BenchCouncil Transactions on Benchmarks, Standards and Evaluations 2 (2022) 100075

Contents lists available at ScienceDirect

BenchCouncil Transactions on Benchmarks, Standards and Evaluations

journal homepage: www.keaipublishing.com/en/journals/benchcouncil-transactions-on-benchmarks-standards-and-evaluations/

Research Article

High fusion computers: The IoTs, edges, data centers, and humans-in-the-loop as a computer

Wanling Gao [a,b], Lei Wang [a,b], Mingyu Chen [a,b], Jin Xiong [a,b], Chunjie Luo [a,b], Wenli Zhang [a,b], Yunyou Huang [c], Weiping Li [d], Guoxin Kang [a,b], Chen Zheng [e], Biwei Xie [a,b], Shaopeng Dai [a,b], Qian He [f], Hainan Ye [f], Yungang Bao [a,b], Jianfeng Zhan [a,b,*]

[a] State Key Lab of Processors, and Research Center for Advanced Computer Systems, Institute of Computing Technology, Chinese Academy of Sciences, No. 6 Kexueyuan South Road, Haidian District, 100190, Beijing, China
[b] University of Chinese Academy of Sciences, No. 19 (A) Yuquan Road, Shijingshan District, 100049, Beijing, China
[c] Guangxi Normal University, No. 15, Yucai Road, Qixing District, Guilin, 541010, Guangxi, China
[d] Civil Aviation Flight University of China, No. 46, Section 4, Nanchang Road, Guanghan, 618307, Sichuan, China
[e] Institute of Software, Chinese Academy of Sciences, No. 4, South 4th Street, Zhongguancun, 100190, Beijing, China
[f] Beijing Institute of Open Source Chip, No. 7 Caoqiao, Northwest Corner of Haidian Bridge, Haidian District, 100080, Beijing, China

ARTICLE INFO

Keywords:
Emerging and future applications
Safety-critical
Mission-critical
IoT
Edge
Data center
Humans-in-the-loop
Open-source computer systems
High Fusion Computers
HFC

ABSTRACT

Emerging and future applications rely heavily upon systems consisting of Internet of Things (IoT), edges, data centers, and humans-in-the-loop. Significantly different from warehouse-scale computers that serve independent concurrent user requests, this new class of computer systems directly interacts with the physical world, considering humans an essential part and performing safety-critical and mission-critical operations; their computations have intertwined dependencies between not only adjacent execution loops but also actions or decisions triggered by IoTs, edge, datacenters, or humans-in-the-loop; the systems must first satisfy the accuracy metric in predicting, interpreting, or taking action before meeting the performance goal under different cases.

This article argues we need a paradigm shift to reconstruct the IoTs, edges, data centers, and humans-in-the-loop as a computer rather than a distributed system. We coin a new term, high fusion computers (HFCs), to describe this class of systems. The fusion in the term has two implications: fusing IoTs, edges, data centers, and humans-in-the-loop as a computer, fusing the physical and digital worlds through HFC systems. HFC is a pivotal case of the open-source computer systems initiative. We laid out the challenges, plan, and call for uniting our community's wisdom and actions to address the HFC challenges. Everything, including the source code, will be publicly available from the project homepage: https://www.computercouncil.org/HFC/.

1. Introduction

The past decades have witnessed solid achievements and ambitious plans on planetary-scale infrastructures. Typical examples include but are not only limited to Grid computing [2], planet-scale data centers hosting internet services or called warehouse-scale computers [3], virtual supercomputers in the cloud [4], interplanetary Internet [5], networked systems of embedded computers [6], planet-scale computing networks [7,8] or planetary computer [9]. However, emerging and future applications raise daunting challenges beyond the reach of the state-of-the-practice systems.

Emerging and future applications rely heavily on systems consisting of IoTs, edges, data centers, and humans-in-the-loop [10]. These networked systems of embedded computers [6] or IoTs, collaborating with data centers, edges, and humans-in-the-loop, can radically change

* Correspondence to: Institute of Computing Technology, Chinese Academy of Sciences, No. 6 Kexueyuan South Road, Haidian District, 100190, Beijing, China
E-mail addresses: gaowanling@ict.ac.cn (W. Gao), wanglei_2011@ict.ac.cn (L. Wang), cmy@ict.ac.cn (M. Chen), xiongjin@ict.ac.cn (J. Xiong), luochunjie@ict.ac.cn (C. Luo), zhangwl@ict.ac.cn (W. Zhang), huangyunyou@gxnu.edu.cn (Y. Huang), weiping2012c@gmail.com (W. Li), kangguoxin@ict.ac.cn (G. Kang), zhengchenjason@hotmail.com (C. Zheng), xiebiwei@ict.ac.cn (B. Xie), daishaopeng@ict.ac.cn (S. Dai), heqian@bosc.ac.cn (Q. He), yehainan@bosc.ac.cn (H. Ye), baoyg@ict.ac.cn (Y. Bao), zhanjianfeng@ict.ac.cn (J. Zhan).

[1] According to [1] a safety-critical system is "a system whose failure may result in injury, loss of life or serious environmental damage, e.g., a control system for a chemical manufacturing plant". https://ifs.host.cs.st-andrews.ac.uk/Books/SE9/Web/Dependability/CritSys.html

[2] According to [1] a mission-critical system is "a system whose failure may fail some goal-directed activity, e.g., a navigational system for a spacecraft"; a business-critical system is "a system whose failure may result in very high costs for the business using that system, e.g., customer accounting system in a bank". https://ifs.host.cs.st-andrews.ac.uk/Books/SE9/Web/Dependability/CritSys.html

https://doi.org/10.1016/j.tbench.2022.100075
Received 20 September 2022; Received in revised form 26 October 2022; Accepted 26 October 2022
Available online 28 October 2022

the way people interact with the physical world and perform safety-critical,[1] or mission-critical,[2] tasks. This new class of systems appears in many forms and continues to expand: "implemented as a kind of digital nervous system to enable instrumentation of all sorts of spaces, ranging from in situ environmental monitoring to surveillance of battlespace conditions" [6], e.g., climate change monitoring and defense systems; embodied as integrated instrumentation, operation, maintenance, and regulation facilities of critical physical infrastructure, e.g., industrial digital twin, energy infrastructure management and civil aviation regulation; "employed in personal monitoring strategies (both defense-related and civilian), synthesizing information from sensors on and within a person with information from laboratory tests and other sources" [6], e.g., medical emergency applications; augmented as an extension of the real-world life for entertainment, education, and social activities, e.g., Metaverse; instrumented as a kind of digital sensing and autonomic control systems to perform safety-critical or mission-critical tasks, e.g., automotive driving and interplanetary explorations.

Significantly different from warehouse-scale computers that non-stop serve independent concurrent user requests [3], the new class of computer systems has three unique requirements. First, they directly interact with the physical world — considering humans an essential part: human-in-the-loop, performing safety-critical and mission-critical operations, and a significant fraction of actions may have an irreversible effect. The role of humans and their interactions with the other system components cannot be ignored in the final impact on the physical world. Second, unlike Internet services that process independent concurrent requests across data centers, their computations have intertwined dependencies between not only adjacent execution loops (which we call internal dependencies) but also actions or decisions triggered by IoTs, edge, data centers, or humans-in-the-loop (which we call external dependencies), and traverse different paths through and around IoTs, edges, and data centers. Third, under this highly entangled state, the systems must first satisfy the quality metric before meeting the performance goal under different conditions, like worst-case, average-case, and best-case. The quality metric measures the accuracy of an application, task, or algorithm in predicting, interpreting, or taking action.

Even considering a simple IoT application — 95% queries reporting the status and 5% user queries accessing the database, serving 10-billion devices needs about one thousand to one hundred thousand nodes under the threshold of 50 ms response time. Further, considering the three unique system requirements mentioned above and the sea change in computing, data access, and networking patterns, this new class of computer systems demand resources that are several orders of magnitude beyond the reach of the state-of-the-practice systems, which raises daunting challenges.

This article argues that we need a paradigm shift to rebuild the IoTs, edges, data centers, and humans-in-the-loop as a computer rather than a distributed system. We coin a new term, high fusion computers (HFCs), to describe this new class of computer systems. According to the Oxford English Dictionary, fusion means "the process or result of joining two or more things together to form a single entity". Fusion in HFCs has two-fold meanings: fusing IoTs, edges, data centers, and humans-in-the-loop as a computer, fusing the physical world and digital world. Fig. 1 shows a concept viewpoint of HFCs. Our intuition in rebuilding the HFC systems is simple: We explicitly value the role of humans-in-the-loop in different contexts and consider them as essential components of the system; we aggressively embrace co-design from vertical and horizontal dimensions, and will co-explore the design space from the algorithms, runtime systems, resource management, storage, memory, networking, and chip systems from a vertical dimension; Meanwhile, we will consider the close collaboration among IoTs, Edges, data centers, and humans-in-the-loop from a horizontal dimension, and ponder how to facilitate the interactions of humans-in-the-loop with other hardware and software systems.

To dismantle the complexity of building the systems and improve the efficiency, we use the funclet abstraction, architecture, and methodology, inspired by the philosophy of building large systems out of smaller functions [11,12]. The funclet abstract represents "the common proprieties of basic building blocks: each funclet is a well-defined, independently deployable, testable, evolvable, and reusable functionality with modest complexity; funclets interoperate with each other through well-defined interconnections" [12]. Four kinds of funclets form the four-layer funclet architecture: chiplet, HWlet, envlet, and servlet — at the chip, hardware, environment management, and service layers, respectively [12].

We take HFC as a pivotal example of the open-source computer system plan. We abstract reusable functions (funclets) across system stacks among IoTs, edges, and data centers. Based on funclets, we rebuild the IoTs, edges, data centers, and humans-in-the-loop as a computer in a structural manner, with full-fledged functions of autonomic resource discovery, management, programming, workload scheduling, and coordinated collaboration between software, hardware and human components. Our plans are three-fold. First, we value the importance of benchmarks and funclet-based standards in evaluating and building the systems. Second, we emphasize the methodology and tool to facilitate the workload-driven exploration of the system and architecture design space. Third, we will provide the first open-source implementation of the funclet architecture of HFC systems.

We organize the rest of this paper as follows. Section 2 explains the motivation. Section 3 illustrates the HFC challenges. Section 4 explores the HFC software and hardware design space. Section 5 describes our plan. Section 6 summarizes the related work. Section 7 concludes.

2. Motivation

In this section, we first analyze seven typical emerging and future applications' unique requirements, then explain why we need to build an HFC system.

2.1. The requirements of emerging and future applications

This subsection analyzes seven emerging and future applications. Table 1 characterizes those applications. I detail two applications specifically as follows.

2.1.1. Medical emergency management

According to the data from the World Bank, there are more than 723 million people over the age of 65 in the world in 2020, accounting for 9.321% of the world's total population [14]. What is worse, despite the slowdown in world population growth, the proportion of people over the age of 65 is growing rapidly, which will account for 16% of the total population by 2050 [15,16]. Due to the decline of physical function and pathological changes, the elderly will experience many unplanned emergencies in terms of companionship, nursing, medical treatment, etc., bringing massive pressure to the emergency medical care of the entire lifecycle in the future [17]. In addition, the surge of patients will rapidly overwhelm the overcrowded medical facilities when a disaster occurs [18].

Many computing technologies (e.g., IoT, AI, cloud computing) are introduced to support the health system to overcome current and future dilemmas of the elderly emergency medical care [19–22]. However, as shown in Fig. 2, Many elderly emergency medical care issues of the entire lifecycle remain unsolved, posing enormous challenges to the computer systems sustaining emergency medical care applications.

