**Pearson correlation coefficient** [20] between resource utilization and time.
Predictability	**Variance** of resource utilization for different containers of the same service.
Resource correlation	**Multiple correlation coefficient** [21] between each resource and other resources.

Figure 12, the generator can simulate resource utilization fluctuations for each resource type. The generated data incorporates randomness while adhering to the distribution observed in real data.

5 Exploring user requests

In addition to ASPs, C-ESP also needs to pay attention to the users. These users use the services of ASP by accessing the edge server deployed in C-ESP and generating a large number of requests to the edge server. We first analyze the characteristics of servers and users, then analyze the geographical distribution of requests and resources, and finally carry out quantitative modeling for the characteristics of requests, hoping to provide a basis for other research teams' research.

5.1 Distribution of requests

At present, C-ESP hosts more than a dozen different types of ASPs. For our analysis, we focus on the service log of a specific ASP as an illustrative example. This log comprises one hour of data per day (at different periods) spanning six days, resulting in a total of six hours of data. The dataset contains 10,159,851 logs from 96,209 users, with these requests being processed by 2,359 edge servers. Specifically, each data entry includes: (i) the approximate geographic location of the request sender (user); (ii) the approximate geographic location of the request receiver (edge server); (iii) the request generation time; and (iv) unique identification for both the request sender and receiver. It should be noted that the geographic locations provided in the data are approximations based on IP addresses, ensuring the privacy of user information.

To begin our analysis, we calculate the number of requests sent/received by each user/edge server during the data collection period, aiming to uncover the distribution patterns of requests from both the sender and receiver perspectives. Subsequently, we sort each user/edge server based on the number of requests sent/received, from the smallest to the largest, as depicted in Figure 13. Notably, we employ a logarithmic transformation for the y-axis in Figure 13. The figure reveals that the request distribution among users follows an exponential trend,

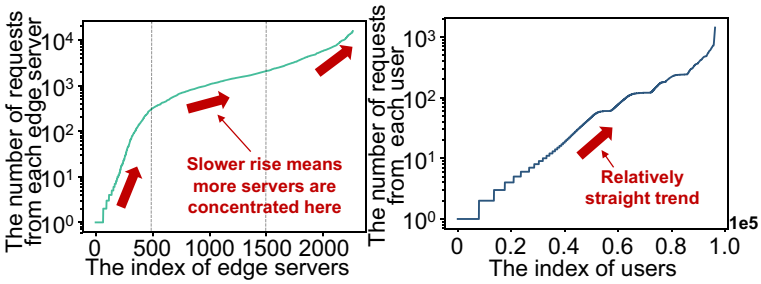

Figure 13 Frequency analysis of (a) edge servers (left) and (b) users (right)

whereas the request distribution among edge servers appears to be more balanced. This distinction arises from the natural distribution of requests among users, which remains unaffected by human intervention, while the distribution among edge servers is influenced by server configurations and load-balancing policies. Therefore, Figure 13 uncovers a significant phenomenon: **the number of requests per user in the collected data follows an exponential distribution.**

5.2 Analysis of spatial features

Based on the observations from Figure 14, we notice that **the number of resources in each area does not align with the number of generated requests**. Some areas have a surplus of resources but generate few requests, while others experience the opposite scenario. This imbalance poses a challenge to efficiency since processing only locally generated requests in each area would result in underutilization of resources. Moreover, certain areas generate requests without deploying the corresponding types of services. To address this, it becomes necessary to develop request scheduling algorithms that can effectively distribute and schedule requests from different areas. As depicted in Figure 14, the inclusion of requests from different areas leads to a more balanced ratio between the number of requests and available resources in each area, thus validating the effectiveness of C-ESP's request scheduling algorithm.

In summary, unlike the centralized mode of cloud computing platforms, edge platforms face the challenge of **matching resource distribution with request distribution** due to the distributed nature of resources.

