Chinese Journal of Electronics
Vol.29, No.5, Sept. 2020

Transparent Computing: Development and Current Status

ZHANG Yaoxue[1], DUAN Sijing[2], ZHANG Deyu[2] and REN Ju[2]

(1. *Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China*)

(2. *School of Computer Science and Engineering, Central South University, Changsha 410083, China*)

Abstract — Current terminal devices, such as Personal computers (PCs), diskless workstation, mobile terminals and embedded Internet of things (IoT) devices, are facing several challenges, including limited computing capabilities and storage resources, complex maintenance and management of the software and data, as well as secure service provisioning. Facing these challenges, Cloud computing (CC) and Edge computing (EC) encourage the terminal devices to shift compute-intensive works to the cloud servers or edge nodes. Different from CC and EC, Transparent computing (TC) is a novel user-centric computing paradigm in which the hardware and software are separated in different places. Specificaly, all the software, including Operating systems (OSes), applications programs and management tools, are deposited on the servers and transmitted on demand to terminal devices in a transparent way, while the computation is performed on terminals. In this article, we give a broad survey on transparent computing about the development and current status and compare the differences with CC and EC. To realize the provisioning of the cross-platform service, TC shields hardware differences of heterogeneous devices and builds standardized hardware and software interfaces. From the perspective of security, TC entails high-level security mechanisms and the Meta OS can detect malware and OS-level attacks. Specifically, we first review the development courses of network computing paradigms and the motivation of TC. Then, we present a comparison of several emerging computing paradigms and TC from three perspectives, *i.e.*, virtualization, location for computing, and location for storage. We also show the basic architecture of TC, development process, current status, and key enabling technologies of TC for PCs and lightweight terminals. Lastly, several new challenges and future research directions are indicated to inspire continuous investigations. We believe that this survey can help readers to obtain complete information about TC.

Key words — Transparent computing, Network computing, Survey.

I. Introduction

In the last decades, computing paradigms, network communication, and big data have evolved in a spiral trend. The advances of computing paradigms have undergone several periods, from the early single-computer computing to centralized mainframe-centric computing. By the 1980s, the local area network technology has gradually grown mature, computers can share resources through the network, which forms a distributed client/server computing paradigm. Subsequently, mobile devices and digital home appliances emerge, the size of computers is further miniaturized and the types become diversified. To further satisfy the diverse demands of the information era, Xerox PARC Lab proposed ubiquitous network and pervasive computing, aiming at establishing an intelligent environment where computing anywhere and anytime. However, how to realize a user-centric service sharing is still a difficult issue, which has aroused extensive researches from academia and industry. For example, several companies including VMWare, HP, and Intel propose Virtual machine (VM) computing[1] aiming to provide multiple Operating systems (OSes) and application services on a single hardware platform. However, this computing model requires very high-performance computing machines and a high bandwidth network, so it is not suitable for small devices and low bandwidth network environments. Furthermore, the grid computing proposed at the beginning of the 21st century also integrates the resources of multiple computers from the perspective of resource sharing. But it is difficult to popularize to ordinary end users. In recent years, the number of computing devices and data traffics are growing rapidly, Cloud computing (CC) has received extensive attention. CC provides powerful

Manuscript Received Mar. 31, 2020; Accepted June 13, 2020. This work is supported by the National Key R&D Program of China (No.2019YFA0706403), 111 Project (No.B18059), the National Natural Science Foundation of China (No.61702561), and the Natural Science Foundation of Hunan Province (No.2020JJ5774).

 DOI:10.1049/cje.2020.07.001

remote computing resources and scalable storage spaces to end-users through a high-speed network[2]. However, transmitting massive data to the cloud server consumes long round-trip latency and high bandwidth. And as the size of computers become smaller and the types become more diverse, CC cannot solve hardware heterogeneity.

Furthermore, the development of CC has spawned many applications, accompanied by the explosive growth of big data[3]. Cisco predicts that the number of global mobile terminals will reach to 12.3 billion, and the monthly data traffic is expected to reach 77 exabytes by 2022[4]. And Artificial intelligence (AI) is also growing rapidly. A variety of intelligent applications penetrate our lives, for example, image recognition[5], speech recognition[6], intelligent robotic[7] and autonomous vehicle[8], *etc.* These emerging applications usually require real-time data processing and fast responses. But CC presents many limitations in supporting smart terminals in real-time, such as the computation and storage are executed on remote servers, high data transmission overhead, and low security, *etc.* These limitations bring difficulties to meet the service requirements for delay-sensitive users, and cannot provide cross-platform services on-demand for intelligent terminals. Meanwhile, real-time requirements also pose challenges to network communications.

To satisfy the real-time requirements of these emerging applications, the development of network communication has also evolved over a long period. The clients and servers were connected via a dedicated wire or telephone line in the early days of network computing paradigms. With the increasing number of smartphones, we are entering the era of mobile Internet and Internet of things (IoT). The network has gradually shifted from wired to wireless systems, and the communication technology has also evolved from 2G, 3G to 4G/LTE. Currently, the 5G wireless system is considered as a promising way to satisfy the increasing requirements of wireless communications, by leveraging high-frequency communication technologies, ultra-dense networks, and MIMO[9]. Compared with 4G, 5G will greatly increase the data transmission and spectrum utilization rates, together with low latency, low power consumption, and high-bandwidth communication services and mobile Internet experiences for billions of mobile and IoT devices.

The above advances have promoted the development of network computing, some novel computing architectures are also proposed successively in recent years, such as cloudlet, Fog computing (Fog), Edge computing (EC), the same idea behind them is to employ small-scale servers that close to end-users and provide timely services without the long distances data transmission between terminals and remote servers[10]. In Fog, MEC, and cloudlet architecture, users usually adopt task offloading schemes and the data are processed at the nearby edge servers to decrease the delay and improve user experience. However, all of these computing paradigms cannot solve hardware heterogeneity and provision cross-platform services. Although the high-speed networks accelerate data transmission, but still cannot avoid data privacy leakage problems. Therefore, we need a novel computing paradigm named Transparent computing (TC) to provide safe, cross-platform and real-time services for heterogeneous smart terminals in the current intelligent computing era.

The motivation of TC is to isolate the computation and storage for the servers and terminals. Specifically, all the software, including Operating systems (OSes), applications programs, and management tools are deposited on the servers, while the computation is performed on terminals. TC allows users to fetch the necessary OSes and code blocks in a streaming way from servers to local terminals, without caring about the information of service provision, such as update and management of software[11]. In such a way, the computing abilities of intelligent terminals could be utilized and the storage space of local devices is reduced. Furthermore, TC addresses the heterogeneities of various IoT terminals and supports cross-platform and on-demand services, which is regarded as a promising way to address various challenges in the current IoTs and mobile Internet.

In this paper, we will give a comprehensive survey on the development and current status of TC. The remainder of the survey is presented as follows. Section II compares the differences with several emerging computing paradigms, *viz.*, TC, CC, and EC. Section III introduces a basic TC architecture and three key technologies of TC. Section IV reviews the development process of key technologies for TC and the different architectures designed by TC for heterogeneous terminals, including TC for PCs, TC for lightweight mobile and IoT devices, security and privacy. Section V summarizes a novel transmission mode in TC named block-streaming as a service. Several new challenges and future directions are outlined in Section VI. Finally, we give a conclusion in Section VII.

II. Comparison of Several Emerging Computing Paradigms

In this section, we compare the differences among the aforementioned emerging computing paradigms and TC. The common features of CC and EC are to unify computing and storage, to decrease the network latency, and to enhance service efficiency. While TC paradigm emphasizes the separation of computation and storage and designs a Meta OS to realize the virtualization instead

of VM and container which are mainly used in CC and EC. In the following, we will compare the differences from three aspects in Table 1, *viz.*, virtualization, location for computing, and location for storage.

Table 1. Comparison of various computing paradigms

Paradigms	Virtualization	Location for computing	Location for storage
TC	Meta OS	Proximal end	TC server
CC	Hypervisor and containers	Centralized cloud server	Centralized cloud server
EC	Hypervisor and containers	Edge servers	Edge servers

1. Virtualization

The purpose of the TC architecture is to solve the diversity of different terminal devices and provide a united environment for various instance OSes and cross-platform application programs[11]. Therefore, the complication and expenses of hardware development are decreased and the harmony of software are strengthened. Controversially, the motivation of CC and EC paradigms is to handle computation tasks at the centralized cloud servers or the nearby edge servers to decrease the delay and enhance the service experience and efficiency.

Actually, CC, EC, and TC use different technologies to achieve virtualization. TC realizes virtualization by designing a super-OS control platform named Meta OS that can instantiate terminal OSes, thus shield the hardware differences of terminal devices and build standardized hardware and software interfaces. To support the separation of program execution and storage, Meta OS enables the terminal devices to choose an instance OS on-demand and load code blocks to the local devices[12]. However, CC and EC mainly adopt the hypervisor[13] and containers[14] to provide virtualization services, respectively. The hypervisor is a middle tier that operates between the basic physical devices and OSes. It enables various OSes to share physical resources. These OSes come as Virtual machines (VMs)[15], and the hypervisor acts as a Virtual machine monitor (VMM) to manage these VMs and allocate hardware, memory, CPU, and disk to each VM. And the container technique is a kind of kernel lightweight OS layer virtualization technique that runs on the physical host OS. It is easier to deploy and resource-efficient than the hypervisor. Compared with TC, the hypervisor and containers used in CC and EC introduce overhead because of the need to virtualize all computing devices, both of them realize virtualization by exploiting physical computing resources, like the CPU or memory, which is suitable to run on servers with strong computing capabilities[16]. But the Meta OS of TC can flexibly schedule and load resources from remote servers, as well as providing central management and monitoring functions for lightweight intelligent terminals, which can greatly reduce the extra overhead brought by full virtualization and improve system performance.

2. Location for computing

From the perspective of location for computing, the TC paradigm highlights computing at the proximal end. Specifically, the computing is performed at the terminal devices by obtaining the instance OSes and app programs from the TC server[17]. The motivation is three aspects. The first is the application requirements, most computations are lightweight that terminals can handle. For the computing resources aspect, CC emphasizes centralized computing, whereas the resources of proximal end are idle. TC can fully utilize the capabilities and resources of terminal devices to finish suitable computations and provide low-latency services. Thirdly, in the era of big data, TC emphasizes shipping code to data and block streaming services, without transmitting massive data to the cloud server, which consumes long round-trip latency and high bandwidth. On the contrary, CC usually processes computation tasks on the centralized cloud data center. While the purpose of EC is to supply the proximate computing resources of edge servers for end-users to decrease the latency of data transmission and enhance service experiences[18].

Leveraging the block streaming, the TC terminals do not need to load the whole software codes and resources to the front-end for computing, but only the code blocks and data related to the requested service are delivered to terminal devices in a transparent way[19]. Therefore, the streaming computing in TC can realize the adaptive segmentation of code blocks, thus significantly reducing communication and computing costs. But in CC, the computation works are executed on a resource pool composed of many computing units, enabling various applications to obtain compute power, storage spaces, and services as needed. CC provides three levels of computing service models, *i.e.*, IaaS, PaaS, and SaaS[20]. For EC, edge servers provide nearby computing resources for terminal devices. The terminal users can improve their computing productivity by shifting some compute-intensive works to edge or cloud servers. Therefore, computation offloading is essential in EC via flexible task separation among cloud servers, edge nodes, and end devices. In recent years, a large amount of research works study program partitioning, computation allocation, resource management, and distributed execution in EC[21], and this computing paradigm is suitable for the delay-sensitive application scenarios.

