# From Cloud to Edge: A First Look at Public Edge Platforms

Mengwei Xu[1], Zhe Fu[2], Xiao Ma[1], Li Zhang[1], Yanan Li[1], Feng Qian[3], Shangguang Wang[1], Ke Li[4], Jingyu Yang[4], Xuanzhe Liu[5]
Beijing University of Posts and Telecommunications[1]
Tsinghua University[2], University of Minnesota - Twin Cities[3]
Unaffiliated[4], Peking University[5]

## ABSTRACT

Public edge platforms have drawn increasing attention from both academia and industry. In this study, we perform a first-of-its-kind measurement study on a leading public edge platform that has been densely deployed in China. Based on this measurement, we quantitatively answer two *critical* yet *unexplored* questions. First, from end users' perspective, what is the performance of commodity edge platforms compared to cloud, in terms of the end-to-end network delay, throughput, and the application QoE. Second, from the edge service provider's perspective, how are the edge workloads different from cloud, in terms of their VM subscription, monetary cost, and resource usage. Our study quantitatively reveals the status quo of today's public edge platforms, and provides crucial insights towards developing and operating future edge services.

## CCS CONCEPTS

• **Networks** → **Network measurement**; • **Computer systems organization** → **Grid computing**.

## KEYWORDS

Measurement Study, Edge Computing, Workloads Analysis

**ACM Reference Format:**
Mengwei Xu[1], Zhe Fu[2], Xiao Ma[1], Li Zhang[1], Yanan Li[1], Feng Qian[3], Shangguang Wang[1], Ke Li[4], Jingyu Yang[4], Xuanzhe Liu[5]. 2021. From Cloud to Edge: A First Look at Public Edge Platforms. In *ACM Internet Measurement Conference (IMC '21), November 2–4, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3487552.3487815

## 1 INTRODUCTION

By bringing computation and storage closer to end users, edge computing is expected to benefit a wide range of applications such as auto-driving, AR/VR, IoTs, and smart cities. Edge computing can be instantiated by various paradigms such as cloudlet [82] and MEC [51]. This work targets at *public edge platforms* (or edge clouds), which deploy massive yet lightweight datacenters (DCs) decentralized at different geographical locations and provide hardware resources to third-party customers. Such platforms are increasingly popular, because they inherit the key spirits from commercial cloud computing that has proved its tremendous success in the past decade. For example, major cloud providers are building their public edge platforms such as Azure Edge Zone [13] and AWS Local Zones [11].

| Platform | Regions / Coverage | | Density ($10^6 mi^2$) | Platform | Regions / Coverage | | Density ($10^6 mi^2$) |
|---|---|---|---|---|---|---|---|
| AWS EC2 | 24 | Global | 0.13 | MS Azure | 33 | Global | 0.17 |
| | 6 | U.S. | 1.58 | | 8 | U.S. | 2.11 |
| Google Cloud | 24 | Global | 0.13 | Alibaba Cloud | 23 | Global | 0.12 |
| | 8 | U.S. | 2.10 | | 12 | China | 3.23 |
| Azure Edge Zones | 5 | U.S. | 1.32 | Huawei Cloud | 5 | China | 1.35 |
| AWS Wavelength + Local Zones | 14 | U.S. | 3.70 | NEP (our study) | >500 | China | >135 |

**Table 1: A comparison of NEP's deployment with other popular cloud/edge services. Dated to May. 26, 2021.**

Edge platforms offer several major advantages compared to classic cloud computing, such as much lower network latency, improved application performance, and potentially reduced operational costs. Although these benefits are qualitatively known, their *quantitative characteristics* in operational environments are far from being comprehensively studied. In this paper, we conduct to our knowledge the first measurement study of a commercial public edge platform in the wild, from both the end users' and the edge providers' perspectives. For the former, we investigate key metrics that are perceivable by edge customers (who deploy their apps on edges), such as end-to-end latency, throughput, and application QoE; for the latter, we take a closer look at the edge workload dynamics. Such a "dual" approach helps reveal a complete landscape of the edge ecosystem.

**Challenges** We face several challenges in this study.

First, edge servers are more geographically distributed compared to traditional cloud servers (Table 1), thus requiring more effort on conducting measurements at a large number of vantage points. To this end, we perform a country-wide crowd-sourced study involving 158 participants, who run our custom testing tool on their mobile devices. We obtained meaningful results from 41 cities in China over diverse access networks (WiFi/LTE/5G). We also develop two user applications that can benefit from edge computing: cloud gaming and live video streaming, and deploy them over commercial edge/cloud services. Our own implementations allow us to instrument the apps and obtain detailed QoE information.

Second, obtaining an insider's view of operational edge service providers is difficult. In this study, we collaborate with a commercial, multi-tenant edge service provider, referred to as NEP (**N**ext-generation **E**dge **P**latform[1]). As a leading edge service provider in China, NEP has operated for more than 3 years, serving a wide spectrum of applications used by millions of users. The deployment scale of NEP is significantly larger than the aforementioned edge

[1]NEP is commercially known as Alibaba ENS [15].


*IMC '21, November 2–4, 2021, Virtual Event, USA*

ACM ISBN 978-1-4503-9129-0/21/11…$15.00
https://doi.org/10.1145/3487552.3487815

platforms or popular cloud platforms as summarized in Table 1. We collected detailed usage traces of *all* Infrastructure-as-a-Service (IaaS) VMs in NEP's DCs for three months, and use them to profile the edge workloads.

Third, ideally we would like to quantitatively compare edge to cloud in terms of their performance and server workload. Through active measurements, we compare NEP's performance with Alibaba Cloud ECS (AliCloud) [10], a leading cloud provider in China. The server workload comparison is much more challenging as few cloud providers release their workload traces. To this end, we compare our NEP dataset with the Azure cloud dataset [38] collected in 2019 – the only touchable, full cloud workload that we are aware of [2]. We admit that this comparison is not perfectly apples-to-apples as the Azure data is mainly for the U.S. market and the data collection period is different. Yet, this is the best we can achieve due to a lack of publicly available workload traces. We therefore draw our conclusions in a conservative and cautious manner – only when we are confident that the disparities we show between the NEP and Azure datasets are very likely attributed to the inherent differences between cloud and edge.

**Findings** We summarize our key findings below.

*(1)* ***Network latency*** *(§3.1)* is the key metric that edges are expected to improve. Our crowd-sourced results show that NEP offers a lower network delay: the median RTTs between users and the nearest edge DC are 10.5ms/34.2ms/11.7ms for WiFi/LTE/5G networks, which are 1.89×/1.42×/1.35× lower than the nearest cloud DC of AliCloud, respectively. The reduced network jitter is even more significant (~5×). However, NEP still cannot (or barely) meet the requirements of delay-critical applications like cloud VR/AR (5ms–20ms) [8] and auto-driving (10ms) [7]. This is because the nearest server of NEP is still 5–12 (median: 8) hops away from end users, instead of 1–2 hops as commonly envisioned for edge computing [51, 82]. The network performance of NEP can therefore be further improved by a denser deployment of DCs and by sinking DCs into the ISP's core networks or even cellular base stations.

*(2)* ***Network throughput*** *(§3.2)* We find that by bringing servers closer to users, NEP improves network throughput only when the last-mile bandwidth capacity is high enough, e.g., >200Mbps like 5G downlink. Otherwise, e.g., for WiFi and LTE, the end-to-end network throughput is bottlenecked by the wireless hop instead of the Internet; therefore edges exhibit no improvements over remote clouds. Considering the rare use cases where such high throughput is demanded by today's application and its incurred high operational cost, we believe that throughput is not a primary advantage of NEP-like edges at this moment. However, this situation may change in the near future, when 5G shifts the bottleneck from the last mile to the wired Internet.

*(3)* ***Application QoE*** *(§3.3)* Through controlled experiments, we observe that placing the gaming backend on nearby NEP sites can noticeably improve the response delay compared to remote clouds (91ms vs. 145ms). To further enhance the QoE, optimizations shall focus on server-side gaming execution, e.g., through higher CPU parallelism or hardware acceleration. For live streaming, NEP only brings modest improvement (up to 24% of streaming latency) and the streaming delay remains high (400ms without a jitter buffer). The reasons are twofold: (i) NEP's edge resources can effectively reduce the propagation delay, but not necessarily the transmission delay; (ii) more importantly, the bottleneck is oftentimes the content processing/computation rather than the network. Future efforts should thus focus on improving the hardware capacities (e.g., the camera's image signal processor) and the system-software stacks.

*(4)* ***Characteristics of NEP's edge VMs*** *(§4.1)* We find that NEP's VMs and their incurred workloads are noticeably different from those of Azure. NEP's VMs are often equipped with more resources, including both CPU (median: 8 vs. 1) and memory (median: 32GBs vs. 4GBs). However, NEP's resource utilization is much lower than Azure (6× lower for mean CPU usage), indicating that its customers may over-provision the hardware resources. We identify two possible reasons for that: (i) NEP apps are mostly delay-critical and their usage patterns exhibit high temporal variations, forcing its customers to reserve more resources to ensure a consistently good QoE; (ii) it remains difficult for edge customers to forecast the fluctuating resource demands at different locations. Our findings suggest that existing resource allocation challenges are amplified when apps are migrated from clouds [39, 67] to edges.

*(5)* ***Resource usage*** *(§4.2)* is a critical piece of information that an edge service provider needs to closely keep track of, and ***(6) load balancing*** (§4.3) facilitates edge services' SLA by adapting resource allocation to applications' needs. We find that for NEP, its resource usage is highly unbalanced across servers (up to 14× from the same site), across sites (up to 731× in the same province), and across the VMs hosting the same app (up to 3×). These observations indicate possible imperfections of NEP's VM placement and selection strategies. We also identify important factors to consider when designing load balancers for NEP-like edge platforms, including the fluctuating usage patterns, the geo-sensitive resource demand, and the decoupled VM placement and end-user request scheduling strategies. Fortunately, our further experiments of ***(7) resource prediction*** (§4.4) show that NEP workloads have stronger seasonality and are easier to predict as compared to Azure. It offers a good opportunity for more fine-grained, intelligent resource management.

*(8)* ***Monetary cost*** *(§4.5)* is a critical dimension of commercial edge services but is rarely studied by prior literature. We find that the apps deployed on NEP are mostly bandwidth-hungry, which often constitutes most of their billing cost. Since data is generated by nearby users, deploying applications over NEP is indeed much cheaper compared to AliCloud (recall that both are Chinese providers) – about 45% of cost reduction on average, and up to 98% for network cost reduction, making it one of the strongest incentives to move from cloud to NEP. However, we also discover that two types of apps may not get financial benefits if deployed on NEP: (i) apps with high hardware demand but low network demand, as NEP charges slightly higher on hardware resources; (ii) apps with high temporal network usage variance, as NEP adopts a very coarse-grained billing model.

**Contributions** This work presents a first-of-its-kind measurement study of a major public edge platform in China from both the end users' and the edge operators' perspectives. Our contributions consist of detailed characterizations of the performance, workload, and billing of the edge platform. Based on our findings, we summarize

[2] §2.2 summarizes the publicly available workload traces and clarifies the reason why Azure dataset is the only appropriate one for comparison.

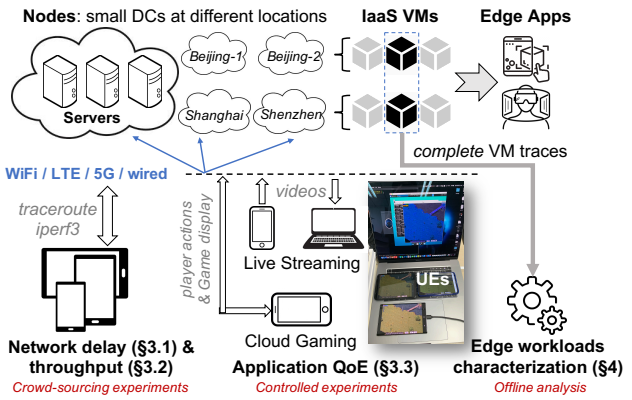



**Figure 1: The overall organization of NEP platform (top) and our measurement methodologies (bottom).**

the key lessons learned in §6. Given the increasing prevalence of edge computing (in particular fueled by 5G), our work provides crucial insights towards improving future edge services. Meanwhile, our results also provide an important "baseline" for studying how it evolves in the future.

**Open source** The edge workloads traces we collected are available at https://github.com/xumengwei/EdgeWorkloadsTraces.

