electronics

MDPI

Article

Research on Terminal-Side Computing Force Network Based on Massive Terminals

Jianhua Gu [1], Jianhua Feng [1], Huiyang Xu [2,*] and Ting Zhou [2]

[1] Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China; gujh19@mails.tsinghua.edu.cn (J.G.); fengjianhua@tsinghua.edu.cn (J.F.)
[2] China Mobile Group Device Co., Ltd., Beijing 100053, China; zhouting@cmdc.chinamobile.com
* Correspondence: xuhuiyang@chinamobile.com

Abstract: With the explosive growth of the demand for computing power in the era of digital economy and the continuous enhancement of the computing power of terminals, how to provide high bandwidth, low-latency, and low-cost service by leveraging the user devices' computing, storage, and network resources has become a research hotspot. However, due to differences in magnitude, architecture, and performance, the existing technologies for cloud computing and edge computing need to overcome many challenges such as android-based devices not supporting container technologies. In this paper, a terminal-side computing force network (TSCFN) architecture is proposed, which realizes the unified computing power management of massive user devices by layered and distributed architecture with highly dynamic and domain-federated deployments. At the same time, we propose a cloud-native container resource scheduling scheme based on the Android system to enhance the scalability of TSCFN. Taking the CDN service as a use case, the experiment results show that services provided by TSCFN can reduce latency and improve resource utilization, especially in an unstable network status. Compared with traditional CDN, delay duration of HomeCDN based on TSCFN is decreased by 96% in a bad network environment.

Keywords: terminal-side computing force network; container technology; resource scheduling; resource awareness and abstraction; service orchestration

Citation: Gu, J.; Feng, J.; Xu, H.; Zhou, T. Research on Terminal-Side Computing Force Network Based on Massive Terminals. *Electronics* **2022**, *11*, 2108. https://doi.org/10.3390/electronics11132108

Academic Editors: Jorge Ortiz, Zhiwei Zhao and Guohao Lan

Received: 14 June 2022
Accepted: 4 July 2022
Published: 5 July 2022

1. Introduction

Computing Power is fast becoming a key instrument in digital economy. Only with stronger computing power, can smart industry, smart city, smart transportation, smart medical, and other application scenarios continue to change from ideal to reality, so as to achieve continuous expansion of industrial scale and continuous economic and social progress. With the continuous penetration of digitalization into thousands of industries, the demand for computing power is also surging. In 2020, total computing power reached 135EFLOPS in China. In order to meet the growing demand for computing power, Industry must focus on building large-scale and ultra-large-scale data centers. However, the centralized cloud computing model has obvious shortcomings in terms of delay, bandwidth, and flexibility. The edge computing model is closer to the data source. The delay-sensitive tasks are deployed to the edge nodes, without uploading to cloud data centers through long links. It realizes the rapid response of key tasks and effectively relieves the computing pressure of cloud data centers. With the gradual increase in the computing power of personal computers, many distributed computing projects have used the idle computing power of volunteer computers around the world, such as the SETI@Home and Folding@home projects. However, they all focus on homogeneous terminal resources, and do not solve the resource utilization problem of heterogeneous massive terminals. At the same time, the number of intelligent terminal devices is growing exponentially. According to IDC's forecast, the number of global terminal devices will reach 100 billion in 2025. The rapid development of large-scale integrated circuits and semiconductor technologies has led to

a rapid increase in the computing power of terminal equipment. Taking the Apple A15 chip as an example, it is equipped with a faster neural network engine and image signal processor, and the AI computing power has reached 15.8TOPS. According to a Gartner report, in 2022, 74% of the world's data will be analyzed and calculated on the terminal side. With the continuous enrichment of terminal-side computing resources, whether the massive terminal computing resources can be utilized more effectively has become a new research field. Table 1 shows the comparison of three kinds of computing models.

Table 1. Comparison of cloud, edge, and device computing.

Issues	Cloud	Edge	Device
Computing Mode	Centralized	Distributed	Hyperdistributed
Distribution Characteristics	Network Center	Network Edge	Service Occurrence Site
Hardware resources	Sufficient and belongs to the company	Relatively sufficient and belongs to the company	Fragmented and limited and belongs to the user
Delay	High	Low	Very Low
Location Awareness	No	Regional Awareness	Precise Awareness
Bandwidth Requirements	Sensitive	Insensitive	Insensitive
Service type	batch large tasks	small tasks	lightweight small tasks
Privacy level	low	low	high

Existing work tends to achieve centralized computing power scheduling through network capabilities and promote the ubiquity of computing power. For example, China Mobile, China Telecom, China Unicom have all released white papers on computing power networks. However, current solutions basically do not consider how to use the computing power of massive terminals, but only use terminals as a carrier of business, rather than as a provider of business. At the same time, the traditional cloud-native technology system faces challenges when managing terminal devices because they are massive and lightweight, with different architecture and high fragmentation, compared with the data center.

In this paper, we leverage customized cloud native technology to help in the management and scheduling of dispersive computing resources. We propose a terminal-side computing force network (TSCFN) architecture that includes resource scheduling layer, resource management layer, orchestration and scheduling layer, and the capability opening layer. Through this architecture, idle terminal computing power resources can be aggregated and scheduled to support upper-level services with container technology. However, a large number of Android devices do not support container technology. We introduce a cloud-native container resource scheduling scheme based on the Android system. This scheme can make cloud management technology easier to deploy and manage on Android devices. In addition, we research the application scenarios based on TSCFN, and built a demonstration environment to verify the solution from both the function and performance aspects.

In summary, the main contributions are the following:

- We propose a TSCFN architecture to achieve unified allocation, scheduling, and management of massive lightweight computing resources, to bring computing closer to the terminal side, while pooling idle terminal computing power to better meet the user experience.
- We propose a cloud-native container resource scheduling scheme based on the Android system. This allows a large number of Android devices to join TSCFN, thereby greatly expanding the availability and scalability of the network.
- We propose typical application scenarios based on TSCFN. In addition, we experimentally verified that TSCFN-based HomeCDN service can provide better performance compared to traditional CDN.

