A Holistic QoS View of Crowdsourced Edge Cloud Platform

Shihao Shen[†], Yicheng Feng[†], Mengwei Xu[‡], Cheng Zhang[§], Xiaofei Wang[†*], Wenyu Wang[¶], Victor C.M. Leung[‖††]
[†] College of Intelligence and Computing, Tianjin University, Tianjin, China
[‡] Beijing University of Posts and Telecommunications, Beijing, China
[§] Institute of Technology, Tianjin University of Finance and Economics, Tianjin, China
[¶] Paiou Cloud Computing (Shanghai) Co., Ltd., Shanghai, China
[‖] College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China
[††] Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, Canada
{shenshihao, yichengfeng}@tju.edu.cn, mwx@bupt.edu.cn, zccode@gmail.com, xiaofeiwang@tju.edu.cn,
wayne@pplabs.org, vleung@ieee.org

***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: (1) First, we look at how much hardware resources are provided by edge servers, how much time they are available for service deployment, and what are the major abnormal behaviors. (2) Second, we analyze the factors affecting service stability and quantify the resource utilization pattern of containerized services hosted on those edge servers. (3) 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.**

***Index Terms*—edge computing, performance analysis, network measurement**

I. Introduction

Contents delivery, AR/VR, and so on are increasingly adopting edge computing paradigm [1], [2], [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 service quality [4], [5]. According to Gartner, around 75% of enterprise-generated data will be processed at the edge by 2025 [6].

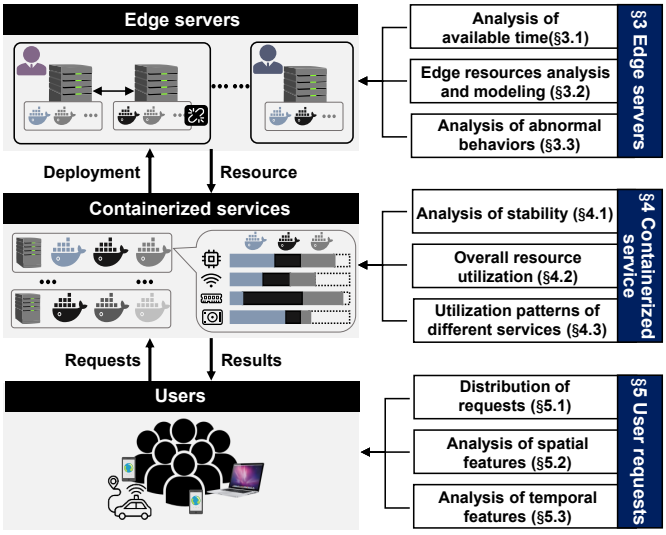

Figure 1. The organization of this work.

There are many forms of edge deployment. Major cloud resource providers are building their edge infrastructure at a state or city level such as Azure Edge Zone [7] and AWS Local Zones [8]. Some aim to sink the edge servers into buildings or base stations [9], [10]. In neither way, those edge sites' hardware and software are both maintained by the edge service providers under full control, which benefits from high Quality of Service (QoS).

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

Mengwei Xu was supported by the National Key R&D Program of China (Grant 2021ZD0113001). Victor C.M. Leung was supported by 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), and the Canadian Natural Sciences and Engineering Research Council (Grant RGPIN-2019-06348). Xiaofei Wang was supported by 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). Corresponding author: Xiaofei Wang (email: xiaofeiwang@tju.edu.cn).

979-8-3503-9973-8/23/$31.00 ©2023 IEEE

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 Fig 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 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.

Edge servers often contribute modest hardware resources in an ephemeral manner. Servers connect and disconnect frequently. Per day, about 8% of total registered edge servers have connect/disconnect records. Among all the online sessions, more than half of them last less than one hour. It challenges the C-ESP in utilizing those ephemeral server time with tight resource constraints, as the service deployment often takes non-trivial time.

The available edge resource quantity at a geographical location relates to population/GDP. The higher population or GDP, the more edge servers and corresponding resources are expected. Based on these observations, we design a simple modeling generator to estimate the edge resources distribution at an arbitrary region, which could help edge researchers and developers to evaluate their systems/algorithms under a realistic and diversified geographical setting.

The network is the primary cause of Service-Level Agreement (SLA) violation, while the device leads to the most fines. For each SLA violation, C-ESP traces down the reason and records the resulting fines to be paid to Application Service Providers (ASPs). Among the 34,091 SLA violations collected, the network contributes almost 66%. We find the device causes the most fines, as it is often more harmful to the service quality. To mitigate such costs, we should prioritize the handling of device failures in the operation and maintenance.

The resource usage of containerized services is highly heterogeneous and changes significantly over time. This observation highlights the need to bin-pack services with different usage patterns into the same server. Fortunately, we observe a few fixed patterns that can well match different services, e.g., high-demand resources, peak periods, and rapid change. Leveraging those patterns as tags of each service, we can easily cluster the services into different types and better consolidate them in servers.

The amount of requests generated and the available edge resources do not match geographically. For instance, we observe some areas[3] with a large number of requests generated but few resources available. Consequently, simply scheduling the requests to nearby edge services could lead to unbalanced resource usage. To this end, a globally resource-aware requests scheduler is demanded.

The number of user requests generated across time nearly follows Poisson distribution. Based on the data collected, we build a statistic model that can simulate real-world user requests generated. This model could help edge

[1]Abbreviated as edge platform in the following

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

[3]"Area" in this paper refers to the largest administrative region of the country, such as a state or province.

researchers to evaluate their algorithms or systems in a more realistic simulation manner.

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.

II. C−ESP

In the cloud computing platform, 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 so on. They construct centralized large-scale cloud computing infrastructures which provide services across a large geographic area. However, C−ESP, which adopts edge computing, is different from them. To further reduce the geographic distance between users and services, it adopts a decentralized architecture to deploy the computing infrastructure to the network edge.