## 2 THE NEP EDGE PLATFORM

**Context and terminology** The primary differences between NEP and cloud providers, e.g., Alibaba Cloud (AliCloud) and AWS EC2, are how physical servers are located, organized, and maintained. While cloud providers also build their large data centers across different geographical locations, edge providers take a step further and treat such geo-distribution as their first-class target. We call data centers at different locations **sites**. A site consists of many **servers**, and each server hosts many VMs. The customers of NEP typically subscribe to one or multiple VMs, on which they operate applications or services. In this study, we assume the VMs that use the same system image and belong to the same user serve the same application (**edge app**). Figure 1 shows the overall organization of NEP and our measurement methodology as will be discussed in the following subsection.

**NEP overview** While still at its early stage, NEP has now become a leading edge platform in China. Compared to cloud platforms that typically have less than 10 sites in one country, NEP's site number is about two orders of magnitudes larger and the number is still fast-growing. Such a difference leads to a significant chain reaction in other aspects of the platforms such as app performance, resource usage, and so on as we will characterize in the following sections. A site in cloud computing often hosts thousands or even millions of servers and the number is highly scalable; while a NEP site typically hosts only tens or hundreds of servers as constrained by the physical infrastructure, e.g., space and electricity. While NEP supports many types of services (e.g., PaaS and FaaS), the current dominant usage is Infrastructure-as-a-Service (IaaS) VMs. Thus, this paper mainly targets at IaaS VMs hosted in NEP for workload analysis. The physical servers of NEP come from many sources. The majority of them are built atop Alibaba CDN PoPs. Some are cooperatively managed by NEP and other third-part IDCs or network operators. NEP also provides business customers with edge infrastructures that are hosted on the customers' own hardware. Nevertheless, the current form of NEP is mainly based on micro datacenters and has not generally sunk into cellular core networks as envisioned by MECs [51].

**NEP operation** Just as cloud, deploying an app on NEP takes two main stages. (1) **VM placement by edge provider.** The customers first submit their resource requirements at different geographical locations to NEP administrators. For example: "I need 10 virtual machines in Guangdong province, each with 16 CPU cores and 32GB memory." Generally speaking, NEP only exposes a relatively coarse spatial granularity for customers to subscribe (e.g., province instead of site). This is to ensure an elastic resource allocation strategy, as the resources available on each site are very limited. Once a subscription request arrives, NEP returns one feasible allocation. While there are often thousands of options, NEP favors the servers that are low in usage in terms of the sales ratio and actual CPU usage (mean and max). (2) **End-user traffic scheduling by edge customers.** Once NEP allocates the VMs, customers take over the whole control of those VMs. They are also in charge of scheduling the requests from end users to a given VM. Similar to traffic routing in content delivery network (CDN), edge customers typically route user requests to their nearby sites based on DNS or HTTP 302.

### 2.1 Measurement Methodology

We collect two kinds of datasets from NEP: (i) edge performance (§2.1.1), for which we *actively* build benchmark tools and obtain testing results through crowdsourcing and controlled experiments; (ii) edge workloads (§2.1.2), for which we *passively* log the edge VMs' activities and traces.

*2.1.1 Edge Performance Data Collection.* We actively collected three kinds of data: network latency, network throughput, and application-level performance, for both edges and clouds. The first two were obtained by crowdsourcing, while the other one is performed in controlled settings.

**Edge and cloud servers** (1) For latency, we set up one VM on each edge site of NEP and each cloud region of AliCloud. Those VMs are used as the ping destinations. (2) For throughput, we set up 20 NEP VMs at different cities, each with 1Gbps bandwidth capacity. We didn't use AliCloud or all NEP regions because the experiments impose too much traffic overhead. However, the 20 VMs are enough to draw our key conclusions as will be later presented. (3) For application QoE, we set up 1 nearest edge VM and 3 cloud VMs at different locations that are 670Km/1300Km/2000Km away from where the experiments are performed. Each VM has 8 vCPUs (2.5GHz), 16GBs memory, and sufficient bandwidth.

**User equipments (UE)** We use several commodity off-the-shelf UEs for crowd-sourced network measurements. For application QoE testing, we used one laptop (MacBook Pro, 2019 version, 16-inch) and three smartphones: Samsung Note 10+ (Snapdragon 855, 5G-supported), Xiaomi Redmi Note 8 (Qualcomm Snapdragon 665), and Nexus 6 (Qualcomm Snapdragon 805). We mainly used Qualcomm chipsets because the GamingAnywhere [53] framework cannot utilize the built-in codec hardware for other chips.

We mainly focus this study on smartphones because (1) smartphone is often regarded as the major type of UE for accessing edge resources, and (2) the wifi speed on smartphones and laptops are similar as we have measured.

**Testing tools and applications** We mainly used traceroute (ICMP) and iPerf3 (TCP) to obtain the network latency and throughput performance. We also built two QoE-testing apps, which are commonly envisioned to be (future) killer apps in the era of edge computing. (1) *Cloud gaming*: we adopted three desktop games (*Battle Tanks* [1], *Pingus* [2], and *Flare* [9]) to be cloud-powered based on GamingAnywhere [53], the state-of-the-art cloud gaming platform. Edge/cloud servers are to receive player actions from UEs, perform game logic, render the images, and finally encode and send them back to the UE for display. (2) *Live streaming*: we built a live streaming app based on real-time messaging protocol (RTMP) with Nginx [20] (server side, Ubuntu), EasyRTMP-Android [14] (sender UE, Android device), and MPlayer [19] (receiver UE, Mac Laptop). In this application, edge/cloud servers are to pull the videos from the sender UE, (optionally) transcode the videos, and push them to the receiver UE.

**Testing process** (1) For latency, we recruited volunteers in China using Android devices. We installed our speed-testing app on their devices, and asked them to run the tests. During testing, the app will obtain the round-trip time (RTT) to each edge/cloud VM we set up and the intermediate hops if visible. Each IP testing is repeated by 30 times. Once finished, the testing results will be encrypted and uploaded to our server, along with the network condition (WiFi/LTE/5G), testing time, and the city name. In total, we received 385 testing results (>2M pings) from Jun. 1st to Aug. 1st in 2020[3]. The results come from 158 users, covering 20 provinces, and 41 cities in China. For network type, 59%/34%/7% of the testings are performed under WiFi/LTE/5G. During each test, we ask the participants to keep their smartphones in a stationary context, e.g., no WiFi/4G switching or 4G handoff. This is ensured by our testing script that monitors the network condition and physical motions of the devices. (2) For throughput, we selected 25 volunteers at different cities, a subset from the above, to run our testing script. The script used iPerf3 to get both downlink/uplink throughput to each of the 20 edge VMs we selected, where iPerf3 runs for 15 seconds per connection. (3) The application QoE experiment was performed by the authors. Each testing was repeated across 4 different locations in the same city: campus indoor/outdoor and office building indoor/outdoor.

*2.1.2 Edge Workloads Data Collection.* This dataset contains information about every VM running on NEP from June 1st to Sep 1st, 2020. More specifically: (1) a VM table, with each VM's placement information (which server and site it's hosted at), customer information (whom it belongs to), and system information (the image id, os type, kernel number, etc); (2) the resource size (capacity) in terms of maximum CPU cores, memory, and disk for each VM and server; (3) the CPU usage reported every 1 minute for each VM; (4) the bandwidth usage reported every 5 minutes for each VM, including both private (intra-site) and public traffic.

[3]The volunteers recruited are not affiliated with NEP. All participants are paid for their efforts and the traffic data consumed in the experiments.

## 2.2 Selecting cloud workloads for comparison

The goal of this work is to compare NEP with cloud platforms to reveal their disparity, and therefore showcase the key benefits brought by NEP. Regarding the workloads comparison (§4), we investigate the cloud workloads datasets that are publicly available and summarize them in Table 2. Next, we describe these datasets in detail and explain why we choose to use or not use each of them for comparison.

- **Azure dataset** [38] is the most representative counterpart of NEP on public cloud platforms and thus comprehensively compared in this work (we used the 2019 version).
- **AliCloud dataset** [3] is not compared because: (1) It only contains the usage of containers instead of VMs, while the major form of NEP is VM; (2) Its time range is too short for certain analysis (8 days), e.g., resource usage profiling and prediction.
- **Google dataset** [92] is not compared because: (1) Its resources are not available to public but only to Google's internal developers, making it not representative of public cloud platforms. (2) The dataset access is through Google's BigQuery interface, which doesn't support complicated usage such as ML-based prediction.
- **GWA-T-12 dataset** [88] is not compared because it's too small-scale and out-of-date.

## 2.3 Ethics

When conducting this study, we take careful steps to protect user privacy and preserve the ethics of research. (i) For collecting the edge performance dataset, the data collection was approved by the Research Ethical Committee of the institutes that the authors are currently affiliated with; the collection was also approved by the participants ahead of experiments through informed consent; we collected no sensitive data from the participants except their residential city, which was input by the participants themselves. (ii) For collecting the edge workloads dataset of NEP, the data collection was approved by its customers through the service agreement; no customer identifiable information was collected during the study. When exported, the customer ID of the dataset is anonymized.

# 3 DEMYSTIFYING EDGE PERFORMANCE

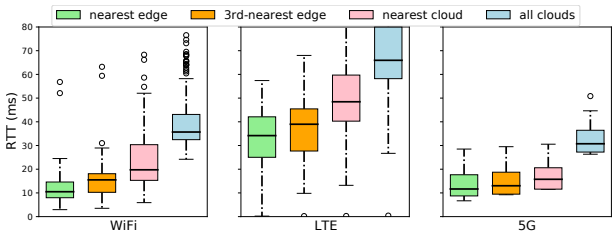
## 3.1 End-to-end Network Latency

Based on the collected data (§2.1.1), we first calculate the median network delay among each user and NEP site, and then aggregate the results across users. This is to eliminate the impacts from heavy users who have run our testing multiple times.

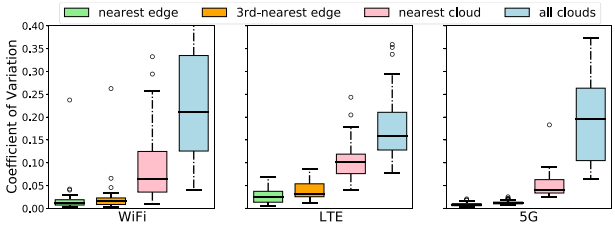
For simplicity, we define the "nearest edge/cloud" as the edge/-cloud site that has the smallest median RTT to an end user. Besides the nearest edge/cloud that represents the optimal network performance available in the current deployment of NEP/AliCloud, we include two other baselines: (i) the 3rd-nearest edge, for which we will show that there are multiple edges that are close to each user; (ii) all clouds, which is averaged across all the sites of AliCloud. This baseline reflects the performance of deploying on a centralized server for users of a nation (China in our case), a common tradeoff among economic and performance perspectives.

| | Platform | Duration | Scale | Customers | Why it's not compared? |
|---|---|---|---|---|---|
| **Azure Dataset** [38] | Azure Cloud | 1 month in 2017<br>1 month in 2019 | 2.0M VMs<br>2.7M VMs | public | The 2019 version is used. |
| **AliCloud Dataset** [3] | AliCloud ECS | 12 hours in 2017<br>8 days in 2018 | 1.3k servers<br>4.0k servers | public | Only containers' usage are included. |
| **Google Dataset** [92] | Google Borg | 1 month in 2011<br>1 month in 2019 | 12.6k servers<br>96.4k servers | Google developers | Only support BigQuery. Not public platform. |
| **GWA-T-12** [88] | Bitbrains | 3 months in 2013 | 1.75k VMs | Enterprises | Old. Not publicaly available. Small scale. |
| **Our Dataset** | NEP | 3 months in 2020 | Complete set | public | / |

**Table 2: A comparison of cloud/edge workloads traces that are publicly available. We also explain why we choose Azure as the cloud-side counterpart for head-to-head comparison in this work.**




**(a) Median RTT across users.**




**(b) RTT coefficient of variation (CV) across users.**

**Figure 2: The network delay (median RTT) and jitter (RTT CV) from end users to edge/cloud sites.**

**Overall RTT** Figure 2(a) illustrates the median RTTs across users under different network types. Under WiFi, the median RTT for the nearest edge site is 10.5ms, which is 1.89× (19.8ms) faster than the nearest cloud site, and 3.4× (35.7ms) faster than all clouds on average. The 3rd nearest edge site also provides smaller network latency (15.5ms) than the nearest cloud. Under LTE, the overall improvement decreases: the median latency for the nearest edge is 34.2ms, which is only 1.42×/1.93× faster than the nearest/all cloud sites. The decreased latency reduction comes from that the first 2 hops of LTE network incur much higher latency than WiFi (47.8ms vs. 9.0ms on average). We will give more details of hop-level latency breakdown later in this section.