3. Location for storage

To unify the resource management and distribution, as well as to provision cross-platform services for distributed computing terminals[22]. TC adopts the centralized resource management method that stores OSes, supporting tools and applications on the back-end transparent servers. The front-end of TC is usually bare machine and equipped with necessary protocols, *i.e.*, MRBP (Multi-OS remote booting protocol) and NSAP (Network storage access protocol)[23], and parts of Meta OS components for client startup. The motivation of cloud computing provides end-users with scalable remote storage and computing resources. Unlike cloud storage, EC stores data at the edge servers near the data sources. It has lower network communication overhead, interaction delay, and bandwidth costs, and can provide reliable data storage and access for EC.

In TC, the clients fetch the required resources from the server as required and perform locally via block streaming loading. Rather than pushing the whole application to devices, TC takes the code block of an application as a unit of service and loads them to the terminal in a block streaming manner according to the user's request. After execution, the blocks can be decommissioned and removed if it exceeds the time threshold on the device, without occupying the additional storage space. Therefore, the storage technique of TC is suitable for resource-constrained lightweight devices and PC. However, cloud storage deposits the data and resources in many virtual devices hosted by third-party companies[24]. These companies are required to design cluster technology, distributed file systems, and grid computing to realize collaboration between distributed storage devices. Thus, cloud storage is usually exploited to deposit large volumes of data.

In EC, edge servers supply the proximate storage resources for end-users to store data directly at the data collection point without transferring to a storage center server or cloud storage. The edge storage consists of edge device, edge data center and distributed data center and is able to utilize data de-duplication and approximate storage technology to alleviate the storage pressure of cloud storage[25]. Therefore, edge storage is widely employed in emerging applications, such as industrial IoT, Internet of Vehicles, smart cities, and unmanned aerial vehicles.

III. Key Technologies of Transparent Computing

Since the introduction of TC, its theoretical research and applications have received extensive attention from academia and industry. TC was first proposed in 2004[22], the basic architecture is shown in Fig.1. As an emerging computing paradigm, TC gets rid of the constraints of traditional computers and realizes the centralized management of network/computing resources and distributed computing. In 2007, the researchers designed a super-OS control platform named Meta OS for loading programs through a distributed 4VP+ platform[23]. In 2012, Zhang *et al.* developed an OS in the cloud named TransOS[26], which could replace traditional desktop OS[27]. Moreover, a research team in Intel proposed to abstract the underlying platform details through a software-hardware interface called UEFI (Unified extensible firmware interface) for the TC system[28]. Based on TransOS, a novel service model BaaS (Block-streaming as a service)[29] was proposed. Since we enter the era of IoTs and mobile Internet, researchers began to focus on the mobile transparent computing[30] and IoT-based on TC[31], and designed product of TC in wearable devices called TC Watch[32]. In summary, TC has three key technologies, *i.e.*, Meta OS, MRBP and NSAP, and streaming loading. In the following, we introduce the core idea of these technologies.

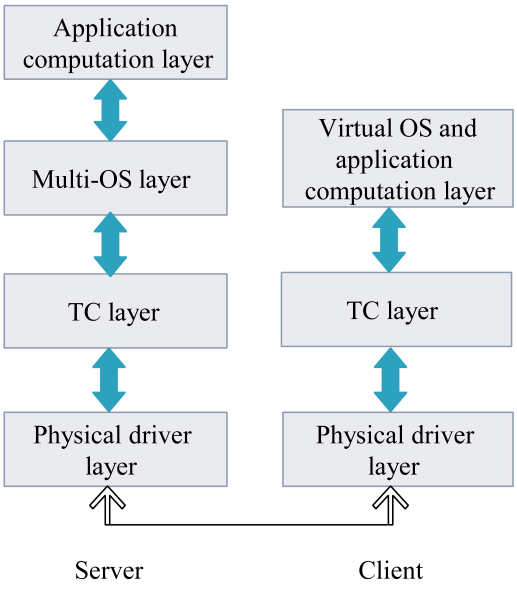

Fig. 1. The basic architecture of TC

1. Meta OS

Meta OS is a super-OS control platform that manages and schedules other instance OSes, as well as hardware and software resources on transparent clients/servers[12]. Meta OS is spatially distributed and locates under the instance OS. The components of Meta OS are mounted on different transparent terminals/servers and works in coordination with network protocols. Any information can be exchanged between various parts of Meta OS, including data information, instructions, and terminal signals, *etc.* We describe a typical Meta OS implementation called 4VP+ (4

Virtual layers and 2 protocols) in Fig.2[23]. In this implementation, the function of Meta OS is divided into two protocols and four virtual layers. Noting that Meta OS is a distributed OS, each protocol and virtual layer has a client and server.

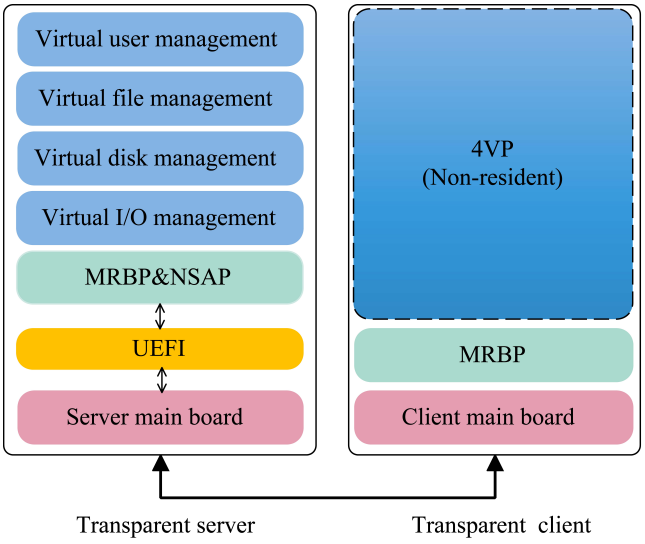

Fig. 2. The composition of 4VP+

The four virtual layers work together to manage and schedule software and hardware resources on transparent terminals and servers.

i) Virtual I/O Management. The main function of virtual I/O is to receive the interrupt requests on terminals and I/O processing requests from users, then analyzing the cause of the interrupt and waking up the response handler, while performing the allocation and buffer processing of related devices and starting the I/O operations. The virtual I/O must handle different kinds of interrupts and I/O requests, as well as requiring to manage multi-level queues and performing synchronous mutex processing of responses between transparent terminals and servers.

ii) Virtual Disk Management. The virtual disk has three functions: the allocation and recovery of virtual equipment; the image mapping and management of transparent terminal and server; the exchange and scheduling between transparent terminals and servers.

iii) Virtual File Management. The functions of the virtual file layer are the distribution and management of file spaces and directories, as well as the redirection of file access. Besides, virtual file management also needs to ensure the uniformity of file access and provide the capabilities of file controlling and searching.

iiii) Virtual User Management. The virtual user layer provides the management of user parameter configuration and the allocation of transparent terminal addresses.

2. MRBP and NSAP

Meta OS contains two protocols, *i.e.*, MRBP and NSAP. The main function of MRBP is to capture the initial startup interrupt signal of the transparent terminal and send a boot request to the server. After finding the specific OS selected by the user, the client MRBP will download and startup the NSAP module to enable a virtual I/O for terminals. The client acquires the instance OS stored in the server through the virtual I/O devices and boot them from the transparent server to execute in the client memory according to the choice of users. The NSAP[33] is a module written by assembly instruction, which is used for transmitting instructions, data, and signals to start virtual I/O between transparent terminal and server. Each client needs to set two request queues, one for the virtual disk requests, and the other for the user services request. Finally, the server responds to the user requests including the necessary data and services. Additionally, to ensure trustworthy data delivery, the server NSAP will confirm the signal to the client disk request, and the NSAP client will use the timeouts mechanism to detect the missing request so that the transparent client and server can verify with each other.

3. Streaming loading

The detailed booting of Meta OS and streaming loading of instance OS are depicted in Fig.3[12]. When the terminal device is powered up and finds that the needed data blocks of Meta OS or instance OS are not deposited in the local memory, the client MRBP sends an interrupt request to the transparent server[23], as depicted in Fig.3(a). After the transparent server listens and receives the request from the transparent terminal, it reads the corresponding instructions or data blocks from the memory or hard disk and sends it to the transparent terminal through the transparent delivery network, as shown in Fig.3(b).

When the transparent terminal receives the OS instructions transmitted from the server, the client starts NSAP to access the virtual disk image in the server repository through the virtual I/O device. The terminal devices then launch the master boot record of the special virtual disk to continue the booting procedure and load them into the memory of the transparent terminal for execution, as shown in Fig.3(b), the blocks $\{M_1\}$, $\{M_2\}$, $\{M_3\}$, $\{M_4\}$ become $\{M_1, M_i\}$, $\{M_2, M_j\}$, $\{M_3, M_k\}$, $\{M_4, M_l\}$, respectively. After loading the instance OS, the Meta OS and JITC (Just-in-time computing) layer continues to load the necessary codes and application programs from servers to the terminal in an on-demand manner, thereafter utilizing local resources (CPU and memory) to execute computation tasks. Since the only required instructions or data blocks are fetched each time, the above process will loop repeatedly between the transparent terminal and the server as computing

proceeds, thereby forming dynamic scheduling. Besides, the virtual disk driver and the virtual file module will clear the cache contents before the client is shut down. In such an extension of the von Neumann architecture, the external memory can be extended from TC clients to servers via wired/wireless network[34].

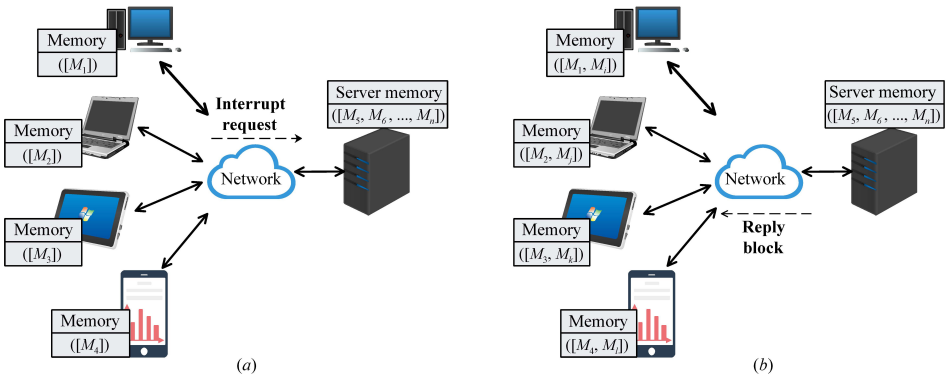

Fig. 3. Distributed scheduling of OSes in TC

IV. The Development Process of Transparent Computing

Over the past 30 years, TC has gradually developed and extends several different architectures to support various heterogeneous terminals. In the following, we will introduce the development process of key technologies for TC and several TC-based architectures, Furthermore, we will also discuss performance optimization, privacy and security of the TC system.

