Contents lists available at ScienceDirect

BenchCouncil Transactions on Benchmarks, Standards and Evaluations

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

# Open-source computer systems initiative: The motivation, essence, challenges, and methodology

Jianfeng Zhan

*Research Center for Advanced Computer Systems, Institute of Computing Technology, Chinese Academy of Sciences, China*

ARTICLE INFO

*Keywords:*
Information technology decoupling
High-end computer systems
Opensource computer systems initiative
Challenges
Funclet
Abstraction
Architecture
Chiplet
HWlet
Envlet
Servlet

ABSTRACT

The global community faces many pressing and uncertain challenges like pandemics and global climate change. Information technology (IT) infrastructure has become the enabler to addressing those challenges. Unfortunately, IT decoupling has distracted and weakened the international community's ability to handle those challenges.

This article initiates an open-source computer system (OSCS) initiative to tackle the challenges of IT decoupling. The OSCS movement is where open-source software converges with open-source hardware. Its essential is to utilize the inherent characteristics of a class of representative workloads and propose innovative abstraction and methodology to co-explore the software and hardware design spaces of high-end computer systems, attaining peak performance, security, and other fundamental dimensions. I discuss its four challenges, including the system complexity, the tradeoff between universal and ideal systems, guaranteeing quality of computation results and performance under different conditions, e.g., best-case, worst-case, or average-case, and balancing legal, patent, and license issues.

Inspired by the philosophy of building large systems out of smaller functions, I propose the funclet abstraction and methodology to tackle the first challenge. The funclet abstraction is a well-defined, evolvable, reusable, independently deployable, and testable functionality with modest complexity. Each funclet interoperates with other funclets through standard bus interfaces or interconnections. Four funclet building blocks: chiplet, HWlet, envlet, and servlet at the chip, hardware, environment management, and service layers form the four-layer funclet architecture. The advantages of the funclet abstraction and architecture are discussed. The project's website is publicly available from https://www.opensourcecomputer.org or https://www.computercouncil.org.

## 1. Introduction

The complex interactions between human activities and the earth's ecosystem lead to two pressing challenges: the COVID-19 pandemic and global climate change. The study, published on 10 March in The Lancet [1], says that the actual number of lives lost to the COVID-19 pandemic by 31 December 2021 was close to 18 million. That far outstrips the 5.9 million deaths that were reported to various official sources for the same period [2]. This dire situation poses a heartbroken challenge to our seniors and children. On the other hand, due to climate change [3], more frequent and intense drought, storms, heatwaves, rising sea levels, melting glaciers, and warming oceans can directly threaten the survival of humans and wild animals.

In addition to the global society's strong support and action, the science and technology society bears the burden of tackling those challenges. Unfortunately, the growing political gaps among people with deviated viewpoints tear apart the science and technology community. Undeniably, human societies have undergone disparate political systems with varying extents of political rights and civil liberties — however, the trend converges. For example, almost every nation abolished slavery in favor of human rights; Almost every country acknowledges the rule of law, though the processes and meaning vary wildly. In the short term, there may be a spikey political gap. However, the gap has been closing for a long time. The temporally increasing political gaps do not justify the technology decoupling. Instead, technology decoupling will weaken the global community's capability to handle the pressing challenges, which adversely shakes the foundation of the worldwide community. IT is one of the enablers underpinning effective plans and actions addressing those challenges [4–7]. IT decoupling threatens the shared IT infrastructure and hence the shared future.

Technology decoupling and export control between nations will increase costs and lower productivity [8]. The dire IT decoupling distracts and weakens human beings' ability to address those pressing challenges. This calls for our wisdom and actions to unify our science and technology community. The open science initiatives [9–12,12,13] partially responded to it.

As shown in Fig. 1, this article initiates an open-source computer system movement (in short, OSCS) where open-source software converges with open-source hardware. High-end computer systems serve

*E-mail address:* zhanjianfeng@ict.ac.cn.
*URL:* http://www.benchcouncil.org/zjf.html.

https://doi.org/10.1016/j.tbench.2022.100038

Available online 25 April 2022

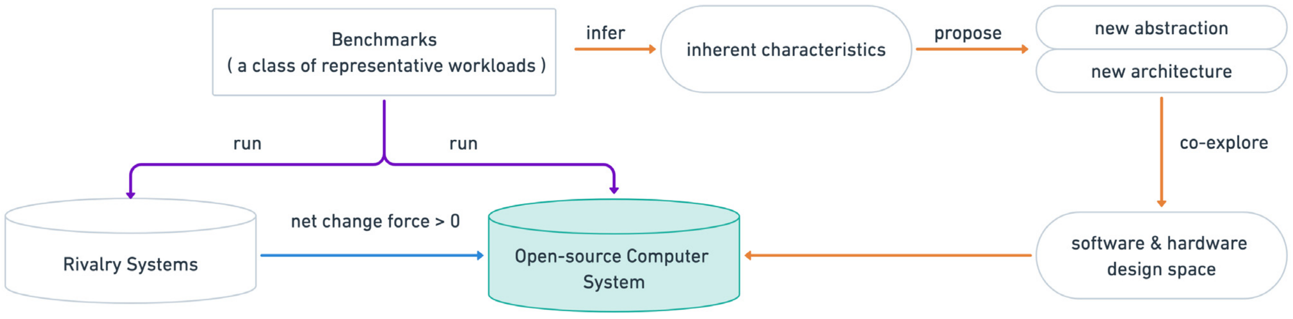

**Fig. 1.** The essential of the open-source computer systems (OSCS) initiative.

as the cornerstone of IT infrastructure, and their components like Chips or Operating Systems are the basis for building the IT infrastructure. The OSCS initiative chooses high-end computer systems as its target to relieve the side effect of IT decoupling. To generate a positive change force to overcome the ecosystem inertia, I use Zhan's laws of technology [8] to guide this project (The laws of technology are summarized in Section Two). On the one hand, the OSCS initiative will generate a positive change force through the open-source movement. On the other hand, it proposes an innovative methodology to improve the efficiency or other fundamental dimensions to generate the change force. The OSCS essential is to utilize the inherent characteristics of a class of representative workloads (benchmarks [14]) and propose innovative abstraction and methodology to co-explore the software and hardware design spaces of high-end computer systems, attaining peak performance, security, and other fundamental dimensions.