- **Task types: emergent and mainly safe-critical.** The elderly are more likely to experience medical emergencies (e.g., stroke, myocardial infarction, falls, etc.) than younger adults, and these events often cause more significant harm to the elderly. When an emergency medical event occurs, it is required that the medical system can handle the safe-critical task in real-time and that

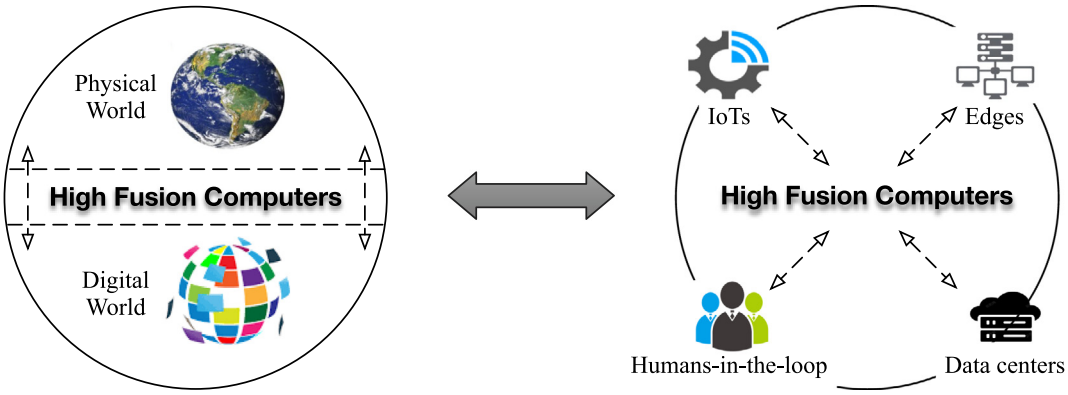

Fig. 1. A Concept Viewpoint of High Fusion Computers (HFCs).

Table 1
Summarization of emerging or future applications.

Application	Critical or typical tasks	Task type	Effect	Metrics	IoT Devices	Dependencies*	Data Management	Access Patterns
Medical emergency management	Emergency detection	Safety-critical	Reversible	Worst-case	Camera, blood pressure monitor, spirometer, gyroscopes, CT scanner, mass spectrometer, etc.	Observe, fuse, recommend, train (Internal & external dependencies)	Image, video, relational data, XML	Real-time write; Non-periodic random read
	Rescue planning	Mission-critical	Reversible	Average-case				
	Rapid diagnosis	Safety-critical	Reversible	Worst-case				
Autonomous driving	Trajectory planning	Safety-critical	Irreversible	Worst-case	Camera, LiDAR, Radar, ultrasonic, GNSS, GPS, etc.	Observe, fuse, act, coordinate (Internal & external dependencies)	Image, video, LAS binary, ASCII [13], text, XML, float matrix, csv	Real-time write; Periodic burst read
	Surrounding object detection	Safety-critical						
	Autonomic control	Mission-critical						
Smart defense systems	Battlespace surveillance	Safety-critical	Irreversible	Worst-case	Seismic, acoustic, magnetic, and imaging sensors or terminal control units, etc.	Observe, orient, decide, act (Internal & external dependencies)	SEG-Y files, MP3, WAV, image	Real-time write; Real-time read
Digital Twin	Smart manufacturing	Mission-critical	Reversible	Average-case	Cameras, sensors, analog-to-digital converter, digital-to-analog converter, etc.	Observe, model, decide, control (Internal & external dependencies)	Image, binary, relational data	Periodic write; Periodic read
	Oil well drilling	Safety-critical	Irreversible	Worst-case				
Civil aviation safety regulation	Airport security	Safety-critical	Irreversible	Worst-case	Cameras, Radars, VHF, Flight-Data Acquisition Unit (FDAU), etc.	Observe, fuse, decide, and alert (Internal & external dependencies)	Image, binary, relational data	Real-time write; Random read
	Air navigation	Mission-critical						
	Anomaly detection	Safety-critical						
Metaverse	Scenario generating	Mission-critical	Reversible	Average-case	Head-mounted display (HMD), Handheld devices (HHDs), etc.	Observe, recognize, fuse, act (Internal & external dependencies)	Dynamic multimedia, relational data	Real-time write; Real-time read
	Avatar maintaining	Mission-critical	Reversible	Average-case				
	Decentralized finance	Mission-critical	Irreversible	Worst-case				
Interplanetary explorations	Knowledge discovery	Mission-critical	Reversible	Best-case	Satellite, Space probe, Robots, etc.	Observe, recognize, infer, control (Internal & external dependencies)	Image, Hierarchical data format(HDF), Network Common Data Form (NetCDF)	Real-time write; Batch transfer; Random read
	Collision avoidance	Safety-critical	Irreversible	Worst-case				
	Space navigation	Mission-critical	Irreversible	Worst-case				

*Internal dependency indicates the dependencies between adjacent execution loops.
*External dependency indicates the dependency between actions or decisions triggered by IoTs, edge, data centers, or humans-in-the-loop.

the medical system can reasonably allocate medical resources for rapid rescue. It is worth noticing that medical experts play a decisive role in the system. Medical emergency management systems consider medical professionals a reliable external component in the control loop, which we call reliable-human-in-the-loop. In this scenario, the system may make recommendations, but the medical expert takes the responsibility, and the decision made by the system is Reversible.

- **Metrics:** In the worst-case and average-case, the quality of computation results is vital. Though the medical experts will take the final responsibility, the systems are valuable only by providing high-quality and interpretable computation results. To provide real-time and safe-critical services, the worst-case performance is important besides the average performance, including the latency and throughput. Since the workloads are often spiky, the systems must gracefully handle overloading.

Due to the specificity of medical care, security and privacy are always the first issues that medical systems have to consider. The future healthcare systems involve a more significant number and variety of devices, a larger population, and more applications, and its complexity brings more significant challenges to security and privacy.

- **Various IoT devices generate a massive volume of heterogeneous data.** Unlike traditional emergency medical care, modern elderly emergency medical care has been extended to their daily lives: before, during, and after the hospital. The emergency medical systems not only rely on a large number of different sensors (such as cameras, gyroscopes, blood pressure monitors, etc.) and also need to access various professional medical equipment (such as CT scanner, mass spectrometer, spirometer, etc.). These devices generate a large amount of heterogeneous data. The emergency medical care system needs to integrate and process large amounts of heterogeneous data in real-time to provide emergency medical services to the elderly throughout their life cycle.
- **Computation patterns, computation dependencies, and interaction patterns:**
  The computation patterns follow the observe, fuse, recommend, and train patterns. The IoT devices observe the data of the patients at different levels. The system may fuse various observations at the edge or data center. The data centers or edges will train and update an AI model through the widely collected and labeled data. The IoT or edge makes a recommendation like alert and further-taken actions. The medical experts make a final decision.
  The computation dependencies are mainly internal dependencies — the dependency between adjacent execution loops of observing, fusing, recommending, and training. For example, a patient's previous diagnosis and treatment recommendations would impact their subsequent recommendations. Besides, external dependencies exist between the decisions triggered by different IoTs, edges, data centers, or humans. Typical examples include new emergency cases reported by the IoTs, the newly trained models, and interventions brought out by the experts.
  Over the other applications, the interaction patterns are simpler. Each IoT works within different conditions, and each computation may trigger different algorithms but only involve local data. Recommendations may be made at edges locally, involving collected data with different spatial–temporal scopes. When training the model, the data are widely collected from IoTs or edges and annotated at the data center for further training. Or in another manner, the labeled data at IoT or edges are distributed training using federation learning techniques [23,24]. Then lightweight models are deployed at IoT and edges.
- **Data management and access pattern:** Patients generate a large number of real-time data from various sensors and professional medical devices. The formats of patient data are diverse — image, video, relational data, XML, etc [25]. Patient data is written into the emergency medical care system in real-time. When an emergency medical event is detected, the emergency medical care system immediately initiates rapid diagnosis and develops rescue planning. Patient data helps medical experts understand how a critical health event unfolds, uncover the geographic characteristics of events, and locate the nearest medical resources. Patient data is accessed only when the patient experiences a medical emergency. Therefore, medical emergencies result in non-periodic random access patterns.

2.1.2. Autonomous driving

Autonomous driving is a promising technology that changes the way people travel. According to the standard of SAE International [26], autonomous driving is classified into six levels—"no driving automation (Level 0), driver assistance (Level 1), partial driving automation (Level 2), conditional driving automation (Level 3), high driving automation (Level 4), and full driving automation (Level 5)" [26]. With its continuous development, self-driving cars will hit the roads and enter into a highly-automated era in the future [27].

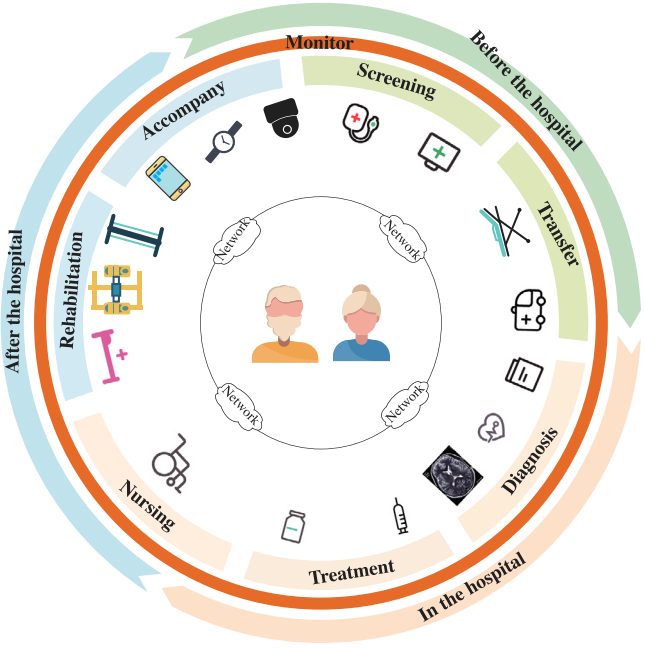

Fig. 2. The healthcare lifecycle of the elderly.

- **Task types: highly-automated and mainly safety-critical.** The future autonomous driving would be highly-automated, even fully-automated, and consider no human in the control loop. The corresponding system needs to perceive and collect multi-source and multi-dimensional data in real-time and respond within several milliseconds. Considering the properties of high autonomy, hard real-time, and potentially destructive effects, the decision and action made by the system are irreversible. It will be a system failure in hard real-time when missing a deadline. In auto-driving, missing a deadline will be catastrophic.
- **Metrics:** In the worst-case and average-case, the quality of computation results is vital as no person takes responsibility. The systems must provide high-quality and interpretable computation results. The worst-case performance of autonomous driving is highly significant. For the safety of cars, pedestrians, and surroundings, an autonomous driving system is demanded to manage and coordinate massive self-driving cars synchronously, assure the performance of almost all vehicles and guarantee the worst-case performance — make the tail latency as low as possible.
  Security and privacy are critical challenges in autonomous driving. Besides the traditional security issue, security also means the car can perceive the environment, make decisions, and take actions correctly.
- **Various IoT devices generate a massive volume of heterogeneous data.** Autonomous driving depends on a large number of sensors; even a single car may deploy multiple kinds of sensors [28,29] including cameras, ultrasonic radar, millimeter-wave radar, lidar, IMU (Inertial Measurement Unit), etc., to obtain the environment information comprehensively. The input data are multi-source and heterogeneous; thus, the system should be able to fuse multi-sensor data for quick processing.

- **Computation patterns, computation dependencies, and interaction patterns:**
  The computation patterns follow the observe, fuse, act, coordinate, and train patterns. The IoT devices observe and detect the data of the weather, surroundings, road lanes, traffic signs, pedestrians, other vehicles, etc. The system fuses various observations, makes a decision, and acts locally at the edge (within each car), including the control of the steering wheel, brake, speed, acceleration, and engine [30]. Meanwhile, vehicles, roads, and surroundings will synchronize their data with each other through data centers and finally coordinate their behaviors. In the background, different models are trained and updated regularly. The computations have severe internal and external dependencies. Internally, a self-driving car's current status and actions would impact its subsequent computations and actions through the observe, fuse, act, and coordinate loop. Externally, a self-driving car's status and action would affect the behaviors of the other vehicles.
- **Data management and access pattern:** The autonomous driving system relies on a large number of sensors to provide comprehensive environmental information, such as traffic signs, pedestrians, and weather. The environment information is continuously written for trajectory planning, surrounding object detection, and autonomous control. The dataset includes LAS binary, float matrix, CSV file, etc [31–33]. Moreover, in the morning and evening rush hour, cars often need real-time path planning to avoid traffic congestion. The autonomous driving system needs to handle periodic burst accesses.