The rest of the paper is organized as follows. Section 2 discusses the related works. Section 3 describes the overall structure of TSCFN and key technologies. Section 4 studies

potential application cases and analyzes the performance of HomeCDN based on TSCFN. Finally, Section 5 concludes this paper and gives future research directions.

2. Related Work

The computing force network is the ubiquitous computing force resource based on the ubiquitous network. It is an intelligent, flexible, and on-demand real-time invoking Network Architecture. The computing force network connects distributed computing nodes through the network. According to the real-time perception of massive computing resources and network resource status, the resources of the computing force network are reasonably coordinated. The computing tasks are uniformly scheduled, allocated, and arranged to meet new business, new applications, and requirements for computing power [1]. At present, the industry and academia are promoting the development and research of the computing force network.

In industry, the three major operators are actively involved in the related work of Computing Force Network. China Mobile leads the international standard for computing power-aware networks in ITU-T, and promotes computing-force-aware networks to become an important direction for the next ITU research period. China Mobile first posted the "White paper on computing power aware network technology" [2]. They also proposed a computing force-aware network based on distributed systems, computing, and network integration, realized joint optimal scheduling of ICT (Information and Communications Technology) systems, provided end-to-end ICT systems and SLA (Service-Level Agreement) experience guarantee, and promoted the computing force network as one of the foundations of the 6G network development in IMT-2030. Furthermore, they released the "Computing Force network White paper" in November 2021 [3] to analyze the concept, architecture, technology, and ecology of the network. China Unicom takes the lead in ITU computing power scheduling technical standards, proposing a network-centric realization of computing force network in CUBE-Net3.0 and a new type of "computing network integration" as the long-term goal of future bearer network development. The "White paper on computing network architecture and technology system" was released [4], which clarifies the key factors of China Unicom's computing network architecture design, functional model, inter-layer interface, and each functional layer. Combined with several scenarios, the application and deployment methods of computing force network are prospected. China Telecom's "Cloud-Network Integration 2030 Technical White Paper" emphasizes cloud-network integration [5], and proposes the technical architecture, three-stage development path, and goals of China Telecom's cloud-network integration, and regards computing force network as one of the key technological innovation areas of cloud-network integration. Huawei proposed CFirstN at the IETF, and ZTE actively conducted technical research and prototype verification. In the computing force network, the computing, storage, and other resources of massive terminal-side devices are an important part of its infrastructure. The research of various enterprises mainly focuses on large-scale data centers, and there is no systematic consideration of massive terminal-side resources. The TSCFN proposed in this paper is different from the traditional data center that is centralized by operators or cloud service providers. It collects and organizes the idle resources of massive user terminals, makes full use of the computing resources of massive terminals, and cooperates with traditional cloud computing and edge computing to achieve complementary advantages in different scenarios.

In academics, with the exposure of problems in cloud computing, the research on edge computing has also been very in-depth, which mainly focuses on computing offloading, service migration, content caching, and cloud-edge-terminal collaboration. In the literature [6], the author focus on three offloading decision scenarios to explore the optimal strategy for offloading scheduling and resource management. In the paper [7], the authors introduce the network organization method of the computing force network, which is oriented to business requirements, and allocates and flexibly schedules computing resources among clouds, networks, and edges on demand. Starting from the computing

force network architecture, in order to realize the unified scheduling and management of a wider range of computing resources, based on the cloud-native resource scheduling mechanism, a lightweight, multi-cluster hierarchical edge resource scheduling scheme is expounded. Based on the lightweight cloud native platform, the unified management of front-end massive edge devices for computing force networks is realized, and can be deployed on embedded platforms of various architectures. Reference [8] designed a public computing power platform architecture for the current situation that machine learning has relatively high requirements on the development environment. Through the technical design and configuration of containerization, the hardware resources of the GPU can be shared. By allocating computing resources on demand, the isolation of the development environment of each artificial intelligence development project can also be achieved. The specific methods of platform implementation, management, and maintenance are proposed. In [9], the author conducts an in-depth study on the key issues of efficient deployment of tasks in the cloud-edge joint data center environment. An efficient task deployment method of the joint cloud computing model in the edge computing environment is proposed, and the efficient deployment method of the task in the cloud-edge joint computing environment is studied. The efficient processing method of large-scale lightweight tasks in the mobile edge computing environment is innovated. In [10], the author analyzes the spatial and temporal differences of edge container cloud loads in a multi-cluster environment and the differences in experimental sensitive requirements of edge applications, and proposes a multi-cluster edge cloud framework managed by master-slave mode. Further, they research the resource scheduling problem of delay-sensitive applications in the framework, and a cross-cluster resource scheduling strategy for delay-insensitive applications is proposed to improve the resource utilization rate of the cluster. In the literature [11], the author proposes a lightweight mobile operating system virtualization architecture for the problems of low efficiency and high cost caused by the current virtualization architecture translating a large number of instructions between the hardware layer and the virtual system. By extending the Driver namespace framework on the basis of the Linux kernel namespace mechanism, multiple virtual Android systems can run simultaneously.

Traditionally, cloud computing technology was applied to the mobile Internet, so that mobile devices such as mobile phones, tablets, and stereos can obtain the required IT resources or information services in an on-demand and easily scalable manner through the mobile network. As to how to fully mine and utilize the resources of the terminal itself after the computing power sinks, as well as the problems of heterogeneous and decentralized computing power [9,11], the industry has begun to explore how to apply the cloud-based container orchestration technology to the edge. Most research still focuses on the orchestration and scheduling of MEC resources. For the orchestration and scheduling of computing power on the terminal side, it is more based on a unified technical system to promote better collaboration between the cloud and the edge, and it has not yet been involved in how to effectively integrate the computing power resources of massive terminals.

3. Terminal-Side Computing Force Network

TSCFN, integrating the computing resources of massive user terminals and devices in the community and buildings, such as phones, smart home terminals, community monitoring devices, and edge nodes, provide users with high-quality computing services on demand. It can unify, manage, and dynamically schedule terminal computing resources, and fully coordinate with edge computing and cloud computing. Although TSCFN focuses on the terminal-side, it does not mean that it is separated from the cloud-side and the edge-side. On the contrary, TSCFN is more closely integrated with the cloud-side and edge-side. A computing task can be calculated in the cloud, edge or TSCFN independently according to the different privacy or latency requirements. It may also need the cloud and edge and TSCFN cooperate to complete the calculation. In addition, the TSCFN can build everyone's private IaaS on the user side. Users can "rent out" the storage and computing resources of the device when the device is idle or after the end of its life cycle, and users

can get income from their own device resources. It will bring a new supply and service way of computing power.