Based on the form of crowdsourcing, C−ESP integrates third-party scattered resources at the network edge and builds its own edge servers to provide high-quality computing power as a guarantee. Moreover, C−ESP builds an edge computing network that can cover almost all areas of China. C−ESP has deployed over 10,000 edge servers in over 1,000 regions to provide resources for ASPs. According to our knowledge, C−ESP is one of the largest enterprises applying crowdsourced edge computing in business scenarios.

In addition, C−ESP is profitable by providing edge computing power to ASPs. It currently hosts services from over a dozen ASPs with tens of millions of users. In addition, to enhance rapid deployment, resource isolation and service compatibility, all of these services have adopted containerization technology. Typically, ASPs provide service containers as well as resource requirements, and C−ESP enables the deployment and operation of service containers under SLA guarantees.

III. Exploring Edge Servers

C−ESP has a large number of edge servers with complex sources, including (i) its own computing servers built and deployed, (ii) large idle computing servers leased from other organizations, and (iii) smaller servers rented from individual users. In terms of QoS for edge servers, we first focus on the available time of crowdsourced servers. After that, we collect data on multiple dimensions and quantify edge server resource comprehensively. Finally, we discuss the abnormal behavior of edge servers.

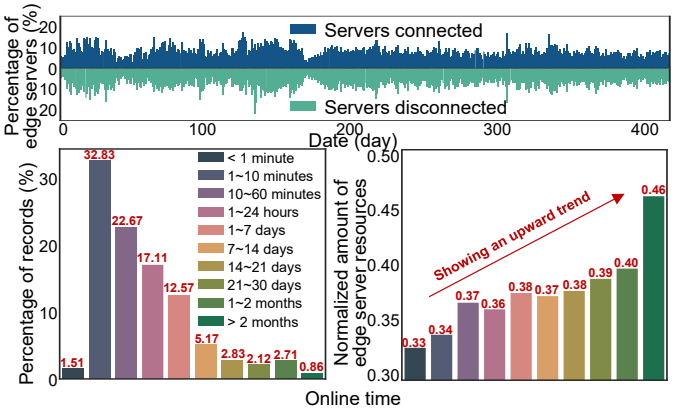

Figure 2. The analysis of (a) daily connect/disconnect distribution (top), (b) online times distribution (left) and (c) average resource distribution (right).

A. Analysis of Available Time

C−ESP uses a crowdsourcing model to leverage idle servers from third parties. However, these servers that are not fully controlled bring more uncertainty to the platform. Therefore, we paid special attention to the available time of the edge servers under the crowdsourcing model. With the help of C−ESP, we collect server connection and disconnection records for up to 418 days, with a total of 185,698 records. The data collected included server ID, timestamp, connection/disconnection, and server resource.

As shown in Fig. 2 (a), the number of server connections and disconnections per day is about equal for C−ESP, thus ensuring a stable number of servers in 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, **frequent changes in server status can lead to frequent adjustments in service deployment**, which causes more costs compared to traditional platforms. To maintain the stability of QoS, C−ESP uses a bonus-based reward and punishment mechanism for maintenance. Since the resource requirement of the service fluctuates regularly (as shown in Sec. IV-B), C−ESP sets high bonus/penalty for connection/disconnection servers during high resource requirement periods such as afternoon peak and evening peak to reduce server status changes, and adjusts the service deployment caused by server connection/disconnection during idle periods.

In addition, we analyze the online time distribution of the servers in the collected data, i.e. Fig. 2 (b). It can be found that more than half of the online time records are less than one hour, while the overall average online time is 337,782 seconds (about four days). Since there is a time cost for service deployment, **a large number of short-term connections can lead to inefficient service deployment**, i.e., services that are deployed with time spent are not available due to server disconnection. In response, C−ESP is planning to design a mechanism to predict whether connected servers will be disconnected soon and whether disconnected servers will be reconnected soon. Further, the prediction results can be used to enhance service deployment and migration policies to improve QoS.

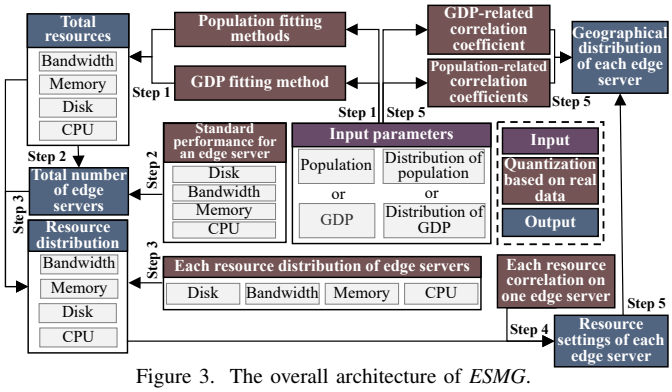

Figure 3. The overall architecture of *ESMG*.

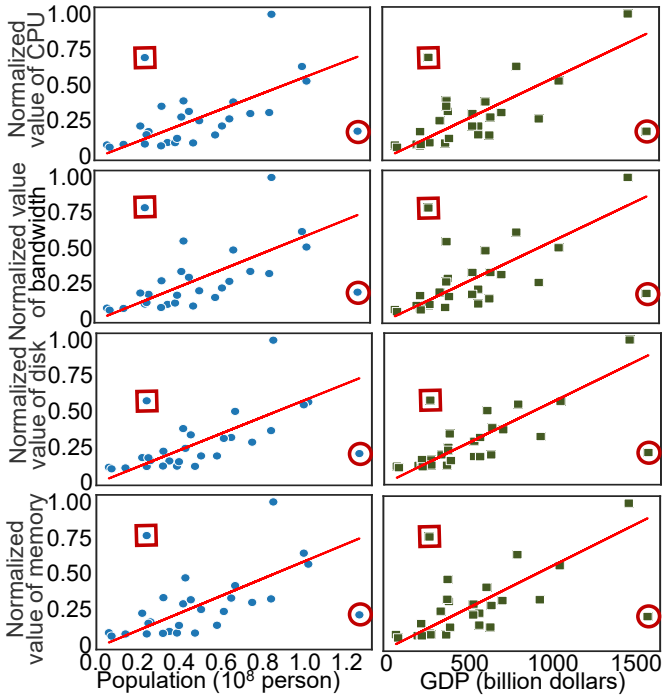

Figure 4. The correlation of various types of resources in each area with population and GDP.