2.2. Why we need to build an HFC system?

2.2.1. The three unique requirements of HFC systems

Significantly different from the warehouse-scale computers that non-stop serve user requests [3], HFC systems have three unique requirements. First, they directly interact with the physical world — considering humans as an essential part: human-in-the-loop [10], and perform safety-critical and mission-critical operations. Each action may have an irreversible effect. Some systems treat humans as "an external and unpredictable element in the control loop" [10], which we call unreliable-human-in-the-loop. For example, many security systems rely on a "human in the loop" to perform security-critical functions [34], but humans are incompetent and often fail in their security roles [34]; In contrast, the other systems consider humans, who make the final decision, a reliable component in the control loop, which we call reliable-human-in-the-loop. For example, medical expert plays a decisive role in medical emergency management systems. Meanwhile, more scenarios "bolster a closer tie with the human through human-in-the-loop controls that consider human skills, intents, psychological states, emotions, and actions inferred through sensory data" [10], which we call collaborative-human-in-the-loop. Collaborative-human-in-the-loop indicates that humans are complements of the other components of the system. Still, an uncoordinated collaboration between a human being and other system components may result in disaster.

Second, unlike Internet services that process independent concurrent requests on planet-scale data center infrastructures, HFC computations have intertwined dependencies between not only adjacent execution loops but also actions or decisions triggered by IoTs, edge, data centers, or humans. Third, under this extremely entangled state, the systems must first satisfy the accuracy metric in predicting, interpreting, or taking action before meeting the performance goal under different conditions, like worst-case, average-case, and best-case.

We take autonomous driving as an example. The systems directly interact with the world. Each action has an irreversible effect. The current status and action of a self-driving car would impact its subsequent computations and actions; a self-driving car's status and action would affect the behaviors of the other vehicles. Autonomous driving must first ensure the accuracy of the decision (quality) and then guarantee the worst-case performance (tail latency).

2.2.2. HFCs demand resources that are several orders of magnitude beyond the reach of the state-of-the-practice systems

Even only taking the worst-case performance metrics – tail latency – as examples, we illustrate that the state-of-the-practice systems are far from satisfying the processing requirements. Even for a much simpler application with simpler computation patterns (our motivating example) compared to those in Table 1 – a simplified smart home application with 95% queries reporting the status and sending heartbeat packets, and 5% queries processing user requests and accessing the Redis database, serving vast concurrent connections is still tricky. Zhang et al. [35] simulate this application using a million-level client load generator (MCC) and evaluate the service capacity of kernel TCP and mTCP v2.1 on an X86 server equipped with Intel Xeon E5645 processor, Centos 7.2, Kernel 3.10.0, and 64 GB memory. Taking 50 ms as the 99th percentile latency threshold, the kernel TCP supports one hundred thousand concurrent connections, while for the user-level mTCP network stack, the number is nine hundred and sixty thousand [35]. Accordingly, to achieve ten billion concurrent connections under the threshold of 50 ms, more than ten thousand nodes, even one hundred thousand nodes, would be needed.

Considering the three unique requirements discussed in Section 2.2.1 and the other factors, HFCs demand resources that are several orders of magnitude beyond the reach of the state-of-the-practice systems. The other factors considered in this rough estimation include scheduling, complex computation and interaction patterns, complex data access patterns, heterogeneous systems and networks, longer communication links, and differentiated processing abilities of IoTs, edges and data centers, which will aggravate the situation exponentially.

We further present several factors in detail. For example, as shown in Fig. 3, a lot of emerging and future applications usually require nearly perfect quality (i.e., predicting and interpreting accuracy in Autonomous driving) and the worst-case tail latency within several milliseconds, which are overlooked in the above motivating example.

From the perspective of task scheduling, different scheduling strategies may significantly impact performance. The previous experiments reveal that different placement and scheduling policies of data and workloads across IoTs, edges, and data centers may substantially affect the overall performance: even with the same infrastructure, the gap may achieve dozens or even hundreds of times considering the total response time [36]. The scheduling strategies are affected by multiple factors, including task complexity, device processing capability, available resources, network condition, pending tasks, etc. However, the state-of-the-practice solutions provide separate management of IoT, edge, and data center, lacking a global perspective, and further hardly to discover the optimal scheduling strategy. Consequently, unified management and efficient collaboration across IoTs, edges, and data centers are required to assure high performance and resource utilization.

Instead of simple status reports in our motivating example, HFC computations are much more complex and intricate, such as object recognition interference, OODA (observe, orient, decide, and act) [37] or even complex Avatar behavior in terms of machine learning or big data processing; An HFC system manages multi-source and heterogeneous data from various IoT devices, manifesting diverse data access patterns in real-time, random, burst, periodic, non-periodic, and batch manners, which also significantly impact the performance; IoT devices have a vast number that substantially outweighs the size of Internet users, with notable discrepant functions.

3. The challenges

This section lays out several HFC challenges.

Organizability and manageability challenges. Unlike a traditional computer system, e.g., a supercomputer or warehouse-scale computer, an HFC system is geographically distributed, consisting of IoTs, edges, data centers, and humans-in-the-loop. Moreover, they are dynamic. For example, in smart defense systems and applications [6],

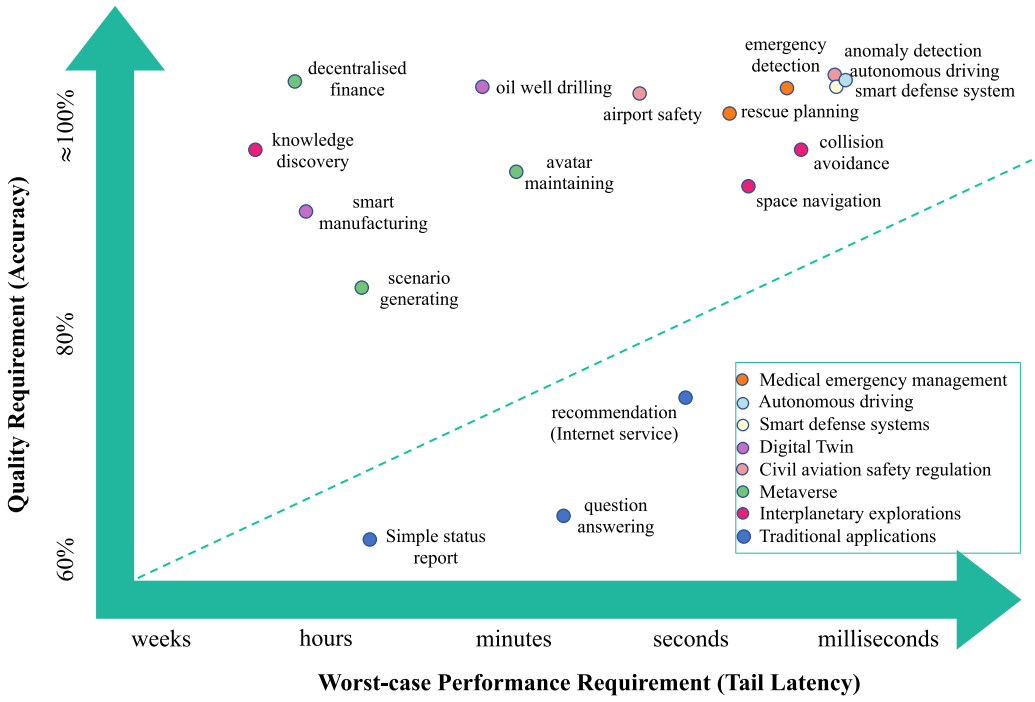

Fig. 3. This figure compares the worst-case quality and performance requirements between the emerging and future applications in Table 1 and the traditional applications. The quality requirement indicates the required model accuracy of the application or tasks; for example, autonomous driving requires nearly 100% accuracy to assure safety. The worst-case performance requirement illustrates the required tail latency of the application. For example, autonomous driving requires extremely low tail latency within several milliseconds.

sensors or terminal control units can be dispersed by airdrop, inserted by artillery, and/or individually placed by an operation team [6]. In an extreme case, the spatial realm even has no bound. For example, unmanned spacecraft may have no bounded destination in interplanetary exploration applications.

So it is challenging to discover the resources and assemble them into a computer system. The challenges lie in managing these resources efficiently, keeping their survivability in the case of highly possible failures, guaranteeing the immunizability from malicious intrusions and attacks, and improving the programmability and schedulability of massive funclets across a large scale of IoTs, edges, and data centers.

Collaboration challenges between software, hardware and people components. More systems exhibit "a closer tie with the human through human-in-the-loop controls" [34]. However, not all these controls are reliable-human-in-the-loop. For example, some systems treat humans as "an external and unpredictable element in the control loop" [10]; thus, the action may have an irreversible effect and result in disaster. The challenge is (1) how do we handle the dilemma of choosing between the system and humans' decision, especially considering "human skills, intents, psychological states, emotions, and actions inferred through sensory data" [10]? (2) how do we support the collaboration between the system and humans? (3) how do we decide the respective responsibility in the partnerships? On the other hand, when the system makes a solo decision, especially for safety-critical functions or worst-case performance, how do we verify and validate its behavior? What is the human' responsibility behind the system? Let us look at these challenges from the perspective of intelligent defense systems described in Table 1. These challenges are not abstract but vivid and concrete in terms of casualties and losses of people and equipment.

Irreversible effect challenges. Most HFC systems or applications perform safety-critical and mission-critical operations, directly interacting with the physical world. Each action may have an irreversible effect under unreliable-human-in-the-loop and collaborative-human-in-the-loop conditions. Even with humans-in-the-loop, considering the human's reaction time, it is hard to make a timely decision and action in an emergency. This irreversible effect demands that the systems' behaviors be verified and validated in advance; the systems can trace the impact of its attributing factors or causes or even achieve interpretability.

Most HFC systems need to explicitly state the quality of computation results and performance constraints in the entire process, including design, implementation, verification, and validation, referring to its target applications' correctness and performance constraints. For example, an autonomous driving application requires a worst-case design that assures high accuracy and tail latency within several milliseconds. In this case, we need to holistically verify and validate the systems and algorithms in terms of quality and performance in different cases like best-case, worst-case, and average-case. We have not gained enough experience in this regard.

Ecosystem wall challenge. An entire HFC ecosystem not only consists of the ensemble of the respective IoT, edge, and data center ecosystems but also involves multifarious meanwhile disparate technologies like processor design, operating system, toolchain, middleware, networking, etc. Moreover, the design of IoT, edge, and data centers follows distinct guidelines and targets according to their unique requirements or constraints, thus generating various ecosystems with different scopes and boundaries, which we call ecosystem walls. For example, the processor design of IoT pursues low energy consumption and a small chip area, while the data center regards performance as the first element. The daunting complexities caution us that we cannot reinvent the wheel. Instead, we need to be compatible with the ecosystem while improving the performance, energy efficiency, and other primary metrics to generate a positive change force that overcomes the ecosystem inertia [12].

The effective evaluation challenges. Generally, we need to deploy a system in a real-world environment and run a real-world application scenario to evaluate the performance and provide optimization guidelines. However, the real-world environment and emerging/future application scenarios are inaccessible and costly in assessing and verifying an HFC system. Hence, benchmarks as proxies of emerging/future

application scenarios and simulators that emulate real-world systems are necessities for designing, evaluating and optimizing an HFC system.

- Benchmarking challenges. The real-world emerging or future application scenarios are incredibly complex, involving the interconnection of IoT, edge, data center, and human-in-the-loop, and including intricate and lengthy execution path [38]. Constructing benchmarks for HFC systems faces significant challenges. First, the benchmarks should reflect the three unique requirements discussed in Section 2.2.1, which are not easily embodied in a benchmark manner. Second, the application designers and providers are concerned about the interaction, interconnection, task assignment, and end-to-end performance across IoT, edge, data centers, and human-in-the-loop. Thus, the reality of a benchmark is fundamental. Considering the complexity and confidential issues, the real-world application scenario is not fit to be used directly as a benchmark; hence, a simplified scenario is necessary. However, the real-world scenario contains hundreds or thousands of modules and components; even a tracing or monitoring tool can hardly figure out the execution path and call graph. Simplifying the real-world scenario while maintaining the critical parts is a big challenge. Moreover, considering the humans-in-the-loop behaviors while constructing the benchmarks are a complex problem.
- Simulation and validation challenges. At the early stage of system and architecture evaluation, the simulator plays a vital role due to the vast manufacturing investment of time and money and the immaturity of the corresponding ecosystem. For example, the effectiveness of the improved processor design, memory access technologies, etc., is evaluated on a simulator. Considering the cost of building a real-world HFC system, a simulation or validation system supporting the whole environment simulation and technology verification is significant. However, the complexity and diversity of application scenarios pose substantial challenges in building such a simulator.

  First, there has no unified interface for different application scenarios or architectures like IoT, edge, and data center. Thus, it is challenging to manage different architectures and support various scenarios.