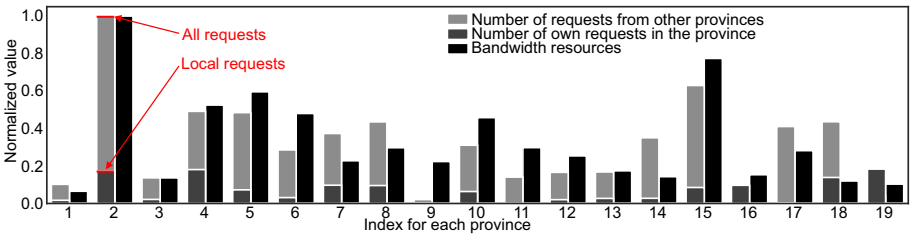

Figure 14 Comparison between the number of requests received and the amount of resources in each area

Table 6 Poisson distribution parameters

Index	λ	N	C	Index	λ	N	C
1	474.650	35688	0.0133	4	438.087	33699	0.0130
2	499.511	34449	0.0145	5	427.671	34770	0.0123
3	473.499	35074	0.0135	6	470.102	34314	0.0137

5.3 Analysis of temporal features

Evaluating system performance often requires simulation modeling of requests since acquiring a large-scale dataset of real-world end-users and their requests for laboratory-level systems is challenging. Thus, it is valuable to share insights into user request modeling based on the quantitative analysis of real large-scale data. Initially, we divide the collected data into six parts based on the dates and process them separately. The Poisson process, widely employed to model the arrival time of events in a system [22–26], is used to fit the request distribution, represented as $P(X = k) = \frac{\lambda^k}{k!}e^{-\lambda}, k = 0, 1, \cdots$.

As described above, once λ is determined, the Poisson distribution can be computed. To analyze λ, we introduce N to represent the number of users. We assume that λ is a constant multiple of N, i.e., $\lambda = C \times N$, which we validate. We calculate the average for each group of data (as presented in Table 6) and utilize it as λ for fitting, as illustrated in Figure 15. Furthermore, we count the number of users N in each of the six collected datasets and incorporate N with λ into $\lambda = C \times N$ to calculate C, as shown in Table 6. Remarkably, we find that the independently obtained C values from the six collected datasets are consistent with each other, validating our assumption that $\lambda = C \times N$. Finally, we take the mean value of C obtained from the six collected datasets as the final result, yielding $\lambda = 0.0134 \times N$.

In system design and theoretical research, it is often challenging to attract a significant number of real users to participate. Therefore, it becomes necessary to simulate and model requests to evaluate system performance. To address this need, we provide a user request generator in the **open source project** (introduced in Section 1) that mimics users' generated requests. By specifying the number of users to be served, the generator can produce the request patterns for each user. Moreover, the generated user request data conforms to the distribution patterns observed in Figures 13 and 15.

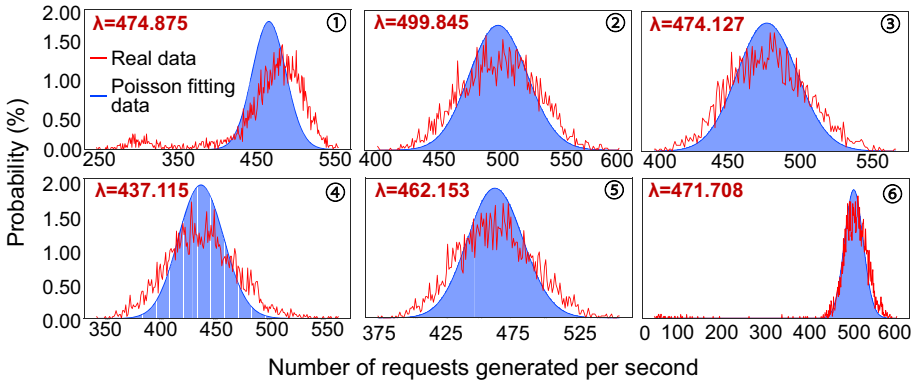

Figure 15 Fitting analysis based on Poisson distribution

6 Related work

Edge computing has emerged as a prominent research topic in both academia and industry. Researchers (i) have explored edge computing architectures and edge-cloud collaborative architectures, while enterprises (ii) have begun adopting edge computing and developing business-oriented edge platforms. Despite these efforts, there is a lack of comprehensive large-scale measurement studies that examine edge computing in real-world business application scenarios from multiple dimensions, including services, servers, and requests.