Finally, we explore the relationship between resources and online time, i.e., Fig. 2 (c). The resource amount of the server is calculated by summing the four metrics of CPU, disk, memory, and bandwidth after normalization using min-max. It can be found that servers with higher resources tend to be online for longer periods. The reason is that low-resource servers usually are from small organizations or individuals whose management of servers often lacks long-term planning.

B. Edge Resources Analysis and Modeling

The high heterogeneity of edge servers in hardware resources and geographical distribution leads to difficulties in practical applications or modeling studies. However, as C-ESP is one of the few enterprises that commercialize edge computing, we are curious about the impact of the heterogeneity in the business scenarios and whether there are potential patterns behind it. For the above question, we collect the data from 13,036 edge servers, including CPU, bandwidth, memory, disk, latitude, and longitude. Based on the above data, we found a set of quantifiable potential patterns and summarized them into a complete architecture, namely *ESMG* (Edge server Model Generation) shown in Fig. 3.

Table I
LINEAR FITTING EQUATIONS FOR POPULATION/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	$y = 8.086 \times 10^{8}x - 7.387 \times 10^{10}, r = 0.80$

In addition, due to commercial constraints, it is difficult for companies to open up complete data. Even if they open up the complete data, it is not flexible enough to support relevant research and practice due to the rigidity of the data in terms of the number of users and the size of the region. However, due to the completeness of *ESMG*, it can not only quantitatively reveal the potential patterns of edge servers, but also provide a reference for future edge computing practical applications or modeling research.

First, users of *ESMG* only need to input the total amount and the geographic distribution of population/GDP in the target region, which can be easily accessed from the local governments or other statistical agencies. After that, *ESMG* can flexibly model the edge server attributes and geographic information of the target region based on the various quantitative models. Next, we focus on Fig. 3 to present the quantifiable patterns found and how these patterns take effect in *ESMG*.

Step 1. The goal of this step is to take the total GDP or population of a region as input to output the total amount of each type of resource for the whole region. First, we quantify the relationship between resources and population/GDP. As shown in Fig. 4, except for the circle-marked area with low resources and the square-marked area with high resources, the resources of other areas have a linear relationship with population/GDP. Consequently, linear fitting is performed after removing the aforementioned two outliers. Finally, we derive the linear fitting equations as shown in Table I, and the unit of each parameter includes GDP (billion dollars), population (10^8 person), CPU (core), bandwidth (Byte/s), disk (Byte), and memory (Byte). In addition, the r in Table I refers to the Pearson correlation coefficient, which is used to quantify the goodness of fit.

Step 2. The goal of step 2 in *ESMG* is to use the total amount of each type of resource in the region as input to output the number of edge servers in this region. First, we calculate the mean based on the data of 13,036 edge servers to provide an average edge server performance: CPU of 11.510 Core, bandwidth of 782.759 MB/s, memory of 17.519 GB, and disk of 2456.536 GB. After that, we can obtain four estimates of the number of edge servers by dividing the total

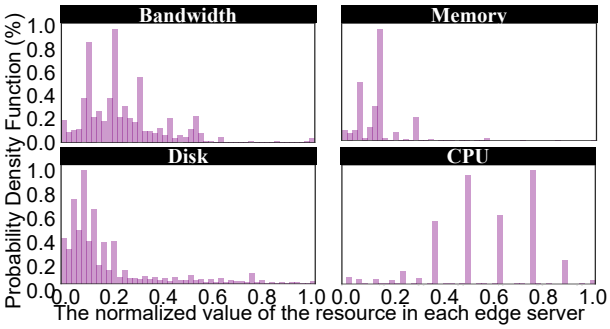

Figure 5. The resource distribution of edge servers.

amount of each type of resource in the region by the above average performance. Finally, we calculate the mean of the four estimates as the output of step 2.

Step 3. The goal of step 3 in *ESMG* is to take the total number of each resource and the number of edge servers in the region as input to output the distribution of each resource in this region. Since the heterogeneity of edge servers cannot be modelled if resources are distributed equally, the heterogeneity distribution of edge servers in each resource needs to be quantified in Step 3. As shown in Fig. 5, it quantifies the resource heterogeneity of real edge servers and can therefore be used to determine the distribution of each resource.

Step 4. The goal of step 4 in *ESMG* is to take the distribution of each resource in the region as input to output the resource configuration for each edge server. Although the above quantifies the distribution of each resource, the different resource associations are still unclear. For example, an edge server with high CPU resources is likely to have abundant resources in memory. Therefore, we used Spearman correlation coefficients to quantify the correlation between the various resources of the edge server, as shown in Table II. It can be found that the correlation between CPU, bandwidth, and memory is higher compared to the correlation between disk and other resources, so the resource correlation constraint shown in Table II needs to be approximated as closely as possible when determining the resource configuration.

Step 5. The goal of step 5 in *ESMG* is to take the geographic distribution of GDP/population and the edge server resource allocation in the region as input to output the geographic distribution of edge servers in this region. Based on Table I, it can be found that the distribution of edge servers is associated with the distribution of GDP/population, but the distribution of both is not exactly the same. Therefore, in addition to using the distribution of GDP/population to map the distribution of edge servers, step 5 also requires some random adjustments to refine the heterogeneity. In this regard, these random adjustments need to be close to the constraints of r in Table I.

Based on the above, it can be found that using GDP/population as the entry point can well reveal edge server heterogeneity in business scenarios. Moreover, since almost all dimensions of edge servers are quantified, other research and practice in related fields can be flexible to generate realistic and arbitrary region-sized edge server configuration.

For better use and understanding by other researchers, we provide the required supporting data and the implementation code of *ESMG* in the **open source project** (introduced in Sec. I). Thereby, it is hoped to help other researchers in advancing related research and performance validation.