3.1. Overall Structure

The overall architecture of the TSCFN is shown in Figure 1, which mainly includes the resource scheduling layer, the resource management layer, the orchestration and scheduling layer, the capability open layer, and the application layer. The resource scheduling layer unifies and manages resource instances in heterogeneous devices with different configurations and different operating systems using container virtualization technologies. TSCFN relies on the resource scheduling layer to sense the network resources, computing resources, and storage resources in the cluster. The resource scheduling layer is the basic form of orchestration and scheduling of fragmented terminal resources—it reports the resources information to the resource management layer. The resource management layer abstracts, maps, and manages the unified fragmented device resources to form the terminal-side computing resource pools and storage resource pools, which shield the underlying differences of the devices. The resource scheduling layer and the resource management layer jointly realize the unified management of device resources of TSCFN. The orchestration and scheduling layer is the brain in TSCFN. It intelligently decomposes applications and reasonably arranges and schedules computing tasks with AI algorithms according to the TSCFN load, node availability, and attributes of upper-layer applications. The capability open layer provides a unified capability set of computing capabilities, storage capabilities, and AI capabilities of TSCFN, and provides standard open APIs for the different application environments. Developers only need to package applications into containerized micro-services, which can be easily deployed, updated, and iterated on TSCFN. At the same time, TSCFN also built an application store for developers and users, which can provide better integrated computing services and enrich the ecology of TSCFN.

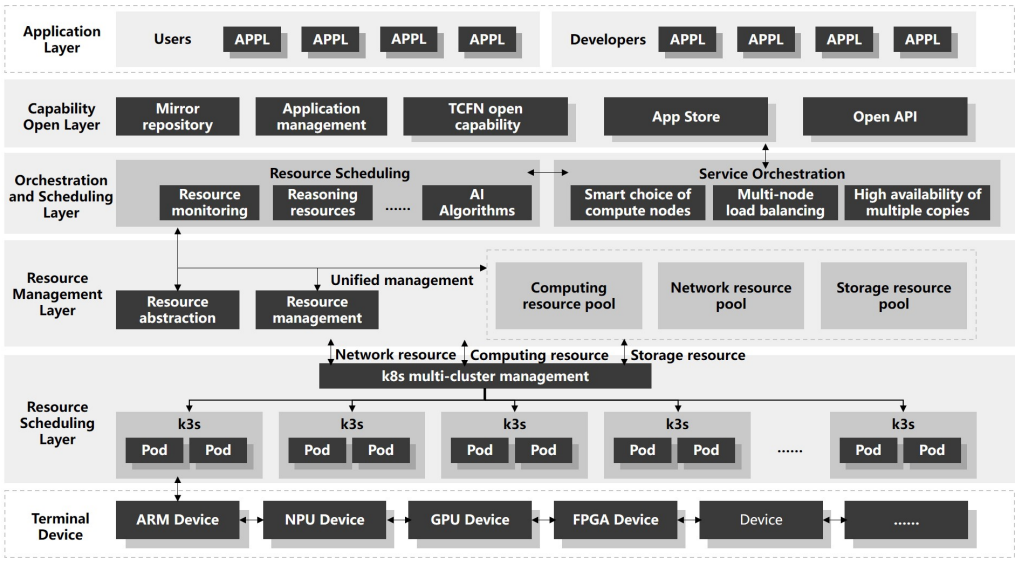

Figure 1. Overall structure.

3.2. Key Technology

TSCFN has certain similarities with cloud computing. Cloud computing aggregates a large number of hardware resources to form a unified resource pool, which can provide the computing services with the dynamic and scalable virtualized resources through the network. Similarly, TSCFN also aggregates massive amounts of individual hardware to form a unified terminal-side resource pool, which can provide the computing or storage services through scheduling and orchestration. Therefore, it has the theoretical basis for using the cloud computing technologies to build TSCFN. At present, the cloud native technologies have become a hot spot in the field of cloud computing. Among them, the

container technology is one of the most active technologies. Containers are a lightweight virtualization alternative to VMs [12] that leverage two Linux kernel features: namespace and cgroups [13]. Containers can reduce overhead compared to VMs by packaging only applications or functions and application-specific operating system dependencies [13,14]. One of the most widely used containerization technologies is Docker due to its ability to provide portability and scalability [15]. Lightweight Docker containers enable portability and real-time deployment of distributed applications. Solutions based on lightweight Docker containers also promote the realization of TSCFN [16,17]. Docker handles the packaging and distribution of applications, while the Kubernetes (K8s) is used to scale, run, and monitor applications [18]. K8s has become the mainstream tool for orchestration and scheduling in cloud-native programs, also known as a container orchestrator. It provides deployment automation, scheduling, scaling, and orchestration of containerized applications [19]. At the same time, considering there are more and more light devices, K3s has gradually become a hot topic, which tailors and optimizes the management plane and control plane of K8s.

However, since the computing resources in TSCFN come from terminal devices, which are massive, lightweight, and heterogeneous compared with dedicated servers. Therefore, the technical solutions of TSCFN have to fully optimize the differentiated characteristics of the devices, which should consult both the experience of cloud resource management and the compatibility with the mainstream cloud native technologies. The key technologies of TSCFN include perception and abstraction of heterogeneous resources, scheduling of massive lightweight resources, and orchestration of services.