1. Transparent computing for PC

Initially, the clients served by TC are mainly fixed computing devices such as PCs, PDAs and digital home appliances. There was no OS, supporting tool and application program running on them. These resources are deposited in the TC server repositories and could be flexibly loaded via the network during execution, thus improving the degree of software sharing and reducing costs as well as the complexity of management. We give an illustration in Fig.4.

TC extends von Neumann architecture by expanding the traditional single-computer mode to networked computers so that the distributed computers can share the application programs from each other[17]. Specifically, the architecture of TC for PCs consists of two components, *i.e.*, TC client and TC server, both of them load and run different parts of Meta OS (a super-OS control platform). The TC server acts as a warehouse, which stores and centrally manages the heterogeneous instance OSes, software, user data, and management tools. The clients must be equipped with two protocols, *i.e.*, the MRBP and NSAP. The BIOS is the basic input and output system which is in charge of connecting hardware and software.

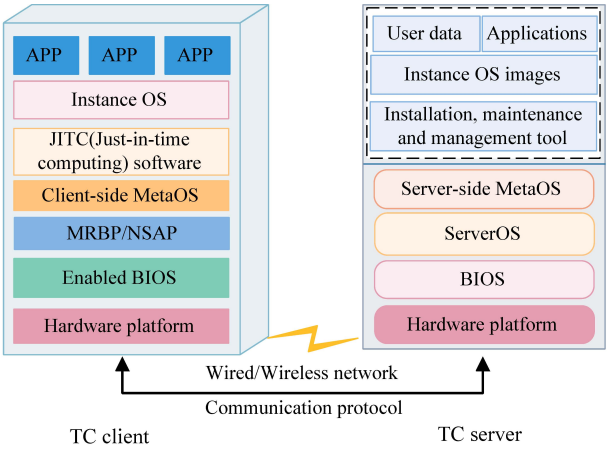

Fig. 4. The illustration of TC architecture for PCs

In recent years, to better serve PCs, the key technologies of TC have been gradually developed and optimized, including MRBP and NSAP, four virtual layers, cache and resource management, virtual machine-based TC.

1) MRBP and NSAP MRBP and NSAP attract numerous research efforts. To enhance the flexibility and availability of terminal devices startup, Zhou *et al.*[35] proposed a customizable remote boot protocol named NCBP (Network-based client boot protocol) for network computing. NCBP can obtain the native identity by extending the DHCP protocol and then load the MenuBatch script execution environment by using the secure APTP protocol. Through the interpretation and execution of the MenuBatch script, the client loads the corresponding OS kernel from the server according to the user's choice, to achieve the remote startup of the device. However, NCBP only supports Linux and Win98 OS. Wang *et al.*[36], proposed a remote boot

mechanism that supports multi-file kernel OS. Another research[37] presented a new remote booting protocol called MRBP2, which was more secure and efficient than MRBP. To address the performance problems of the booting procedure, Guo *et al.*[38] analyzed the workload and booting details during the client startup process, then designed a closed queuing network model. In 2009, the Li *et al.*[39] proposed ENCBP, which improved the process of NCBP protocol interaction by differentiating the types of OS kernel structures that were booted. It also improved the efficiency of the remote boot of the single kernel OS and enhanced the security of the protocol. Moreover, it saved the demands for additional chips by introducing a firmware boot agent in the BIOS chip. For the research on NSAP, Li *et al.*[40] analyzed the process of service startup and continuous service content delivery. They found that the traditional NSAP overlooked the lower-layer performance, for instance, the transport delay and user experience. Therefore, they presented an enhanced version of NSAP called NSAP+ to improve the quality of delivery service.

2) Four virtual layers In Table 2[41–47], we summary the main research work about the optimization of four virtual layers.

3) Cache and resource management For cache and resource management, cache mechanism is one of the crucial component that affects system performance in TC. Since traditional cache methods cannot be directly applied in TC virtual storage system. Wei *et al.*[48] proposed a shared storage cache simulator that supports multi-users to share server storage space. This method extended the traditional stack-distance model by using the first storage block number and two-tuple ID to identify the cache block. Moreover, it manages and schedules the cache block according to the access ratio of different users. The simulation results indicate that this approach can effectively enhance system performance and

Table 2. Optimization of four virtual layers

Virtual layer	References	Description
Virtual I/O	[41]	It uses disk I/O redirection and disk mapping to ensure personalized use of users, which can reduce total costs
	[42]	It adopts a multimedia I/O access control method for TC, which contains three schemes to enhance the multimedia I/O processing performance
Virtual disk	[43]	It builds a VM-based network storage system
	[44]	It designs a queuing model and find the key bottleneck of TC performance, and develops cac he schemes to optimize sever virtual disk access
	[45]	It proposes and implements a virtual storage device driver model for TC, including bus scan ning, virtual block device creation, and I/O request processing
	[46]	It combines the characteristics of TC and CC, and designs a virtual disk-based CC platform, which reduces the system costs
Virtual file and user	[47]	It develops a method for data transmission between computer users in TC system, which imp roves the processing speed and reduces the resource occupation

scalability. Tan *et al.*[49] designed a TC system based on hierarchical caching, aiming at improving I/O performance on both client and server sides. In terms of the cache management strategy, they proposed an improved LRU algorithm called LRU-AFS based on access frequency counting threshold to improve the cache hit ratio. The results show that this algorithm can greatly decrease the client startup time and improve the random read and write throughput while reducing the traffic. To fit different cache sizes and workloads, Gao *et al.*[50] designed a frequency-based multi-priority queue in the server cache, while the LRU was used in the client. However, it remains challenging to assess the effectiveness of a TC system with a specific cache strategy. Liu *et al.*[51] proposed an approach named TranSim for TC, which can assess the effectiveness of multi-level cache hierarchies in the TC system under various cache strategies. With TranSim, the system designers are able to efficiently assess the effectiveness of cache schemes and system performance. Recently, Zhang *et al.*[52] adjusted the remote resource access model in TC to a multi-level cache framework, which effectively reduces the network latency of resource transmission by setting caches at the network edge.

From the perspective of resource management, resource management in TC is mainly the management of data and users. Guo *et al.*[53] investigated data inconsistency problem caused when programs running on different clients modify data at the same location in the virtual hard disk image and proposed a data consistency method based on block granularity. Gao *et al.*[54] also developed a remote resource management approach for TC, which can collect the status of the resources of the terminals by leveraging virtualization techniques.

4) Virtual machine-based transparent computing In the TC architecture, VM technology enables users to use applications across OS platforms, but the software and hardware management of VMs are complex. As shown in Fig.5, Xu *et al.*[55] proposed a VM-based TC system, which redirected the input and output requests

of users to the server through the virtual device model, and realized the loading and application of unmodified OS in the TC environment without concerning the technical details. A prototype system based on the Intel VT platform and XEN virtual machine proved its effectiveness and feasibility. A similar idea was also validated in Ref.[43]. Chen *et al.*[56] implemented a Lightweight virtual machine-based TC (LBTC) system that allowed the user OS to directly access devices other than network ones, such as GPUs, which greatly reduced the overhead caused by full virtualization. At the same time, LBTC adds a virtual storage device model in the service domain and uses a lightweight VM monitor to redirect user OS storage I/O requests to the server, thus improving system performance. In addition, to monitor the network packets in a virtual computing environment, Du *et al.*[57] designed a transparent network monitoring system based on the VM. This system operates in the host system and can configure different monitoring rules according to the service running in each VM. Wang *et al.*[58] proposed a cloudware PaaS platform based on the concept of VM-based TC, which can efficiently deploy cloud software in a cloud environment, thus solve the problem of software cloudification.

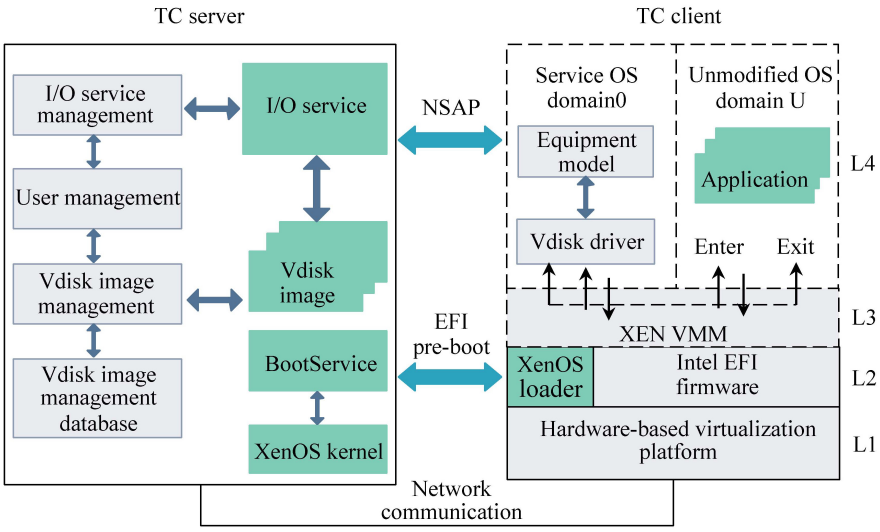

Fig. 5. The VM-based TC system

5) Performance optimization To improve the service quality and user experience of the TC system, performance optimization has always been the main focus of researchers, such as load balance, kernel optimization, resource allocation and so on. Xue *et al.*[59] use stochastic Petri nets to model the performance of TC and provided mathematical analysis on the response latency, throughput, and wait time. In the following, we summarize the related works about the performance optimization of TC in Table 3[60–70].

6) Security and privacy Recently, the rapid development of computer networks has brought many challenges for information security. Although TC can provide advanced security for end users[71], it also faces security risks from various aspects, such as malicious attacks, worms/trojans, and inevitable privacy leakage. At the same time, there are some new security challenges because of the unique characters of this new computing architecture, including multi-OS remote booting, virtual disk sharing, *etc.* Thus, it is essential to design security mechanisms and privacy-preserving model for this architecture of TC. To address these security threats in the TC system, some research works propose effective approaches.

Kuang *et al.*[72] found that there is no verification mechanism when downloading resources via the remote booting protocol. An attacker can attack the TC system by disguising a malicious machine as a server startup service, or intercepting and tampering with the startup code to be downloaded to the end systems, which leaves security risks to the TC system. Therefore, Ref.[72] proposed a Remote start integration service (RBIS) method to solve the security problems during the remote startup of TC. RBIS is a method based on public-private key cryptography and integrity verification, which improves the security of the startup process by verifying the integrity of the startup code obtained by the end devices during the startup process. In addition, Du *et al.*[57] designed a network surveillance model based on VM, to recognize and process security risks. The model was placed in the host system and separated with the target VM to achieve a more secure system. Besides the remote booting protocols and network monitoring, Wang *et al.*[73] summarized the security challenges of TC

system and developed a TC security framework, which allows users to choose the desired security mechanisms to configure controllable security execution environment and enable users to protect their own data and applications. To protect data security, Peng *et al.*[74] developed a multi-level AC mechanism in TC, it provides data security, authentic identity authentication and comprehensive protection against malicious attacks.

From the privacy protection perspective, privacy threats from confidentiality and integrity are key issues. Xue *et al.*[75] developed a system-level privacy-preserving model for TC systems which is based on mandatory AC. This model adopted formal methods to analyze the privacy-preserving problem and had five state transition rules. Therefore, TC system will protect sensitive data. The security proofs proved the validity of the model theoretically.