Table II
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

C. Analysis of Abnormal Behavior

Since edge servers are deployed in a distributed manner, it makes the operation and maintenance more difficult and has a negative impact on QoS. Therefore, we pay heightened attention to the abnormal behaviors of edge servers. For the aforementioned purpose, we collected 428,160 operation and maintenance data from C−ESP for 139 days. Each data includes server ID, the number of abnormal behaviors that occur on an edge server in a day, and the kind (categorized as shown in Table. III).

Since detection is performed every five minutes in C−ESP, each abnormal behavior can be considered as lasting five minutes. As shown in Table. III, we found that the average value of *Machine line drop times* is very high. This is due to C−ESP uses link aggregation to make a server have multiple network lines. Therefore, if the part line is unavailable, it will be counted as *Machine line drop times*, while the entire server is unavailable will be counted as *Offline times*. Furthermore, 7.85% of *Machine line drop times* recorded as 288, which is the whole day. Therefore, even though 76.41% of the data is 0, the average value reaches 33.92. In addition, *High I/O load times* is worth noting. We find that although there is no shortage of disk resources for C−ESP (as shown in Sec. IV-B), the average value of *High I/O load times* is high. It indicates that **in addition to disk storage space, disk I/O needs more consideration when optimizing.**

After that, we use the correlation coefficient to quantify the relationship between different abnormal behaviors and expect to explore the causes of them. As shown in Fig. 6, most abnormal behaviors are independent of each other, indicating that **it is difficult to predict the occurrence of abnormal behaviors by correlation.** However, we found two pairs of abnormal behaviors with strong correlation: (i) *Abnormal IP change times* is correlated with *Machine line drop times*, which indicates that it is necessary to pay attention to whether the IP is normal after disconnection and reconnection. (ii) *Offline times* is correlated with *Unavailable time*, which indicates that frequent offline is the key cause of unavailability.

IV. 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

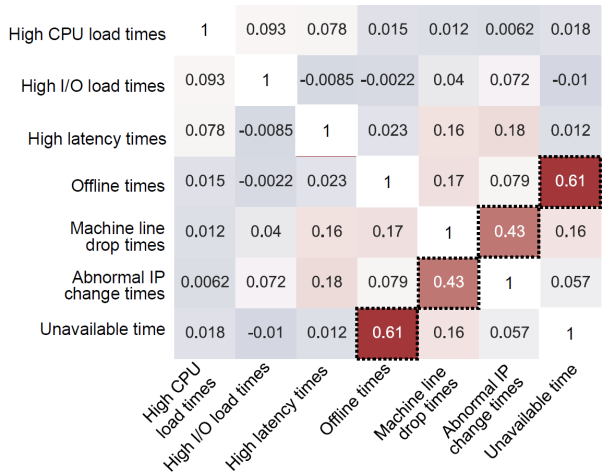

Figure 6. Spearman correlation coefficient of edge servers among deferent abnormal behaviors.

Table III
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 (seconds)	1464	93.52 %

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.

A. Analysis of Stability

Compared with the centralized operation and maintenance of cloud computing, the distributed deployment of edge computing poses challenges to service reliability. However, the core business of C−ESP is to provide high-quality infrastructure resources to ASPs, so the reliability of service is essential. In this regard, we collect data for about two months with the help of C−ESP. Since C−ESP provides a series of SLA guarantees for ASPs and different SLA violations can result in C−ESP handing over different fines to ASPs, our measurements in reliability focus on two main metrics: (i) The number of failures caused by each type of factor, i.e., is used to explore the factors that are most likely to cause failures; (ii) The number of fines resulting from failures caused by each type of factor, i.e., is used to explore the factors that are most harmful to stability.

Since SLAs are confidential business information, we cannot show the full details. Therefore, we fuzzify the data by four categories: (i) **Network** includes factors such as network connection, communication; (ii) **Artificial** includes factors such as adjustment and testing by engineers; (iii) **Device** includes factors such as server disconnection by the owner and hardware failure; (iv) **Service** include factors such as configuration and operation of the container.

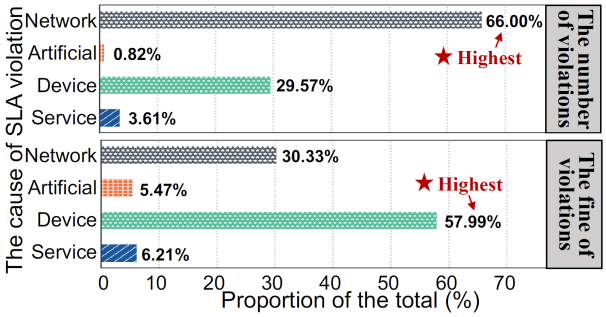

Figure 7. Analysis of (a) the number of violations of service SLA guarantees (top) and (b) the corresponding amount of fines incurred (bottom).

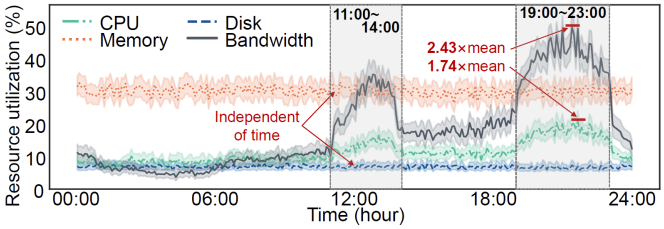

Figure 8. Resource utilization of all edge servers in one day.

As shown in Fig. 7, the causes of SLA violations are divided into four categories. Among them, **the network is the factor that causes the most SLA violations**, accounting for 66.00% of the overall. The reasons for this phenomenon include: (i) C−ESP's edge servers sink to some remote areas with unstable networks, which are prone to SLA violations; (ii) C−ESP uses link aggregation on some edge servers to aggregate bandwidth resources, so the number of network links is larger than the number of servers and the larger number also leads to more SLA violations.