For 5G[4], the median RTT of the nearest edge is only 10.4ms. Its significant improvement over LTE mainly attributes to the flatten architecture of 5G and the improved fiber fronthaul/backhaul [6, 97]. The improvement over all clouds is also tremendous (2.64×). However, the improvement is much smaller (1.35×) compared to the nearest cloud. We dig into our trace and find out that almost all our 5G testing results are from Beijing due to very limited 5G coverage in other regions in China. Since AliCloud also deploys a site in Beijing, the difference in accessing NEP and AliCloud is trivial. We expect the network improvement brought by NEP to be more significant when 5G infrastructures become more widely deployed and accessible to more end users.

[4]In China, 5G network operates at 3.5 GHz frequency. Note that comparing 5G to LTE is not the focus of this study, for which we refer readers to [73, 97].

| | Nearest edge site | | Nearest cloud site | |
|---|---|---|---|---|
| | 1st-2nd-3rd hop | Rest | 1st-2nd-3rd hop | Rest |
| **WiFi** | 44.2%-10.3%-15.1% | 30.2% | 30.1%-5.0%-11.5% | 52.5% |
| **LTE** | 10.2%-70.1%-9.4% | 10.3% | 10.1%-51.6%-13.1% | 25.2% |
| **5G** | 97.9% in total | 2.1% | 82.2% in total | 17.8% |

**Table 3: Hop-level breakdown of network delay**

We also analyze the average RTT and physical distance to the nearest edge/cloud site across users based on their locations, i.e., whether they are co-located with an edge/cloud site in the same city. The results are summarized in Table 4. In our experiments, most users (69%) are not co-located with any edge/cloud site. In such a circumstance, the average RTT is reduced from 34.97ms (to the nearest cloud) to 22.37ms (to the nearest edge) with NEP, and the geographical distance is reduced from 351km to 130km. The reduction is much more significant when the users are co-located with a NEP site but not a cloud site (18%), i.e., 47.06ms to 18.45ms. For cases where users are co-located with both edge/cloud sites, we find NEP edge can still improve the RTT. The reason is that NEP deploys multiple sites in a few cities, e.g., Beijing, so that the users in those cities can access nearer resources. In summary, while NEP delivers lower network delay to end users through resources in proximity, the benefits vary across different locations of endpoints.

**Network jitter** Many network-sensitive tasks like live streaming are required to deliver consistent, predictable user experience. To quantify the network jitter, we measure the RTT coefficient of variation (CV for short, measured as stddev/mean) during our repetitive tests (30 times) for each experiment. As illustrated in Figure 2(b), edge platform has significantly lower RTT CV (i.e., higher stability) compared to cloud platform. Under WiFi/LTE/5G, the median RTT CV is only 1.1%/2.3%/0.7% for the nearest edge and 1.5%/3.2%/1.7% for the 3rd nearest edge. Taking the nearest edge as baseline, the nearest cloud site has 5.8×/3.9×/5.7× higher median RTT CV, and the average numbers across all sites can be up to 30×. Such a low network jitter is critical to provide service-level agreement (SLA) to edge customers.

**Per-hop latency breakdown** Table 3 illustrates the hop-level breakdown of the end-to-end RTT. We highlight the latency of the first 3 hops and combine the rest. For WiFi, the first wireless hop contributes to 44.2%/30.2% of the end-to-end latency to the nearest edge/cloud. For LTE, the second hop contributes the most latency, e.g., 70.1% to the nearest edge. This is because the 2nd hop contains the network delay *accumulated* from multiple physical hops in the GTP-U tunnel, where data packets are encapsulated in GTP Protocol Data Units and the hop count is not changed during

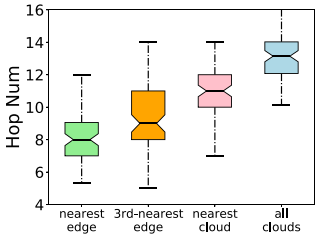



**Figure 3: Hop numbers**

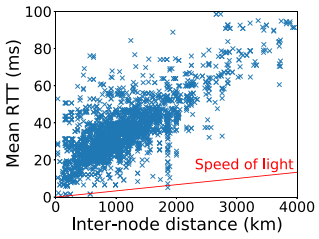



**Figure 4: Inter-site RTTs**

the transmission [24]. Therefore, the "hop" latency is longer. Such an observation is consistent with a recent measurement study [97]. For 5G, our collected trace doesn't contain the latency of the first 2 hops, possibly because the ICMP service is disabled by the operator. Instead, we report the latency of the first 3 hops in total, and find they dominate the end-to-end latency, i.e., 98% for the nearest edge. Note that, compared to cloud platforms, the current deployment of NEP mainly reduces the inter-city transmission delay (i.e., the backbone network). The traffic still needs to travel through the core network within a city to reach the edges.

**Hop number** Figure 3 illustrates the number of hops between end devices and edge/cloud servers, averaged across all network types. It shows that the hop number to the nearest edge (5–12) is much fewer than the clouds (10–16). The reduced hop number leads to lower network latency and jitter. To further reduce the hop distance, NEP needs to increase the site density and sink the resources into the core network by collaborating with operators as aforementioned.

**Inter-site RTT** We also measure the network latency between NEP's sites. We obtain the RTT between every site pair every 5 minutes in a day of June 2020, and average the results. Figure 4 illustrates the geographical distances (x-aixs) and network latency (y-axis) between edge sites. Overall, the RTTs increase with the inter-site distances, and reach 100ms when two sites are 3000km away. More importantly, it shows there are many nearby edge sites that have very low RTT, thanks to the deployment density of NEP. For each site, there are 1/3/11 nearby sites that are within 5ms/10ms/20ms RTTs on average. It promises fine-grained resource and user request scheduling between edge sites.

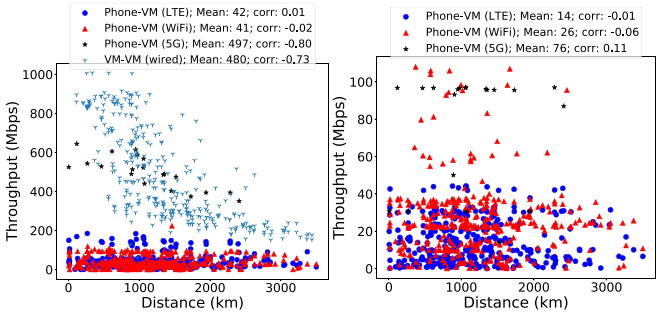
**Implications** *NEP delivers noticeably lower and more stable network delay for end users compared to AliCloud. Despite that, NEP hasn't fully reached the envisioned prospects of edge computing (even with current 5G), e.g., sub-10ms delay and 1–2 hops distance to access edge resources [51]. The last-mile hops (1st for WiFi and 2nd for LTE) become the bottleneck of network delay. To step forward, NEP needs to deploy denser sites and collaborate with operators to sink the edge resources into ISP's core networks or even cellular base stations, i.e., Mobile Edge Computing [12, 51].*

## 3.2 End-to-end Network Throughput

Between a cloud/edge VM and a client, the network throughput is bounded by the poorest link among them, e.g., the first hop for wireless access as commonly believed. Based on the data collected through crowdsourcing (§2.1.1), this section dives into the question: how does the geographic distance (or hop number) affect the

| U/E/C Locations | RTT (ms) Nearest-E | RTT (ms) Nearest-C | Dist (KM) Nearest-E | Dist (KM) Nearest-C |
|---|---|---|---|---|
| U/E & U/C co-located (13%) | 10.96 | 15.64 | 0 | 0 |
| U/E co-located (18%) | 18.45 | 47.06 | 0 | 973 |
| None co-located (69%) | 22.37 | 34.97 | 130 | 351 |

**Table 4: Average RTT and physical distance to the nearest edge/cloud server across different locations of users. "U/E/C": user, edge site, cloud site. "co-located" means the user is in a city where at least one edge/cloud site is deployed. When calculating the distance, we look at the geographic distance at city level. Numbers averaged across WiFi/4G/5G.**




**(a) Downlink throughput** **(b) Uplink throughput**

**Figure 5: The TCP-based network throughput against geographical distance. Each point represents a 15-sec iPerf-tested result. "corr" is the Pearson correlation coefficient (-1 to 1) between distance and throughput.**

network throughput between end users and data centers. By answering this question, we can learn whether or how edge platforms like NEP can improve the network throughput compared to remote clouds.

Figure 5 illustrates the overall results of our throughput testing. The key observation is that, when throughput is low e.g., ≤100Mbps for LTE and WiFi, the correlation between the distance and throughput is negligible, as indicated by Pearson correlation coefficient lower than 0.2 [28]. Noting that the 5G uplink bandwidth (mean: 52Mbps) is strictly capped by asymmetric time slot ratio in the ISP's configuration following Rel-15 TS 38.306 [24], thus its correlation with distance is also negligible. Only when the throughput reaches high, e.g., for 5G downlink (mean: 497Mbps) and wired access (mean: 480Mbps), the correlation becomes significant (corr>0.7). In such cases, the throughput degrades observably as physical distance increases. The reason is that, with LTE/WiFi access, the network throughput is usually bounded by the bandwidth capacity at wireless hop, therefore has little correlation with the distance. When the capacity is high, e.g., for 5G downlink, the bottleneck resides at the Internet link which directly correlates with the distance (or RTT [66]). It is also confirmed by our observations of the TCP congestion window size and the packet loss rate during experiments.

Note that, to have perceivable benefits from the geographically closer edge resources, two more factors need to be satisfied besides the high bandwidth capacity: (1) Applications that can generate high-volume traffic at more than 200Mbps. We find that few today's applications can do that: for example, streaming video at 4K

| | Edge | Cloud-1 | Cloud-2 | Cloud-3 |
|---|---|---|---|---|
| WiFi | 11.4ms | 16.6ms | 40.9ms | 55.1ms |
| LTE | 22.2ms | 25.6ms | 54.6ms | 63.2ms |
| 5G | 18.1ms | 22.8ms | 49.5ms | 60.8ms |

**Table 5: The RTTs of edge/cloud VMs used for QoE experiments in §3.3, averaged across different locations.**

resolution and 60FPS consumes only less than 100Mbps [27]. (2) Equally-high or even higher bandwidth needs to be allocated to the edge VMs so that the DC gateway doesn't become the bottleneck. Such high bandwidth usage, however, can be prohibitively expensive to developers.

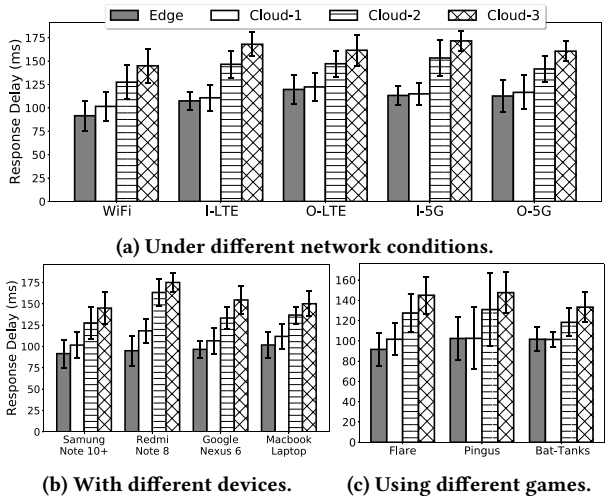
**Implications** *Bringing resources closer to users improves network throughput on NEP only with high bandwidth capacity at the last mile, e.g., for 5G downlink and wired access. Such an advantage over cloud computing, however, is weakened by the absence of ultra-bandwidth-hungry applications and the cost considerations from developers' perspectives. Given that, we conclude that improving network throughput is not a primary incentive of current edge applications on NEP. In the future, however, we believe that the throughput improvement will benefit more emerging, bandwidth-hungry edge applications.*

## 3.3 Application Performance (QoE)

This section presents the experiment results of application-level QoE. Recall that (§2.1.1) we use one nearest edge VM and 3 cloud VMs with different distances from the area where the experiments are carried out. For reference, Table 5 shows the average RTTs to those VMs in this experiment. For simplicity, we term the tests as "Cloud-1/2/3" from the nearest to the farthest. Note that the locations and resource characteristics of the servers are described in §2.1.1.

*3.3.1 Cloud Gaming.* By hosting game execution and rendering on backend servers, cloud gaming promises mobile devices the ability to play games at a lower cost [58]. Cloud gaming systems have stringent response delay requirements, as gamers may demand less than 100ms response delay [36]. With cloud servers as the backend, it is difficult for players to attain real-time interactivity in the face of wide-area network latency. With much lower latency, edge computing is expected to significantly improve the game experience [21]. We now validate our cloud gaming systems built with edge/cloud backend.