3.2.1. Heterogeneous Resource Awareness and Abstraction Technologies

TSCFN is a highly heterogeneous environment, covering different hardware, software architectures, and various chips (the resources and processing capabilities of various chips are different, for example, some of the processor cores support a large cache structure, and some support a small cache structure). Some cores are sequential pipelines; some are out-of-order pipelines. Such a highly heterogeneous environment requires TSCFN to be aware of the resource requirements of different programs on different chips (due to the different attributes of computing tasks, different computing tasks have different requirements for hardware resources, such as computing-intensive tasks, accessing memory-intensive tasks or branch-intensive tasks). At the same time, the resource scheduling decisions also need to sense the load of TSCFN in real time, and comprehensively consider the migration cost between different cores. Containerized transformation of the terminal system can shield the interface details of the hardware platform and the differentiation of operating system versions. In TSCFN, the resource management layer provides abstract services for the upper layers with APIs, so that the application does not need to know the details of the device, which greatly reduces the developer's dependence on system understanding and development complexity.

3.2.2. Resource Scheduling Technologies

Considering the characteristics of devices, such as mass and lightweight, this paper adopts a converged architecture of "K8s+K3s" to achieve unified terminal-side resource scheduling management. The terminal cluster uses the lightweight K3s cloud native platform to manage the resources, which is more suitable for devices with limited resources. On the cell or edge side, the K8s cloud native platform is used to unify, schedule, and manage the multi-clusters. This converged architecture based on cell slicing can fully match the capabilities and characteristics of nodes, and realize efficient collaboration between cloud, edges, and terminals with the unified platform interfaces [20]. Now, the main operating system is Android, but K3s cannot support Android, due to the tailoring of some functions of Linux by the Android. This paper proposes a cloud-native container resource scheduling scheme based on the Android system, which ensures TSCFN can coverage mainstream terminal devices, the scheme is detailed in Section 3.3.

3.2.3. Service Orchestration Technologies

The traditional cloud computing environment has homogenized computing nodes and reliable network connections, while the computing nodes and network connections in TSCFN are highly heterogeneous. If TSCFN arranges the tasks indiscriminately, it will easily lead to a waste of resources for devices with high computing power, and the longer computing time of devices with low computing power, which makes the service worse. In centralized service orchestration, each service orchestration is relatively simple and straightforward (the main purpose of the service orchestration is obtaining more resources and completing services more efficiently). In TSCFN, computing resources are relatively limited and more fragmented, therefore, it has higher demands for service orchestration technologies. TSCFN adopts cloud-native technologies, and each service is dynamically deployed in the container. Meanwhile, more terminal-side characteristic factors are considered, such as load, delay, network consumption, and service attributes. The orchestration and scheduling layer of TSCFN in this paper comprehensively consider the interaction between the orchestration strategy and the cost, and continuously iterate the overall service orchestration strategy that meets the Nash equilibrium conditions, for example, when there are multiple applications running at the same time, the security and real-time interactive services should be defined as high priority, but the storage and audio-visual services should be defined as low priority.

3.3. A Cloud-Native Container Resource Scheduling Scheme Based on Android System

According to the architecture design principles of K8s and K3s, when the system is running, it needs to isolate the host resources, virtualize the network, and arrange the running containers to realize the management and scheduling of the underlying computing resources. The K8s operating architecture is shown in Figure 2. However, considering the lightweight characteristics of terminal devices, the current Android system has tailored some kernels of the Linux system, including the module supporting the cloud-native network protocols, go language runtime environment, namespace and cgroup process isolation mechanisms, and the orchestration of container runtime. The cropping make it impossible to start the container management service, to realize the management of the full life cycle of the container, and the orchestration of distributed containers; this means it cannot support K8s or K3s.

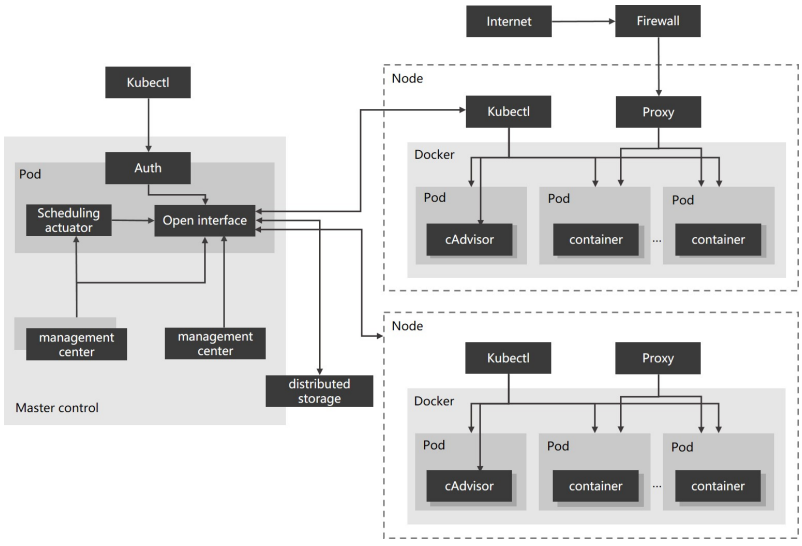

Figure 2. Kubernetes operation diagram.

For the problem of container virtualization in Android, there are some solutions and theoretical achievements, but they are not suitable for containerized management scenarios of massive terminals. The Linux Container (LXC) project team has developed

related components based on the Android library, so that Android can also implement container virtualization through LXC [21]. However, LXC does not follow the OCI Runtime specification, which is formulated by the OCI (Open Container Initiative) organization, while the mainstream container orchestration system and container management engine, such as K8s, K3s and Docker, cannot run container runtimes that do not follow the OCI Runtime specification. The Condroid project of Zhejiang University refactored the LXC tools to the ARM architecture, and proposed an Android system virtualization solution based on Linux containers. Condroid can start multiple containers on the same device and run an independent Android in each container [22,23]. However, in practice, this solution needs to modify the code of each virtual Android system, which faces great difficulties in scalability and compatibility. The Virtualization Research Laboratory of Columbia University proposed a Cells virtualization architecture [24,25]. Unlike the traditional Hypervisor virtualization architecture [26], Cells can run multiple independent virtual Android systems on one Android device, and support resource calls of the same physical device between multiple virtual Android systems. However, Cells also brings more overhead due to its own virtual architecture. The above solutions all solve the problems of container virtualization in the Android system, to some extent, but they all cannot support the current mainstream container orchestration tools, for they do not support the OCI specification or the mainstream container runtimes, etc., so they also cannot realize the containerized management of massive devices.