The device is the factor that causes the most fines of SLA violations. While the network is responsible for the most SLA violations, it only accounted for 30.33% of fines. In contrast, the device accounted for 29.57% of the total SLA violations, but accounted for 57.99% of the total fines. Through in-depth investigation, the most frequent SLA violation in the network factor is partial line disconnection. This failure causes only part of the bandwidth resources of edge servers to be unavailable. In contrast, most device failures lead to the unavailability of all resources of the whole server, so SLA violations due to device failures result in more fines. Therefore, it is essential to focus on failures of devices, which can pose the greatest threat to SLA guarantees.

Thus, it is necessary to take methods to deal with device failures in advance. On the one hand, it is advised to optimize server operation and maintenance to prevent SLA violations. On the other hand, it is recommended to establish a recovery mechanism in the edge cluster to reduce the impact of failures.

B. Overall Resource Utilization

Since most of the containerized services deployed in C−ESP are used to respond to requests from users, the behavior of users tends to change with a periodic pattern over time. This change can directly affect the resource requirements of the

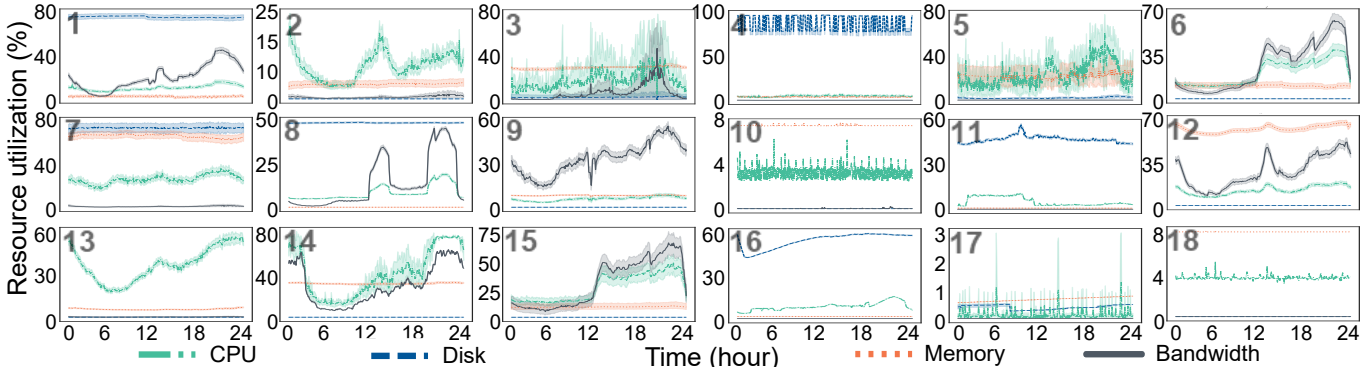

Figure 9. The resource utilization changes of multiple types of service containers in one day.

service, thus potentially resulting in time-dependent patterns in the resource requirements of the containerized services. In order to reflect the real performance of containerized services, we collected monitoring data of over 10,000 edge servers and the service containers on them with the help of C−ESP.

The data contains the monitoring logs of over 10,000 edge servers and the service containers on them for a single day. Edge servers collect information every 5 minutes, i.e., each edge server and container generate 288 logs in one day. Besides, the data includes four resources: memory, bandwidth, disk and CPU. After that, we get the real-time utilization changes of various resources, i.e., Fig. 8 and Fig. 9. To enhance the persuasiveness of the data, we overlay the data of all edge servers and containers, so the solid line in the figure indicates the mean, while the shading indicates the variance.

First, Fig. 8 illustrates the resource utilization of containerized services on the edge servers. It can be observed that the resource utilization of memory and disk remains relatively stable within a day, so these two types of resources require little reservation to cope with potential peaks in applications. On the contrary, CPU and bandwidth requirements change significantly throughout a single day, and bandwidth utilization varies the most. In addition, we can find a strong correlation between CPU and bandwidth utilization changes. Simultaneously, there are two apparent peaks of CPU and bandwidth utilization in a day, namely 11:00-14:00 (afternoon peak) and 19:00-23:00 (evening peak), with the evening peak being the highest (CPU/bandwidth utilization is 1.74/2.43 times of the daily average), and the CPU and bandwidth utilization being also significantly higher than at night.

The above shows that although CPU and bandwidth are volatile over time, there are still potential patterns. In practical scenarios, extra attention should be paid to CPU and bandwidth resource utilization during the afternoon and evening peak hours, which are the most likely times for accidents due to insufficient resources. In addition, since CPU and bandwidth utilization fluctuates significantly, they are more difficult to improve than disk and memory utilization: (i) If a large number of resources are reserved to cope with the peak, it can guarantee the quality of service but will lead to resource waste. (ii) If idle resources are utilized and released before the peak, the overall resource utilization can be improved. However, the occurrence probability of resource insufficiency will increase due to the ambiguity and uncertainty of the peak arrival time. For these reasons, **CPU and bandwidth resources face a more severe trade-off between efficiency and quality.**

C. Utilization Patterns of Different Services

Fig. 9 reveals the resource utilization changes of containers from 18 different ASPs. Since C−ESP uses the multi-tenant model, we divide the containers into 18 groups based on different ASPs and show the resource utilization of each group. It can be observed that the distribution patterns of different kinds of resources differ significantly. It creates difficulties for the efficient deployment of service containers on the server, i.e., it can cause the server to run out of one type of resource, but have other types of resources left over. In these cases, because one resource is used up, new containers cannot be deployed due to the exhaustion of one resource, resulting in the waste of other resources.

As shown in Fig. 8 and 9, the resource isolation capability of the container divides the idle resources into two parts, so the optimisation of resource utilisation needs to be considered from two perspectives. On the one hand, the idle resources on the edge servers are **direct resources** that can be reallocated, and resource utilisation can be improved by optimising the scheduling of container deployment. On the other hand, the idle resources in the edge containers are **indirect resources** that cannot be reallocated. Due to the resource isolation capability of the container, these resources need to be transformed through the container scaling policy before they can be reallocated. However, directly using these resources by increasing the load through the request offloading policy is also a feasible method.

Therefore, containerization not only provides resource isolation, but also presents **a new problem of server-container resource requirement matching**, namely how to arrange containers on edge servers to make full use of resources. In addition, the distribution of different containers is uneven for a single resource type. This heterogeneity raises **a new problem of container-container resource requirement matching**, i.e., how to deploy containerized services with complementary requirements on an edge server to improve resource utilization?