**Metrics** We follow prior work [53] to measure the end-to-end performance as the interval between a player issuing a command and the in-game action appearing on the client. We refer to this metric as *response delay*. To obtain this delay, we periodically send touch events to the game menu and wait for the menu window to pop up. In the meantime, we use another device to record the device screen at a high frequency (FPS). At offline, we use FFmpeg [16] to decode the videos and manually obtain the interval between a touch event being invoked and the menu popping up. The response delay is thus obtained as the interval between the two timestamps. The touch event is set to be visible through Android built-in tools.




**(a) Under different network conditions.**

**(b) With different devices.** **(c) Using different games.**

**Figure 6: The cloud gaming performance under different settings. Default setting: Samsung Note 10+, Game *Flare*, and WiFi. "I-": indoor, "O-": outdoor.**

The online experiment is performed automatically through Android Monkey tool [23]. For each testing, we obtain 50 data points.

**Overall results** Figure 6 shows the cloud gaming performance with different network conditions (a), client devices (b), and game types (c). Overall, with a good network condition (WiFi in our case) and nearby VMs (Edge and Cloud-1), cloud gaming can achieve less than 100ms response delay. The distance matters: remote cloud VMs lengthen the delay by up to 60ms. Among the devices, Samsung Note 10+ performs slightly better than others for its high-end chipset, but the improvement is not significant because the hardware-accelerated video decoding is fast enough for all the devices tested, i.e., less than 10ms for the default 800×600 resolution used in GamingAnywhere, and the screen fresh rate is the same (60Hz). Note that decoding is needed as the gaming server encodes the game scenes to video frames before sending to UE. Among different games, *Pingus* experiences slightly higher delay and jitter for its more complex game logic.

**Breakdown** An intuitive question is how we can further improve the 100ms response delay, e.g., within 30ms or 2-3 frames. Our breakdown results show that, the network delay of the nearest edge VM is no longer the bottleneck: the propagation delay is only 11.4ms shown Table 5 and the transmission delay (to send a frame) is less than 10ms according to our measurement. Instead, we find the major portion is on the server-side (game logic execution and rendering), which contributes to around 70ms delay. We further look into this delay and make two interesting observations: (1) While each server VM is equipped with 8 CPU cores, most cores except one run at very low utilization (less than 10%). In other words, increasing CPU cores won't help as it is difficult to parallelize the game logic. (2) Our offline experiment that hosts the server on a Macbook laptop suggests that enabling the GPU rendering can help reduce the delay by around 10ms–20ms.

**Implications** *Placing gaming backend on nearby NEP edges can help achieve less than 100ms response delay. To further enhance the experience, we need to improve the server-side gaming execution.*

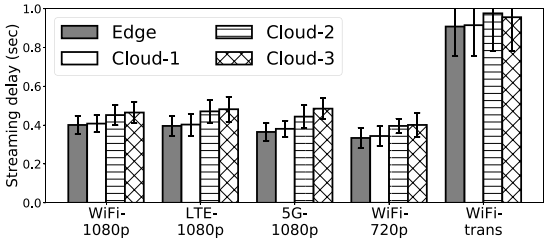



**Figure 7: The live streaming performance on edge/cloud under different experiment conditions.**

*More specifically, adapting the gaming to multi-core systems with high parallelism and applying hardware acceleration (e.g., GPU) are promising approaches.*

*3.3.2 Live Streaming.* A report [5] in 2018 shows that 42% of the population in the U.S. have now live-streamed online content. Live streaming also demands low end-to-end delay, especially for bidirectional streaming scenarios that involve human-to-human interaction, e.g., online meeting. Our workload analysis in §4.1 shows that live streaming is one major application type served by NEP. In this experiment, we assume the sender and receiver are located in the same city, which is common for many streaming scenarios such as online education. Unless otherwise specified, we stream 1080p video without transcoding, and the encoded streaming bitrate is around 5Mbps.

**Metrics** Following prior work [97], we define *streaming delay* as the amount of time between a real-world event and the display of that event on the receiver's screen. To obtain this delay, we use the sender UE to capture a millisecond-level clock (displayed by a third device), and use a fourth device to capture the running clock and the receiver UE's screen simultaneously. We then inspect the difference between the two clocks as the streaming delay. Each testing runs for 20 seconds from which we obtain 50 data points.

**Overall results** Figure 7 shows the live streaming performance under different conditions. Here, "WiFi-trans" indicates transcoding videos from 720p to 1080p on server, while others simply stream videos without transcoding. We draw the following important observations from this figure. (1) Edge servers have limited benefit in reducing the streaming delay, e.g., upmost 24% compared to the farthest cloud under 5G, mostly because the network doesn't constitute the bottleneck as we will show next. (2) Streaming images with a lower resolution can reduce the delay around 67ms (26%) from 1080p to 720p. Note that this reduction not only comes from the reduced network transmission time, but also the rendering on the receiver UE. (3) Transcoding incurs a high overhead: around 400ms (2×) from 1080p to 720p under our WiFi condition. This overhead includes both the transcoding time and server waiting time for a video segment to arrive.

All above experiments are carried out without using a *jitter buffer* on the receiver side. A jitter buffer is commonly used in video streaming to compensate transmission impairments caused by the time-variant packet delays [54, 60]. Our additional experiments show that, with a small jitter buffer (e.g., 2MBs), the streaming delay reaches as high as 2 seconds and the difference between edge/clouds becomes trivial.

**Breakdown** Even without using jitter buffer and transcoding, the streaming delay remains about 400ms. Following the approach in [27], we break down this delay to image capture, local frame processing (frame patch splice, codec, and video rendering), and RTMP network transmission delay. We draw the following findings. (1) Network delay takes around 50ms, which doesn't constitute the major bottleneck. Note that edge reduces only the propagation delay but not transmission delay. The observation is confirmed by another micro experiment, where we repeat the above experiments but deploying the server on a laptop wired to sender/receiver UEs (LAN environment). In such settings, the streaming delay is only reduced by 40ms. (2) By calculating the timestamp difference between the running clock and the sender UE's screen, we estimate the image capture plus rendering to be around 140ms. This delay is sophisticated, including the digital processing by camera image signal processor and the time spent on the system-software stack (Android in our case). (3) The encoding/decoding delays are 25ms/10ms on the sender/receiver UEs, respectively. (4) The software matters: when using FFplay [17] instead of MPlayer [19] on the receiver UE to pull and display the video stream, the streaming delay reduces almost by 90ms.

**Implications** *While NEP brings modest delay reduction to live streaming scenarios by reducing the network delay, the delay spent on image capture, transcoding, jitter buffer, and even the system-software stack remain the bottleneck to achieve real-time human interaction. Thus, it is imperative to improve the hardware capacities and the system software design in order to support the niche edge applications. Even so, live streaming has become a major application type hosted on NEP for its reduced billing cost as we will show in §4.5.*

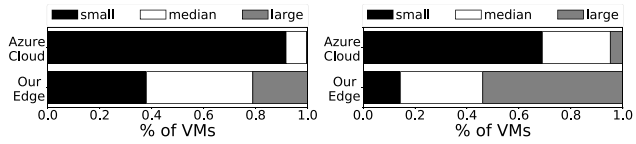
# 4 DEMYSTIFYING EDGE WORKLOADS

In this section, we characterize the workloads based on our collected VM traces (§2.1.2) and the public **Azure Dataset** collected from Azure's *entire* VM sets [38] (2019 version).

## 4.1 Applications and VM Subscription

**Application type** We investigate the major customers of NEP. We classify them into different categories, and the most popular ones are: video live streaming, online education, content delivery, video/audio communication, video surveillance, and cloud gaming. Most of them have two common characteristics: (1) Network-intensive: they stream a lot of data, mostly videos. §4.5 will show that edge platforms like NEP are more budget-friendly to the applications with high bandwidth usage. (2) Delay-critical: user interaction is often involved, either unidirectional or bidirectional. This is because edge services can provide lower and more stable network performance. In fact, those two factors are the major incentives to decentralize cloud applications to edges.

**VM size** We compare the VM sizes, i.e., the amount of resources allocated to each VM on NEP and Azure Cloud. Note that NEP provides very similar VM configuration options in terms of CPU and memory to customers as Azure Cloud does. Illustrated in Figure 8, our key observation is that NEP VMs typically request more resources than Azure VMs. Overall, the median number of CPU cores and memory requested on NEP and Azure are (8 vs. 1) and (32GBs vs. 4GBs). In addition, Azure Cloud has many "low-end" VMs with




(a) # of vCPUs per VM (b) Memory size per VM

**Figure 8: NEP VMs are larger than Azure. "small/median/large": ≤4 / 5–16 / >16 CPU core or GBs memory.**

only a few CPU cores (90% VMs with ≤4 vCPUs) and relatively little memory (70% VMs with ≤4 GBs), while NEP's half VMs have more than 8 CPU cores and 16GBs memory. The median/mean storage size of NEP VMs is 100/650 GBs (not compared as Azure dataset doesn't contain storage information).

The possible reason for NEP VMs subscribing more hardware resources is that NEP's current customers are mostly business-oriented, who need to deploy commercial services or apps that are likely to be delay-critical, while Azure also serves individuals (e.g., researchers, educators) who only need very few resources per VM to complete their jobs.

**Implications** *Large VM size on NEP-like edge platforms may cause severe resource fragmentation, i.e., the bin-packing problem [39, 67], hindering a high sale ratio for each server as we will show next. To mitigate such fragmentation, techniques like dynamic VM migration [70] and resource disaggregation [87] may help.*

**VM numbers per app**

As shown in Figure 9, customers tend to deploy *slightly more* VMs on NEP than Azure. For instance, more than 9.6% apps on NEP deploy at least 50 VMs, while on Azure only 6.1% of apps deploy that many. The largest edge app is a CDN application which comprises of almost 1,000 VMs. There exist two possible reasons: (1) Apps deployed on NEP are more likely to be delay-sensitive than Azure, requiring a larger number of geo-distributed VMs to guarantee low delay and high reliability. (2) As aforementioned, Azure serves more small-scale businesses or individuals who only need very few VMs to operate.

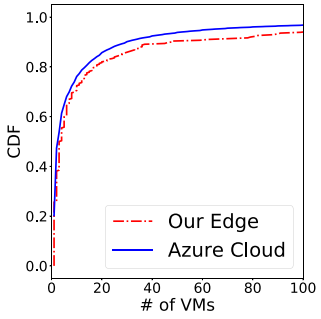
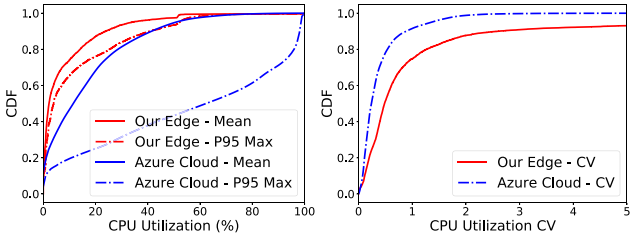



**Figure 9: Per-app VM num.**

**Implications** *Compared to clouds, managing and scheduling a large number of geo-distributed edge VMs is more challenging. First, traditional tools like Kubernetes [18] may not suffice in maintaining such VM orchestration [4]. It motivates us to improve or re-architect those tools. Second, edge customers also need more effort to schedule end-user traffic to the optimal VM in a fine-grained way. §4.3 will show that current edge customers often fail to make a good scheduling decision.*

**Servers/sites sales rate** We also summarize the resource sales rate on NEP (figure not shown), defined as the percentage of CPU/memory resources sold out to customers per site or server. We have two key observations: (1) The sales rate is highly skewed across servers and sites. For example, the 95th-percentile CPU sales rate across sites is about 5× higher than the 5th-percentile. Such skewness stems from that, in edge computing, server resource demand highly depends on the geolocations. (2) Compared to memory, CPU cores are more likely to be saturated: the median sales rate of CPU is almost 2× of the memory.