Considering that a large number of devices are equipped with the Android system, such as mobile phones, set-top boxes, smart speakers, etc., this paper proposes a cloud-native container resource scheduling scheme based on Android and completes the optimization of the bottom layer in native Android. The scheme ensures optimized Android support of the mainstream container orchestration tools, such as K8s and K3s. The optimized Android architecture is shown in Figure 3.

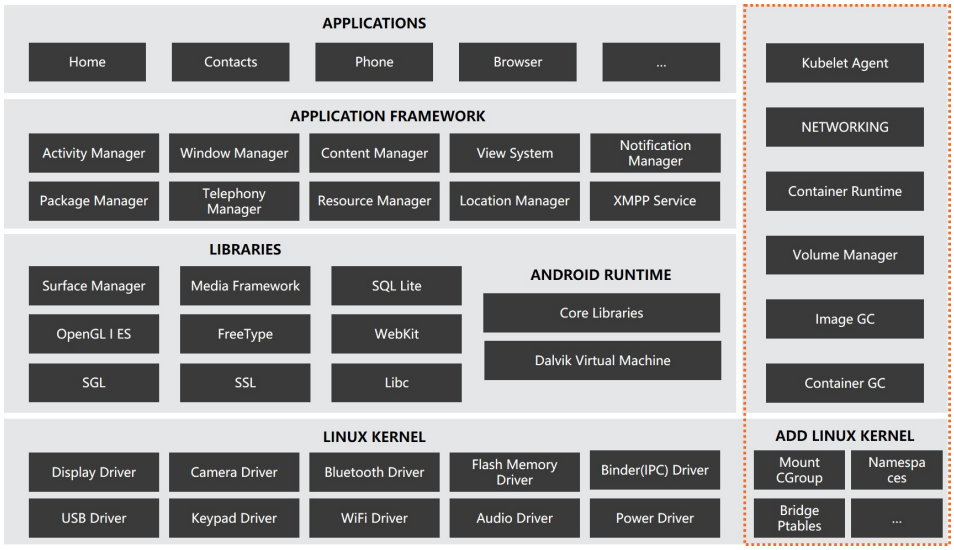

Figure 3. Android system optimization supporting container technology.

First of all, it is necessary to solve the problem of allocation and isolation of network resources and physical resources, it will create a healthy running environment for containers under Android. First, we add virtualization to the kernel module layer and add drivers for kernel config to support cloud-native network protocols. Therefore, the container network can be allocated and managed after the containers are started; it means the K3s nodes can communicate between each other. Second, we add namespaces, cgroups in kernel. We realize the encapsulation, abstraction, and isolation of processes with namespace, so that the processes in each namespace have their own global resources. We further realize the allocation and restriction of underlying physical resources, such as CPU, memory, etc., with

cgroups, so that the containers can run stably. On this basis, we modify the kernel setting parameters to realize the docking with K3s and the supporting for containerd. At last, we realize the pulling of container images, the management of the full life cycle of the container, and the scheduling of computing resources with the supporting for the commands, such as the runc command.

4. Case Studies and Performance Evaluation

As an important part of the computing power network ecosystem, TSCFN can bring users richer experiences. It can provide users with lower-cost and lower-latency bandwidth, storage, computing, and other resource capabilities, upgrading existing scenarios for individuals, families, and industries, and meet the needs of ultra-high-definition video, cloud games, and other high-concurrency, large-throughput, and large-bandwidth business requirements. This case study confirms the importance of idle computing power.

4.1. Homecdn

CDN (Content Delivery Network) is a distributed technology that distributes the content of the source site to all nodes, thereby shortening the delay for users to view content and improving the response speed of user access. It can effectively solve the problems of small network bandwidth, large number of user visits, and uneven distribution of network points. Currently, it is mainly used in video acceleration. With the increasing demand of users for high-definition video, live broadcast, etc., the CDN market is developing rapidly. The current total market size is about 300 T–500 T, which has reached CNY 31.1 billion in 2020 and is expected to reach CNY 95 billion in 2025. Based on TSCFN, idle traffic bandwidth capabilities of massive video boxes, smart speakers, and other lightweight devices can be integrated, and provide users low-cost CDN to enhance the service experience, including OTA upgrades, long and short videos, live broadcasts, game acceleration, etc. Figure 4 presents an overview of Home CDN services based on TSCFN.

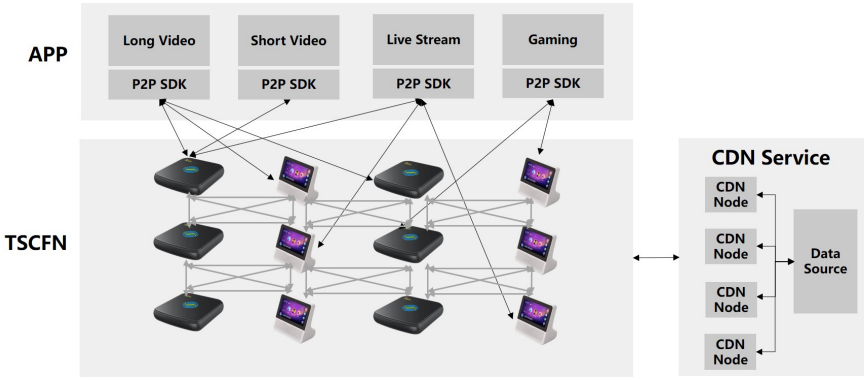

Figure 4. Integrated scheduling of massive terminal-side bandwidth resources—HomeCDN.

4.2. Personalcloud

With the explosive growth of digital content, there is a huge demand for content storage by individuals. According to MarketsandMarkets forecast, the global personal cloud scale was USD 23.7 billion in 2019, and the market size is expected to reach USD 73.4 billion in 2024, with a compound annual growth rate of 25.4%. At the same time, with the increasing demands of privacy and security, the centralized cloud storage model has many shortages such as slow transmission speed, file restrictions, expensive storage price, and so on. The user-centric distributed storage model has become a new trend. Based on TSCFN, idle storage capabilities of smartphones, smart TVs, smart speakers, and other devices can be integrated to provide users low-cost storage for high privacy distributed storage service. Figure 5 shows an overview of personal cloud services based on TSCFN.