2. Transparent computing for lightweight mobile devices

Over the past decades, TC has been successfully applied in the PC-oriented applications and has shown its advantages to provide scalable cross-platform services. As we are entering the era of mobile Internet and IoTs, the computing environment becomes heterogeneous which includes a variety of lightweight intelligent devices, edge servers, and cloud servers. All of these computing devices

Table 3. Overview of performance optimization for TC

Technology	Optimization type	Performance
LBTC[60]	Load balance	i) Reduce average response time ii) Improve system load capacity
E-Eifel[61]	NSAP	i) Reduce the number of pseudo-timeouts ii) Improve the data transmission efficiency of NSAP
ONSA and OFSA[62]	Task scheduling	Achieve 1.1 times acceleration than the optimal solution
2HR-fτ[63]	Routing algorithm	i) Improve the network throughput ii) Reduce deliver delay
HG[64]	File fetch and transmission	Reduce total file fetch time nearly by 50%
TCOS[65]	Kernel and application layer	i) Reduce the time of booting the OS by 33.11% ii) Decrease the memory space usage by 40.43%
SDSCS[66]	Code scheduling	Improve application performance and user experience
DCA[67]	Resource allocation	Improve the TC system performance
TL[68]	Intelligent TC system	i) Integrate TC with machine learning ii) Reduce the training time considerably
HIF[69]	Task offloading	Reduce the offloading energy
TCC[70]	Cache optimization	i) Decrease the usage of server disk I/O ii) Improve startup speed of the terminal devices iii) Decrease the network bandwidth

are interconnected via networks. In such a mixed computing architecture, the service demands of these devices are typically delay-sensitive and context-aware. So the traditional TC is no longer suitable for the current complex computing environment. In recent years, these changes motivate many researchers to investigate the service provisioning of TC for lightweight mobile and IoT devices. In the following, we will discuss several novel architectures and related works of TC in the mobile Internet and the IoT era.

To uniform the resource management and code scheduling of terminals, edge and cloud servers, Zhou *et al.*[30] proposed a mobile TC architecture based on the concept of traditional TC. As depicted in Fig.6, the mobile TC architecture includes five layers, all of them are built on the underlying hardware devices. In the following, we describe the functions of each layer and the corresponding research progresses.

1) Network and communication protocol layer This layer includes underlying network infrastructures and communication protocols, such as WiFi, 3G/4G/5G, the improved MRBP, and NSAP, which aim to provide network transmission between transparent terminals and servers. Typically, the TC mobile terminals load required OS images and application programs from TC servers through the wireless network and customized remote booting protocol. To improve the data transmission efficiency between mobile devices and servers in the WLAN (Wireless Local Area Network), a cross-layer design based dynamic TFTP (trivial file transfer protocol) was designed[76], where the window size varies moderately with the bit rate. In addition, Yin *et al.*[77] applied multicast in mobile TC and developed an efficient multicast grouping scheme for mobile TC over WLAN, which effectively reduced the system launching time and enhanced the QoS.

2) Resource virtualization and management layer This layer is in charge of administering various physical devices, such as local intelligent terminals, proximate computing devices, edge/cloud nodes, as well as the virtual resources including heterogeneous OSes and usage data that are deposited in the TC server.

Different from traditional resource management, some specific characteristics may affect the service performance of mobile TC, *e.g.*, location, mobility, and price. In Refs.[78,79], the authors proposed resources management strategies called SRSM for mobile TC, which was able to adopt to multiple heterogeneous hardware architectures and OSes. In SRSM, the core management layer that consists of multi mobile agents can dynamically schedule and account for the resources and services to achieve efficient resource management. To accelerate the loading of resources from the TC server to mobile terminals and reduce the response time of software, Song *et al.*[80] proposed a two-tiers collaborative caching scheme based on local devices. One is the local cache which was used for store personal user data, and the other was a P2P cache, which deposited softwares. From the perspective of resource storage, Jin *et al.*[81] presented a collaborative storage framework for mobile TC called COAST. With the end-to-end resource sharing technique, COAST enables mobile devices to obtain resources from proximate devices without connecting the Internet.

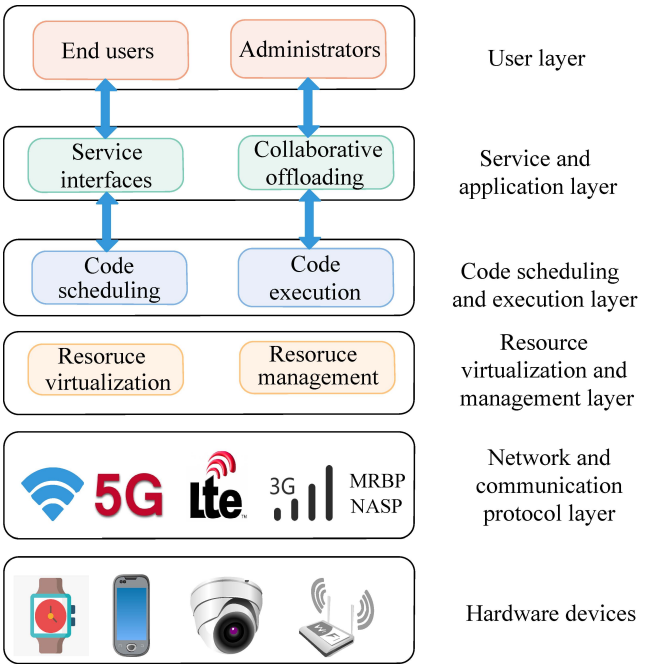

Fig. 6. The illustration of mobile TC

3) Code scheduling and execution layer This layer is responsible for code scheduling from servers to TC terminals and code execution on local devices. Specifically, the lightweight terminals will generate interrupt requests when it needs to execute certain applications and send requests to TC servers. Then the server schedules necessary instance OS and code block according to the instructions to respond to the requests of clients. With this layer, flexible service provisioning on terminals can be realized. Since the TC server serves multiple clients simultaneously, the code scheduling may occur in various computing devices. To realize this scheduling, the entire execution status of the mobile TC system, including user request and response, resource allocation, system performance are required to be monitored in an almost real-time way. Guo *et al.*[82] used NFS (Network file system) protocol to achieve the flexible scheduling of resources under an unstable network wireless condition. They comprehensively discussed the elements that affect the NFS performance such as NFS read/write block size, network bandwidth, *etc.* Finally, the analysis results are used to guide the resource scheduling of mobile users and the performance improvement of NFS. Tang *et al.*[70] found that the frequent I/O operations were the bottleneck of TC, so they proposed a block-level caching optimization approach for both server and client sides, which effectively reduced network traffic and the access frequency for the server. Besides, BaaS[29] was a typical and effective approach to perform code scheduling and execution, which will be discussed in detail in Section V.

4) Service and application layer This layer includes two sub-layers. The service sub-layer provides service interfaces that are used to shield the underlying implementation details of the system, allowing lightweight terminals to interact with the TC server transparently. The transparent way is convenient to use and manage, make decisions smartly and respond adaptively. The application sub-layer can provide many value-added services, such as collaborative computing, service crowdsourcing and so on. Liang *et al.*[83] proposed a cross-platform application design method named CPMTC. CPMTC used the TC server to manage user data and allowed users to access their data on any device. Considering the impact of network quality on mobile user experience, Dong *et al.*[84] developed a transparent agent framework for measuring mobile services, from three perspectives, *i.e.*, underlying network infrastructure, the network-layer behavior, and performance, as well as the service content. Furthermore, Guo *et al.*[85] designed a Context-aware service provisioning (CASP) model in the mobile Internet scenarios. The CASP includes four steps: obtain context parameters at the clients, establish context database and choose service at the server, and push the transparent services to mobile users. The evaluation results validate that this unified framework can greatly reduce software development works.

5) User layer This layer is the highest level of mobile TC which consists diverse types of users. Such as different kinds of smart client users, equipment or infrastructure providers, and system administrators/managers. In 2012, Intel claimed that HTML 5 is a promising technique for client implementation in TC due to its

characteristics of client offline storage, cross-platform support, low development expenses. By integrating HTML 5 with TC, the browser engine can act as the Meta OS layer, which can shield browser differences and achieve cross-platform streaming execution[34]. In addition, the TCWatch, a TC-based wearable platform, is the typical implementation for TC in the user layer[32], which will be described in the following.

3. Transparent computing for IoT devices

In the IoT scenario, IoT devices are often lightweight with weak computing capabilities and limited storage spaces, they are also generally pre-configured specific embedded OSes, such as FreeRTS, RT-Thread, LiteOS, uCOS, and MICO. Since these heterogeneous OSes have different advantages, various vendors develop customized software and services based on them. It makes the IoT devices exposed to scalability and manageability issues. To this end, the resources of IoT devices should be managed centrally and compute in a distributed way without caring the underlying details and incompatibilities caused by heterogeneous terminals. Therefore, how to eliminate the dependence of the hardware and software and provision cross-platform and on-demand services for IoT terminals are challenges in the IoT era.

In Refs.[32,86], the authors leverage TC to establish a scalable IoT framework to address the above challenges. As shown in Fig.7, the TC-based IoT framework contains five layers. We introduce the functions of each layer in detail as follows.

1) End-user layer This layer is comprised of different IoT devices, such as smart wearables, phones, sensors, vehicles, *etc.* All of them can be seen as terminals in the TC-based IoT framework and equipped with resident applications like Meta OS to support resources loading from the edge server layer, remote booting, and cross-platform services.

2) Edge server layer This layer is in charge of capturing users' requests, accessing resources from the cloud servers via the core network, controlling resources scheduling and distribution. Besides, edge devices provide data pre-processing and storage functions for IoT devices. The edge servers can be various facilities or network infrastructures, *e.g.*, high-performance routers at the network edge, base stations, PCs, portable laptops and so on.

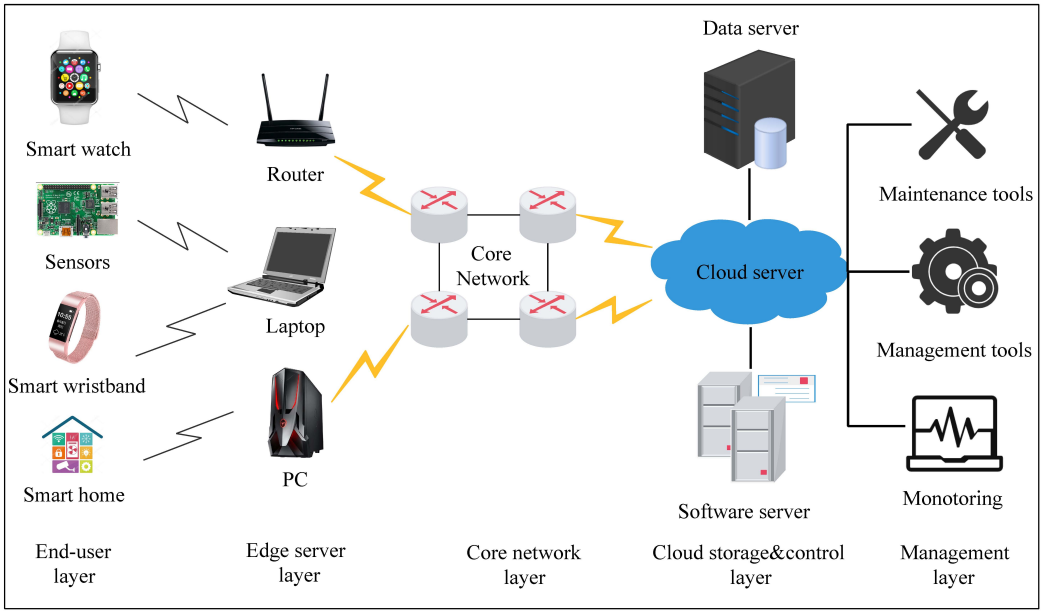

Fig. 7. A scalable framework for IoT based on TC

3) Core network layer This layer is the bridge between the edge server layer and the cloud storage/control layer. It is in charge of establishing connections and forwarding requests. Since the rapid growth if IoT devices in recent years have brought huge pressure on the core network, how to eliminate the pressure on this layer has become the focus of the research.