(a) Mean/Max CPU util. (b) CPU util. variance

**Figure 10: CPU utilization on NEP is lower but more variant than Azure. (a): the overall CPU utilization; (b): CPU utilization variance across time (CV).**

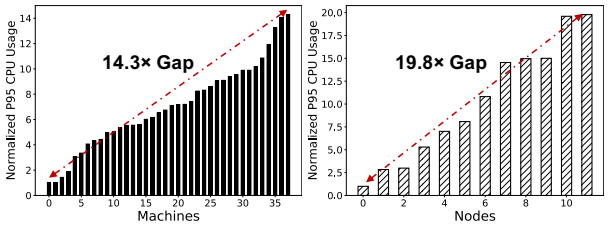
**Implications** *The skewed sales rate across sites can guide NEP providers to locate the regions with higher business returns for future investment. As edge apps increasing, it's also important to invest more in those "hot spots" to ensure good availability and elasticity of computation resources.*

## 4.2 Overall Resource Usage

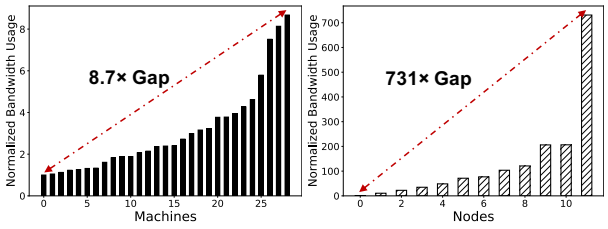
**Overall CPU usage** Figure 10(a) illustrates the per-VM CPU usage on edge/cloud as the Cumulative Distribution Function (CDF) of the average CPU utilization, and the CDF of the 95th-percentile of the maximum CPU utilization (P95 Max). The key observation is that, either by mean or P95 Max, VM CPUs on NEP are much less utilized than Azure Cloud. For example, 74% VMs on NEP have less than 10% CPU utilization on average, while on Azure Cloud only 47% VMs have less than 10% utilization. It indicates that, either unconsciously or purposely, edge customers tend to over-provision the hardware resources for VMs, which also echoes our analysis in §4.1 that edge VMs often subscribe more resources than on cloud. Diving deeper, such resource over-provision may be due to two reasons: (1) Edge apps are more likely to be delay-critical, so that more resources are needed to deliver a good quality of service. (2) It is difficult for edge customers to understand the resource demand at different locations due to the high density of edge server deployment and the temporal dynamics of user requests as will be discussed below. There, they tend to be conservative when provisioning them.

**CPU usage variance across time** We also investigate the across-time resource usage variance of edge/cloud VMs, as indicated by the coefficient of variation (CV = std/mean) of CPU usage. As illustrated in Figure 10(b), edge VMs exhibit more usage variance across time than cloud (median: 0.48 vs. 0.24). The reason is that apps deployed on edge platforms are more likely to be interactive (§4.1) so that the usage highly depends on human activities.

**Implications** *The relatively low but highly skewed CPU usage challenges the NEP's VM management. To better utilize the CPU resources, NEP may borrow existing techniques from cloud computing research, e.g., smart VM placement algorithms based on VM resource usage prediction [35, 46, 65]. An alternative approach is to employ*

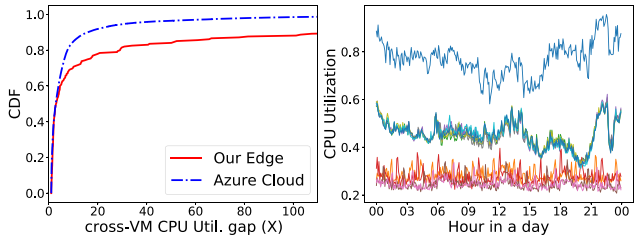



(a) Cross-server CPU usage (b) Cross-site CPU usage




(c) Cross-server NET usage (d) Cross-site NET usage

**Figure 11: The resource usage across machines/sites is highly unbalanced. All sites are randomly sampled from Guangdong Province, and the machines are from a random site. For (a)/(b): a machine's CPU usage is calculated as the weighted (by requested cores) CPU usage of all its hosted VMs, and a site's CPU usage is averaged across all its machines. For (c)/(d): the bandwidth usage of a machine/site is summed across all the VMs hosted in the machine/site. For each figure: all numbers are normalized to the smallest one.**




(a) CPU usage gap between the same app's VMs. (b) CPU usage of 11 VMs from the same edge app.

**Figure 12: The CPU usage of the same app's VMs is highly unbalanced. In (a), the usage gap of each app is measured as the 95th-percentile divided by the 5th-percentile of the mean CPU usage of all its VMs.**

*more elastic computing forms, e.g., containers, together with IaaS VMs on the same server. §6 will further discuss the opportunities and challenges.*

## 4.3 Resource Load Balance

Resource usage balance (or load balance) is critical to the application QoE on multi-tenant cloud platforms [44, 77, 83]. Such importance is further amplified on edge platform that hosts delay-critical apps and needs to provide high uptime SLA. For example, excessively high CPU usage will cause compute tasks to be delayed and high bandwidth usage may cause traffic congestion and long network delay. Therefore, we investigate the load balance of NEP and Azure, from the perspectives of both physical servers/sites and apps.

**Load balance from servers/sites perspective** [5] We find the resource usage is highly skewed. Figure 11 shows a case of 11 sites from the same Province in China and the servers in one of the selected sites. As observed, across servers in the same site the bandwidth usage gap can be up to 19.8×. The gap is even more significant across sites, i.e., 8.7× for P95 Max CPU usage and 731× for bandwidth usage.

Note that load balance is one of the major targets in NEP's VM placement strategy (§2). The above unbalanced load can be accounted to three main reasons. (1) Even for a given VM, its resource usage can change dramatically over time. Figure 13 shows an example of 4 random VMs' usage within three months. For 2 VMs among them ("VM-1" and "VM-2"), the weekly-averaged bandwidth usage varies in a dramatic and unpredictable way. In the extreme case, a VM's (black line) bandwidth usage goes down from almost 12Gbps to 4Gbps, and then 0.2Gbps within three consecutive weeks (6th-8th), and then goes up back to 4Gbps in the 12th week. (2) As NEP is still evolving rapidly, new sites are added to NEP frequently. This also explains why the resource usage skewness is more severe across sites than servers. With the arrival of both sites and VM subscriptions, it becomes difficult to balance the resource usage. (3) The strategies of VM allocation and end-user request scheduling are made by edge providers and customers independently. Such a separation hinders load balancing.

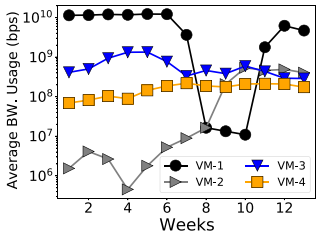
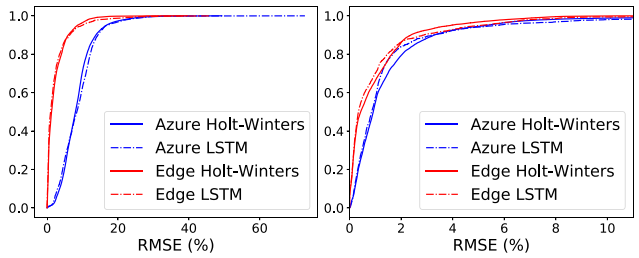



**Figure 13: The bandwidth usage of VMs may vary significantly across time.**

**Implications** *Given the observed dynamics and complexity of VM resource usage, we found it's extremely difficult to design an effective static resource allocation strategy. Instead, we envision that dynamic VM migration [35, 65] can better balance the across-server resource usage. Perhaps a more fundamental approach is changing the way of resource allocation from VM-based to more elastic ones (e.g., FaaS or serverless [22, 56]).*

**Load balance from app perspective** We further investigate the resource usage of VMs from the same app. The results are illustrated in Figure 12, where subplot (a) shows how unbalanced the VM loads from the same app are. Our major observation is that there are much more apps with highly unbalanced cross-VM usage on edge than cloud. For instance, 16.3% of edge apps have more than 50× cross-VM usage gap (defined as the 95th-percentile divided by the 5th-percentile of the mean CPU usage of all the VMs that an app uses) on NEP, while on Azure only 0.1% of apps have that large unbalanced CPU usage. Figure 12(b) zooms into one app and shows its 11 VMs' CPU utilization in a day (one curve corresponds to one VM). As observed, there's one VM running at very high load, e.g., e.g., for more than 33% of the time, the CPU utilization is higher

[5] Since the Azure dataset doesn't contain the VM placement information, we cannot make the comparison here.




**(a) Maximal CPU prediction** **(b) Mean CPU prediction**

**Figure 14: Edge VMs' CPU usage is easier to predict than Cloud VMs. Results accumulated across all VMs. Prediction time window length: half hour.**

than a typical safe threshold of 80%. In contrast, some other VMs' CPU utilization is constantly below 30%.

**Implications** *Given the importance of load balance, however, our results demonstrate that current edge apps deployed on NEP often fail to deliver this goal. Indeed, achieving load balance on edge platforms is difficult, because the VMs are geo-distributed and their resource usage patterns may change over time as aforementioned (Figure 13). To handle this challenge, we can resort to: (1) Dynamically adapting the resources of each VM to what's needed. This approach, however, requires rebooting the VM that can take tens of seconds or even minutes, as current edge/cloud platforms don't support hot resource scaling. (2) Load-aware traffic scheduling from end users to VMs considering the current loads on each available VM. This is similar to global server loading balancing (GSLB) commonly used in web traffic management and application data delivery [42]. However, edge customers face unique challenges in load balancing because inter-site request scheduling may increase the user-perceived network delay. Even so, we believe a load balancer is useful in edge platforms as the network delay between nearby edge sites is already small (§3.1).*

## 4.4 VM Usage Prediction

VM usage prediction is a critical feature in data center management [30, 38, 45]. In this section, we compare the difficulties of VM usage prediction for edge and cloud. To be fair, we use 1-month data from both Azure and NEP, with each VM's data split to ML training (3 weeks) and testing (1 week). The task is to predict the max/mean CPU usage of the next half-hour window based on the historical data. We try two algorithms, Holt-Winters [32] and LSTM [50], which are commonly used for workloads prediction [48]. The LSTM model has 1 layer and 24 units (2496 weights). The two models are trained and tested on each separated VM for predicting maximal and mean usage, respectively. We use root mean square error (RMSE) as the metric to obtain the prediction accuracy.

Figure 14 shows the prediction accuracy of max and mean CPU usage. Both algorithms achieve higher accuracy on the edge workloads. For example, the Holt-Winters algorithm achieves a 2.4% error in predicting the maximal CPU usage on NEP's workloads, much lower than on Azure Cloud (8.5%). To predict the mean CPU usage, NEP also has higher accuracy than Azure Cloud but the difference is smaller. The reason is that the prediction accuracy is already high enough for both workloads (median error ≤2%).

| **Baselines normalized to NEP (in times ×)** | | *On-demand, by bandwidth* | *On-demand, by quantity* | *Pre-reserved (fixed)* |
|---|---|---|---|---|
| **vCloud-1** | Range: | 0.50×–6.88× | 0.60×–14.98× | 1.03×–41.02× |
| | Mean: | 1.82× | 2.76× | 4.93× |
| | Median: | 1.21× | 1.97× | 3.84× |
| **vCloud-2** | Range: | 0.64×–6.43× | 0.60×–14.97× | 1.03×–14.87× |
| | Mean: | 1.76× | 2.66× | 4.82× |
| | Median: | 1.25× | 1.97× | 3.56× |

**Table 6: NEP can significantly reduce the monetary cost compared to two cloud counterparts. The cost includes both hardware and bandwidth. The numbers are summarized over 50 heaviest apps.**

The notable difference shown in Figure 14 comes from the disparate characteristics of edge and cloud. We dig into the reason by calculating the VMs' seasonality [96], an indicator of the strength of the usage patterns across time. It shows that edge VMs experience stronger seasonality (mean: 0.42) than cloud VMs (mean: 0.26). Apparently, with stronger seasonality, ML algorithms can better predict the usage based on historical data. The high seasonality is possibly attributed to the fact that more services deployed on edges follow end users' daily activities.

**Implications** *With stronger seasonality and better predictability compared to cloud VMs, edge VMs offer a good opportunity for more fine-grained, smarter resource management. For example, knowing the future CPU usage can guide VM allocation and migration, thus help avoid server malfunction or even crash induced by CPU overload or network congestion.*

## 4.5 Monetary Cost of Edge Apps

In this subsection, we investigate the monetary cost billed to edge customers for deploying edge apps on NEP. For comparison, we also estimate the cost for the same hardware subscribed and workloads incurred on clouds.

**Baseline** We use "virtual baselines" that simulate the situation if NEP's edge apps were deployed on cloud platforms. It works by clustering and merging the VMs' usage (both hardware and bandwidth) of NEP into the site distribution of cloud platforms based on geographical distances. Here, we use two most popular cloud platforms in China: AliCloud (*vCloud-1*) and Huawei Cloud (*vCloud-2*).