  Second, simulation accuracy is a crucial metric. High accuracy means the simulator can reflect similar running characteristics to the real world and exhibit running differences under different system environments. Considering the difficulties of multiple-level or multiple-scale simulation, including hardware level, e.g., processor chip, cache, memory hierarchy, disk, and software level, e.g., operation system, ensuring the simulation accuracy is necessary but challenging.

4. Exploring the solution space

We adopt a funclet methodology and architecture to facilitate exploring the HFC software and hardware design space. According to [12], the funclet represents "the common proprieties of basic building blocks" across the systems. Each funclet has the following characteristics [12]:"each contains a well-defined and evolvable functionality with modest complexity; each can be reusable in different contexts; each can be independently tested and verified before integrating; each can be independently deployable; each can interoperate with other funclets through a well-defined bus interface or interconnection".

The funclet architecture consists of four layers: chiplet, HWlet, envlet, and servlet. A chiplet is "an integrated circuit (IC) with modest complexity, providing well-defined functionality" [39,40]; it is "designed to be susceptible to integration with other chiplets, connected with a die-to-die interconnect scheme" [39,40]. A servlet is "an independently deployable and evolvable component that serves users with a well-defined and modest-complexity functionality" [12]. Microservices [41] or cloud functions [42] are two forms of servlet. An HWlet is "an independently deployable, replaceable, and accessible hardware component, e.g., CPU, memory, storage" [12]. An envlet is "an independently deployable and evolvable environment component with well-defined functionality that supports the management of servlets" [12] .

The funclet architecture uses a three-tuple: {funclet set architecture (FSA), organization, system specifics} [12] methodology to describe the architecture. According to [12], the FSA refers to "the actual programmer-visible function set [43], serving as the boundary between two adjacent layers and among different funclets in the same layer"; The organization includes "the high-level aspects of how funclets in the same layer and adjacent layers collaborate"; The system specifics describe "the design and implementation of a system built from funclets" [12]. The advantages of the funclet architecture are discussed in [12]: it increases technology openness, improves productivity, and lowers cost; relieves the complexity of building systems; improves reusability and reliability.

Fig. 4 illustrates the reference HFC architecture and components. For the chiplet layer, we focus on workload-driven chiplet designs. For the HWlet layer, we pay attention to the message interface-based memory system and data path network processor. For the envlet layer, we concentrate on the HFC operating systems and the performance-deterministic distributed storage systems. Also, we provide several tools. A system design tool suite is provided to help designers explore the HFC design space across chiplet, HWlet, envlet, and servlet. A scenario simulator is built across IoTs, edges, and data centers to accelerate innovative technologies' deployment and verification; A full-stack optimization tool is provided. We construct a series of benchmarks and microservices for various HFC applications for the servlet layer. Several scenario benchmarks are proposed as the proxy of real-world application scenarios, aiming to support the whole-stack evaluation and provide feedback to scenario simulators or even real-world scenarios.

Fig. 5 shows a funclet-based HFC architecture in terms of {funclet set architecture (FSA), organization, system specifics}, which is derived from an open-source computer system initiative [12]. Each layer of chiplet, HWlet, envlet, and servlet specifies a set of *FSAs* for IoTs, edges, and data centers. The *FSAs* are composable and collaborative through the same-layer and adjacent layer. The organization specifies "the high-level aspects of how funclets in the same layer and adjacent layers collaborate" [12], including IoT–IoT, Edge–Edge, Datacenter–Datacenter, IoT–Edge, IoT–Datacenter, Edge–Datacenter, and IoT–Edge–Datacenter collaborations. The *system specifies* shows how to implement an HFC system. The different layers for IoTs, edges and data centers are interconnected through the funclet-based open standards, including interconnection, interface, protocol, and networking across IoTs, edges, data centers, and humans-in-the-loop.

4.1. Benchmarks

As the foundation of system design and optimization, benchmarks are of great significance for developing HFC systems. We aim to propose a series of benchmarks that reflect three unique characteristics and other important factors of HFC computations for designing and evaluating HFC systems.

Scenario benchmarks. From the perspective of the whole-stack system benchmarking, e.g., the interconnection and communication of IoTs, edges, data centers, and humans-in-the-loop, we propose scenario benchmarks as the proxy of the real-world application scenario. The construction follows a scenario-distilling methodology that formalizes a real-world application scenario as a graph model and distills it into a combination of essential tasks and components [38]. This methodology identifies "the critical path and primary modules of a real-world scenario since they consume the most system resources and are the core focuses for system design and optimization" [38], thus reducing complexity.

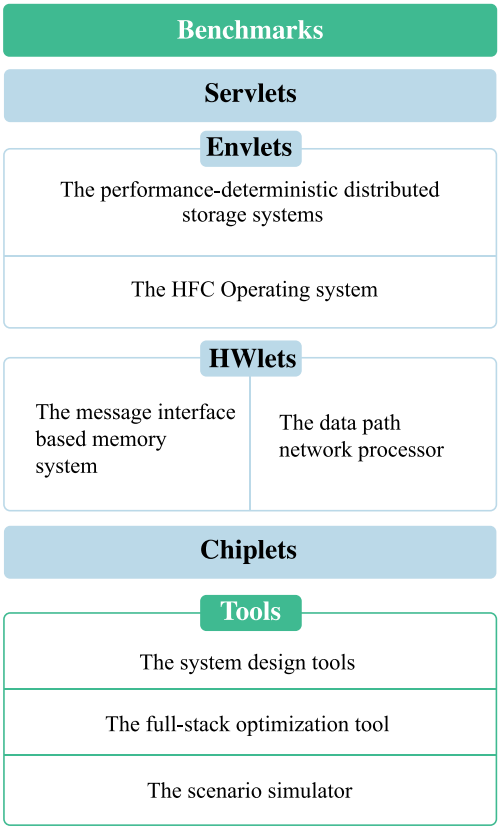

Fig. 4. The overview of the reference architecture.

IoT Benchmarks. From the perspective of middle-level modular benchmarking, we construct IoT benchmarks to evaluate mobile and embedded devices. The IoT benchmarks include the lightweight IoT workloads and light-weight AI workloads.

CPU Benchmarks. We present CPU benchmarks covering typical workloads from emerging and future application scenarios from the low-level architecture benchmarking perspective. Constructing CPU benchmarks adopts a traceable methodology, managing the traceable processes from problem definition, problem instantiation, solution instantiation, and measurement [44].

4.2. Chiplets

This subsection presents the workload-driven chiplet design methodology to explore the ideal architecture for each class of emerging and future applications. We aim to provide reusable building blocks considering the different PPA (performance, power, area) requirements for IoT, edge, and data center.

For the first step, we perform comprehensive workload characterization on a broad spectrum of tasks within target applications. For each application in Table 1, we analyze the computation patterns drilling down into the critical computations within the execution loop, like OODA in smart defense scenarios across IoT, edge, and data center. Then we analyze the interaction patterns covering the interactions between IoT–IoT, IoT–edge, IoT–datacenter, edge–edge, edge–datacenter, and datacenter–datacenter. Above all, the analysis contains computation, memory access, networking, and other characteristics. According to the results, on a single level of IoT, edge, or data center, we classify their characteristics into several classes for each pattern, like computation. After that, we will obtain several classes, including (IoT, computation), (IoT, memory access), (IoT, networking), (edge, computation), (edge, memory access), (edge, networking), (data center, computation), (data center, memory access), (data center, networking), etc.

For the second step, we attempt to define the ideal chiplet architecture for different IoT, edge, data center layers and different analyzed patterns. Specifically, according to the classifications in Step 1, we define the computation, memory, networking chiplets for IoT, edge, and data center, respectively. Each layer of IoT, edge, or data center will contain multiple chiplets for different patterns of computation, memory access, networking, etc. Additionally, each pattern may contain multiple chiplet designs according to the classifications of workload characteristics.

For the third step, we validate the chiplet architecture design and further performs improvements according to the feedback. We adopt FPGA-based simulation and evaluate the scenario, IoT, and CPU benchmarks to conduct the functionality and performance validation. Further, we explore the upgrades and design optimizations based on the validation results.

The chiplet architecture design contains a loop of workload characterization, chiplet design, and validation until the output designs satisfy the application requirements.

4.3. Hwlets

This subsection presents the HWlets solutions, primarily focusing on two innovative HWlets: a message interface-based memory system and a data path network processor.

4.3.1. The message interface based memory system

As the boundary between internal and external memory is blurred, a computer system may face different memory devices with various latency, bandwidth, granularity, and capacity. Thus, the challenge is providing a universal memory interface and a unified memory system so that the programmers do not need to switch between byte-level load/store CPU instructions (for internal memory) or block-level read/write I/O operations.

We have proposed a message-interface-based memory access approach to solve various “memory wall” problems [45]. We plan to extend the message interface to include internal and external memory to build a unified memory system. The system assumes a high bandwidth, high concurrency, and low latency network, which we believe will come soon.

Instead of only using a fixed command format and address for a memory request, we propose to use a message that contains rich semantics to express a memory request. The semantic information consists of size, sequence, priority, process id, persistence, etc., or even array, link pointer, and locks. Furthermore, the memory resource provider is no longer a simple dumb device but with different local computation capabilities to service a message request.

The message interface base memory system decouples the data access from the data organization. A client does not need to know the details or memory resource organization like banks and rows of an SDRAM, even the exact location of the data. The message interface-based memory system also decouples the data access from data transfer. Small data requests can be combined into a large network message. A large data request can be divided into multiple messages.

There are three critical components to implementing a message interface-based memory system. (1) a CPU core generates concurrent memory requests with semantic information. Traditional load/store-based instructions are too simple to express a rich-semantic memory request. Instead, we need kinds of asynchronous and operand-variable instructions. (2) a memory controller with a message interface. The controller should group and assemble memory requests from internal CPU cores into coarse-grained messages to exchange with different memory servers through the network. (3) various message-interfaced memory servers. The memory server manages local storage media and accepts message requests and responses after specific local processing. We will implement the previous two components as chiplets and the message-interfaced memory servers as HWlets.

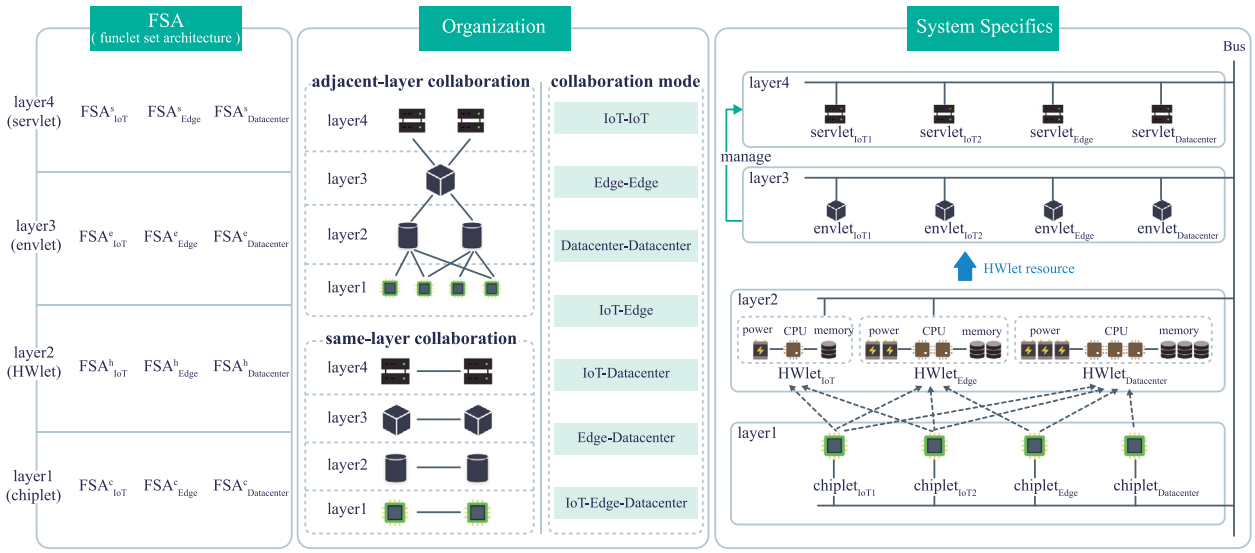

Fig. 5. An HFC instance of the four-layer funclet architecture based on [12].