Edge server Several measurement studies have focused on characterizing server performance by analyzing network bandwidth [27], traffic [28], latency [29, 30], resource utilization [31], and robustness [32, 33]. However, most of these studies primarily focus on cloud computing platforms and do not consider the distributed deployment and performance heterogeneity of edge computing servers. Recently, a study investigated the network latency, throughput, and Quality of Experience (QoE) of edge servers [34]. However, there is still a lack of large-scale measurements on edge servers in crowdsourced edge platforms.

Containerized service With the increasing network requirements of services like IoT, streaming media, and cloud gaming, these services are transitioning from cloud computing to edge computing [35–39]. To cater to the needs of other ASPs, some enterprises are building edge platforms with containerized service capabilities from the perspective of Platform as a Service (PaaS), such as KubeEdge [40], OpenYurt [41], and Baetyl [42]. Nevertheless, there is a dearth of measurement studies that specifically focus on containerized services in edge platforms.

User request The measurement of user requests heavily relies on commercial enterprises with large-scale user bases. In the realm of cloud computing, numerous efforts have been made to characterize requests generated by business users, including studies conducted on Azure cloud [43], Google cloud [44], and Alicloud [45]. However, these studies primarily concentrate on user requests within centralized cloud clusters, lacking features such as user-server geographic relationships and load balancing in the edge platform.

7 Conclusions

We have carried out a large-scale measurement of QoS for a commercial crowdsourced edge platform based on three dimensions: edge servers, containerized services and user requests. Specifically, we have analyzed geographical distribution, resource distribution, reliability, and many other aspects. Further, we have designed an open source project to provide a near-realistic simulation environment. Based on the above research, we aim to provide realistic experience for related research and to promote solutions to the problems mentioned in this paper.

Author Contributions Yicheng Feng and Shihao Shen were primarily responsible for the writing of the paper, conducting experimental tests, and plotting experimental graphs. Mengwei Xu, Cheng Zhang, Xin Wang, Xiaofei Wang, Wenyu Wang and Victor C.M. Leung provided guidance on the design and writing of the paper. All authors reviewed the manuscript.

Funding This research was supported by the National Key R&D Program of China (Grant 2021ZD0113001), the Guangdong Pearl River Talent Recruitment Program (Grant 2019ZT08X603), the Guangdong Pearl

River Talent Plan (Grant 2019JC01X235), Shenzhen Science and Technology Innovation Commission (Grant R2020A045), the Canadian Natural Sciences and Engineering Research Council (Grant RGPIN-2019-06348), the National Science Foundation of China (Grant 62072332), the China NSFC (Youth) (Grant 62002260), the China Postdoctoral Science Foundation (Grant 2020M670654), and the Tianjin Xinchuang Haihe Lab (Grant 22HHXCJC00002).

Data Availibility Not applicable.

Code Availability Codes related to the model generator in this research are available.

Declarations

Ethical approval This article does not contain any studies involving human participants and/or animals by any of the authors.

Competing interests The authors declare that they have no competing interests.