**Billing model** Appendix A elaborates the detailed difference of their billing models. To summarize, NEP and clouds charge the hardware resources (CPU, memory, storage) in a similar way. For network, most cloud platforms support 3 kinds of billing models: by bandwidth (on-demand), by traffic quantity (on-demand), and by pre-reserved fixed bandwidth. NEP currently only supports the first one, and even for this method, there are two notable differences.

- NEP's network billing is much cheaper than AliCloud in unit price, up to 13× depending on the geo-locations. This is because edge servers handle requests from nearby locations, which means the traffic won't travel far along the network path. This reduces the Internet backbone traffic, and leads to reduced operational cost for NEP and henceforth for the edge customers, compared to cloud platforms.
- NEP charges by the peak bandwidth usage per day, while AliCloud charges in a more fine-grained way, i.e., peak bandwidth usage per minute. This is in line with the billing model of NEP's

ISP, likely because the ISP wants to mitigate the burstiness of the edge traffic that often exhibits high variance over time as shown in §4.2.

To put it simply, NEP charges network usage in a cheaper yet less elastic way than clouds due to its traffic characteristics.

**Overview** Table 6 summarizes the cost difference if the edge apps were moved from NEP to the cloud platforms (using the NEP's cost as the baseline). Among the three network billing models, on-demand by bandwidth often costs less. However, even compared to this model, NEP can significantly reduce the cost. The average cost saving against vCloud-1/vCloud-2 is 45% (= 1 − 1/1.82)/43% and up to 85%/84%. This is because NEP has a cheaper bandwidth unit price than AliCloud. Only a few apps NEP charges more than AliCloud. Diving deeper, we find those apps either have high hardware resource demand or high bandwidth variance across time. For example, an online education app has most of its traffic from 9:00 AM-12:00 PM. Its peak (max) bandwidth usage is more than 10× higher than its average usage, while for other apps the variance is mostly between 1.5× and 4×. For those education apps, AliCloud is more cost-friendly as it charges by minute while NEP charges by the peak bandwidth usage per day as aforementioned.

**Breakdown** We then break down the bill to hardware and network bandwidth cost. NEP often charges slightly more than AliCloud (3%–20%) for each app's hardware resources, because of the relatively higher hardware maintaining cost on NEP currently. However, NEP can significantly reduce the network bandwidth cost (up to 90%), and the network resource often dominates the cost, i.e., 76% on average and up to 96%. Overall, the current customers of NEP are mostly video-related (as discussed in §4.1), of which the bandwidth cost is much higher than hardware resources. In fact, cost efficiency is one of the major incentives for those customers to move their services from clouds to NEP.

**Implications** *For NEP customers, deploying apps on its servers can significantly reduce the monetary cost due to cheaper bandwidth cost. However, two kinds of apps may be exceptions: (1) apps with high hardware resource demand but less network demand; (2) apps with very high network usage variance across time. For edge providers, it remains challenging to offer good billing elasticity to customers because of the high traffic variance across time.*

## 5 IMPLICATIONS

We summarize the key implications from our measurements. Note that some of our recommendations to improve NEP may be common practices in cloud datacenters. Yet, they remain an open problem in large-scale geo-distributed edge platforms as we have confirmed with the NEP development team.

**Killer apps for NEP-like edges** In the past decade, the research of edge computing is far ahead of its commercialization [71], mixed with real demands and hype. Looking into NEP, we find reducing network cost is, for now, a key incentive to move applications from clouds to edge platforms like NEP, making *video-related applications* major customers, e.g., live streaming (§4.1). Other network-level metrics such as network latency and throughput also get improved with NEP to some extent. The improvement delivered to application-level QoE, however, can be diminished by many other practical factors beyond the network as demonstrated in §3.3. To avoid this, the hardware/software stacks of the edge infrastructure also need to be enhanced. This will allow NEP-like edges to better serve emerging computation-heavy applications such as AR/VR and autonomous driving.

**Sites as an integrated cluster** Unlike cloud platforms, edge platforms have very limited per-site resources but a high density of site deployment, which necessitates cross-site coordination. Such a need has been recently recognized by the clouds as well [91]. Treating edge sites as an integrated cluster facilitates the overall infrastructure management, but also brings unique challenges due to its nature of geo-distribution and ultra-low delay requirement. Many of our observations (e.g., in §4.2 and §4.3) motivate cross-site VM migration for balancing the resource usage and reducing resource fragmentation, etc. While the technique has been extensively explored on both cloud [35, 49, 65] and edge [75, 81], it remains challenging because of the high migration delay and the impacts on the app QoS [26, 29].

**Decomposing edge services** While heavy IaaS VMs dominate the current usage of NEP, we believe the future of public edge platforms should embrace more elastic computing paradigms, e.g., microservices [72] and serverless computing [22, 56]. They help facilitate flexible resource management and fine-grained billing, which can benefit NEP as highlighted in §4.3 and §4.5. However, such elasticity comes at a price. For example, serverless computing has been criticized for its slow cold start [86, 95]. Existing solutions to mitigate such slow start, e.g., highly optimized function loader and executor [25, 43, 74], can barely meet the requirements for ultra-low-delay edge applications.

**Cross-sites traffic scheduling** Given massive, decentralized sites, which one should be responsible to handle a certain request from users? Such a traffic scheduling strategy should not only satisfy the QoE of each application, but also consider cross-site load balance. The current scheduling policy of NEP, which is demonstrated to be oftentimes ineffective in §4.3, is owned by the developers (as clouds typically do). On the opposite side, if the scheduling is done by the platform, it's difficult to guarantee the application QoE due to a lack of application-specific information. We believe this is a new and open problem faced by edges that differ from clouds in many aspects, including the application programming model, resource allocation policy, and business model.

**Workloads profiling and prediction** have been extensively studied for cloud platforms [30, 38, 45]. Directly applying them to NEP may not suffice considering the distinct workloads running atop them. As a concrete example, scheduling VMs to different sites based on the *past* resource usage prediction will likely lead to the *future* usage change of that VM itself, as the usage pattern of edge VM highly depends on the geo-locations. Such change may lead the prediction to be invalid shortly.

## 6 LIMITATIONS AND FUTURE WORK

**Dataset representativeness** Despite the best efforts we committed, the cloud-side workload dataset (Azure) still doesn't perfectly align with our NEP dataset, i.e., they were collected from different countries and at different times. Thus, we tend to draw our conclusions conservatively. Even so, we believe that the lessons learned from the study are valid for two reasons. First, Azure is a global

CSP with sites also deployed in China as NEP does. Second, the Azure cloud workloads are relatively stable from its 2017 version to 2019 version; we therefore expect the workloads to exhibit similar characteristics in 2020 when the NEP dataset was collected.

**Experimental settings** Apart from the workloads traces, our actively collected dataset was collected by us through crowdsourcing-based controlled experiments. We identify the following imperfections in carrying out those experiments.

- In §3.1, we use ICMP-based ping instead of TCP to measure the network delay mainly because it's a built-in feature of non-rooted Android devices. However, TCP-based ping is regarded more representative of normal workloads as ICMP is often treated with different priority by cloud providers than regular TCP/UDP traffic [52].
- In §3.3 we implement two edge applications following their typical design and using state-of-art supporting libraries. Despite that, we admit that our (best-effort) implementation and deployment may not perfectly reflect those of commercial edge apps. In the future, we will benchmark more applications and their diverse deployment configurations to comprehensively study how edge computing boosts the application QoE.
- The number of endpoints (users) participating in our user study is limited. We plan to further scale it up in future work.
- We did not investigate the network-layer characteristics (e.g., routing) of NEP traffic. Understanding them can facilitate network-edge cooperation through, for example, improved traffic engineering.

**NEP as an early adopter** NEP is now at an early stage (3 years since release) and not all its characteristics match what researchers commonly envision for "ideal" edge computing, e.g., in regard to the deployment density, customer diversity, and the VM elasticity feature. Nonetheless, NEP is a leading edge platform whose number of sites is about two orders of magnitudes larger than a typical cloud provider, with large-scale adoption by a wide spectrum of commercial applications. Our measurements already suggest striking differences between NEP and cloud platforms. Also, as commercial edge computing has recently made its debut, our results provide an important "baseline" for studying how it evolves in the future. Note that this limitation is shared by other studies of emerging technologies such as 5G [73, 97].

## 7 RELATED WORK

**Commercial edge/cloud platforms** While the concept of "pushing computations and services closer to users" is generally accepted by edge researchers and practitioners, it still remains an open problem on how to bring the edges to reality. In this work, we focus on a public edge platform NEP, which is regarded as an extension of traditional clouds but more diversely geo-distributed. Other major cloud providers are also building their multi-tenant edge platforms, e.g., AWS Local Zones [11] and Azure Edge Zone [13]. However, those platforms are at a very early stage compared to NEP (Table 1), and there are no comprehensive measurements on them yet. Major data providers like Facebook [33, 84, 85] and Google [98] have built edge infrastructure (CDN, PoP, etc) to deliver their contents to end users more efficiently. While content delivering is one killer use case in edge, NEP is built beyond the need for that but as a more general-purpose, multi-tenant computing platform like existing cloud providers.

**Measurements of edge/cloud platforms** (1) At network performance aspect, the wide-area network (WAN) performance has been extensively studied from the viewpoint of cloud providers, including the network latency [47, 89], throughput [40, 59], and resource demand volatility [57]. A few recent studies [55, 71] specifically target geo-distributed datacenters but are still at the cloud level. Partly inspired by those work, we are the first to quantify the network and application performance of a real edge platform that has much denser DC deployment than traditional cloud platforms (shown in Table 1). (2) At workloads aspect, we are not aware of any prior work characterizing the workloads on edge platforms. Some work [41, 79] analyze the first-party, container-based workloads on cloud platforms, which are orthogonal to ours that targets multi-tenant, VM-based workloads of NEP. The most related work is performed on Azure [38] cloud, which is directly compared in this work. As a key observation, we find that the edge workloads are indeed different from cloud. [37] also performs large-scale measurements on the network performance among end users (8,000 RIPE Atlas probes) and datacenters (189 in total from many cloud providers). Their study on global cloud platforms is orthogonal to ours on a much denser, nationwide edge platform.

**Edge systems and applications** have been built to bridge the gap between low-end devices and far-away clouds. The key use cases include smart homes and cities [34, 90, 93], autonomous driving [61, 62, 64], video analytics for smartphones [63, 78], surveillance cameras [31, 99, 100], and drones [94]. Those scattered thoughts can be regarded as motivations to build NEP that relieves edge developers from deploying and maintaining the edge hardware, just as the way cloud computing helps developers in the last twenty years. Beyond specific use cases, there have been system-level optimizations towards edge performance and security [68, 69, 76, 80]. Those techniques are orthogonal to NEP.

## 8 CONCLUSIONS

We have performed the first comprehensive measurement on a commercial, multi-tenant edge platform. Our study quantitatively answers two key questions: what is the edge performance perceived by end users and what are the edge workloads experienced by the edge operator. Our findings reveal critical differences between cloud and edge platforms; they also lead to insightful implications for designing future edge platforms and edge-based applications.

## ACKNOWLEDGMENTS

Mengwei Xu was supported by National Key R&D Program of China under grant number 2020YFB1805500, the Fundamental Research Funds for the Central Universities, and National Natural Science Foundation of China under grant number 61922017. Xuanzhe Liu was supported in part by Alibaba University Joint Research Program. We hereby give special thanks to Alibaba Group for their contribution to this paper. We also thank our shepherd, Aaron Schulman, and the anonymous IMC reviewers for their useful suggestions. Shangguang Wang is the corresponding author of this work.