4.3.2. The data path network processor

High-speed networks have been an indispensable part of modern computer systems. There are two distinct technical routes for network acceleration: offloading [46–48] and onloading [49]. Offloading means to offload part of the network processing to an external accelerator card, namely smart NICs [46,50], and save CPU resources for application logic. For example, the checksum of TCP packets can be computed on high-end NICs [48]. Although various smart NICs have been proposed, none of them has occupied the dominant position in the market. More recently, smart NICs have a new name — DPU (Data processing unit) [51]. On the other hand, onloading means pushing all network packet processing onto general-purpose CPU cores to utilize the ever-increasing computing power of the CPU core. DPDK by Intel [52] is the industry standard for onloading with the help of hardware features built-in Intel CPUs.

Whether offloading or onloading, there are many functions in a network stack that can be accelerated by hardware, for example, checksum, encryption/decryption, table lookup, keyword matching, queuing, ordering, etc. Additionally, several functions can only be efficiently processed by the CPU, such as fragments, lists, buffers, order, and other complex data structures.

Hence, we argue that the critical point is organizing these accelerating resources efficiently. Equipping a general-purpose CPU with accelerators accessed via the I/O bus is not the most efficient solution. Likewise, putting a powerful general-purpose multi-core CPU into a NIC will not change much. The challenge here is how to design efficient control and data path of accelerators inside a CPU to combine both the accelerator units' special functions and the CPU cores' general processing ability.

For general-purpose processors, NIC is always an "external" device. The processor has to initialize a DMA operation to move data from the I/O bus to the local memory or cache to access packet data. Most built-in processors in smart NICs still have such structures. That results in uncontrollable processing latency, so these processors are only suitable for processing control paths. Only hardware logic like FPGA or special function-limited core like P4 engine can process line-speed data paths. They can reach the line speed only because their functions are simple and deterministic enough. Generally, they cannot process complex semantic information in data paths that need complex data structures to store the state of many concurrent transactions.

We propose to design a processor for line-speed processing data paths, which we call the data path network processor (a datapath processor). We will implement the datapath processor as an accelerator. However, the network packet will be the first-class citizen in the datapath processor. The packet stream leaves the register file of the primary CPU and arrives at the datapath processor directly without going through a complex memory hierarchy. The datapath processor has full functionality, including access to the cache and main memory. Thus, the datapath processor can hold and process complex state information necessary for the data plane. The datapath processor also has accelerating units on the local bus; data exchange between the datapath processor core and accelerator can be low latency, highly paralleled, and fine-grain. We have not seen such the structure of the datapath processor before, but fortunately, open-source processors, like RISC-V based, allow us to design a novel processor architecture freely.

4.4. Envlets

This subsection presents the Envlets solutions, primarily focusing on the HFC OS and the HFC distributed storage systems.

4.4.1. The HFC operating system

In the HFC scenarios, computing is ubiquitous, consisting of geographically distributed, heterogeneous hardware devices with different performance and power consumption constraints. Hence, we need a more efficient way to improve the organizability and manageability of the HFC systems. Our OS solutions aim to provide the following features:

(1) The new OS should have a flexible system structure. For different devices and workloads, specific OS capabilities need to be built. OS is no longer limited to a single kernel running directly in the local node but a distributed OS architecture that adapts to hardware resources in different computing nodes. OS needs to be able to reconfigure or rebuild itself in run-time to adapt to various scenarios.

(2) To efficiently manage the distributed heterogeneous hardware, the new OS should rebuild a general and intelligent device driving framework to discover, identify, register, access, and drive the massive hardware resources automatically. In addition, it can establish a soft bus connection for interactions with the immunizability from the malicious intrusions and attacks.

(3) To meet different HFC workloads' performance targets, the new OS needs to build fine-grained resource metering and application profiling features to facilitate efficient scheduling for improving performance and resource utilization.

(4) The HFC ecological boundary is open. As a result, OS faces security challenges, such as end-to-end device authentication access, identification issues in an open environment, and security isolation. Under the premise of "zero trust", we need to embody security enhancement strategies in the native OS kernel.

4.4.2. The performance-deterministic distributed storage systems

The emerging and future applications heavily rely on advanced techniques like big data and AI, performing hybrid and concurrent tasks with different requirements, e.g., latency-critical and throughput-critical, and pursuing the worst, average, or best-case performance. However, these hybrid tasks usually adopt different systems and architectures, and have distinct data access patterns and requirements [53,54]. In addition, the worst-case tail latency poses great challenges even for Internet services in data centers [55–57], let alone much more complex applications across IoTs, edges, and data centers. Thus, to efficiently serve these applications and tasks, building a single distributed storage system (DSS) that provides *deterministic performance and high throughput* is an urgent demand [58–60]. Note that the deterministic performance means that a DSS should enforce differentiated tail latency SLOs for concurrent latency-critical tasks. Throughput means the total QPS (requests per second) or bandwidth of a DSS.

The design of a DSS needs to consider the characteristics of storage devices. We conclude two development trends of storage devices. On the one hand, the devices will be increasingly faster with microsecond-scale or even lower latency. Storage devices have experienced several technological breakthroughs in the past twenty years, such as the development of commercial SSD products, NVM-based SSD products (Intel Optane SSD [61]), and persistent memory products (Intel Optane PM [62]). Compared to HDD and ordinary SSD, NVM-base devices have much lower latency. In addition, emerging fast networks (e.g., 200 Gbps and 400 Gbps Infiniband) have round-trip latency of less than 1 μs [63]. These low latency devices put forward high demand to the storage systems [64]. On the other hand, the devices will contain enhanced computation capacities, such as computational storage drives [65,66], SmarkNICs [67], and programmable switches [68]. Many studies propose to offload several tasks to the devices and have shown the performance advantages, like offloading query processing and data (de-)compression to SmartSSDs [69–72], data replication and file system functions to SmartNIC [73,74], and global memory management, load balance and data cache to programmable switches [75–77]. Hence, the design of a DSS needs to make full use of these in-device computing resources.

Considering the application requirements and device characteristics, building a DSS faces serious challenges. First, due to the distinct states of different machines/threads, latency spikes [55,78], schedulability issues [79], load burst [80–83], and resource contention inside storage devices and the network stack [84–86], it is extremely hard to guarantee deterministic performance and high throughput. Second, many technologies have been proposed to achieve low latency, including using poll instead of interrupts [87], kernel-bypass I/O like user-space communication mechanism [52,88], user-space device drivers [89,90], and user-space file systems [91,92]. However, the previous DSS only adopt a single technology, and it is challenging to integrate and benefit from all these technologies in a single DSS. Third, there are increasing computation capacities inside devices through either FPGA or low-energy embedded processors. Previous work attempts to offload partial computation to in-device computing logic [93–97]. We argue that offloading partial infrastructure software like DSS [74,98] and SQL engine [69,71] rather than user applications are better [74]. However, due to the complicated functionalities of DSS, it is a tough thing.

Our solutions for a novel DSS include the following innovations. First, a new DSS should exploit the leading technologies for emerging devices with μs-scale latencies, including polling device events, user-space I/O, and run-to-completion request processing. Existing efforts focus on one individual technology and fail to integrate all of them into a single system in a systematic way for efficiency. Second, a new DSS should embed scheduling along the whole I/O path from the clients to the servers and storage devices. Schedulable architectures should be used at each layer along the I/O path, including client-side, network, server-side, and storage devices. Moreover, exploiting in-device computing power through software–hardware co-design controls resource utilization at the fine granularity and reduces latency. Third, a new DSS should offload some functionalities to the devices without reducing total throughput and impairing the performance of individual applications. Existing efforts exploit one type of in-device computing power, either computational storage, SmartNIC, or programmable switches. Unlike them, a new DSS should make full use of all these in-device computing powers to achieve better performance and energy efficiency.

4.5. Tools

This section presents our tools: the system design tools, the full-stack optimization tool, and the scenario simulator.

4.5.1. The system design tools

We propose system design tools to help designers explore the HFC design space. The system design tools include the chiplet design tool, the HWlet design tool, the envlet design tool, and the servlet design tool, which correspond to the funclet architecture. Each design tool provides the simulation–validation–development tool suite. For example, for the chiplet design tool, we propose a whole-picture simulation to explore the co-design space of the chiplet across stacks. Unlike the traditional microarchitecture-level simulation, such as the GEM5, our whole-picture simulation is across full system stack levels, including three hierarchical levels: the IR (intermediate representation) level, ISA, and microarchitecture levels [99]. The whole-picture simulation combines the IR, ISA, and microarchitecture level simulations. The design decisions at IR, ISA, and microarchitecture levels are ISA-independent, microarchitecture-independent, or specific to the actual processor's microarchitecture. Combining the design decisions from the IR, ISA, and microarchitecture, the user can explore the co-design space across stacks. Furthermore, we propose cycle-accurate and bit-accurate circuit simulations and verification tools for the validation. The validation tool is based on general x86 computing and heterogeneous FPGA resources and provides an on-demand service for chiplet validation. For the development, we propose AI-based open-sourced EDA tools for integrated circuit design and development.

4.5.2. The full-stack optimization tool

The full-stack optimization tool has the following challenges:

(1) The optimization object is uncertain. Finding the performance bottleneck in an HFC system is non-trivial. Users may feel confused about the optimization objects because of the optimization possibilities on IoTs, edges, or data centers and the complex hierarchies of algorithms, frameworks, software, and hardware.

(2) The optimization space is vast. There are thousands of optimization dimensions of the algorithm, software, and hardware, and the values of the variate vary in an extensive range. As a result, the optimization space is exceptionally huge.

(3) The optimization target is diverse. Different application scenarios have additional user requirements. In addition to the vital importance of accuracy, some applications are sensitive to latency; some require high throughput, and some are concerned with energy consumption. Therefore, different application scenarios have other optimization goals.

The tool covers the optimization from vertical and horizontal dimensions. From the vertical dimension, we will co-explore the optimization space from the algorithm, software, and hardware. For example, for deep learning applications, jointly optimizing the network's architecture and the hardware accelerators is promising in improving performance and reducing energy consumption. We will

consider the close collaboration among IoTs, Edges, data centers, and humans-in-the-loop from the horizontal dimension. For example, we will automatically offload whole or partial deep neural network computations from end devices to more powerful devices, such as edges or data centers.

Automatically co-optimization is non-trivial because of the vast optimization space. There are thousands of optimization dimensions of the algorithm, software, and hardware, and the values of the variate vary in an extensive range. As a result, the optimization space is too huge to complete the search. Reinforcement learning has shown powerful capabilities for the problem of searching for optimal policies in a vast space. Evaluating is expensive in co-optimizing the algorithm, software, and hardware across the IoTs, edges, and data center. We will investigate the state-of-the-art learning algorithms and evaluation strategies and develop the corresponding tools for automatic optimization.

4.5.3. The scenario simulator

The scenario simulator is a miniature of the real system, which contains unified interfaces and replaceable components to enable rapid deployment and verification of innovative technologies. The scenario simulator manages the whole environment of a computer system, e.g., processor chip, operating system, memory, network, etc. It covers the complete execution and interaction across IoTs, edges, data centers, and humans-in-the-loop. It can demonstrate the effects visually, e.g., running results, performance, and power consumption, under different technologies, deployments, or parameter settings, for example, the latency performance of an autodrive scenario using other memory devices and network protocols. For the first step, we plan to provide a network simulator that simulates the communication patterns of representative scenarios like big data and artificial intelligence, supporting different networking technologies. The involved components are replaceable, and we can easily use emerging technology to replace the existing one and verify its effectiveness. We will expand the scenario simulator to the whole HFC environment for the next step.

5. Our plan

We aim to define a new paradigm — IoTs, edges, data centers, and humans-in-the-loop as a computer and launch an open-source high fusion computer (HFC) system initiative. The goal is to vastly enhance the system capabilities under specific energy and cost constraints for most emerging and future applications.

We abstract reusable functions (funclets) across system stacks among IoTs, edges, and data centers to guide the HFC system design and evaluation. We first propose to define a series of benchmarks and funclet-based standards and then build the tools to facilitate the workload-driven exploration of the system and architecture design space. Finally, we provide open-source implementations of an HFC system. We will perform system co-design from the vertical and horizontal dimensions throughout the process. Vertically, we comprehensively explore the algorithms, runtime systems, resource management, storage, memory, networking, and chip technologies. Horizontally, we deeply discover the collaboration and interaction among IoT, edges, data centers, and humans-in-the-loop.

We plan to build the first open-source implementation of an HFC system using an iterative and evolving way. Fig. 6 shows the development milestone of our HFC system. We first focus on one or two typical application scenarios and essential funclets across IoT, edge, and data centers to reduce the complexity. Then we expand the focus and update or replace the technologies gradually. A scenario simulator is beneficial to the expansion, update, and replacement. Finally, we will summarize the experience and lessons during this period and dedicate ourselves to the contributions of a useful HFC system and related advanced technologies.