References

1. Lv, Z.: Virtual reality in the context of internet of things. Neural Comput. Appl. **32**(13), 9593–9602 (2020)
2. Ren, P., Liu, L., Qiao, X., Chen, J.: Distributed edge system orchestration for Web-based mobile augmented reality services. IEEE Trans. Serv, Comput (2022)
3. Khan, M.A., Sayed, H.E., Malik, S., Zia, T., Khan, J., Alkaabi, N., Ignatious, H.: Level-5 autonomous driving—are we there yet? a review of research literature. ACM Comput. Surv. **55**(2) (2022)
4. Shen, S., Ren, Y., Ju, Y., Wang, X., Wang, W., Leung, V.C.: Edgematrix: A resource-redefined scheduling framework for sla-guaranteed multi-tier edge-cloud computing systems. IEEE J. Sel, Areas Commun (2022)
5. Liu, Z., Song, J., Qiu, C., Wang, X., Chen, X., He, Q., Sheng, H.: Hastening stream offloading of inference via multi-exit dnns in mobile edge computing. IEEE Trans. Mob, Comput (2022)
6. Meulen, R., et al.: What edge computing means for infrastructure and operations leaders. Smarter with Gartner (2018)
7. Azure MEC (2020). https://docs.microsoft.com/en-us/azure/private-multi-access-edge-compute-mec/overview. Accessed 1 Apr 2022
8. AWS Local Zones (2020). https://aws.amazon.com/cn/about-aws/global-infrastructure/localzones/. Accessed 1 Apr 2022
9. How an IoT Edge device can be used as a gateway (2022). https://learn.microsoft.com/en-us/azure/iot-edge/iot-edge-as-gateway?view=iotedge-1.4. Accessed 1 Apr 2022
10. Analytics on the edge using IBM Cloud Pak for Data (2020). https://www.ibm.com/blogs/journey-to-ai/2020/05/analytics-on-the-edge-using-ibm-cloud-pak-for-data/?_ga=2.207148885.771206213.1610462457-287655082.1610462457. Accessed 1 Apr 2022
11. Ercan, M., Malmodin, J., Bergmark, P., Kimfalk, E., Nilsson, E.: Life cycle assessment of a smartphone. In: ICT for Sustainability 2016, pp. 124–133 (2016). Atlantis Press
12. Dream, build, and transform with Google Cloud (2011). https://cloud.google.com/. Accessed 1 Apr 2022
13. Amazon Web Services (2004). https://aws.amazon.com/. Accessed 1 Apr 2022
14. Cidon, I., Khamisy, A., Sidi, M.: Analysis of packet loss processes in high-speed networks. IEEE Trans. Inf. Theory **39**(1), 98–108 (1993)
15. Docker overview (2021). https://docs.docker.com/get-started/overview/. Accessed 1 Apr 2022
16. Kubernetes Documentation (2022). https://kubernetes.io/docs/home/. Accessed 1 Apr 2022
17. Hedhli, A., Mezni, H.: A survey of service placement in cloud environments. J. Grid Comput. **19**(3), 1–32 (2021)
18. Chang, W., Wang, P.: Write-aware replica placement for cloud computing. IEEE J. Sel. Areas Commun. **37**(3), 656–667 (2019)
19. Slimani, S., Hamrouni, T., Ben Charrada, F.: Service-oriented replication strategies for improving quality-of-service in cloud computing: a survey. Clust. Comput. **24**(1), 361–392 (2021)
20. Benesty, J., Chen, J., Huang, Y., Cohen, I.: Pearson correlation coefficient. In: Noise Reduction in Speech Processing, pp. 1–4. Springer
21. Abdi, H.: Multiple correlation coefficient. Encyclopedia of measurement and statistics **648**, 651 (2007)