# REFERENCES


[1] Game battle tanks. http://btanks.sourceforge.net/blog/, 2010.
[2] Game pingus. https://pingus.seul.org/, 2015.
[3] Alibaba cluster trace program. https://github.com/alibaba/clusterdata, 2018.
[4] Scaling kubernetes to 2,500 nodes. https://openai.com/blog/scaling-kubernetes-to-2500-nodes/, 2018.
[5] U.s. video 360 report 2018. https://www.nielsen.com/us/en/insights/report/2018/video-360-2018-report/#, 2018.
[6] 3gpp org. 2019. https://www.3gpp.org/release-15, 2019.
[7] C-v2x use cases methodology, examples and service level requirements. https://5gaa.org/wp-content/uploads/2019/07/5GAA$_1$91906$_W$P$_C$V2X$_U$Cs$_v$1-3-1.pdf, 2019.
[8] Cloud ar/vr whitepaper. https://www.gsma.com/futurenetworks/wiki/cloud-ar-vr-whitepaper/, 2019.
[9] Game flare. https://flarerpg.org/, 2019.
[10] Alibaba cloud elastic compute service. https://www.alibabacloud.com/product/ecs, 2020.
[11] Aws local zones. https://aws.amazon.com/about-aws/global-infrastructure/localzones/, 2020.
[12] Aws wavelength. https://aws.amazon.com/wavelength/, 2020.
[13] Azure edge zone. https://docs.microsoft.com/en-us/azure/networking/edge-zones-overview, 2020.
[14] Easyrtmp-android. https://github.com/tsingsee/EasyRTMP-Android, 2020.
[15] Extending the boundaries of the cloud with edge computing. https://www.alibabacloud.com/blog/extending-the-boundaries-of-the-cloud-with-edge-computing$_5$94214, 2020.
[16] Ffmpeg. https://ffmpeg.org/, 2020.
[17] ffplay documentation. https://ffmpeg.org/ffplay.html, 2020.
[18] Kubernetes (k8s). https://kubernetes.io/, 2020.
[19] Mplayer. http://www.mplayerhq.hu/design7/news.html, 2020.
[20] nginx. https://nginx.org/en/, 2020.
[21] Powered by sa: 5g mecbased cloud game innovation practice. https://www.gsma.com/futurenetworks/wp-content/uploads/2020/03/Powered-by-SA-5G-MEC-Based-Cloud-Game-Innovation-Practice-.pdf, 2020.
[22] Serverless computing and applications. .https://aws.amazon.com/serverless/, 2020.
[23] Ui/application exerciser monkey. https://developer.android.com/studio/test/monkey, 2020.
[24] User equipment (ue) radio access capabilities. https://www.3gpp.org/ftp/specs/archive/38$_s$eries/38.306/, 2020.
[25] Istemi Ekin Akkus, Ruichuan Chen, Ivica Rimac, Manuel Stein, Klaus Satzke, Andre Beck, Paarijaat Aditya, and Volker Hilt. {SAND}: Towards high-performance serverless computing. In 2018 {Usenix} Annual Technical Conference ({USENIX}{ATC} 18), pages 923–935, 2018.
[26] Sherif Akoush, Ripduman Sohan, Andrew Rice, Andrew W Moore, and Andy Hopper. Predicting the performance of virtual machine migration. In 2010 IEEE international symposium on modeling, analysis and simulation of computer and telecommunication systems, pages 37–46, 2010.
[27] Ghufran Baig, Jian He, Mubashir Adnan Qureshi, Lili Qiu, Guohai Chen, Peng Chen, and Yinliang Hu. Jigsaw: Robust live 4k video streaming. In The 25th Annual International Conference on Mobile Computing and Networking, pages 1–16, 2019.
[28] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. In Noise reduction in speech processing, pages 1–4. Springer, 2009.
[29] David Breitgand, Gilad Kutiel, and Danny Raz. Cost-aware live migration of services in the cloud. SYSTOR, 10:1815695–1815709, 2010.
[30] Rodrigo N Calheiros, Enayat Masoumi, Rajiv Ranjan, and Rajkumar Buyya. Workload prediction using arima model and its impact on cloud applications' qos. IEEE Transactions on Cloud Computing, 3(4):449–458, 2014.
[31] Christopher Canel, Thomas Kim, Giulio Zhou, Conglong Li, Hyeontaek Lim, David G. Andersen, Michael Kaminsky, and Subramanya R. Dulloor. Scaling video analytics on constrained edge nodes. In Proceedings of the 2nd SysML Conference, 2019.
[32] Chris Chatfield. The holt-winters forecasting procedure. Journal of the Royal Statistical Society: Series C (Applied Statistics), 27(3):264–279, 1978.
[33] David Chou, Tianyin Xu, Kaushik Veeraraghavan, Andrew Newell, Sonia Margulis, Lin Xiao, Pol Mauri Ruiz, Justin Meza, Kiryong Ha, Shruti Padmanabha, et al. Taiji: managing global user traffic for large-scale internet services at the edge. In Proceedings of the 27th ACM Symposium on Operating Systems Principles, pages 430–446, 2019.
[34] Franco Cicirelli, Antonio Guerrieri, Giandomenico Spezzano, and Andrea Vinci. An edge-based platform for dynamic smart city applications. Future Generation Computer Systems, 76:106–118, 2017.
[35] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. Live migration of virtual machines. In Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2, pages 273–286, 2005.
[36] Mark Claypool and Kajal Claypool. Latency and player actions in online games. Communications of the ACM, 49(11):40–45, 2006.
[37] Lorenzo Corneo, Maximilian Eder, Nitinder Mohan, Aleksandr Zavodovski, and Suzan BayhanZ. Surrounded by the clouds. In The Web Conference, 2021.
[38] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In Proceedings of the 26th Symposium on Operating Systems Principles, pages 153–167, 2017.
[39] Christina Delimitrou and Christos Kozyrakis. Quasar: resource-efficient and qos-aware cluster management. In Rajeev Balasubramonian, Al Davis, and Sarita V. Adve, editors, Architectural Support for Programming Languages and Operating Systems, ASPLOS '14, Salt Lake City, UT, USA, March 1-5, 2014, pages 127–144. ACM, 2014.
[40] Haotian Deng, Chunyi Peng, Ans Fida, Jiayi Meng, and Y Charlie Hu. Mobility support in cellular networks: A measurement study on its configurations and implications. In Proceedings of the Internet Measurement Conference 2018, pages 147–160, 2018.
[41] Sheng Di, Derrick Kondo, and Walfredo Cirne. Characterization and comparison of cloud versus grid workloads. In 2012 IEEE International Conference on Cluster Computing, pages 230–238, 2012.
[42] John Dilley, Bruce Maggs, Jay Parikh, Harald Prokop, Ramesh Sitaraman, and Bill Weihl. Globally distributed content delivery. IEEE Internet Computing, 6(5):50–58, 2002.
[43] Dong Du, Tianyi Yu, Yubin Xia, Binyu Zang, Guanglu Yan, Chenggang Qin, Qixuan Wu, and Haibo Chen. Catalyzer: Sub-millisecond startup for serverless computing with initialization-less booting. In Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems, pages 467–481, 2020.
[44] Rohan Gandhi, Hongqiang Harry Liu, Y Charlie Hu, Guohan Lu, Jitendra Padhye, Lihua Yuan, and Ming Zhang. Duet: Cloud scale load balancing with hardware and software. ACM SIGCOMM Computer Communication Review, 44(4):27–38, 2014.
[45] Zhenhuan Gong, Xiaohui Gu, and John Wilkes. Press: Predictive elastic resource scaling for cloud systems. In 2010 International Conference on Network and Service Management, pages 9–16, 2010.
[46] Ori Hadary, Luke Marshall, Ishai Menache, Abhisek Pan, Esaias E Greeff, David Dion, Star Dorminey, Shailesh Joshi, Yang Chen, Mark Russinovich, et al. Protean:{VM} allocation service at scale. In 14th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 20), pages 845–861, 2020.
[47] Osama Haq, Mamoon Raja, and Fahad R Dogar. Measuring and improving the reliability of wide-area cloud paths. In Proceedings of the 26th International Conference on World Wide Web, pages 253–262, 2017.
[48] Antony S. Higginson, Mihaela Dediu, Octavian Arsene, Norman W. Paton, and Suzanne M. Embury. Database workload capacity planning using time series analysis and machine learning. In David Maier, Rachel Pottinger, AnHai Doan, Wang-Chiew Tan, Abdussalam Alawini, and Hung Q. Ngo, editors, Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020, pages 769–783.
[49] Michael R Hines, Umesh Deshpande, and Kartik Gopalan. Post-copy live migration of virtual machines. ACM SIGOPS operating systems review, 43(3):14–26, 2009.
[50] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.
[51] Yun Chao Hu, Milan Patel, Dario Sabella, Nurit Sprecher, and Valerie Young. Mobile edge computing—a key technology towards 5g. ETSI white paper, 11(11):1–16, 2015.
[52] Zi Hu, Liang Zhu, Calvin Ardi, Ethan Katz-Bassett, Harsha V Madhyastha, John Heidemann, and Minlan Yu. The need for end-to-end evaluation of cloud availability. In International Conference on Passive and Active Network Measurement, pages 119–130. Springer, 2014.
[53] Chun-Ying Huang, Kuan-Ta Chen, De-Yu Chen, Hwai-Jung Hsu, and Cheng-Hsin Hsu. Gaminganywhere: The first open source cloud gaming system. ACM Trans. Multim. Comput. Commun. Appl., 10(1s):10:1–10:25, 2014.
[54] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In Proceedings of the 2014 ACM conference on SIGCOMM, pages 187–198, 2014.
[55] Yuchen Jin, Sundararajan Renganathan, Ganesh Ananthanarayanan, Junchen Jiang, Venkata N Padmanabhan, Manuel Schroder, Matt Calder, and Arvind Krishnamurthy. Zooming in on wide-area latencies to a global cloud provider. In Proceedings of the ACM Special Interest Group on Data Communication, pages 104–116. 2019.

[56] Eric Jonas, Johann Schleier-Smith, Vikram Sreekanti, Chia-Che Tsai, Anurag Khandelwal, Qifan Pu, Vaishaal Shankar, Joao Carreira, Karl Krauth, Neeraja Yadwadkar, et al. Cloud programming simplified: A berkeley view on serverless computing. arXiv preprint arXiv:1902.03383, 2019.

[57] Cinar Kilcioglu, Justin M Rao, Aadharsh Kannan, and R Preston McAfee. Usage patterns and the economics of the public cloud. In Proceedings of the 26th International Conference on World Wide Web, pages 83–91, 2017.

[58] Kyungmin Lee, David Chu, Eduardo Cuervo, Johannes Kopf, Yury Degtyarev, Sergey Grizan, Alec Wolman, and Jason Flinn. Outatime: Using speculation to enable low-latency continuous interaction for mobile cloud gaming. In Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services, pages 151–165, 2015.

[59] Fangfan Li, Arian Akhavan Niaki, David Choffnes, Phillipa Gill, and Alan Mislove. A large-scale analysis of deployed traffic differentiation practices. In Proceedings of the ACM Special Interest Group on Data Communication, pages 130–144. 2019.

[60] Guanfeng Liang and Ben Liang. Effect of delay and buffering on jitter-free streaming over random vbr channels. IEEE transactions on multimedia, 10(6):1128–1141, 2008.

[61] Liangkai Liu, Baofu Wu, and Weisong Shi. A comparison of communication mechanisms in vehicular edge computing. In 3rd {USENIX} Workshop on Hot Topics in Edge Computing (HotEdge 20), 2020.

[62] Liangkai Liu, Yongtao Yao, Ruijun Wang, Baofu Wu, and Weisong Shi. Equinox: A road-side edge computing experimental platform for cavs. In 2020 International Conference on Connected and Autonomous Driving (MetroCAD), pages 41–42, 2020.

[63] Luyang Liu, Hongyu Li, and Marco Gruteser. Edge assisted real-time object detection for mobile augmented reality. In The 25th Annual International Conference on Mobile Computing and Networking, pages 1–16, 2019.

[64] Shaoshan Liu, Liangkai Liu, Jie Tang, Bo Yu, Yifan Wang, and Weisong Shi. Edge computing for autonomous driving: Opportunities and challenges. Proceedings of the IEEE, 107(8):1697–1716, 2019.

[65] Ali José Mashtizadeh, Min Cai, Gabriel Tarasuk-Levin, Ricardo Koller, Tal Garfinkel, and Sreekanth Setty. Xvmotion: Unified virtual machine migration over long distance. In 2014 {USENIX} Annual Technical Conference ({USENIX}{ATC} 14), pages 97–108, 2014.

[66] Matthew Mathis, Jeffrey Semke, Jamshid Mahdavi, and Teunis Ott. The macroscopic behavior of the tcp congestion avoidance algorithm. 27(3):67–82, 1997.

[67] David Meisner, Brian T Gold, and Thomas F Wenisch. The powernap server architecture. ACM Transactions on Computer Systems (TOCS), 29(1):1–24, 2011.

[68] Hongyu Miao, Myeongjae Jeon, Gennady Pekhimenko, Kathryn S McKinley, and Felix Xiaozhu Lin. Streambox-hbm: Stream analytics on high bandwidth hybrid memory. In Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, pages 167–181, 2019.

[69] Hongyu Miao, Heejin Park, Myeongjae Jeon, Gennady Pekhimenko, Kathryn S McKinley, and Felix Xiaozhu Lin. Streambox: Modern stream processing on a multicore machine. In 2017 {USENIX} Annual Technical Conference ({USENIX}{ATC} 17), pages 617–629, 2017.