4) Cloud layer This layer consists of many servers with abundant computing and storage resources. For example, the data servers are used to store and process user data, while the software servers store heterogeneous OSes and application program files. Besides, the cloud layer can work with the second layer to provision delay-sensitive and context-aware services for end-users.

5) Management layer This layer mainly provides management appliances to manipulate the whole IoT platform, as well as the maintenance of software and monitoring the operating status of the entire platform.

In Ref.[32], Ren *et al.* developed a TC-based wearable product called TCwatch, which can be applied in various domains, such as body area networks, electronic medical, smart education, *etc.* Fig.8 illustrates the framework for the TC-based wearable platform. It includes TCwatch, a mobile phone, and a high-performance PC. TCwatch interacts with the mobile phone through the bluetooth low energy protocol, while the mobile phone links to the cloud via WiFi. They design a lightweight OS which plays the same role as the Meta OS, called LiteTCOS, to remotely fetch resources from mobile phones. And mobile phones can provide computing and data storage services for TCwatch. The cloud server deposits all the software and data and is responsible for the development and updating of new version Apps for TCwatch to access. If the requested resources are not stored in the mobile phones, the TCWatch will send requests to the cloud server. In addition, the TCwatch can offload the computation-intensive tasks to the cloud. The authors describe the processes of dynamically carrying out a service on TCwatch in detail in the literature. And the evaluation results show that TCwatch can efficiently realize flexible service provisioning compared with traditional smartwatches. Moreover, Yi *et al.*[87] developed a similar scalable system for wearable devices named TC-based intelligent devices (TCID), which exploited TC technology to improve the terminals' scalability.

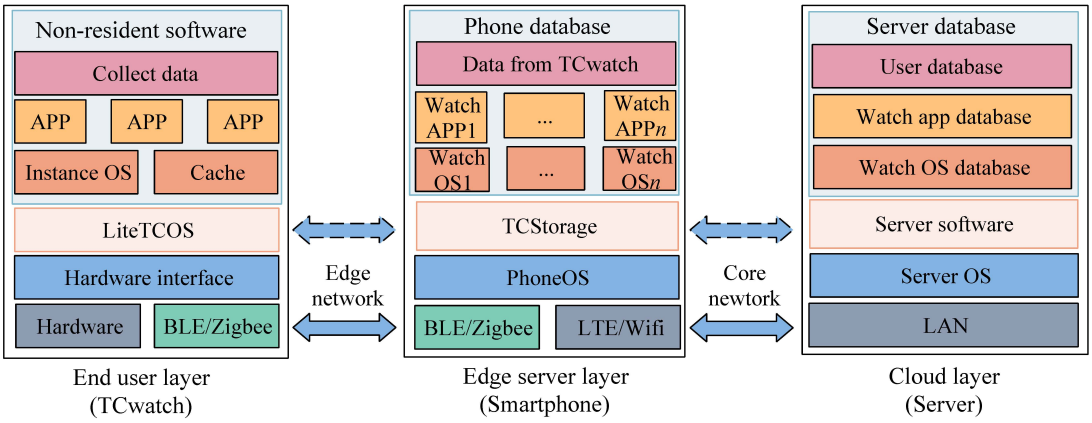

Fig. 8. The framework for TC-based wearable platform

V. Block-Streaming as a Service: The Extension and Application of TC

With the wide application of lightweight mobile and IoT devices in recent years, the user demands for real-time and diversified service have increased significantly. The cloud-centric computing mode is difficult to provide context-aware and delay-sensitive service requirements for lightweight IoT devices due to high round-trip latency. In addition, lightweight devices are often resource-constrained. Traditional approaches have limitations on flexible service provisioning, such as pre-installing applications on IoT devices, or in a manner that packaging all system software and programs as a whole. Actually, full app delivery is not always necessary because a user may not use all of the functions of a certain app. Besides, apps do not need to load all content available in an app package to offer a single functionality[88]. The experimental results in Ref.[19] showed that most of the users often spend 80% of the time on the 10% functions, and the remaining 90% codes are idle. The analysis using 22 of 70 activities in the Six Flags app indicates that a user can get all relevant info about a particular water park using only 5.4% of the apk. If the clients only download the necessary part of code blocks for service provisioning instead of the entire programs, the QoS and system performance will be greatly enhanced. To better understand the motivation of BaaS, we provide an illustration. Let us take an example of Six Flags app on Google Player. The size of the entire app is 57MB, including 70 activities such as maps, tickets, parking, cart, *etc.* If a user who does not need to park and only wants to order lunch in the restaurant, he does not need to download and install the total app on his smartphone.

Therefore, we need a new service model to provide block-stream services for resource-constrained lightweight devices through the wireless network. He *et al.*[29] proposed BaaS on the basis of the TC concept. Rather than pushing the whole application to devices, BaaS takes the code block of an application as a unit of service and loads them to the terminal in a block streaming manner according to the user's request. After execution, the blocks can be decommissioned and removed if it expires. In such a way, we do not need to maintain the latest version of the code block or install security patches on the terminal. And the code block can also be customized based on the usage history and the capabilities of devices.

Compared with the application store, BaaS is suitable for providing flexible and adaptive services for IoTs and mobile devices. It also can be combined with CC, EC and other computing paradigms to make network computing more secure and efficient.

1. The architecture of BaaS

We present the BaaS architecture in Fig.9. As shown in the figure, the BaaS works in a client-server model. The server works as a warehouse that maintains various kinds of code blocks, including App, middleware, library, and OS. The underlying block management engine is responsible for managing the execution, maintenance, and update of all code blocks, and defining the block streaming load policies, such as choosing the execution environment, how the server interacts with the terminals, determine which app blocks execute in which devices and so on.

BaaS can divide an application into multi-code blocks according to their functions and dependency relationship, each contains one or more microservices. In the beginning, the clients are generally bare metal with the Meta OS layer and some booting protocols. When the clients request certain microservices from the server, the servers will discover the hardware types and service content, then decide which blocks will be loaded to the clients by block management engine on the server. After obtaining the required blocks, the Meta OS on the terminal device rebuild the software stack and activate itself. The code blocks loaded from the server can be deleted immediately after use or keep them on the client for a while. If it needs to update the application programs in runtime, it will choose the new blocks to add in the software stack and remove the old-version blocks. Therefore, BaaS can adapt to different sizes of devices in the IoTs and mobile Internet environment.

In addition, with the purpose of load balancing, BaaS allows two duplicates of the same block to perform simultaneously on both client and server. For instance, the block PB1 can be executed simultaneously on the client and environment on the server, and collaborate with PB4 on the client at the same time to finish the complete App execution. Furthermore, BaaS is not limited to the client and server, but can also be applied among clients. By this means, each device deployed in the network environment can contribute its idle computing power, which fully reflects the concept of TC and persuasive computing.

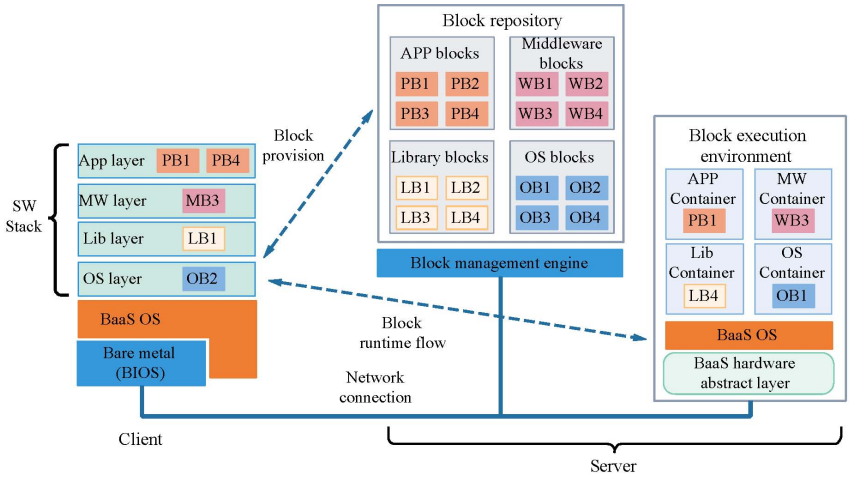

Fig. 9. The BaaS architecture

2. The implementation of BaaS

BaaS is a promising approach to provide dynamic service and secure code management for lightweight mobile and IoT devices. However, there are many challenges in the implementation of BaaS. First, because different elements of a program are correlative dependence from each other and hard to be performed separately, how to correctly partition the code blocks of the application so that they can be executed on the lightweight devices is a challenge. Second, the local request and I/O interrupts will be redirected to the server via the network, how to study a reliable I/O virtualization technique for BaaS is another challenge. Third, how does the software stack of clients organize code blocks into usable functions, and differentiate which hardware environment the different blocks execute in should also be considered? Furthermore, for the server-side, how to manage the dependency of code blocks, as well as block upgrades, classifying, adding, removing and transmission security. Finally, how to achieve the load balance among servers and clients. This problem needs to consider the current network conditions, processors loads on both sides, power consumption and so on.

To solve these problems, Peng *et al.*[19] proposed BOAT, a block streaming program execution strategy for TC, to sustain efficient service provisioning for lightweight

terminals. BOAT can partition code automatically and divide the entire program into many function blocks based on executable and link files. Furthermore, a novel I/O virtualization method and a relocation mechanism are also proposed. Fig.10(a) shows the module diagram of BOAT, the APP managers administers the app life cycle. In particular, the *Block manager* is used to decide the residence time of the code blocks on the client according to the user preference, network condition and so on. The Block loader fetches the blocks into memory and plans for execution at runtime, where the *Relocation module* is applied to perform relocation operations. The Block detector is used to detect whether the client has a certain block. If the block is not stored on the client, a "miss" occurs, otherwise a "hit" occurs. The Virtual I/O protocol module uses suitable network protocols to obtain the requested resources from servers. The Block generator is in charge of generating blocks according to the users' requirements, where the *App partitioner* dynamically separates the APK into blocks. However, due to the dependency of blocks from each other, the suffixes for instructions and function modules are named .text, each code block also requires a corresponding resource file, *i.e.*, res. Therefore, the *App partitioner* only separates .text and res into blocks as required. The divided blocks of .text and res are called .textlet and reslet, respectively.

As illustrated in Fig.10(b), an app execution procedure includes two stages, *i.e.*, initialization and block-streaming execution. Specifically, when a software starts up, the key blocks are firstly loaded to a different location of the client, including the ELF-Head, Section header table, the .data, .bss, .rodata files and other auxiliary data. During the block-streaming execution phase, the necessary .textlet and reslet are loaded on demand. Then the block detector on the client will detect whether a block is saved locally. For example, the .textlet 4 and reslet are required. Because both of them are not stored at the local device, two virtual I/O requests are then produced and redirected to the server. The server captures the requests, performs blocks retrieve and other block preparation works, finally schedules the required blocks to clients. The evaluation outcomes prove that the designed mechanism can effectively enhance the performance of lightweight terminals with limited resources.