In summary, while the similar problem exists in the centralized cloud computing paradigm [14], [15], [16], the problem of server-container resource requirement matching becomes more

Table IV
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** [17] between resource utilization and time.
Predictability	**Variance** of resource utilization for different containers of the same service.
Resource correlation	**Multiple correlation coefficient** [18] between each resource and other resources.

important and challenging in the edge computing paradigm due to (i) the distributed deployment of edge servers, (ii) the resource limitations of a single edge server, (iii) the heterogeneity of edge servers, (iv) the resource isolation brought about by the containerization (if containerization is used). Therefore, this problem has become a critical problem hindering the efficient utilization of edge resources.

To address these issues, we need to determine whether two services complement each other and quantify resource fluctuations. In this regard, we take Fig. 9 as an example to propose a classification method, which can be used to summarise service resource characteristics from six dimensions:

- **High-demand resources**. It represents the type of resource in high demand, such as disk (1, 4, etc.)[4], memory (10, 12, etc.), CPU (2, 13, etc.), bandwidth (6, 9, etc.);
- **Peak period**. It represents the peak period of the service, such as the afternoon peak (12), evening peak (1, 14, etc.), and both (6, 15, etc.);
- **Rapid change**. It represents the rate of change in resource requirements, such as some change fast (8, 15, etc.) and some change slow (1, 9, etc.);
- **Time-dependent**. It indicates whether the resource requirement varies with time, such as some related to time (6, 14, etc.) and some not (4, 7, etc.);
- **Predictability**. It indicates whether different containers of the same service have consistent resource utilization, i.e., the shaded range in the Fig. 9, such as some are consistent (1, 6, etc.), some are not (3, 5, etc.);
- **Resource correlation**. It indicates whether different resource changes are correlated, e.g. some services have similar CPU and bandwidth changes (6, 14, etc.) and some are not (2, 13, etc.).

For further quantification, we provide a set of quantitative methods for the above six dimensions as an example, namely Table IV. Therefore, the six dimensions can be used as six labels to tag each kind of service. Based on these labels, researchers can comprehensively quantify the resource demand patterns of various service containers, which is helpful for modeling and algorithm design.

[4]The numbers here and below correspond to the indexes in Fig. 9.

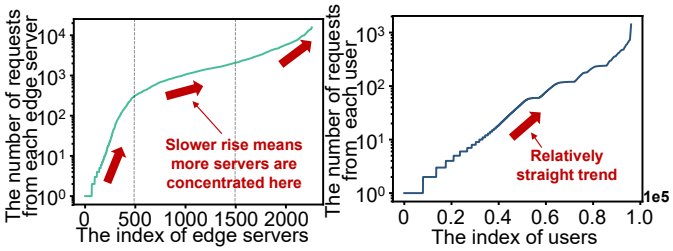

Figure 10. Frequency analysis of (a) edge servers (left) and (b) users (right).

In addition, to provide a container resource fluctuation close to the real scenario, we provide a model generator for containerized services in the **open source project** (introduced in Sec. I). Specifically, by entering a list of desired container types as shown in Fig. 9, it can generate resource utilization fluctuations for each type of resource. In addition, the generated data has some randomness, but it is satisfied with the distribution of the real data.

V. 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.

A. Distribution of Requests

Currently, C-ESP hosts more than a dozen different types of ASPs, and we base our analysis on the service log of an ASP as an example, which consists of one hour of data per day (at different periods) for six days, i.e., a total of six hours. The data contains 10,159,851 logs from 96,209 users, and these requests are sent to 2,359 edge servers for processing. Specifically, each data contains: (i) the fuzzy geographic location of the request sender (user); (ii) the fuzzy geographic location of the request receiver (edge server); (iii) the generation time of requests; (iv) unique identification of the request sender and the request receiver. It should be noted that the geographic locations in the data are fuzzy locations based on the IP address, so as to ensure the privacy of user information.

First, we count the number of requests sent/received by each user/edge server during the data collection period to explore the distribution pattern of requests on the sender and receiver sides. After that, we sort each user/edge server by the number of requests sent/received from smallest to largest, as shown in Fig. 10. Notably, we use log transformation for the y-axis in Fig. 10, and the request distribution of users shows an exponential trend, while the request distribution of edge servers is more balanced. This is because the request distribution for users is natural and unaffected by human intervention, whereas the request distribution for edge servers is interfered with by server configuration and load-balancing policies. Therefore, Fig. 10 reveals a significant phenomenon.

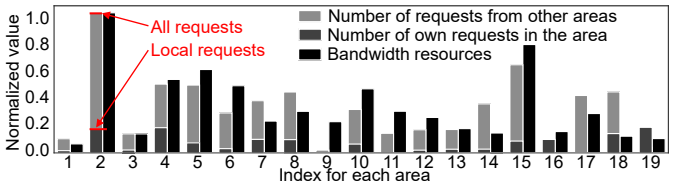

Figure 11. Comparison between the number of requests received and the amount of resources in each area.

Table V
POISSON DISTRIBUTION PARAMETERS.

Index	λ	N	C	Index	λ	N	C
1	474.875	35688	0.0133	4	437.115	33699	0.0130
2	499.845	34449	0.0145	5	462.153	34770	0.0123
3	474.127	35074	0.0135	6	471.708	34314	0.0137

Namely, **the number of requests per user in the collected data follows the exponential distribution.**

B. Analysis of Spatial Features

Based on Fig. 11, we can find that **the number of resources in each area is inconsistent with the number of requests generated**, i.e. some areas have a high amount of resources but generate few requests, and some areas the opposite. In this case, if each area only processed its own generated requests, it would significantly reduce efficiency. In addition, some areas only generate requests without deploying the corresponding type of service. Therefore, it is necessary to develop request scheduling algorithms to schedule requests from different areas. As shown in Fig. 11, the addition of requests from different areas makes the number of requests and the number of resources relatively consistent for each area, which verifies the effectiveness of C-ESP's request scheduling algorithm.