6. Related work

The development of the computer industry witnessed a series of computer systems concepts and implementations, as shown in Fig. 7. In 1936, Turing proposed to invent the single machine to compute any computable sequence, a concept of a "universal" computing device [100,101]. This kind of "universal" would incur additional performance, cost, or energy overhead called "Turing Tax" – a fundamental question that computer architects aim to reduce [100]. John Gage first proposed the phrase "the network is the computer" in 1984 [102,103]. In 1985, Lewis proposed the concept of the "Internet of things" in a speech to the Congressional Black Caucus Foundation 15th Annual Legislative Weekend [104]. In the mid-1990s, grid computing was proposed to provide computing power, data, and software on-demand, through standardizing the protocols [2]. In 2001, EmNets [6] were proposed and referred to as networked systems of embedded computers. In 2005, cyber–physical systems were proposed to "bridge the cyber-world of computing and communications with the physical world" [105–107]. In 2009, Google proposed the concept of "the data center as a computer", or called warehouse-scale computers (WSCs), to efficiently deliver good levels of Internet service performance [108]. In 2009, the Chinese Academy of Sciences predicted that human–cyber–physical ternary computing would be a development trend in the next 50 years [109]. In 2012, the "Industrial Internet of Things (IIoT)" concept, also known as "Industrial Internet", was proposed to integrate the latest technologies, intelligence systems, and devices and apply them to the entire industrial economy [110,111]. In 2016, the director of Storage SRE at Google illustrated how they do planet-scale engineering for a planet-scale infrastructure — keep all its services up and running and reduce the downtime [3]. In 2017, Li et al. pointed out that human–cyber–physical ternary intelligence is the leading technology and the main driving force of the new economy in the next 15–20 years [112]. In 2021, Mike Warren's group worked with the EC2 team, launching a virtual supercomputer in the cloud – 4,096 EC2 instances with 172,692 cores. This run achieved 9.95 PFLOPS (actual performance), ranking at 40th on the June 2021 TOP500 list [4]. In 2021 and 2022, Wang et al. and Xu et al. pointed out "a new era of human–cyber–physical ternary computing with diverse, intelligent applications over trillions of devices" [7,8] and further proposed the concept of "Information Superbahn", to achieve high system goodput and application quality of service [7,8].

7. Conclusion

We call attention to the fact that more and more emerging and future applications rely heavily upon systems consisting of Internet of Things (IoT), edges, data centers, and humans-in-the-loop. We characterized this new class of systems and coined a new term, high fusion computers (HFCs), to describe them. Significantly different from warehouse-scale computers that non-stop serve independent concurrent user requests, HFCs directly interact with the physical world, considering humans an essential part and performing safety-critical and mission-critical operations; their computations have intertwined dependencies between not only adjacent execution loops but also actions or decisions triggered by IoTs, edge, data centers, or humans-in-the-loops; the systems must first satisfy the accuracy metric in predicting, interpreting, or taking action before meeting the performance goal under different cases. HFCs raise severe challenges in system evaluation, design, and implementation.

We summarize several HFC challenges: organizability and manageability, collaborations between software, hardware, and people components, irreversible effect, ecosystem wall, and effective evaluation. To tackle the above challenges, we propose reconstructing IoTs, edges, data centers, and humans-in-the-loop as a computer rather than a distributed system; we adopt a funclet methodology of building large systems out of smaller functions and exploring HFC design space in a structural manner. We will provide the first open-source implementation of the funclet architecture of HFC systems. The source code will be publicly available from the project homepage: https://www.computercouncil.org/HFC/.

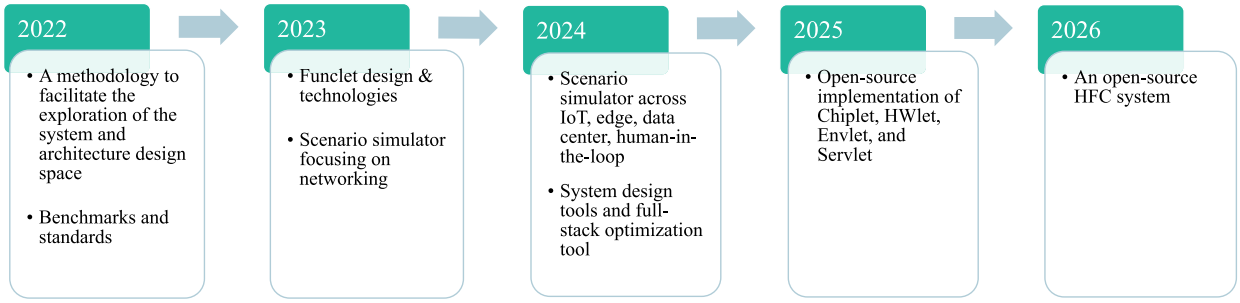

Fig. 6. Development milestone of HFC system.

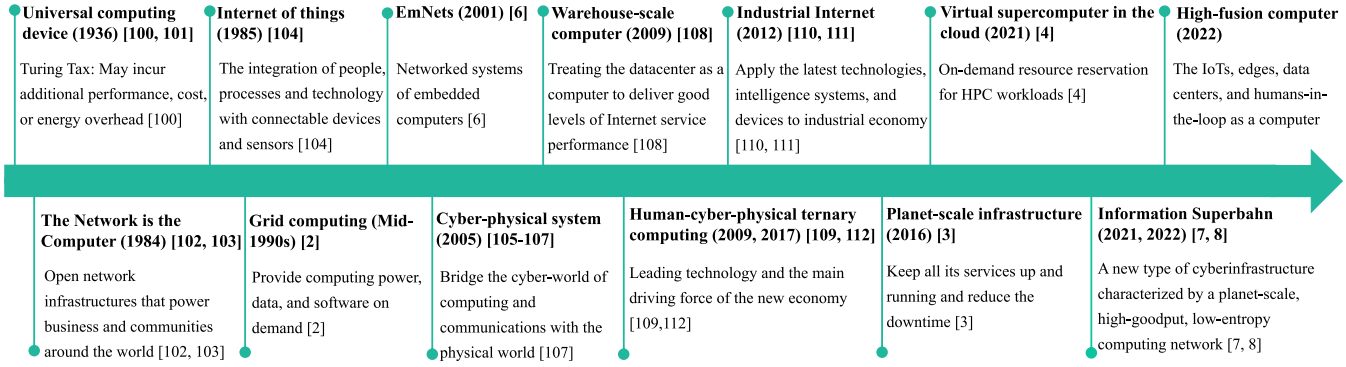

Fig. 7. An overview of the related work.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Dr. Jianfeng Zhan contributes the concept of the IoTs, edges, data centers, and humans-in-the-loop as a computer. Dr. Yungang Bao proposes considering networking as a critical part. Dr. Jianfeng Zhan and Dr. Lei Wang co-coin the term: high fusion computers (HFCs). Dr. Jianfeng Zhan contributes the abstract and Section 1. Section 2.1.1 is contributed by Dr. Jianfeng Zhan, Dr. Yunyou Huang, and Mrs. Guoxin Kang. Section 2.1.2 is contributed by Dr. Wanling Gao, Dr. Jianfeng Zhan, and Mrs. Guoxin Kang. Dr. Wanling Gao contributes the analysis of autonomous driving, civil aviation safety regulation, and interplanetary explorations in Table 1. Dr. Jianfeng Zhan contributes the analysis of medical emergency management, autonomous driving, and smart defense systems in Table 1. Dr. Yunyou Huang contributes the analysis of medical emergency management in Table 1. Dr. Chunjie Luo and Mr. Hainan Ye contribute the analysis of Metaverse in Table 1. Dr. Lei Wang contributes the analysis of digital twin in Table 1. Dr. Weiping Li contributes the analysis of civil aviation safety regulation in Table 1. Mrs. Guoxin Kang contributes all the analysis of data management and access patterns in Table 1. Section 2.2.1 is contributed by Dr. Jianfeng Zhan and Dr. Wanling Gao. Section 2.2.2 is contributed by Dr. Wanling Gao, Dr. Jianfeng Zhan, and Dr. Wenli Zhang. Section 3 is contributed by Dr. Jianfeng Zhan and Dr. Wanling Gao. Dr. Yungang Bao contributed the organizability and manageability challenge. Section 4.1 is contributed by Dr. Wanling Gao and Dr. Jianfeng Zhan. Section 4.2 is contributed by Dr. Wanling Gao, Dr. Jianfeng Zhan, and Dr. Biwei Xie. Section 4.3 is contributed by Dr. Mingyu Chen. Section 4.4.1 is contributed by Dr. Chen Zheng and Dr. Lei Wang. Section 4.4.2 is contributed by Dr. Jin Xiong. Section 4.5.1 is contributed by Dr. Lei Wang. Section 4.5.2 is contributed by Dr. Chunjie Luo and Dr. Biwei Xie. Section 4.5.3 is contributed by Dr. Wanling Gao. Section 5, Section 6, and Section 7 are contributed by Dr. Jianfeng Zhan and Dr. Wanling Gao. Figs. 1, 3, 6, 7 is contributed by Dr. Wanling Gao. Fig. 2 is contributed by Dr. Yunyou Huang. Fig. 4 is contributed by Dr. Lei Wang, Mr. Shaopeng Dai, and Mr. Qian He. Fig. 5 is contributed by Dr. Wanling Gao, Dr. Chunjie Luo, and Mr. Qian He. Dr. Jianfeng Zhan, Dr. Wanling Gao, and Dr. Lei Wang proofread throughout the paper. We are very grateful to Dr. Sa Wang, Dr. Yisong Chang, Dr. Di Zhao, Dr. Mi Zhang, Mrs. Fan Zhang, and Mr. Hongxiao Li for their discussions and contributions.

References

[1] Ian Sommerville, nineth ed., Addison-Wesley, 2011.
[2] Ian Foster, Yong Zhao, Ioan Raicu, Shiyong Lu, Cloud computing and grid computing 360-degree compared, in: 2008 Grid Computing Environments Workshop, IEEE, 2008, pp. 1–10.
[3] Melissa Binde, How Google does planet-scale engineering for planet-scale infrastructure, 2016, https://www.youtube.com/watch?v=H4vMcD7zKM0.
[4] Jeff Barr, Planetary-scale computing - 9.95 PFLOPS & position 40 on the TOP500 list, 2021, https://aws.amazon.com/cn/blogs/aws/planetary-scale-computing-9-95-pflops-position-41-on-the-top500-list/.
[5] Scott Burleigh, Vinton Cerf, Robert Durst, Kevin Fall, Adrian Hooke, Keith Scott, Howard Weiss, The InterPlaNetary internet: A communications infrastructure for Mars exploration, Acta Astronaut. 53 (4–10) (2003) 365–373.
[6] National Research Council, et al., Embedded, Everywhere: A Research Agenda for Networked Systems of Embedded Computers, National Academies Press, 2001.
[7] Zhiwei Xu, Guojie Li, Ninghui Sun, Information superbahn: Towards new type of cyberinfrastructure, Bull. Chin. Acad. Sci. (Chinese Version) 37 (1) (2022) 46–52.
[8] Xiaohong Wang, Sa Wang, Hongwei Tang, Xiaohui Peng, Building substrate for national strategy of new infrastructure construction-practice and thought of information superbahn testbed, Bull. Chin. Acad. Sci. (Chinese Version) 36 (9) (2021) 1066–1073.

[9] Microsoft, A planetary computer for a sustainable future, 2022, https://planetarycomputer.microsoft.com/.

[10] David Sousa Nunes, Pei Zhang, Jorge Sá Silva, A survey on human-in-the-loop applications towards an internet of all, IEEE Commun. Surv. Tutor. 17 (2) (2015) 944–965.

[11] Gordon E. Moore, et al., Cramming More Components onto Integrated Circuits, McGraw-Hill, New York, 1965.

[12] Jianfeng Zhan, Open-source computer systems initiative: The motivation, essence, challenges, and methodology, BenchCouncil Trans. Benchmarks Standards Eval. 2 (1) (2022) 100038.

[13] Qi Chen, Airborne lidar data processing and information extraction, Photogramm. Eng. Remote Sens. 73 (2) (2007) 109.

[14] The World Bank, Population ages 65 and above, total, 2021, https://data.worldbank.org/indicator/SP.POP.65UP.TO. (Last Accessed on 4 February 2022).