3. The theoretical modeling of BaaS

Besides, there are several research works performing theoretical modeling for BaaS. Zhang *et al.*[89] utilized model estimation, dual learning and block serving techniques to study the proactive service model. The response delay was reduced by proactively streaming the blocks to clients. Shen *et al.*[90] designed a proactive service provisioning approach. This approach can predict the requirements of code blocks and requires lightweight devices to pre-fetch blocks so as to decrease loading delay. The experiment results demonstrated the performance of the method. Wang *et al.*[91] studied the privacy problems on the Internet of vehicles (IoVs) where data sharing is the premise. However, the drivers need to protect information privacy while sharing with others, but existing protocols have deficiencies regarding security, efficiency, and availability. The authors proposed a secure lightweight RFID (Radio frequency identification devices) mutual authentication protocol based on TC and synchronous secret information to protect the privacy of the drivers. Besides, it also adopted the OS-level monitoring and attacked tracking technology to further improve system security, real-time, and scalability.

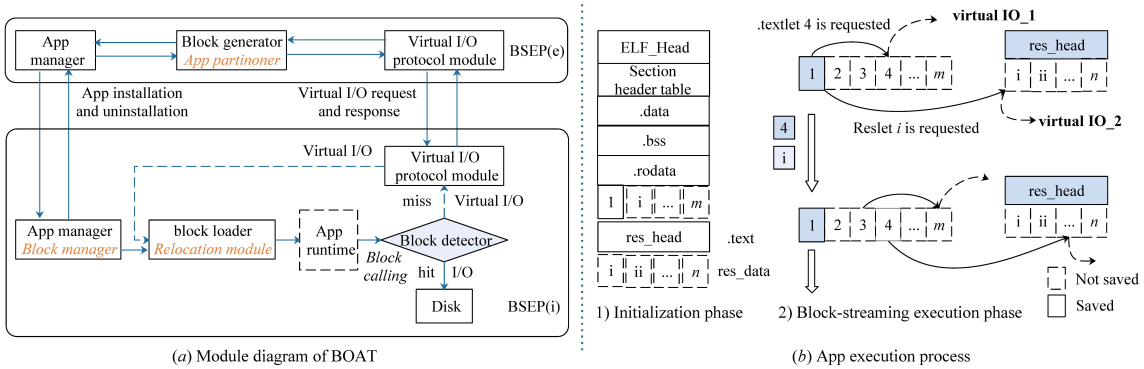

Fig. 10. The module diagram of BOAT and app execution process

VI. New Challenges and Future Directions

Over the past decades, TC has been widely applied in the PC, mobile devices and IoT terminals. Despite the existing studies have performed extensive investigations on TC from various perspectives, TC still faces many challenges and merit further researches to drive the development of smart applications in

Artificial intelligence (AI), as well as the advance of emerging network computing paradigms. In this section, we introduce some new challenges and potential research directions.

1. Big data collection and analysis for transparent computing

The number of IoT and desktop devices from different cyber-physical systems have increased dramatically and produced a huge volume of data. In TC paradigm, the task computing is performed at the front-end devices. Due to the heterogeneity of these hardware devices and software applications, data diversity (*e.g.*, temperature, sound, image or video, *etc.*) brings difficulties for data collection. In addition, the data collected from transparent terminals contain some incomplete information or error data. Therefore, how to efficiently collect heterogeneous data and improve the reliability of data collection is a potential challenge. The second challenge is big data analysis. TC is well suited for applications that use lightweight terminal devices, such as mobile action recognition, speech recognition, and autonomous driving. The streaming data generated in these application scenarios is usually highly unstructured and semi-structured and requires real-time collection and processing, which is hard to be processed and analyzed with the traditional relational database. Moreover, the data from large-scale wireless networks are usually spatially and temporally correlated. Thus, how to establish an IoT big data platform based on TC to manage and extract valuable information from data collected by transparent terminals, and provide an open API interface to ensure that the IoT terminal devices from various fields can be safely accessed is a very important research direction.

2. Cross-platform integration for transparent computing

Currently, various heterogeneous terminals coexist in the TC paradigm, including PCs, mobile/IoT terminals. To provision multi-platform and on-demand services for these terminals, one promising method is to adopt cross-platform integration to fuse the data, resources, functions, and applications. Although some emerging OSes propose to integrate heterogeneous resources to perform unified management for IoT devices, such as Fuchsia OS[92], Harmony OS[93]. None of them can be compatible with embedded devices, smartphones, and PCs simultaneously. To realize cross-platform integration for the TC paradigm, there is a significant need for developing flexible APIs. A well-designed API comprises communication protocols, interaction interfaces, and tools for establishing different systems or platform connections. Besides, the APIs designed for the seamless connectivity between resource-constrained mobile/IoT devices and compute-intensive edge or cloud servers need to consider their capabilities and resource capacities. However, because of the dynamics of resource changes, the design of cross-platform integration APIs faces some challenges. Firstly, establishing adaptive connections between mobile/IoT devices and distributed servers under hybrid system architectures. Secondly, designing suitable cross-platform integration technologies that can utilize and coordinate heterogeneous resources for managing data and services. In addition, the data from different platforms and devices are usually varied structures. Thus, further researches are needed for the integration and analysis of heterogeneous data from different platforms.

3. Security and privacy preservation mechanisms for transparent computing

Although TC has shown its capability to enhance system security by supervising the system defects and malicious attacks. From the perspective of end devices, IoT and smart mobile devices may face abnormal intrusion and various kinds of attacks in complex network environments. However, due to the constrained computing capabilities, limited storage space and battery power, the traditional security mechanism cannot provide reliable protection for the transparent terminals. It is a promising research direction to design lightweight encryption schemes and privacy preservation models for mobile and IoT devices. Another approach is to use edge servers as auxiliary devices to provide enhanced security for transparent terminals. For example, the edge nodes serve as agents and are in charge of access authentication and information encryption.

Furthermore, one possible concern is that the privacy stealer may infer from the individual privacy information which leads to the risk of privacy leakage, especially in the mobile TC architecture. And the conventional privacy-preserving data analysis methods are not applicable to the network data with big volume, heterogeneous structures, diverse types, and spatio-temporal correlations. Therefore, more efforts are needed for designing efficient and reliable privacy-preservation schemes according to the characteristics of real systems for addressing privacy problems, while considering the balance between privacy and efficiency.

4. Transparent computing service provisioning in high-speed network communication

High-speed network communication will provide high-bandwidth, ultra-reliable, and low latency services for TC application scenarios. Such as 5G, a new cellular mobile communication technology that can provide better mobile performance with millisecond delay and ultra-high density connections. It becomes more faster for TC terminals to fetch resources from servers. Therefore, how to ensure reliable data transmission for

TC under future high-speed network is an open research issue. Moreover, we live in a heterogeneous network environment, including 5G, 4G, WiFi, future 6G and so on, each of them utilizes its characteristics to provide reliable services for transparent end devices. So how to design adaptive data transmission protocol and efficient network switch mechanisms is a potential and promising direction.

In addition, the concept of TC has been applied in many fields, such as cloud services[26], wearable devices[32], smart education[94], industry IoTs[95,96] and IoVs[91,97]. However, the high-speed network communication architecture will bring new development potential to existing delay-sensitive applications, such as VR/AR, unmanned cars, smart cities and health care (*e.g.*, telesurgery). Therefore, how to integrate 5G and TC paradigm will empower existing applications or generate new application scenarios merits further investigation.

VII. Conclusions

In this paper, we conducted a comprehensive survey for a promising computing paradigm called TC that can decouple the software from the heterogeneous hardware of terminals to achieve unified management of resources. Specifically, we first introduced the development process of network computing. We then introduced the motivation of TC and compared the differences among CC, EC, and TC paradigm from three perspectives, *i.e.*, virtualization, location for computing, location for storage. Furthermore, we outlined the basic TC architecture and several key technologies for TC. Beyond that, we thoroughly surveyed the development process and several variant architectures based on TC, including TC for PCs, TC for lightweight mobile and IoT devices. Lastly, we summarized several new challenges and future directions for continuous investigations on TC. We hope this survey is able to stimulate fruitful discussions and inspire further research on TC.

References

[1] G. Sven, P. Jim and S. Sharad, "Making the utility data center a power station for the enterprise grid", *Technical Report*, HPL-2003-53, HP Laboratories, Vol.131, 2003.

[2] Amazon, "Announcing amazon elastic compute cloud (amazon ec2)-beta", *https://aws.amazon.com/cn/about-aws/whats-new/2006/08/24/ announcing-amazon-elastic-compute-cloud -amazon-ec2—beta*, 2006-08-24.

[3] Y.X. Zhang, J. Ren, J.G. Liu, *et al.*, "A survey on emerging computing paradigms for big data", *Chinese Journal of Electronics*, Vol.26, No.1, pp.1–12, 2017.

[4] Cisco Systems, "Cisco visual networking index: Global mobile data traffic forecast update, 2017–2022 White Paper", *https://www.cisco.com/c/en/us/solutions/collateral/service -provider/visual-networking-index-vni/white-paper-c11-7384 29.html*, 2020-05-09.

[5] S. Zhang, Y.X. Wu, C.G. Men, *et al.*, "Tiny YOLO optimization oriented bus passenger object detection", *Chinese Journal of Electronics*, Vol.29, No.1, pp.132–138, 2020.

[6] Y.K. Zhang, P.Y. Zhang and Y.H. Yan, "Language model score regularization for speech recognition", *Chinese Journal of Electronics*, Vol.28, No.3, pp.604–609, 2019.

[7] Z.Y. Guo, W.J. Yang, M.L. Li, *et al.*, "ALLIANCE-ROS: A software framework on ROS for fault-tolerant and cooperative mobile robots", *Chinese Journal of Electronics*, Vol.27, No.3, pp.467–475, 2018.

[8] S.D. Liu, Y. Wu, Y.M. Ji, *et al.*, "Research on security of key algorithms in intelligent driving system", *Chinese Journal of Electronics*, Vol.28, No.1, pp.33–42, 2019.

[9] T.E. Bogale and L.B. Le, "Massive MIMO and mmWave for 5G wireless HetNet: Potential benefits and challenges", *IEEE Vehicular Technology Magazine*, Vol.11, No.1, pp.64–75, 2016.

[10] J. Ren, D.Y. Zhang, S.W. He, *et al.*, "A Survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet", *ACM Computing Surveys* (*CSUR*) , Vol.52, No.6, Article No.125, 2019.

[11] Y.X. Zhang and Y.Z. Zhou, "Transparent computing: A new paradigm for pervasive computing", *International Conference on Ubiquitous Intelligence and Computing*, Wuhan, China, pp.1–11, 2006.

[12] Y.X. Zhang and Y.Z. Zhou, "From resource sharing to service sharing: Opportunities and challenges of transparent computing", *Engineering Sciences*, Vol.11, No.8, pp.10-17, 2009. (in Chinese)

[13] M. Roberto, K. Jimmy and K. Miika, "Hypervisors vs. lightweight virtualization: A performance comparison", *2015 IEEE International Conference on Cloud Engineering*, Tempe, USA, pp.386–393, 2009.