In summary, different from the centralized mode in the cloud computing platform, resources in the edge platform are distributed, which brings a new challenge: **the matching problem of resource distribution and request distribution**.

C. Analysis of Temporal Features

Due to the difficulty of obtaining a large number of real-world end-users and requests for laboratory-level systems, simulation modeling of requests is required to evaluate system performance. Therefore, it is meaningful to provide experience in user request modeling based on quantitative analysis of real large-scale data. First, we divided the collected data into six parts based on the date and processed them separately. As Poisson process is a stochastic process widely used to model the time of arrival into the system [19], we use it to fit the request, i.e. $P(X=k)=\frac{\lambda^k}{k!}e^{-\lambda}, k=0,1,\cdots$.

As shown above, the Poisson distribution can be computed once λ is determined. To analyze λ, we use N to represent the number of users. Further, we assume that λ is related to N as a constant multiple, i.e., $\lambda = C \times N$, which will be verified. After that, We can calculate the average of each group data (as shown in Table V) and use it as λ for fitting, i.e. Fig. 12. Further, we can count the number of users N in each of the six collected data and bring N with λ into $\lambda = C \times N$ to calculate

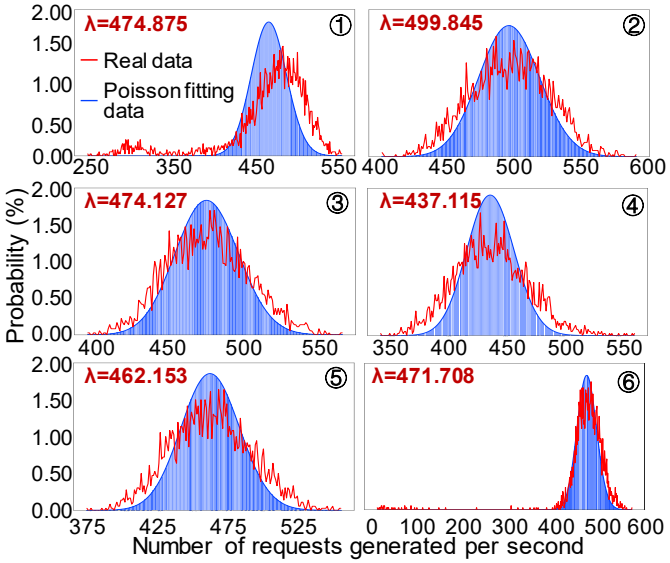

Figure 12. Fitting analysis based on Poisson distribution.

C, i.e., as shown in Table V. We find the C obtained from the six collected data independently are close to each other, which verifies our assumption that $\lambda = C \times N$. Finally, we take the mean value of C obtained from the six collected data as the final result, i.e., $\lambda = 0.0134 \times N$.

In system design and theoretical research, it is often difficult to attract a large number of real users to participate, so it is necessary to simulate and model the requests to evaluate the system performance. In response, we provide a user request generator in **open source project** (introduced in Sec. I) to mimic users generated requests. Specifically, by inputting the number of users to be served, it can generate the request situation of each user. Moreover, the generated user request data satisfies the distribution pattern in Fig. 10 and Fig. 12.

VI. RELATED WORK

Edge computing has become a key topic in academia and industry. (i) Many researchers base their research on edge computing architectures or edge-cloud collaborative architectures; (ii) Many enterprises are starting to drive edge computing adoption and building business-oriented edge platforms. However, there is still a lack of large-scale measurement studies on edge computing in real business application scenarios from multiple dimensions of services, servers and requests.

Edge server. There are many measurement efforts for server performance characterization that deeply analyze the network bandwidth [20], traffic [21], latency [22], [23], resource utilization [24], and robustness [25], [26] of servers, but most of them stay in the cloud computing platform and do not consider the distributed deployment and performance heterogeneity of edge computing servers. In addition, a recent study has measured the network latency, throughput, and QoE (Quality of Experience) of edge servers [27], but it lacks comprehensive measurement in the large-scale crowdsourced platform. Therefore, there is still lacking measurement for large-scale edge servers in the crowdsourced edge platform.

Containerized service. As the network requirements for numerous services increase, such as IoT, streaming media, and cloud gaming, these services are transitioning from cloud computing to edge computing [28], [29], [30], [31], [32]. To supply edge computing to other ASPs, some enterprises build edge platforms from PaaS (Platform as a Service) perspective combined with containerized service capabilities, such as KubeEdge [33], OpenYurt [34] and Baetyl [35]. However, to the best of our knowledge, there is still a lack of measurement of containerized services in the edge platform.

User request. The measurement of user requests relies heavily on commercial enterprises with real large-scale users. In cloud computing, there are many efforts to characterize the requests generated by business users, such as Azure cloud [36], Google cloud [37], and Alicloud [38]. However, these studies all focus on the user request of centralized cloud clusters while lacking features such as user-server geographic relationships and load balancing in the edge platform.

VII. Conclusions

We carry 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 analyze geographical distribution, resource distribution, reliability, and many other aspects. Further, we design 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 promote solutions to the problems mentioned in this paper.

References

[1] Z. Lv, "Virtual reality in the context of internet of things," *Neural Comput. Appl.*, vol. 32, no. 13, pp. 9593–9602, 2020.

[2] P. Ren, L. Liu, X. Qiao, and J. Chen, "Distributed edge system orchestration for web-based mobile augmented reality services," *IEEE Trans. Serv. Comput.*, 2022.

[3] M. A. Khan, H. E. Sayed, S. Malik, T. Zia, J. Khan, N. Alkaabi, and H. Ignatious, "Level-5 autonomous driving—are we there yet? a review of research literature," *ACM Comput. Surv.*, vol. 55, no. 2, 2022.

[4] S. Shen, Y. Ren, Y. Ju, X. Wang, W. Wang, and V. C. Leung, "Edgematrix: A resource-redefined scheduling framework for sla-guaranteed multi-tier edge-cloud computing systems," *IEEE J. Sel. Areas Commun.*, 2022.