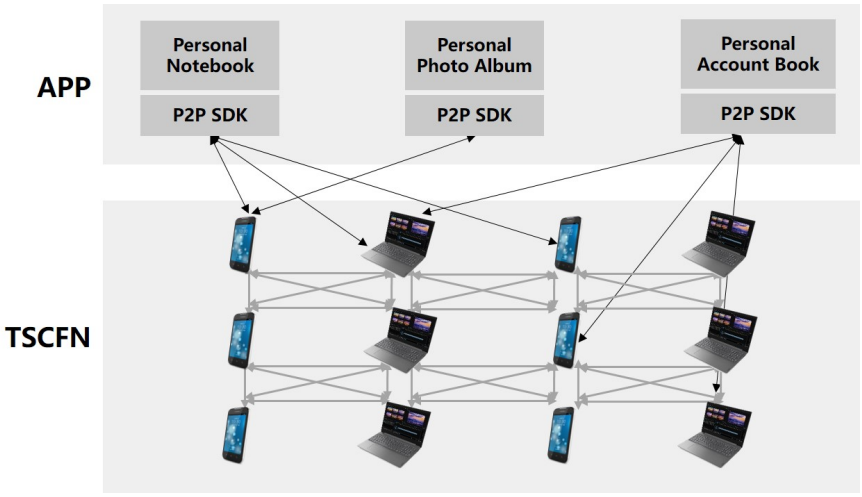

Figure 5. Integration and scheduling of massive terminal-side storage resources—PersonalCloud.

4.3. Localcomputing

With the increase of smart devices and the enhancement of computing power, the terminal will not only be a simple device for data collection and task execution, but also provide external computing power. On the one hand, as the closest computing site for data processing, faster data analysis, and local application processing can be achieved to improve data security, protect user privacy, and effectively reduce bandwidth resource consumption based on TSCFN. With the maturity of interactive technologies such as virtual reality, augmented reality, image processing, and video rendering, the boundary between virtualization and reality will gradually be broken. A kind of Fat terminal with stronger computing power can provide enhanced AI capabilities to empower lightweight devices in the home to achieve a richer experience and realize higher computing scenarios, such as Metaverse. Figure 6 illustrates the overview of local computing services based on TSCFN.

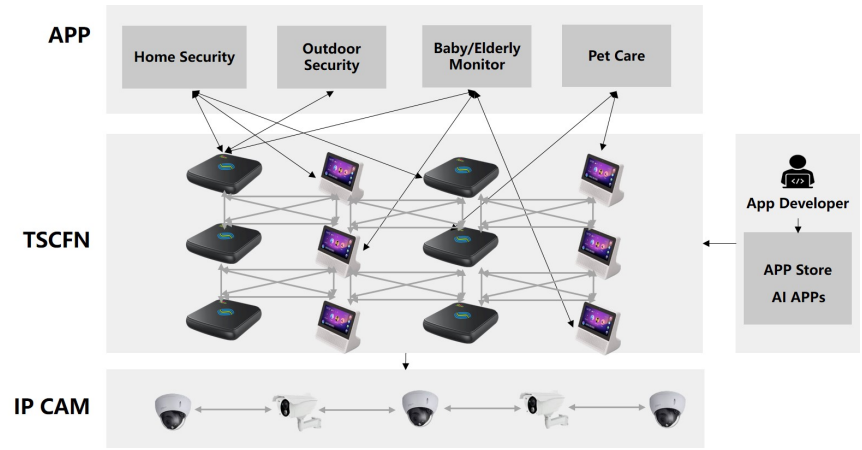

Figure 6. Integrated scheduling of field level computing resources—LocalComputing.

4.4. Experiment and Analysis

In order to verify the function and performance of the TSCFN, this section uses the HomeCDN scenario as a case to build an experimental environment. The terminals such as smart speakers and set-top boxes are selected to build TSCFN. Each device is used as the data source of the download task, and the idle bandwidth resources and storage resources on the terminal devices are integrated through this network to supply CDN service. Due to the adoption of an Android-based containerized scheduling scheme, these computing tasks can run independently using pre-allocated system resources in an isolated environment.

4.4.1. Function Test

Firstly, in order to verify the Android-based container scheduling scheme, an Android mobile phone (CPU MT7661, 4 GB + 64 GB memory) was selected to build a test environment. Firstly, we start the container in the test phone, and deploy and run the NGINX service in the container. The test results feedback that the container in the Android device can run successfully, and K3s can manage and schedule the container. Figures 7 and 8 show the operation of K3s Server and Agent.

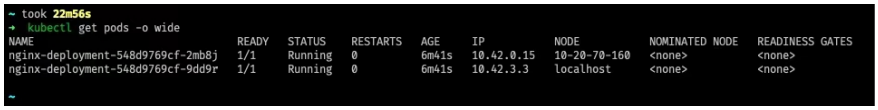

Figure 7. K3s-sever running list.

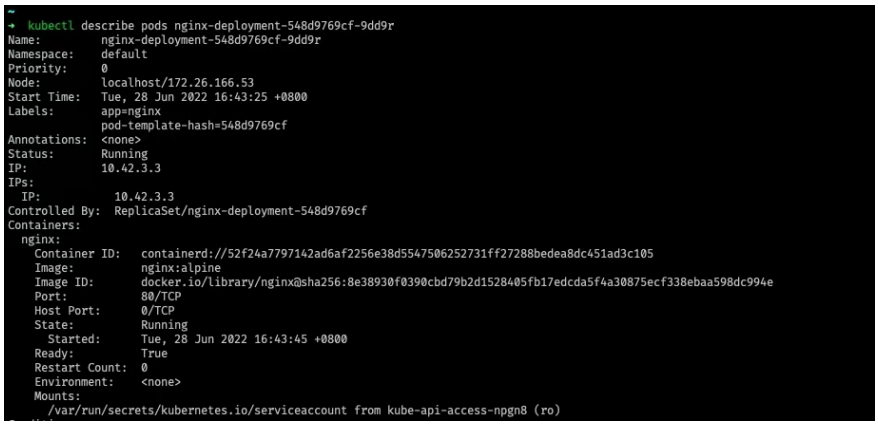

Figure 8. Running status of K3s-agent.