[70] Mayank Mishra, Anwesha Das, Purushottam Kulkarni, and Anirudha Sahoo. Dynamic resource management using virtual machine migrations. IEEE Communications Magazine, 50(9):34–40, 2012.

[71] Nitinder Mohan, Lorenzo Corneo, Aleksandr Zavodovski, Suzan Bayhan, Walter Wong, and Jussi Kangasharju. Pruning edge research with latency shears. In Proceedings of the 19th ACM Workshop on Hot Topics in Networks, pages 182–189, 2020.

[72] Irakli Nadareishvili, Ronnie Mitra, Matt McLarty, and Mike Amundsen. Microservice architecture: aligning principles, practices, and culture. 2016.

[73] Arvind Narayanan, Eman Ramadan, Jason Carpenter, Qingxu Liu, Yu Liu, Feng Qian, and Zhi-Li Zhang. A first look at commercial 5g performance on smartphones. In Proceedings of The Web Conference 2020, pages 894–905, 2020.

[74] Edward Oakes, Leon Yang, Dennis Zhou, Kevin Houck, Tyler Harter, Andrea Arpaci-Dusseau, and Remzi Arpaci-Dusseau. {SOCK}: Rapid task provisioning with serverless-optimized containers. In 2018 {USENIX} Annual Technical Conference ({USENIX}{ATC} 18), pages 57–70, 2018.

[75] Opeyemi Osanaiye, Shuo Chen, Zheng Yan, Rongxing Lu, Kim-Kwang Raymond Choo, and Mqhele Dlodlo. From cloud to fog computing: A review and a conceptual live vm migration framework. IEEE Access, 5:8284–8300, 2017.

[76] Heejin Park, Shuang Zhai, Long Lu, and Felix Xiaozhu Lin. Streambox-tz: secure stream analytics at the edge with trustzone. In 2019 {USENIX} Annual Technical Conference ({USENIX}{ATC} 19), pages 537–554, 2019.

[77] Parveen Patel, Deepak Bansal, Lihua Yuan, Ashwin Murthy, Albert Greenberg, David A Maltz, Randy Kern, Hemant Kumar, Marios Zikos, Hongyu Wu, et al. Ananta: Cloud scale load balancing. ACM SIGCOMM Computer Communication Review, 43(4):207–218, 2013.

[78] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen. Deepdecision: A mobile deep learning framework for edge video analytics. In IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, pages 1421–1429, 2018.

[79] Charles Reiss, Alexey Tumanov, Gregory R Ganger, Randy H Katz, and Michael A Kozuch. Heterogeneity and dynamicity of clouds at scale: Google trace analysis. In Proceedings of the Third ACM Symposium on Cloud Computing, pages 1–13, 2012.

[80] Yuxin Ren, Guyue Liu, Vlad Nitu, Wenyuan Shao, Riley Kennedy, Gabriel Parmer, Timothy Wood, and Alain Tchana. Fine-grained isolation for scalable, dynamic, multi-tenant edge clouds. In 2020 {USENIX} Annual Technical Conference ({USENIX}{ATC} 20), pages 927–942, 2020.

[81] Tiago Gama Rodrigues, Katsuya Suto, Hiroki Nishiyama, and Nei Kato. Hybrid method for minimizing service delay in edge cloud computing through vm migration and transmission power control. IEEE Transactions on Computers, 66(5):810–819, 2016.

[82] Mahadev Satyanarayanan, Paramvir Bahl, Ramón Caceres, and Nigel Davies. The case for vm-based cloudlets in mobile computing. IEEE pervasive Computing, 8(4):14–23, 2009.

[83] Jörg Schad, Jens Dittrich, and Jorge-Arnulfo Quiané-Ruiz. Runtime measurements in the cloud: observing, analyzing, and reducing variance. Proceedings of the VLDB Endowment, 3(1-2):460–471, 2010.

[84] Brandon Schlinker, Ítalo S. Cunha, Yi-Ching Chiu, Srikanth Sundaresan, and Ethan Katz-Bassett. Internet performance from facebook's edge. In Proceedings of the Internet Measurement Conference, IMC 2019, Amsterdam, The Netherlands, October 21-23, 2019, pages 179–194. ACM, 2019.

[85] Brandon Schlinker, Hyojeong Kim, Timothy Cui, Ethan Katz-Bassett, Harsha V Madhyastha, Italo Cunha, James Quinn, Saif Hasan, Petr Lapukhov, and Hongyi Zeng. Engineering egress with edge fabric: Steering oceans of content to the world. In Proceedings of the Conference of the ACM Special Interest Group on Data Communication, pages 418–431, 2017.

[86] Mohammad Shahrad, Rodrigo Fonseca, Iñigo Goiri, Gohar Chaudhry, Paul Batum, Jason Cooke, Eduardo Laureano, Colby Tresness, Mark Russinovich, and Ricardo Bianchini. Serverless in the wild: Characterizing and optimizing the serverless workload at a large cloud provider. In Ada Gavrilovska and Erez Zadok, editors, 2020 USENIX Annual Technical Conference, USENIX ATC 2020, July 15-17, 2020, pages 205–218. USENIX Association, 2020.

[87] Yizhou Shan, Yutong Huang, Yilun Chen, and Yiying Zhang. Legoos: A disseminated, distributed os for hardware resource disaggregation. In 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI), pages 69–87, 2018.

[88] Siqi Shen, Vincent van Beek, and Alexandru Iosup. Statistical characterization of business-critical workloads hosted in cloud datacenters. In 2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pages 465–474, 2015.

[89] Neil Spring, Ratul Mahajan, and Thomas Anderson. The causes of path inflation. In Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, pages 113–124, 2003.

[90] Tarik Taleb, Sunny Dutta, Adlen Ksentini, Muddesar Iqbal, and Hannu Flinck. Mobile edge computing potential in making cities smarter. IEEE Communications Magazine, 55(3):38–43, 2017.

[91] Chunqiang Tang, Kenny Yu, Kaushik Veeraraghavan, Jonathan Kaldor, Scott Michelson, Thawan Kooburat, Aravind Anbudurai, Matthew Clark, Kabir Gogia, Long Cheng, et al. Twine: A unified cluster management system for shared infrastructure. In 14th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 20), pages 787–803, 2020.

[92] Muhammad Tirmazi, Adam Barker, Nan Deng, Md E Haque, Zhijing Gene Qin, Steven Hand, Mor Harchol-Balter, and John Wilkes. Borg: the next generation. In Proceedings of the Fifteenth European Conference on Computer Systems, pages 1–14, 2020.

[93] Rahmadi Trimananda, Ali Younis, Bojun Wang, Bin Xu, Brian Demsky, and Guoqing Xu. Vigilia: Securing smart home edge computing. In 2018 IEEE/ACM Symposium on Edge Computing (SEC), pages 74–89, 2018.

[94] Junjue Wang, Ziqiang Feng, Zhuo Chen, Shilpa George, Mihir Bala, Padmanabhan Pillai, Shao-Wen Yang, and Mahadev Satyanarayanan. Bandwidth-efficient live video analytics for drones via edge computing. In 2018 IEEE/ACM Symposium on Edge Computing, SEC 2018, Seattle, WA, USA, October 25-27, 2018, pages 159–173, 2018.

[95] Liang Wang, Mengyuan Li, Yinqian Zhang, Thomas Ristenpart, and Michael Swift. Peeking behind the curtains of serverless platforms. In 2018 {USENIX} Annual Technical Conference ({USENIX}{ATC} 18), pages 133–146, 2018.

[96] Xiaozhe Wang, Kate A. Smith, and Rob J. Hyndman. Characteristic-based clustering for time series data. Data Min. Knowl. Discov., 13(3):335–364, 2006.

[97] Dongzhu Xu, Anfu Zhou, Xinyu Zhang, Guixian Wang, Xi Liu, Congkai An, Yiming Shi, Liang Liu, and Huadong Ma. Understanding operational 5g: A first measurement study on its coverage, performance and energy consumption. In Henning Schulzrinne and Vishal Misra, editors, SIGCOMM '20: Proceedings

of the 2020 Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication, Virtual Event, USA, August 10-14, 2020, pages 479–494. ACM, 2020.

[98] Kok-Kiong Yap, Murtaza Motiwala, Jeremy Rahe, Steve Padgett, Matthew Holliman, Gary Baldus, Marcus Hines, Taeeun Kim, Ashok Narayanan, Ankur Jain, et al. Taking the edge off with espresso: Scale, reliability and programmability for global internet peering. In Proceedings of the Conference of the ACM Special Interest Group on Data Communication, pages 432–445, 2017.

[99] S. Yi, Z. Hao, Q. Zhang, Q. Zhang, W. Shi, and Q. Li. Lavea: Latency-aware video analytics on edge computing platform. In 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), pages 2573–2574, 2017.

[100] Tan Zhang, Aakanksha Chowdhery, Paramvir (Victor) Bahl, Kyle Jamieson, and Suman Banerjee. The design and implementation of a wireless video surveillance system. In Proceedings of the 21st Annual International Conference on Mobile Computing and Networking, MobiCom '15, pages 426–438, 2015.

# A PRICING MODEL COMPARISON

Table 7 shows a detailed comparison among the billing models of NEP and 2 popular cloud platforms (Alibaba Cloud and Huawei Cloud), referred to vCloud-1 and vCloud-2 in §4.5, in detail. We focus on monthly cost, as it's generally supported by both cloud and edge platforms.

In NEP, the network traffic of VMs located in the same site will be combined and charged together. The bandwidth charged by ENS is the 95-th percentile daily peak bandwidth of the month. In other words, NEP will first record the peak bandwidth usage per day, and then use the 4th highest one from all the daily peak usage in this month to generate the bill. As previously discussed, such a billing method is less elastic as compared to cloud platforms that charge bandwidth by its average usage per minute or even second.

| Unit: RMB | CPU | Memory | Storage (SSD) | Network | | |
|---|---|---|---|---|---|---|
| | | | | Sub-category | Method | Example |
| **Alibaba Cloud** | 2CPU + 4GB: 187/month<br>2CPU + 8GB: 240/month;<br>2CPU + 16GB: 318/month;<br>etc.. | | 1/GB/month | Pre-reserved (fixed) | 1Mbps: 23/Mbps/month;<br>2Mbps: 46/Mbps/month;<br>3Mbps: 71/Mbps/month;<br>4Mbps: 96/Mbps/month;<br>5Mbps: 125/Mbps/month;<br>>5Mbps: 80/Mbps/month. | 2Mbps: 46/month;<br>7Mbps: 125 + (7 - 5) * 80 = 285/month. |
| | | | | On-demand, by bandwidth | 1∼5Mbps: 0.063/Mbps/hour;<br>>5Mbps: 0.248/Mbps/hour. | 2Mbps: (24 * 30) * (2 * 0.063) = 90.72/month;<br>7Mbps: (24 * 30) * [(2 * 0.063) + (7 - 5) * 0.248] = 447.84/month. |
| | | | | On-demand, by quantity | 0.8/GB | 1GB: 1 * 0.8 = 0.8 |
| **Huawei Cloud** | 1CPU + 1GB: 32.2/month;<br>1CPU + 2GB: 72.2/month;<br>2CPU + 4GB: 152.2/month;<br>2CPU + 8GB: 251.6/month;<br>etc.. | | 0.7/GB/month | Pre-reserved (fixed) | 1∼5Mbps: 23/Mbps/month;<br>>5Mbps: 80/Mbps/month. | 2Mbps: 46/month;<br>7Mbps: 23*5 + (7 - 5) * 80 = 275/month. |
| | | | | On-demand, by bandwidth | 1∼5Mbps: 0.063/Mbps/hour;<br>>5Mbps: 0.25/Mbps/hour. | 2Mbps: (24 * 30) * (2 * 0.063) = 90.72/month;<br>7Mbps: (24 * 30) * [(5 * 0.063) + (7 - 5) * 0.25] = 586.8/month. |
| | | | | On-demand, by quantity | 0.8/GB | 1GB: 1 * 0.8 = 0.8 |
| **NEP** | 65/CPU/month | 20/GB/month | 0.35/GB/month | Telecom or Unicom | 25–50/Mbps/month | guangzhou-telecom 2Mbps: 50*2=100/month;<br>chengdu-telecom 2Mbps: 25*2=50/month. |
| | | | | CMCC | 15–30/Mbps/month | guangzhou-cmcc 2Mbps: 30*2=60/month;<br>chengdu-cmcc 2Mbps: 15*2=30/month. |

**Table 7: A detailed comparison of the billing models of NEP and two popular cloud platforms in China. The price of edge network bandwidth varies across different cities and operators.**