22. Gallager, R.: Poisson processes. In: Discrete Stochastic Processes, pp. 31–55. Springer
23. Sadeghi, M., Barati, M.: Performance analysis of poisson and exponential distribution queuing model in local area network. In: 2012 International Conference on Computer and Communication Engineering (ICCCE), pp. 499–503 (2012). https://doi.org/10.1109/ICCCE.2012.6271237
24. Tyagi, R.R., Aurzada, F., Lee, K.-D., Reisslein, M.: Connection establishment in lte-a networks: Justification of poisson process modeling. IEEE Systems Journal **11**(4), 2383–2394 (2017). https://doi.org/10.1109/JSYST.2014.2387371
25. Hagihara, S., Fushihara, Y., Shimakawa, M., Tomoishi, M., Yonezaki, N.: Web server access trend analysis based on the poisson distribution. In: Proceedings of the 6th International Conference on Software and Computer Applications, pp. 256–261 (2017)
26. Rajaram, S., Graepel, T., Herbrich, R.: Poisson-networks: A model for structured poisson processes. In: International Workshop on Artificial Intelligence and Statistics, pp. 277–284 (2005). PMLR
27. Narayanan, A., Zhang, X., Zhu, R., Hassan, A., Jin, S., Zhu, X., Zhang, X., Rybkin, D., et al.: A variegated look at 5g in the wild: performance, power, and qoe implications. In: ACM SIGCOMM, pp. 610–625 (2021)
28. Wang, Z., Li, Z., Liu, G., Chen, Y., Wu, Q., Cheng, G.: Examination of wan traffic characteristics in a large-scale data center network. In: ACM IMC, pp. 1–14 (2021)
29. Schlinker, B., Cunha, I., Chiu, Y., Sundaresan, S., Katz-Bassett, E.: Internet performance from facebook's edge. In: ACM IMC, pp. 179–194 (2019)
30. Mok, R., Zou, H., Yang, R., Koch, T., Katz-Bassett, E., Claffy, K.: Measuring the network performance of google cloud platform. In: ACM IMC, pp. 54–61 (2021)
31. Johnson, M., Liang, J., Lin, M., Singanamalla, S., Heimerl, K.: Whale watching in inland indonesia: Analyzing a small, remote, internet-based community cellular network. In: WWW, pp. 1483–1494 (2021)
32. Xu, E., Zheng, M., Qin, F., Xu, Y., Wu, J.: Lessons and actions: What we learned from 10k SSD-Related storage system failures. In: USENIX ATC, pp. 961–976 (2019)
33. Fida, M., Acar, E., Elmokashfi, A.: Multiway reliability analysis of mobile broadband networks. In: ACM IMC, pp. 358–364 (2019)
34. Xu, M., Fu, Z., Ma, X., Zhang, L., Li, Y., Qian, F., Wang, S., Li, K., Yang, J., Liu, X.: From cloud to edge: a first look at public edge platforms. In: ACM IMC, pp. 37–53 (2021)
35. Rafique, W., Qi, L., Yaqoob, I., Imran, M., Rasool, R., Dou, W.: Complementing iot services through software defined networking and edge computing: A comprehensive survey. IEEE Commun. Surv. Tutor. **22**(3), 1761–1804 (2020)
36. Zhang, Y., Liu, J., Wang, C., Wei, H.: Decomposable intelligence on cloud-edge iot framework for live video analytics. IEEE Internet Things J. **7**(9), 8860–8873 (2020)
37. Jiang, X., Yu, F., Song, T.n., Leung, V.: A survey on multi-access edge computing applied to video streaming: some research issues and challenges. IEEE Commun. Surv. Tutor. **23**(2), 871–903 (2021)
38. Mu, P., Zheng, J., Luan, T., Zhu, L., Dong, M., Su, Z.: Amis: Edge computing based adaptive mobile video streaming. In: IEEE INFOCOM, pp. 1–10 (2021). IEEE
39. Gao, Y., Zhang, C., Xie, Z., Qi, Z., Zhou, J.: Cost-efficient and quality of experience-aware player request scheduling and rendering server allocation for edge computing assisted multiplayer cloud gaming. IEEE Internet Things J. (2021)
40. KubeEdge: Kubernetes native edge computing framework (project under CNCF) (2019). https://github.com/kubeedge/kubeedge. Accessed 1 Apr 2022
41. OpenYurt: Extending your native kubernetes to edge (2020). https://github.com/alibaba/openyurt. Accessed 1 Apr 2022
42. Baetyl: Extend cloud computing, data and service seamlessly to edge devices (2019). https://github.com/baetyl/baetyl. Accessed 1 Apr 2022
43. Cortez, E., Bonde, A., Muzio, A., Russinovich, M., Fontoura, M., Bianchini, R.: Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In: Symposium on Operating Systems Principles, pp. 153–167 (2017)
44. Borg cluster traces (2019). https://github.com/google/cluster-data. Accessed 1 Apr 2022
45. Alibaba Cluster Trace (2020). https://github.com/alibaba/clusterdata. Accessed 1 Apr 2022

Authors and Affiliations

Yicheng Feng[1] · Shihao Shen[1] · Mengwei Xu[2] · Cheng Zhang[3] · Xin Wang[1] · Xiaofei Wang[1] · Wenyu Wang[4] · Victor C. M. Leung[5,6]

Yicheng Feng
yichengfeng@tju.edu.cn

Shihao Shen
shenshihao@tju.edu.cn

Mengwei Xu
mwx@bupt.edu.cn

Cheng Zhang
zccode@gmail.com

Xin Wang
wangx@tju.edu.cn

Wenyu Wang
wayne@pplabs.org

Victor C. M. Leung
vleung@ieee.org

1 College of Intelligence and Computing, Tianjin University, Tianjin, China

2 Beijing University of Posts and Telecommunications, Beijing, China

3 Institute of Technology, Tianjin University of Finance and Economics, Tianjin, China

4 Paiou Cloud Computing (Shanghai) Co., Ltd., Shanghai, China

5 College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China

6 Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, BC, Canada