Although the container will bring a certain performance overhead to the terminal, the overall impact is controllable. Taking memory consumption as an example, from the test results, when the process of K3s-agent is started, its memory accounts for 5%, and the process of Containerd memory accounts for 1.5%, and the process of Runc memory accounts for 0.4%. The overall memory usage of the scheme takes little impact on the service experience.

Secondly, we take the test of the success rate of NAT Hole Punching to verify the functionality of HomeCDN based on TCSFN, while considering that the success rate is the basis for ensuring the smooth operation of the HomeCDN Service. The specific equipment list in the experimental environment is shown in Table 2.

Table 2. List of functional test equipment.

Device Type	Device Model	Amount
xiaodu intelligent screen	1A	16
Android phone	Redmi Note 8 Pro	1
leguang router	K200	1
xiaodu router	XD-INA12-2001	1

The experimental plan divides the cloud seed into 32 shares and pushes them to 16 Xiaodu 1A devices as data sources. Different types of network environments are simulated through router configuration, and Android phones are used as clients to download relevant data. We test the download success rate under different network environments. The experimental results show that the NAT hole punching requirements can be successfully implemented in most network environments to ensure the stable operation of HomeCDN services. It is feasible to form a computing power network by lightweight devices. The test results are shown in Table 3.

Table 3. Functional test results.

ID	Client NAT	Server NAT	1st Attempt	2nd Attempt	3rd Attempt
1	NAT1	NAT1	pass	pass	pass
2	NAT2	NAT1	pass	pass	pass
3	NAT3	NAT1	pass	pass	pass
4	NAT4	NAT1	pass	pass	pass
5	NAT1	NAT2	pass	pass	pass
6	NAT2	NAT2	pass	pass	pass
7	NAT3	NAT2	pass	pass	pass
8	NAT4	NAT2	pass	pass	pass
9	NAT1	NAT3	pass	pass	pass
10	NAT2	NAT3	pass	pass	pass
11	NAT3	NAT3	pass	pass	pass
12	NAT4	NAT3	pass	pass	pass
13	NAT1	NAT4	pass	fail	pass
14	NAT2	NAT4	fail	fail	fail
15	NAT3	NAT4	pass	fail	pass
16	NAT4	NAT4	pass	fail	fail

4.4.2. Performance Test

In order to verify the performance of HomeCDN, the video streaming service that requires high real-time data is selected as the test scenario. We compare HomeCDN with traditional CDN as a benchmark through two sets of comparative experiments. In the first set, we use three Xiaodu smart speakers to build a two-layer HomeCDN network, in which a 1S device and a 1C device form a Level1 network, and the other 1S device acts as a Level2 network. The video data in the Level1 network is pulled from the CDN service, and the Level2 network data is pulled from the Level1 network. In the second set, a 1S device is directly connected to the CDN node to push the video.

QoS is the most concerning aspect for video streaming services, mainly reflected by the delay frequency and the delay duration. The impact of the delay duration on user experience is more obvious. When the delay duration is small, the delay frequency can be ignored. For accurate calculating, delay frequency, and delay duration, we use an external camera to record the pictures of two 1S devices, and use the python script to calculate the result according to the QR code timestamp. We can calculate the delay duration and the delay frequency by comparing the time difference between each frame of the captured video and the original video. The experimental environment is shown in Figure 9.

Figure 9. Play push videos from different sources.

4.4.3. Experimental Results

In this subsection, we evaluate the performance of HomeCDN based on TSCFN. To better verify the performance in different network environments, three rounds of tests were performed by simulating different external network status to record the playback delay. The simulated network environments are normal network status, network status with a packet loss rate of 10%, and network status with a packet loss rate of 30%, respectively. At the same time, considering that different delays will have different impacts on user

experience, we made a detailed comparison between the delay of over 500 ms and over 1000 ms. The performance comparisons are shown in Tables 4–6.

Table 4. Playback latency under normal network.

Data Source	Latency > 500 ms		Latency > 1000 ms	
	Delay Frequency	Delay Duration	Delay Frequency	Delay Duration
Centralized CDN	0	0 s	0	0 s
TSCFN based HomeCDN	0	0 s	0	0 s

Table 5. Playback latency when network delay is 100 to 300 ms.

Data Source	Latency > 500 ms		Latency > 1000 ms	
	Delay Frequency	Delay Duration	Delay Frequency	Delay Duration
Centralized CDN	9	210.6 s	9	210.6 s
TSCFN based HomeCDN	0	0 s	0	0 s

Table 6. Playback latency with 30% packet loss rate.

Data Source	Latency > 500 ms		Latency > 1000 ms	
	Delay Frequency	Delay Duration	Delay Frequency	Delay Duration
Centralized CDN	1	271.0 s	1	271.0 s
TSCFN based HomeCDN	12	10.9 s	4	4.3 s

Table 4 shows that under a normal network environment, the performance of HomeCDN is comparable to traditional CDN. The playback is very smooth without any delay. With the deterioration of the network status, it can be seen that the stability of the HomeCDN network obviously surpasses traditional CDN from Table 5. With the CDN service, the delay occurs four times in total and the playback delay is as long as 210 s, which seriously affects user experience. At the same time, the playback of HomeCDN is still smooth without any delay.

As the network environment continues to deteriorate, it proves that HomeCDN has obvious advantages. Table 6 shows that compared with traditional CDN, the delay duration of HomeCDN is decreased by 96% in a network environment with a packet loss rate of 30%. In particular, the long delay period (over 1000 ms) has been significantly improved to ensure the user experience. Although the delay of HomeCDN occurs more often, each time is short enough to have little impact on the user experience.

5. Conclusions

In this paper, considering the increasing of terminal computing power, we propose a Terminal-side Computing Force Network architecture, namely, TSCFN. Based on cloud-native enabling technologies, TSCFN reconstructs the idle terminal computing power resources and builds a terminal computing power resource pool. The distinctive feature of TSCFN is the support of mainstream Android devices by customized operation system based on an open-source platform, which further expands the scope of computing nodes and extends to a large number of smart devices at home. We verify the functionality and analyze the performance improvement effect of TSCFN. The experimental results show that the CDN service provided by TSCFN can reduce latency, especially in an unstable network status. Due to the limitations of the experimental environment, we only take the CDN service as a test scenario. The application of the proposed TSCFN on a wider scale has yet to be further verified.