[15] U.N. Desa, World Population Prospects 2019: Highlights, Vol. 11, no. 1, United Nations Department for Economic and Social Affairs, New York (US), 2019, p. 125.

[16] Eduardo Teixeira, Hélder Fonseca, Florêncio Diniz-Sousa, Lucas Veras, Giorjines Boppre, José Oliveira, Diogo Pinto, Alberto Jorge Alves, Ana Barbosa, Romeu Mendes, et al., Wearable devices for physical activity and healthcare monitoring in elderly people: A critical review, Geriatrics 6 (2) (2021) 38.

[17] Debajyoti Pal, Suree Funilkul, Nipon Charoenkitkarn, Prasert Kanthamanon, Internet-of-Things and smart homes for elderly healthcare: An end user perspective, IEEE Access 6 (2018) 10483–10496.

[18] JeongGil Ko, Jong Hyun Lim, Yin Chen, Rvăzvan Musvaloiu-E, Andreas Terzis, Gerald M Masson, Tia Gao, Walt Destler, Leo Selavo, Richard P Dutton, MEDiSN: Medical emergency detection in sensor networks, ACM Trans. Embedded Comput. Syst. (TECS) 10 (1) (2010) 1–29.

[19] Sathiyabhama Balasubramaniam, Rajeswari Kurubarahalli Chinnasamy, IoT-based noninvasive wearable and remote intelligent pervasive healthcare monitoring systems for the elderly people, Intell. Pervasive Comput. Syst. Smarter Healthc. (2019) 141–158.

[20] Adriana Alexandru, Dora Coardos, Eleonora Tudora, IoT-Based healthcare remote monitoring platform for elderly with fog and cloud computing, in: 2019 22nd International Conference on Control Systems and Computer Science, CSCS, IEEE, 2019, pp. 154–161.

[21] Sumit Majumder, Emad Aghayi, Moein Noferesti, Hamidreza Memarzadeh-Tehran, Tapas Mondal, Zhibo Pang, M. Jamal Deen, Smart homes for elderly healthcare-recent advances and research challenges, Sensors 17 (11) (2017) 2496.

[22] Ying Liu, Lin Zhang, Yuan Yang, Longfei Zhou, Lei Ren, Fei Wang, Rong Liu, Zhibo Pang, M Jamal Deen, A novel cloud-based framework for the elderly healthcare services using digital twin, IEEE Access 7 (2019) 49088–49101.

[23] Yuan Liang, Yange Guo, Yanxia Gong, Chunjie Luo, Jianfeng Zhan, Yunyou Huang, FLBench: A benchmark suite for federated learning, in: BenchCouncil International Federated Intelligent Computing and Block Chain Conferences, Springer, 2020, pp. 166–176.

[24] Yunyou Huang, Xiuxia Miao, Ruchang Zhang, Li Ma, Wenjing Liu, Fan Zhang, Xianglong Guan, Xiaoshuang Liang, Xiangjiang Lu, Suqing Tang, et al., Training, testing and benchmarking medical AI models using clinical AIBench, BenchCouncil Transactions Benchmarks Standards Eval. (2022) 100037.

[25] Tom J. Pollard, Alistair E.W. Johnson, Jesse D. Raffa, Leo A. Celi, Roger G. Mark, Omar Badawi, The eICU collaborative research database, a freely available multi-center database for critical care research, Sci. Data 5 (1) (2018) 1–13.

[26] Sergej S. Shadrin, Anastasiia A. Ivanova, Analytical review of standard Sae J3016 "taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles" with latest updates, Avtomobil'. Doroga. Infrastruktura. 3 (21) (2019) 10.

[27] NTT DATA, When the car takes over - A glimpse into the future of autonomous driving, 2020, https://us.nttdata.com/en/-/media/assets/white-paper/mfg-autonomous-cars-white-paper.pdf.

[28] De Jong Yeong, Gustavo Velasco-Hernandez, John Barry, Joseph Walsh, Sensor and sensor fusion technology in autonomous vehicles: A review, Sensors 21 (6) (2021) 2140.

[29] Jorge Vargas, Suleiman Alsweiss, Onur Toker, Rahul Razdan, Joshua Santos, An overview of autonomous vehicles sensors and their vulnerability to weather conditions, Sensors 21 (16) (2021) 5397.

[30] Jelena Kocić, Nenad Jovičić, Vujo Drndarević, An end-to-end deep neural network for autonomous driving designed for embedded automotive platforms, Sensors 19 (9) (2019) 2064.

[31] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Scharwächter, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, Bernt Schiele, The cityscapes dataset, in: CVPR Workshop on the Future of Datasets in Vision, Vol. 2, sn, 2015.

[32] Will Maddern, Geoffrey Pascoe, Chris Linegar, Paul Newman, 1 year, 1000 km: The Oxford RobotCar dataset, Int. J. Robot. Res. 36 (1) (2017) 3–15.

[33] Andreas Geiger, Philip Lenz, Christoph Stiller, Raquel Urtasun, Vision meets robotics: The kitti dataset, Int. J. Robot. Res. 32 (11) (2013) 1231–1237.

[34] Lorrie F. Cranor, A Framework for Reasoning About the Human in the Loop, Advanced Computing Systems Professional and Technical Association, 2008.

[35] Wenli Zhang, Ke Liu, Yifan Shen, Yazhu Lan, Hui Song, Mingyu Chen, Yuanfei Chen, Labeled network stack: A high-concurrency and low-tail latency cloud server framework for massive IoT devices, J. Comput. Sci. Tech. 35 (1) (2020) 179–193.

[36] Tianshu Hao, Jianfeng Zhan, Kai Hwang, Wanling Gao, Xu Wen, AI-oriented workload allocation for cloud-edge computing, in: 2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing, CCGrid, IEEE, 2021, pp. 555–564.

[37] Chet Richards, Boyd's OODA Loop, FHS, Sjøkrigsskolen, 2020.

[38] Wanling Gao, Fei Tang, Jianfeng Zhan, Xu Wen, Lei Wang, Zheng Cao, Chuanxin Lan, Chunjie Luo, Xiaoli Liu, Zihan Jiang, Aibench scenario: Scenario-distilling ai benchmarking, in: 2021 30th International Conference on Parallel Architectures and Compilation Techniques, PACT, IEEE, 2021, pp. 142–158.

[39] Tao Li, Jie Hou, Jinli Yan, Rulin Liu, Hui Yang, Zhigang Sun, Chiplet heterogeneous integration technology-status and challenges, Electronics 9 (4) (2020) 670.

[40] Mark Lapedus, The Good and Bad of Chiplets, 2020, https://semiengineering.com/the-good-and-bad-of-chiplets/.

[41] Irakli Nadareishvili, Ronnie Mitra, Matt McLarty, Mike Amundsen, Microservice Architecture: Aligning Principles, Practices, and Culture, " O'Reilly Media, Inc.", 2016.

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

[43] John L. Hennessy, David A. Patterson, Computer Architecture: A Quantitative Approach.

[44] Jianfeng Zhan, A BenchCouncil view on benchmarking emerging and future computing, BenchCouncil Trans. Benchmarks Standards Eval. 2 (2) (2022) 100064.

[45] Licheng Chen, Mingyu Chen, Yuan Ruan, Yongbing Huang, Zehan Cui, Tianyue Lu, Yungang Bao, MIMS: Towards a message interface based memory system, J. Comput. Sci. Tech. 29 (2) (2014) 255–272.

[46] Ming Liu, Tianyi Cui, Henry Schuh, Arvind Krishnamurthy, Simon Peter, Karan Gupta, Offloading distributed applications onto smartnics using ipipe, in: Proceedings of the ACM Special Interest Group on Data Communication, 2019, pp. 318–333.

[47] Yanfang Le, Hyunseok Chang, Sarit Mukherjee, Limin Wang, Aditya Akella, Michael M Swift, TV Lakshman, UNO: Uniflying host and smart NIC offload for flexible packet processing, in: Proceedings of the 2017 Symposium on Cloud Computing, 2017, pp. 506–519.

[48] YoungGyoun Moon, SeungEon Lee, Muhammad Asim Jamshed, KyoungSoo Park, {AccelTCP}: Accelerating network applications with stateful {TCP} offloading, in: 17th USENIX Symposium on Networked Systems Design and Implementation, NSDI 20, 2020, pp. 77–92.

[49] Matthew GF Dosanjh, Ryan E Grant, Patrick G Bridges, Ron Brightwell, Re-evaluating network onload vs. offload for the many-core era, in: 2015 IEEE International Conference on Cluster Computing, IEEE, 2015, pp. 342–350.

[50] Yan Yan, Arash Farhadi Beldachi, Reza Nejabati, Dimitra Simeonidou, P4-enabled smart nic: Enabling sliceable and service-driven optical data centres, J. Lightwave Technol. 38 (9) (2020) 2688–2694.

[51] Idan Burstein, Nvidia data center processing unit (DPU) architecture, in: 2021 IEEE Hot Chips 33 Symposium, HCS, IEEE, 2021, pp. 1–20.

[52] DPDK, Data plane development kit, 2022, https://www.dpdk.org/.

[53] Fengfeng Pan, Yinliang Yue, Jin Xiong, Daxiang Hao, I/O characterization of big data workloads in data centers, in: Big Data Benchmarks, Performance Optimization, and Emerging Hardware, Cham, 2014, pp. 85–97.

[54] Steven W.D. Chien, Stefano Markidis, Chaitanya Prasad Sishtla, Luis Santos, Pawel Herman, Sai Narasimhamurthy, Erwin Laure, Characterizing deep-learning I/O workloads in TensorFlow, in: 2018 IEEE/ACM 3rd International Workshop on Parallel Data Storage & Data Intensive Scalable Computing Systems, PDSW-DISCS, IEEE, 2018, pp. 54–63.

[55] Jeffrey Dean, Luiz André Barroso, The tail at scale, Commun. ACM 56 (2) (2013) 74–80.

[56] Aad Van Moorsel, Metrics for the internet age: Quality of experience and quality of business, in: Fifth International Workshop on Performability Modeling of Computer and Communication Systems, Vol. 34, no. 13, Citeseer, 2001, pp. 26–31.

[57] Tobias Hobfeld, Raimund Schatz, Martin Varela, Christian Timmerer, Challenges of QoE management for cloud applications, IEEE Commun. Mag. 50 (4) (2012) 28–36.

[58] Andrew Wang, Shivaram Venkataraman, Sara Alspaugh, Randy Katz, Ion Stoica, Cake: Enabling high-level SLOs on shared storage systems, in: Proceedings of the Third ACM Symposium on Cloud Computing, SoCC '12, New York, NY, USA, 2012.

[59] Ana Klimovic, Heiner Litz, Christos Kozyrakis, ReFlex: Remote flash ≈ local flash, ACM SIGARCH Comput. Archit. News 45 (1) (2017) 345–359.

[60] Liuying Ma, Zhenqing Liu, Jin Xiong, Dejun Jiang, QWin: Enforcing tail latency SLO at shared storage backend, 2021, arXiv preprint arXiv:2106.09206.

[61] Intel, Intel optane™DC SSD series, 2020, https://www.intel.com/content/www/us/en/products/details/memory-storage/data-center-ssds/optane-dc-ssd-series.html.

[62] Intel, Intel optane™persistent memory, 2020, https://www.intel.com/content/www/us/en/products/details/memory-storage/optane-dc-persistent-memory.html.

[63] Timothy Prickett Morgan, InfiniBand innovation is about more than bandwidth and latency, 2021, https://www.nextplatform.com/2021/09/30/infiniband-innovation-is-about-more-than-bandwidth-and-latency/.

[64] Luiz Barroso, Mike Marty, David Patterson, Parthasarathy Ranganathan, Attack of the killer microseconds, Commun. ACM 60 (4) (2017) 48–54.

[65] Samsung, SmartSSD, 2000, https://semiconductor.samsung.com/ssd/smart-ssd/.

[66] ScaleFlux, ScaleFlux computational storage drive, 2022, https://www.scaleflux.com/.

[67] Mellanox Technologies, NVIDIA mellanox BlueField SmartNIC for InfiniBand & ethernet, 2020, https://network.nvidia.com/files/doc-2020/pb-bluefield-vpi-smart-nic.pdf.

[68] Intel, Intel Tofino™programmable ethernet switch ASIC, 2022, https://www.intel.com/content/www/us/en/products/network-io/programmable-ethernet-switch/tofino-series.html.