[14] B.I. Ismail, E.M. Goortani and E. Ehsan, "Evaluation of docker as edge computing platform", *2015 IEEE Conference on Open Systems*, Melaka, Malaysia, pp.130–135, 2015.

[15] Y. Wang, W. Shi and M.L. Hu, "Virtual servers co-migration for mobile accesses: Online versus off-line", *IEEE Transactions on Mobile Computing*, Vol.14, No.12, pp.2576–2589, 2015.

[16] B. David, "Containers and cloud: From LXC to docker to kubernetes", *IEEE Cloud Computing*, Vol.1, No.3, pp.81–84, 2014.

[17] Y.X. Zhang and Y.Z. Zhou, "Transparent computing: Spatio-temporal extension on von Neumann architecture for cloud service", *Tsinghua Science and Technology*, Vol.18, No.1, pp.10–21, 2013.

[18] A. Nasir, Y. Zhang, T. Amir, *et al.*, "Mobile edge computing: A survey", *IEEE Internet of Things Journal*, Vol.5, No.1, pp.450–465, 2017.

[19] X.H. Peng, J. Ren, L. She, *et al.*, "BOAT: A block-streaming app execution scheme for lightweight IoT devices", *IEEE Internet of Things Journal*, Vol.5, No.3, pp.1816–1829, 2018.

[20] P. Mell and T. Grance, "The NIST definition of cloud computing", *NIST Special Publication* 800-145, Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, 2011.

[21] C.F. Jiang, X.L. Cheng, H.H. Gao, *et al.*, "Toward computation offloading in edge computing: A survey", *IEEE Access*, Vol.7, pp.131543–131558, 2019.

[22] Y.X. Zhang, "Transparence computing: Concept, architecture and example", *Acta Electronica Sinica*, Vol.32, No.12A, pp.169-174, 2004. (in Chinese)

[23] Y.X. Zhang and Y.Z. Zhou, "4VP: A novel meta OS approach for streaming programs in ubiquitous computing", *21st International Conference on Advanced Information Networking and Applications*, Niagara Falls, Canada, pp.394–403, 2007.

[24] J.Y. Wu, L.D. Ping, X.P. Ge, *et al.*, "Cloud storage as the infrastructure of cloud computing", *2010 International Conference on Intelligent Computing and Cognitive Informatics*, Kuala Lumpur, Malaysia, pp.380–383, 2010.

[25] Y. Wu, J. Yang, R.P. Liu, *et al.*, "Survey on approximate storage techniques", *Journal of Computer Research and Development*, Vol.55, pp.2002–2015, 2018.

[26] Y.X. Zhang and Y.Z. Zhou, "TransOS: A transparent computing-based operating system for the cloud", *International Journal of Cloud Computing*, Vol.1, No.4, pp.287–301, 2012.

[27] TransOS, "An operating system in the cloud: TransOS could displace conventional desktop operating systems", 2012-10-09.

[28] M. Wu, "Analysis and a case study of transparent computing implementation with UEFI", *International Journal of Cloud Computing*, Vol.1, No.4, pp.312–328, 2012.

[29] J. He, Y.X. Zhang, J. Lu, *et al.*, "Block-stream as a service: A more secure, nimble, and dynamically balanced cloud service model for ambient computing", *IEEE Network*, Vol.32, No.1, pp.126–132, 2018.

[30] Y.Z. Zhou, B.W. Yang, C. Wu, *et al.*, "Mobile transparent computing: A novel user-centric approach to unify device, edge, and cloud", *IEEE Network*, Vol.33, No.2, pp.132–137, 2019.

[31] H. Guo, J. Ren, D.Y. Zhang, *et al.*, "A scalable and manageable IoT architecture based on transparent computing", *Journal of Parallel and Distributed Computing*, Vol.118, pp.5–13, 2018.

[32] J. Ren, H. Guo, C.G. Xu, *et al.*, "Serving at the edge: A scalable IoT architecture based on transparent computing", *IEEE Network*, Vol.31, No. 5, pp.96–105, 2017.

[33] W.Y. Kuang, Y.X. Zhang, Y.Z. Zhou, *et al.*, "NSAP: A network storage access protocol for transparent computing", *Journal of Tsinghua University (Science and Technology)*, Vol.49, No.1, pp.106–109, 2009. (in Chinese)

[34] Y.X. Zhang, K.H. Guo, J. Ren, *et al.*, "Transparent computing: A promising network computing paradigm", *Computing in Science and Engineering*, Vol.19, No.1, pp.7–20, 2017.

[35] Y.Z. Zhou, Y.X. Zhang, Y. Wang, *et al.*, "A customizable boot protocol for network computing", *Journal of Software*, Vol.14, No.3, pp.538–546, 2003. (in Chinese)

[36] X. Wang, N. Xia, H. Yang, *et al.*, "A network storage protocol based operating system remote boot mechanism", *Computer Engineering and Applications*, Vol.42, No.20, pp.95–97/212, 2006. (in Chinese)

[37] H.J. Yang, Y.X. Zhang, X.H. Wang, *et al.*, "MRBP2: A transparence computing based remote booting protocol", *Mini-Micro Systems*, Vol.27, No.9, pp.1657–1660, 2006. (in Chinese)

[38] G.F. Guo, Y.X. Zhang, Y.Z. Zhou, *et al.*, "Performance modeling and analysis of the booting process in a transparent computing environment", *2008 Second International Conference on Future Generation Communication and Networking*, Hainan, China, pp.83–88, 2008.

[39] W. Li, G.B. Xu, Y.X. Zhang, *et al.*, "ENCBP: An extended multi-OSs remote-booting method", *Journal of Computer Research and Development*, Vol.46, No.6, pp.905–912, 2009. (in Chinese)

[40] S. Li, Y.Z. Zhou and Y.X. Zhang, "NSAP+: Supporting transparent computing applications with a service-oriented protocol", *Computing in Science and Engineering*, Vol.19, No.1, pp.21–28, 2017.

[41] N. Xia, Y.X. Zhang and S.L. Yang, "IOMan: An I/O management method supporting multi-OS remote boot and running", *Computer Research and Development*, Vol.44, No.2, pp.317–325, 2007. (in Chinese)

[42] H.L. Tan, Y. Wang and J.H. Sim, "A dynamic rebuild strategy of multi-host system volume on transparence computing mode", *10th IEEE International Conference on High Performance Computing and Communications*, Dalian, China, pp.981–986, 2008.

[43] Y. Gao, Y.X. Zhang and Y.Z. Zhou, "Building a virtual machine-based network storage system for transparent computing", *2012 International Conference on Computer Science and Service System*, Nanjing, China, pp.2341–2344, 2012.

[44] Y. Gao, Y.X. Zhang and Y.Z. Zhou, "Performance analysis of virtual disk system for transparent computing", *2012 9th International Conference on Ubiquitous Intelligence and Computing and 9th International Conference on Autonomic and Trusted Computing*, Banff, Canada, pp.470–477, 2012.

[45] Y. Gao, Y.X. Zhang and Y.Z. Zhou, "TVSDM: A virtual storage device driver model for transparent computing", *J Tsinghua Univ* (*Sci & Technol*), Vol.53, No.7, pp.1064–1068, 2013. (in Chinese)

[46] Y.Z. Zhou, Y.X. Zhang and Y.L. Xie, "TransCom: A virtual disk-based cloud computing platform for heterogeneous services", *IEEE Transactions on Network and Service Management*, Vol.11, No.1, pp.46–59, 2014.

[47] G.B. Xu, W.Y. Kuang and Y.Z. Zhou, "CSDT: Method for data transfer between client devices of transparent computing systems", *Computer Engineering*, Vol.34, No.17, pp.1–3, 2008. (in Chinese)

[48] L. Wei, Y.X. Zhang and Y.Z. Zhou, "Simulation analysis and validation of cache performance in transCom systems", *J Tsinghua Univ* (*Sci & Technol*), Vol.49, No.10, pp.1700-1703, 2009. (in Chinese)

[49] C.H. Tan, L. Yang, J.D. Wen, *et al.*, "Transparent computing system based on hierarchical cache", *Computer Engineering*, Vol.37, No.5, pp.270–272, 2011. (in Chinese)

[50] Y. Gao, Y.X. Zhang and Y.Z. Zhou, "A cache management strategy for transparent computing storage system", *International Conference on Trustworthy Computing and Services*, Beijing, China, pp.651–658, 2012.

[51] J.Z. Liu, Y.Z. Zhou and D. Zhang, "Transim: A simulation framework for cache-enabled transparent computing systems", *IEEE Transactions on Computers*, Vol.65, No.10, pp.3171–3183, 2016.

[52] D. Zhang, Y.Z. Zhou and Y.X. Zhang "A multi-level cache framework for remote resource access in transparent computing", *IEEE Network*, Vol.32, No.1, pp.140–145, 2018.

[53] G.F. Guo and Y.Z. Zhou, "Block-based data consistency method for transparent computing", *J Tsinghua Univ* (*Sci & Technol*), Vol.49, No.10, pp.150–153, 2009. (in Chinese)

[54] Y. Gao, Y.X. Zhang and Y.Z. Zhou, "A remote resource management method for transparent computing", *2012 International Conference on Computer Science and Information Processing*, Xi'an, China, pp.1378–1381, 2012.

[55] G.B. Xu, Y.X. Zhang, Y.Z. Zhou, *et al.*, "Design and implementation of a virtual machine-based transparent computing system", *J Tsinghua Univ* (*Sci & Technol*), Vol.48, No.10, pp.1675–1678, 2008. (in Chinese)

[56] C.C. Chen, Y.X. Zhang, Y.Z. Zhou, *et al.*, "Lightweight virtu-

al machine-based transparent computing system", *Computer Engineering*, Vol.36, No.11, pp.39–41/44, 2010. (in Chinese)

[57] P.C. Du and X.Y. Li, "T-Netm: Transparent network monitoring on virtual machine", *2016 2nd International Conference on Cloud Computing and Internet of Things*, Dalian, China, pp.64–67, 2016.

[58] C. Wu and Y.X. Zhang, "Toward efficient transparent computing for IoT apps by on-chip kernel offload", *IEEE Internet of Things Journal*, Vol.6, No.4, pp.4085–4097, 2019.

[59] C. Xue and C. Lin, "Performance modelling for transparent computing using stochastic Petri nets", *International Journal of Ad Hoc and Ubiquitous Computing*, Vol.21, No.2, pp.81–94, 2016.

[60] H.J. Yang, Y.X. Zhang, Y.Z. Zhou, *et al.*, "A dynamic load balancing algorithm based on transparence computing", *Computer Engineering*, Vol.32, No.13, pp.133–135/163, 2006. (in Chinese)

[61] X.H. Wang, N. Wang and W. Li, "RTO algorithm for transparent computing systems", *J Tsinghua Univ* (*Sci & Technol*), Vol.47, No.10, pp.1696–1699, 2007. (in Chinese)

[62] X.H. Wang, N. Wang and W. Li, "Analysis on the scheduling problem in transparent computing", *2013 IEEE 10th International Conference on High Performance Computing and Communications and 2013 IEEE International Conference on Embedded and Ubiquitous Computing*, Zhangjiajie, China, pp.1832–1837, 2013.