In the future, the main research direction is the scheduling mechanism optimization because existing open-source platforms lack efficient scheduling algorithms for our scenario. We will design an adaptive mechanism for dynamic task scheduling to better match the characteristics of terminals and improve resiliency and performance. What is more, we will attempt to introduce a zero-trust model to enhance the security of the system on this topic as well.

Author Contributions: Conceptualization, J.G.; methodology, J.F. and H.X.; software, T.Z.; validation, H.X.; resources, T.Z.; data curation, H.X.; writing—original draft preparation, J.G.; visualization, H.X. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lei, B.; Liu, Z.; Wang, X.; Yang, M.; Chen, Y. Computing network: A new multi-access edge computing. *Telecommun. Sci.* **2019**, *35*, 44–51.
2. CMRI; Huawei Technology Co., Ltd. *White Paper on Computing Power Aware Network Technology*; Huawei Technology: Shenzhen, China, 2019; Volume 1, p. 23.
3. CMCC. *Computing Force Network White Paper*; CMCC: Guangzhou, China, 2021; Volume 1, p. 31.
4. Institute, C.U.G.R. *White Paper on Computing Network Architecture and Technology System*; Institute, C.U.G.R.: Beijing, China, 2020; Volume 1, p. 220.
5. Corporation, C.T. *Cloud Convergence 2030 Technology White Paper*; Corporation, C.T.: Guangzhou, China, 2020; Volume 1, p. 41.
6. Zhan, W. Computation Offloading Scheduling and Resource Management Strategy in Mobile Edge Computing. Ph.D. Thesis, School of Information and Software Engineering, Sichuan, China, 2020.
7. Li, M.; Cao, C.; Tang, X.; He, T.; Li, J.; Liu, Q. Research on Edge Resource Scheduling Solutions for Computing Power Network. *Front. Data Comput.* **2020**, *2*, 12.
8. Pu, S.; Zhao, W.; Luo, J. Research and design of computing power sharing platform based on k8s. *J. Coll. Electron. Eng.* **2019**, *8*, 216.
9. Dong, Y. Research on Key Problems of Task Deployment in Cloud-Edge Joint Datacenter Environment. Ph.D. Thesis, Jilin University, Changchun, China. 2021.
10. Wang, B. Research and Implementation of Multi Cluster Container Cloud Resource Scheduling Mechanism Based on Edge Computing. Ph.D. Thesis, Shandong University, Jinan, China, 2019.
11. Liu, B.; Gu, N.; Gu, D.; Su, J. Implementation of OS-level virtualization technology for Android on mobile platform. *Comput. Eng. Appl.* **2017**, *53*, 7.
12. Gillani, K.; Lee, J.H. Comparison of Linux virtual machines and containers for a service migration in 5G multi-access edge computing. *ICT Express* **2020**, *6*, 1–2. [CrossRef]
13. Shah, S.; Gregory, M.A.; Li, S.; Fontes, R. SDN Enhanced Multi-Access Edge Computing (MEC) for E2E Mobility and QoS Management. *IEEE Access* **2020**, *8*, 77459–77469. [CrossRef]
14. Chae, M.S.; Lee, H.M.; Lee, K. A performance comparison of linux containers and virtual machines using Docker and KVM. *Clust. Comput.* **2019**, *22*, 1765–1775. [CrossRef]
15. Guru, S.K.; Patil, M.V.T.; Dhus, A. Survey on docker. *Nat. J. Comput. Appl. Sci.* **2019**, *2*, 5–9.
16. Morabito, R.; Cozzolino, V.; Ding, A.Y.; Beijar, N.; Ott, J. Consolidate IoT Edge Computing with Lightweight Virtualization. *IEEE Netw.* **2018**, *32*, 102–111. [CrossRef]
17. Avino, G.; Malinverno, M.; Malandrino, F.; Casetti, C.; Chiasserini, C.F. Characterizing Docker Overhead in Mobile Edge Computing Scenarios. In Proceedings of the Workshop on Hot Topics in Container Networking and Networked Systems, Los Angeles, CA, USA, 25 August 2017; Association for Computing Machinery: New York, NY, USA, 2017.
18. Shah, J.; Dubaria, D. Building Modern Clouds: Using Docker, Kubernetes & Google Cloud Platform. In Proceedings of the 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 7–9 January 2019.
19. Casalicchio, E. *Container Orchestration: A Survey*; Systems Modeling: Methodologies and Tools; Springer: Cham, Switzerland, 2019.
20. Wang, H.; Wang, Y.; Bai, S. Research on edge computing technology based on k3s in ubiquitous power IoT. *Sci. Technol. Inf.* **2019**, *17*, 3.
21. Gu, D.; Gu, N.; Liu, B.; Su, J.; He, A. Virtualization Technology of Android System Based on LXC. *Comput. Syst. Appl.* **2017**, *26*, 6.
22. Wu, J. Design and Implementation of Key Technologies of Android System Virtualization Based on LXC. Ph.D. Thesis, Zhejiang University, Hangzhou, China, 2014.
23. Chen, W.; Xu, L.; Li, G.; Xiang, Y. A Lightweight Virtualization Solution for Android Devices. *IEEE Trans. Comput.* **2015**, *64*, 2741–2751. [CrossRef]

24. Columbia University. ceClub: A Virtual Mobile Smartphone Architecture. In Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles, Online, 23 October 2011.
25. Detken, K.O.; Oberle, A.; Kuntze, N.; Eren, E. Simulation environment for mobile Virtualized Security Appliances. In Proceedings of the IEEE International Symposium on Wireless Systems, San Francisco, CA, USA, 25–28 June 2012.
26. Uhlig, R.; Neiger, G.; Rodgers, D.; Santoni, A.L.; Martins, F.; Anderson, A.V.; Bennett, S.M.; Kagi, A.; Leung, F.H.; Smith, L. Intel virtualization technology. *Computer* **2005**, *38*, 48–56. [CrossRef]