[5] Z. Liu, J. Song, C. Qiu, X. Wang, X. Chen, Q. He, and H. Sheng, "Hastening stream offloading of inference via multi-exit dnns in mobile edge computing," *IEEE Trans. Mob. Comput.*, 2022.

[6] R. van der Meulen *et al.*, "What edge computing means for infrastructure and operations leaders," *Smarter with Gartner*, 2018.

[7] "Azure mec," 2020. [Online]. https://docs.microsoft.com/en-us/azure/private-multi-access-edge-compute-mec/overview

[8] "Aws local zones," 2020. [Online]. https://aws.amazon.com/cn/about-aws/global-infrastructure/localzones/

[9] "How an iot edge device can be used as a gateway," 2022. [Online]. https://learn.microsoft.com/en-us/azure/iot-edge/iot-edge-as-gateway?view=iotedge-1.4

[10] "Analytics on the edge using ibm cloud pak for data," 2020. [Online]. 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

[11] M. Ercan, J. Malmodin, P. Bergmark, E. Kimfalk, and E. Nilsson, "Life cycle assessment of a smartphone," in *ICT for Sustainability 2016*. Atlantis Press, 2016, pp. 124–133.

[12] "Dream, build, and transform with google cloud," 2011. [Online]. https://cloud.google.com/

[13] "Amazon web services," 2004. [Online]. https://aws.amazon.com/

[14] A. Hedhli and H. Mezni, "A survey of service placement in cloud environments," *J. Grid Comput.*, vol. 19, no. 3, pp. 1–32, 2021.

[15] W. Chang and P. Wang, "Write-aware replica placement for cloud computing," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 656–667, 2019.

[16] S. Slimani, T. Hamrouni, and F. Ben Charrada, "Service-oriented replication strategies for improving quality-of-service in cloud computing: a survey," *Clust. Comput.*, vol. 24, no. 1, pp. 361–392, 2021.

[17] J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," in *Noise reduction in speech processing*. Springer, 2009, pp. 1–4.

[18] H. Abdi, "Multiple correlation coefficient," *Encyclopedia of measurement and statistics*, vol. 648, p. 651, 2007.

[19] R. Gallager, "Poisson processes," in *Discrete stochastic processes*. Springer, 1996, pp. 31–55.

[20] A. Narayanan, X. Zhang, R. Zhu, A. Hassan, S. Jin, X. Zhu, X. Zhang, D. Rybkin *et al.*, "A variegated look at 5g in the wild: performance, power, and qoe implications," in *ACM SIGCOMM*, 2021, pp. 610–625.

[21] Z. Wang, Z. Li, G. Liu, Y. Chen, Q. Wu, and G. Cheng, "Examination of wan traffic characteristics in a large-scale data center network," in *ACM IMC*, 2021, pp. 1–14.

[22] B. Schlinker, I. Cunha, Y. Chiu, S. Sundaresan, and E. Katz-Bassett, "Internet performance from facebook's edge," in *ACM IMC*, 2019, pp. 179–194.

[23] R. Mok, H. Zou, R. Yang, T. Koch, E. Katz-Bassett, and K. Claffy, "Measuring the network performance of google cloud platform," in *ACM IMC*, 2021, pp. 54–61.

[24] M. Johnson, J. Liang, M. Lin, S. Singanamalla, and K. Heimerl, "Whale watching in inland indonesia: Analyzing a small, remote, internet-based community cellular network," in *WWW*, 2021, pp. 1483–1494.

[25] E. Xu, M. Zheng, F. Qin, Y. Xu, and J. Wu, "Lessons and actions: What we learned from 10k {SSD-Related} storage system failures," in *USENIX ATC*, 2019, pp. 961–976.

[26] M. Fida, E. Acar, and A. Elmokashfi, "Multiway reliability analysis of mobile broadband networks," in *ACM IMC*, 2019, pp. 358–364.

[27] M. Xu, Z. Fu, X. Ma, L. Zhang, Y. Li, F. Qian, S. Wang, K. Li, J. Yang, and X. Liu, "From cloud to edge: a first look at public edge platforms," in *ACM IMC*, 2021, pp. 37–53.

[28] W. Rafique, L. Qi, I. Yaqoob, M. Imran, R. Rasool, and W. Dou, "Complementing iot services through software defined networking and edge computing: A comprehensive survey," *IEEE Commun. Surv. Tutor.*, vol. 22, no. 3, pp. 1761–1804, 2020.

[29] Y. Zhang, J. Liu, C. Wang, and H. Wei, "Decomposable intelligence on cloud-edge iot framework for live video analytics," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8860–8873, 2020.

[30] X. Jiang, F. Yu, T. Song, and V. Leung, "A survey on multi-access edge computing applied to video streaming: some research issues and challenges," *IEEE Commun. Surv. Tutor.*, vol. 23, no. 2, pp. 871–903, 2021.

[31] P. Mu, J. Zheng, T. Luan, L. Zhu, M. Dong, and Z. Su, "Amis: Edge computing based adaptive mobile video streaming," in *IEEE INFOCOM*. IEEE, 2021, pp. 1–10.

[32] Y. Gao, C. Zhang, Z. Xie, Z. Qi, and J. Zhou, "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.

[33] "Kubeedge: Kubernetes native edge computing framework (project under cncf)," 2019. [Online]. https://github.com/kubeedge/kubeedge

[34] "Openyurt: Extending your native kubernetes to edge," 2020. [Online]. https://github.com/alibaba/openyurt

[35] "Baetyl: Extend cloud computing, data and service seamlessly to edge devices," 2019. [Online]. https://github.com/baetyl/baetyl

[36] E. Cortez, A. Bonde, A. Muzio, M. Russinovich, M. Fontoura, and R. Bianchini, "Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms," in *Symposium on Operating Systems Principles*, 2017, pp. 153–167.

[37] "Borg cluster traces," 2019. [Online]. https://github.com/google/cluster-data

[38] "Alibaba cluster trace," 2020. [Online]. https://github.com/alibaba/clusterdata