[63] Y.J. Fang, Y.Z. Zhou and X.Y. Jiang, "A delay constrained two-hop relay algorithm for transparent computing in manets", *2013 IEEE 10th International Conference on High Performance Computing and Communications and 2013 IEEE International Conference on Embedded and Ubiquitous Computing*, Zhangjiajie, China, pp.2337–2341, 2013.

[64] K.H. Guo, Y.Y. Tang, J.H. Ma, *et al.*, "Optimized dependent file fetch middleware in transparent computing platform", *Future Generation Computer Systems*, Vol.74, pp.199–207, 2017.

[65] W.H. Zhang, H. Song and D.C. Wang, "Performance optimization of underlying operating system in transparent computing", *2016 IEEE International Conferences on Big Data and Cloud Computing, Social Computing and Networking, Sustainable Computing and Communications*, Atlanta, USA, pp.1–6, 2016.

[66] Y.Z. Zhou, W.J. Tang, D. Zhang, *et al.*, "Software-defined streaming-based code scheduling for transparent computing", *2016 International Conference on Advanced Cloud and Big Data*, Chengdu, China, pp.296–303, 2016.

[67] J.Z. Wang, A.F. Liu, T. Yan, *et al.*, "A resource allocation model based on double-sided combinational auctions for transparent computing", *Peer-to-Peer Networking and Applications*, Vol.11, No.4, pp.679–696, 2018.

[68] K.H. Guo, Z.H. Liang, R.H. Shi, *et al.*, "Transparent learning: An incremental machine learning framework based on transparent computing", *IEEE Network*, Vol.32, No.1, pp.146–151, 2018.

[69] F. Shan, J.Z. Luo, J.H. Jin, *et al.*, "Offloading delay constrained transparent computing tasks with energy-efficient transmission power scheduling in wireless IoT environment", *IEEE Internet of Things Journal*, Vol.6, No.3, pp.4411-4422, 2019.

[70] Y.Y. Tang, K.H. Guo and B. Tian, "A block-level caching optimization method for mobile transparent computing", *Peer-to-Peer Networking and Applications*, Vol.11, No.4, pp.711–722, 2018.

[71] Y.X. Zhang, L.T. Yang, Y.Z. Zhou, *et al.*, "Information security underlying transparent computing: Impacts, visions and challenges", *Web Intelligence and Agent Systems: An International Journal*, Vol.8, No.2, pp.203–217, 2010.

[72] W.Y. Kuang, Y.X. Zhang, Y.Z. Zhou, *et al.*, "RBIS: Security enhancement for MRBP and MRBP2 using integrity check", *Journal of Chinese Computer Systems*, Vol.28, No.2, pp.251–254, 2007. (in Chinese)

[73] G.J. Wang, Q. Liu, Y. Xiang, *et al.*, "Security from the transparent computing aspect", *2014 International Conference on Computing, Networking and Communications*, Honolulu, USA, pp.216–220, 2014.

[74] T. Peng, Q. Liu, G.J. Wang, *et al.*, "A multilevel access control scheme for data security in transparent computing", *Computing in Science and Engineering*, Vol.19, No.1, pp.46-53, 2016.

[75] H.W. Xue and Y.Q. Dai, "A privacy protection model for transparent computing system", *International Journal of Cloud Computing*, Vol.1, No.4, pp.367–384, 2012.

[76] J. Li, W. Yang and G.J. Wang, "A cross layer design for improving trivial file transfer protocol in mobile transparent computing", *2013 IEEE 10th International Conference on High Performance Computing and Communications and 2013 IEEE International Conference on Embedded and Ubiquitous Computing*, Zhangjiajie, China, pp.1872–1877, 2013.

[77] J. Li, W. Yang and G.J. Wang, "Wireless multicast for mobile transparent computing", *2013 IEEE 10th International Conference on High Performance Computing and Communications and 2013 IEEE International Conference on Embedded and Ubiquitous Computing*, Zhangjiajie, China, pp.1884–1889, 2013.

[78] Y.H. Xiong, S.Z. Huang and M. Wu, "Shared resource and service management for mobile transparent computing", *2013 IEEE 10th International Conference on High Performance Computing and Communications and 2013 IEEE International Conference on Embedded and Ubiquitous Computing*, Zhangjiajie, China, pp.1846–1853, 2013.

[79] Y.H. Xiong, S.Z. Huang and M. Wu, "A novel resource management method of providing operating system as a service for mobile transparent computing", *The Scientific World Journal*, Vol.2014, pp.1–12, 2014.

[80] H. Song, D.C. Wang, J.X. Wang, *et al.*, "TC-CCS: A cooperative caching strategy in mobile transparent computing system", *2017 IEEE Visual Communications and Image Processing* (*VCIP*), pp.1–4, 2017.

[81] J.H. Jin, J.Z. Luo, Y.H. Li, *et al.*, "COAST: A cooperative storage framework for mobile transparent computing using device-to-device data sharing", *IEEE Network*, Vol.32, No.1, pp.133–139, 2018.

[82] S.M. Guo, W. Yang and G.J. Wang, "NFS protocol performance analysis and improvement for mobile transparent computing", *2013 IEEE 10th International Conference on High Performance Computing and Communications and 2013 IEEE International Conference on Embedded and Ubiquitous Computing*, pp.1878–1883, 2013.

[83] W. Liang, Y.H. Xiong and M. Wu, "A cross platform computing method and its application for mobile device in transparent computing", *2013 IEEE 10th International Conference on High Performance Computing and Communications and 2013 IEEE International Conference on Embedded and Ubiquitous Computing*, pp.1838–1845, 2013.

[84] J.Q. Dong, H. Yin, H. Li, *et al.*, "TAM: A transparent agent architecture for monitoring mobile applications", *Computing in Science and Engineering*, DOI: 10.1109/MCSE.2016.48, 2016.

[85] K.H. Guo, Y.J. Huang, L. Kuang, *et al.*, "CASP: A context-aware transparent active service provision architecture in a mobile internet environment", *Computing in Science and Engineering*, Vol.19, No.1, pp.38–45, 2016.
[86] H. Guo, J. Ren, D.Y. Zhang, *et al.*, "A scalable and manageable IoT architecture based on transparent computing", *Journal of Parallel and Distributed Computing*, Vol.118, pp.5–13, 2018.
[87] L.T. Yi, J.B. Li, Y.X. Zhang, *et al.*, "Improving the scalability of wearable devices via transparent computing technology", *Computing in Science and Engineering*, Vol.19, No.1, pp.29–37, 2016.
[88] B. Ketan, S. Matt, J. Nikita, *et al.*, "Serving mobile apps: A slice at a time", *Proceedings of the Fourteenth EuroSys Conference 2019*, pp.1–15, 2019.
[89] D.Y. Zhang, R.Y. Shen, J. Ren, *et al.*, "Delay-optimal proactive service framework for block-stream as a service", *IEEE Wireless Communications Letters*, Vol.7, No.4, pp.598–601, 2018.
[90] R.Y. Shen, D.Y. Zhang, J. Ren, *et al.*, "Learning-aided proactive block provisioning in block-stream as a service for lightweight devices", *2018 IEEE International Conference on Communication Systems*, San Francisco, USA, pp.274–279, 2018.
[91] K. Fan, W. Wang, W. Jiang, *et al.*, "Secure ultra-lightweight RFID mutual authentication protocol based on transparent computing for IoV", *Peer-to-Peer Networking and Applications*, Vol.11, No.4, pp.723–734, 2018.
[92] "Fushia OS", *https://fuchsia.dev/*, 2020-4-2.
[93] "Harmony OS", *https://en.wikipedia.org/wiki/HarmonyOS*, 2020-1-4.
[94] K.H. Guo, Y.Z. Xiao and G.H. Duan, "A cost-efficient architecture for the campus information system based on transparent computing platform", *International Journal of Ad Hoc and Ubiquitous Computing*, Vol.21, No.2, pp.95–103, 2016.
[95] W.M. Li, B. Wang and J.F. Sheng, "A resource service model in the industrial IoT system based on transparent computing", *Sensors*, Vol.18, No.4, DOI: 10.3390/s18040981, 2018.
[96] R.Y. Shen, D.Y. Zhang, Y.M. Zhang, *et al.*, "A block prefetching framework for energy harvesting IoT devices", *IEEE Internet of Things Journal*, Vol.7, No.4, pp.3427–3440, 2020.
[97] D.Y. Zhang, L. Tan, J. Ren, *et al.*, "Near-optimal and truthful online auction for computation offloading in green edge-computing systems", *IEEE Transactions on Mobile Computing*, Vol.19, No.4, pp.880–893, 2020.

ZHANG Yaoxue was born in 1956. He received the B.S. degree from the Northwest Institute of Telecommunication Engineering, Xi'an, China, in 1982, and the Ph.D. degree in computer networking from Tohoku University, Sendai, Japan, in 1989. He is currently a Professor with the Department of Computer Science, Central South University, Changsha, China, and also a Professor with the Department of Computer Science and Technology, Tsinghua University, Beijing, China. He has published over 200 technical papers in international journals and conferences, as well as nine monographs and textbooks. His current research interests include computer networking, operating systems, ubiquitous/pervasive computing, transparent computing, and big data. Prof. Zhang is currently serving as the Editor-in-Chief of the Chinese Journal of Electronics. He is a Fellow of the Chinese Academy of Engineering. (Email: zhangyx@tsinghua.edu.cn)

DUAN Sijing was born in 1994. She is a Ph.D. candidate of School of Computer Science and Engineering, Central South University, Changsha, China. Her research interests include transparent computing and big data analysis. (Email: dsjyfd012@csu.edu.cn)

ZHANG Deyu (corresponding author) was born in 1987. He received the B.S. degree in communication engineering from PLA Information Engineering University, Zhengzhou, China, in 2005, the M.S. degree in communication engineering from Central South University, Changsha, China, in 2012, and the Ph.D. degree in computer science from Central South University. From 2014 to 2016, he was a Visiting Scholar with the Department of Electrical and Computer Engineering, University of Waterloo, ON, Canada. He is currently an Associate Professor with the School of Computer Science and Engineering. His current research interests include stochastic resource allocation in wireless sensor networks and cloud radio access networks, edge computing, and transparent computing. (Email: zdy876@csu.edu.cn)

REN Ju was born in 1987. He received the B.S. (2009), M.S. (2012), Ph.D. (2016) degrees all in computer science from Central South University, China. During 2013-2015, he was a visiting Ph.D. student in the Department of Electrical and Computer Engineering, University of Waterloo, Canada. Currently, he is a Professor with the School of Computer Science and Engineering, Central South University, China. His research interests include Internet-of-things, wireless communication, network computing and cloud computing. He is the recipient of the best paper award of IEEE IoP 2018 and IEEE ICC 2019, as well as the most popular paper award (2015-2018) of Chinese Journal of Electronics. He currently serves/served as an associate editor for IEEE Transactions on Vehicular Technology and Peer-to-Peer Networking and Applications, and a TPC member of many international conferences including IEEE INFOCOM19/18, Globecom17, WCNC17, WCSP16, etc. He also served as the TPC chair for IEEE BigDataSE'19, a poster co-chair for IEEE MASS'18, a track cochair for IEEE ICCC'19, I-SPAN'18 and IEEE VTC17 Fall, and an active reviewer for over 20 international journals. (Email: renju@csu.edu.cn)