[69] Jaeyoung Do, Yang-Suk Kee, Jignesh M Patel, Chanik Park, Kwanghyun Park, David J DeWitt, Query processing on smart SSDS: Opportunities and challenges, in: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, 2013, pp. 1221–1230.

[70] Boncheol Gu, Andre S. Yoon, Duck-Ho Bae, Insoon Jo, Jinyoung Lee, Jonghyun Yoon, Jeong-Uk Kang, Moonsang Kwon, Chanho Yoon, Sangyeun Cho, et al., Biscuit: A framework for near-data processing of big data workloads, ACM SIGARCH Comput. Archit. News 44 (3) (2016) 153–165.

[71] Insoon Jo, Duck-Ho Bae, Andre S Yoon, Jeong-Uk Kang, Sangyeun Cho, Daniel DG Lee, Jaeheon Jeong, YourSQL: A high-performance database system leveraging in-storage computing, Proc. VLDB Endow. 9 (12) (2016) 924–935.

[72] Mohammadamin Ajdari, Pyeongsu Park, Joonsung Kim, Dongup Kwon, Jangwoo Kim, CIDR: A cost-effective in-line data reduction system for terabit-per-second scale SSD arrays, in: 2019 IEEE International Symposium on High Performance Computer Architecture, HPCA, IEEE, 2019, pp. 28–41.

[73] Daehyeok Kim, Amirsaman Memaripour, Anirudh Badam, Yibo Zhu, Hongqiang Harry Liu, Jitu Padhye, Shachar Raindel, Steven Swanson, Vyas Sekar, Srinivasan Seshan, Hyperloop: Group-based NIC-offloading to accelerate replicated transactions in multi-tenant storage systems, in: Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication, 2018, pp. 297–312.

[74] Jongyul Kim, Insu Jang, Waleed Reda, Jaeseong Im, Marco Canini, Dejan Kostić, Youngjin Kwon, Simon Peter, Emmett Witchel, LineFS: Efficient SmartNIC offload of a distributed file system with pipeline parallelism, in: Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles, 2021, pp. 756–771.

[75] Seung-seob Lee, Yanpeng Yu, Yupeng Tang, Anurag Khandelwal, Lin Zhong, Abhishek Bhattacharjee, MIND: In-network memory management for disaggregated data centers, in: Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles, SOSP '21, New York, NY, USA, 2021, pp. 488–504.

[76] Jialin Li, Jacob Nelson, Ellis Michael, Xin Jin, Dan R.K. Ports, Pegasus: Tolerating skewed workloads in distributed storage with In-Network coherence directories, in: 14th USENIX Symposium on Operating Systems Design and Implementation, OSDI 20, 2020, pp. 387–406.

[77] Zaoxing Liu, Zhihao Bai, Zhenming Liu, Xiaozhou Li, Changhoon Kim, Vladimir Braverman, Xin Jin, Ion Stoica, DistCache: Provable load balancing for large-scale storage systems with distributed caching, in: Proceedings of the 17th USENIX Conference on File and Storage Technologies, FAST '19, USA, 2019, pp. 143–157.

[78] Jialin Li, Naveen Kr. Sharma, Dan R.K. Ports, Steven D. Gribble, Tales of the tail: Hardware, OS, and application-level sources of tail latency, in: Proceedings of the ACM Symposium on Cloud Computing, SoCC '14, New York, NY, US, 2014, pp. 1–14.

[79] Suli Yang, Jing Liu, Andrea C Arpaci-Dusseau, Remzi H Arpaci-Dusseau, Principled schedulability analysis for distributed storage systems using thread architecture models, in: 13th USENIX Symposium on Operating Systems Design and Implementation, OSDI 18, 2018, pp. 161–176.

[80] Henry Qin, Qian Li, Jacqueline Speiser, Peter Kraft, John Ousterhout, Arachne: Core-aware thread management, in: 13th USENIX Symposium on Operating Systems Design and Implementation, OSDI'18, Carlsbad, CA, 2018, pp. 145–160.

[81] Amy Ousterhout, Joshua Fried, Jonathan Behrens, Adam Belay, Hari Balakrishnan, Shenango: Achieving high {CPU} efficiency for latency-sensitive datacenter workloads, in: 16th USENIX Symposium on Networked Systems Design and Implementation, NSDI 19, 2019, pp. 361–378.

[82] Kostis Kaffes, Timothy Chong, Jack Tigar Humphries, Adam Belay, David Mazières, Christos Kozyrakis, Shinjuku: Preemptive scheduling for μsecond-scale tail latency, in: 16th USENIX Symposium on Networked Systems Design and Implementation, NSDI 19, 2019, pp. 345–360.

[83] Călin Iorgulescu, Reza Azimi, Youngjin Kwon, Sameh Elnikety, Manoj Syamala, Vivek Narasayya, Herodotos Herodotou, Paulo Tomita, Alex Chen, Jack Zhang, et al., {PerfIso}: Performance isolation for commercial {Latency-Sensitive} services, in: 2018 USENIX Annual Technical Conference, USENIX ATC 18, 2018, pp. 519–532.

[84] Mingzhe Hao, Gokul Soundararajan, Deepak Kenchammana-Hosekote, Andrew A Chien, Haryadi S Gunawi, The tail at store: A revelation from millions of hours of disk and {ssd} deployments, in: 14th USENIX Conference on File and Storage Technologies, FAST 16, 2016, pp. 263–276.

[85] Shiqin Yan, Huaicheng Li, Mingzhe Hao, Michael Hao Tong, Swaminathan Sundararaman, Andrew A Chien, Haryadi S Gunawi, Tiny-tail flash: Near-perfect elimination of garbage collection tail latencies in NAND SSDs, ACM Trans. Storage (TOS) 13 (3) (2017) 1–26.

[86] Shine Kim, Jonghyun Bae, Hakbeom Jang, Wenjing Jin, Jeonghun Gong, Seungyeon Lee, Tae Jun Ham, Jae W Lee, Practical erase suspension for modern low-latency {SSDs}, in: 2019 USENIX Annual Technical Conference, USENIX ATC 19, 2019, pp. 813–820.

[87] Jisoo Yang, Dave B. Minturn, Frank Hady, When poll is better than interrupt, in: FAST, Vol. 12, 2012, pp. 3–3.

[88] R. Recio, B. Metzler, P. Culley, J. Hilland, D. Garcia, A remote direct memory access protocol specification, 2007, https://datatracker.ietf.org/doc/html/rfc5040.

[89] Hyeong-Jun Kim, Young-Sik Lee, Jin-Soo Kim, NVMeDirect: A User-space I/O Framework for Application-specific Optimization on NVMe SSDs, in: 8th USENIX Workshop on Hot Topics in Storage and File Systems, HotStorage'16, 2016.

[90] Ziye Yang, James R. Harris, Benjamin Walker, Daniel Verkamp, Changpeng Liu, Cunyin Chang, Gang Cao, Jonathan Stern, Vishal Verma, Luse E. Paul, SPDK: A development kit to build high performance storage applications, in: 2017 IEEE International Conference on Cloud Computing Technology and Science, CloudCom, 2017, pp. 154–161.

[91] Jing Liu, Andrea C Arpaci-Dusseau, Remzi H Arpaci-Dusseau, Sudarsun Kannan, File systems as processes, in: 11th USENIX Workshop on Hot Topics in Storage and File Systems, HotStorage 19, 2019.

[92] Jing Liu, Anthony Rebello, Yifan Dai, Chenhao Ye, Sudarsun Kannan, Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau, Scale and performance in a filesystem semi-microkernel, in: Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles, SOSP '21, New York, NY, USA, 2021, pp. 819–835.

[93] Shengwen Liang, Ying Wang, Youyou Lu, Zhe Yang, Huawei Li, Xiaowei Li, Cognitive SSD: A deep learning engine for in-storage data retrieval, in: 2019 USENIX Annual Technical Conferenc, USENIX ATC 19, USENIX Association, Renton, WA, 2019, pp. 395–410.

[94] Vikram Sharma Mailthody, Zaid Qureshi, Weixin Liang, Ziyan Feng, Simon Garcia de Gonzalo, Youjie Li, Hubertus Franke, Jinjun Xiong, Jian Huang, Wen-mei Hwu, DeepStore: In-storage acceleration for intelligent queries, in: Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO '52, New York, NY, USA, 2019, pp. 224–238.

[95] Jaeyoung Do, Victor C Ferreira, Hossein Bobarshad, Mahdi Torabzadehkashi, Siavash Rezaei, Ali Heydarigorji, Diego Souza, Brunno F Goldstein, Leandro Santiago, Min Soo Kim, et al., Cost-effective, energy-efficient, and scalable storage computing for large-scale AI applications, ACM Trans. Storage (TOS) 16 (4) (2020) 1–37.

[96] Amedeo Sapio, Marco Canini, Chen-Yu Ho, Jacob Nelson, Panos Kalnis, Changhoon Kim, Arvind Krishnamurthy, Masoud Moshref, Dan R.K. Ports, Peter Richtárik, Scaling distributed machine learning with in-network aggregation, 2019, arXiv preprint arXiv:1903.06701.

[97] ChonLam Lao, Yanfang Le, Kshiteej Mahajan, Yixi Chen, Wenfei Wu, Aditya Akella, Michael Swift, ATP: In-network aggregation for multi-tenant learning, in: 18th USENIX Symposium on Networked Systems Design and Implementation, NSDI 21, 2021, pp. 741–761.

[98] Jian Zhang, Yujie Ren, Sudarsun Kannan, FusionFS: Fusing I/O operations using CISCOps in firmware file systems, in: 20th USENIX Conference on File and Storage Technologies, FAST'22, Santa Clara, CA, 2022, pp. 297–312.

[99] Lei Wang, Kaiyong Yang, Wanling Gao, Chunjie Luo, Chengxi Wang, Zhongxin Ge, Li Zhang, Fan Zhang, Jianfeng Zhan, WPC: Whole-picture Workload Characterization, Technical Report, Institute of Computing Technology, Chinese Academy of Sciences.

[100] Paul H.J. Kelly, Advanced computer architecture: The stored program concept and the turing tax, 2020, https://www.doc.ic.ac.uk/~phjk/AdvancedCompArchitecture/Lectures/pdfs/Ch01-part4-TuringTaxDiscussion.pdf.

[101] Paul H.J. Kelly, "Turing Tariff" reduction: Architectures, compilers and languages to break the universality barrier, 2020, https://www.doc.ic.ac.uk/~phjk/Presentations/2020-06-24-DoCLunch-PaulKelly-TuringTaxV04.pdf.

[102] Wikipedia, The network is the computer, https://en.wikipedia.org/wiki/The_Network_is_the_Computer.

[103] SunMicrosystemsInc, The network is the computer, 2007, https://www.youtube.com/watch?v=kWNtaUaDxZw.

[104] Chetan Sharma, Correcting the IoT history, 2016, 14 March 2016, http://www.chetansharma.com/correcting-the-iot-history/.

[105] Verband Der, VDI/Vde-Richtlinien, 2005.
[106] Ken Wallace, Capturing, storing and retrieving design knowledge in a distributed environment, in: International Conference on Computer Supported Cooperative Work in Design, Vol. 1, IEEE Computer Society, 2005, pp. 10–Vol.
[107] Ragunathan Rajkumar, Insup Lee, Lui Sha, John Stankovic, Cyber-physical systems: The next computing revolution, in: Design Automation Conference, IEEE, 2010, pp. 731–736.
[108] Luiz André Barroso, Urs Hölzle, The datacenter as a computer: An introduction to the design of warehouse-scale machines, in: Synthesis Lectures on Computer Architecture, vol. 4, (no. 1) Morgan & Claypool Publishers, 2009, pp. 1–108.
[109] Chinese Academy of Sciences Strategic Research Group in the Field of Information, China's Information Technology Development Roadmap to 2050, Science Press, 2009.
[110] Peter C. Evans, Marco Annunziata, Industrial internet: Pushing the boundaries, Gen. Electr. Rep. (2012) 488–508.
[111] Jianqiang Li, F Richard Yu, Genqiang Deng, Chengwen Luo, Zhong Ming, Qiao Yan, Industrial internet: A survey on the enabling technologies, applications, and challenges, IEEE Commun. Surv. Tutor. 19 (3) (2017) 1504–1526.
[112] Guojie Li, Zhiwei Xu, Judging new economy from perspective of information technology trend, Bull. Chin. Acad. Sci. (Chinese Version) 32 (3) (2017) 233–238.