I discuss the four challenges of the OSCS initiative. The first challenge is the daunting system complexity witnessed by the high-end computer system and the processor ecosystem. The second challenge is how to perform the tradeoff between universal and ideal systems. For each class of representative workloads, there should be an ideal system architecture instead of a universal system where the performance, cost, or energy overhead of universality –"Turing Tax", or "Turing Tariffs" – cannot be avoided [15]. The third challenge is to propose the methodology and tools to aid the community in designing systems with guaranteed quality of computation results and performance under different conditions, e.g., best-case, worst-case, or average-case. Last but not least, it is how to balance legal, patent, and license issues of the OSCS initiative.

To tackle the first challenge, I propose the funclet abstraction and methodology. The funclet abstract represents the common proprieties of basic building blocks at different layers: each funclet is a well-defined, evolvable, reusable, independently deployable, and testable functionality with modest complexity; Each funclet interoperates with other funclets through the standard bus interfaces or interconnections. Four basic building blocks are chiplet, HWlet, envlet, and servlet at the chip, hardware, environment management, and service layers, and they form the four-layer funclet architecture. I present a three-tuple (funclet set architecture (FSA), organization, system specifics): the FSA refers to the actual programmer-visible function set [16], 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 the system built from funclets.

The structure of this article is as follows. Section Two presents the background knowledge of Zhan's three laws of technology. Section three justifies the motivation for the OSCS initiative. Section Four discusses the challenges. Section Five presents the funclet methodology. Section Six concludes.

## 2. Background

Zhan's three laws of technology provide a simple theoretical framework to explain and predict the rise or fall of a technology [8]. In this article, I use Zhan's three laws of technology [8] as a theoretical framework to analyze the potential and pitfall of the OSCS initiative. This section briefly introduces this framework.

The first law is on the obstacle to new technology: technology inertia. Not only end-users but also industry users stick to the existing technology, named consumer inertia and ecosystem inertia. The user size will keep constant unless a non-zero net technology change force acts on it. The second law reveals where the power of new technology comes from. The change in user size is proportional to the net technology change force. The corollary of measurement of technology change force is how to measure the net change force. By creating a brand-new technology or improving an existing technology in terms of user experience, costs, efficiency, or other fundamental dimensions by several orders of magnitude can the new technology generate a positive change force. In improving an existing technology, a new or different ecosystem will generate a negative change force $F_{Ecosystem}$, which is the side effect of ecosystem inertia. Meanwhile, different use which results in a learning cost will generate a negative change force $F_{Learn}$. According to the Equation in Table 1, the net change force $F_t$ is the sum of six components: $F_{Learn}$, $F_{Ecosystem}$, $F_{Experience}$, $F_{Cost}$, $F_{Efficiency}$, $F_{Other}$. Table 1 summarizes the three laws and five corollaries together. Table 2 explains the symbols in the formula in Table 1.

Table 3 presents how to use Zhan's laws of technology to analyze the rise or fall of a technology, the details of which are available from [8].

## 3. Motivation

As shown in Fig. 2, this section explains the motivations from two perspectives: Why is IT decoupling not wise? Why launch the OSCS initiative?

### *3.1. Why is IT decoupling not wise?*

IT infrastructure is the backbone of human society and the enabler that copes with the global pandemic and climate change challenges. For example, the scientific and engineering community heavily relies upon supercomputers to find the COVID-19 drugs and model climate change [17–19]. IT decoupling will hinder knowledge sharing and engineering collaboration, weakening our ability to handle pressing challenges. The decoupled IT communities must address severe challenges separately and amortize the unaffordable research and development costs from software and hardware.

The corollary of technology openness of Zhan's three laws of technology [8] clearly stated that contrasted with a closed ecosystem controlled by one entity, allowing the division of labor among contributors who share an open technology ecosystem improves the gross productivity and lowers the cost amortized on each contributor. Supply chain decoupling and technology export control between nations will

**Table 1**
The OSCS initiative uses Zhan's three laws of technology to decide the project's goal and strategy. The three laws and three corollaries used in this article are summarized while omitting the other two corollaries. The full details are shown in [8].

| Law or Corollary name | Formula |
|---|---|
| Law of technology inertia | $\Delta U_t = U_{(t+\Delta t)} - U_t$<br>$\Delta U_t = 0$ , $U$ *is a natural number* |
| Law of technology change force | $\Delta U_t \propto F_t$ , $U$ *is a natural number* |
| Law of technology change action and reaction | $F_{Emerging} = -F_{Existing}$ |
| Corollary of measurement of technology change force | $F_t = F_{Create} + F_{Learn} + F_{Ecosystem}$<br>or<br>$F_t = F_{Experience} + F_{Cost} + F_{Efficiency} + F_{Other} + F_{Learn} + F_{Ecosystem}$ |
| Corollary of technology breakthrough | $B = F/U$ , $U$ *is a natural number* |
| Corollary of technology openness | $P = p * U_i,\ c = C/U_i$<br>$P = p * U_i/M,\ c = C * M/U_i$<br>$P = p * U_i/N,\ c = C * N/U_i$ |

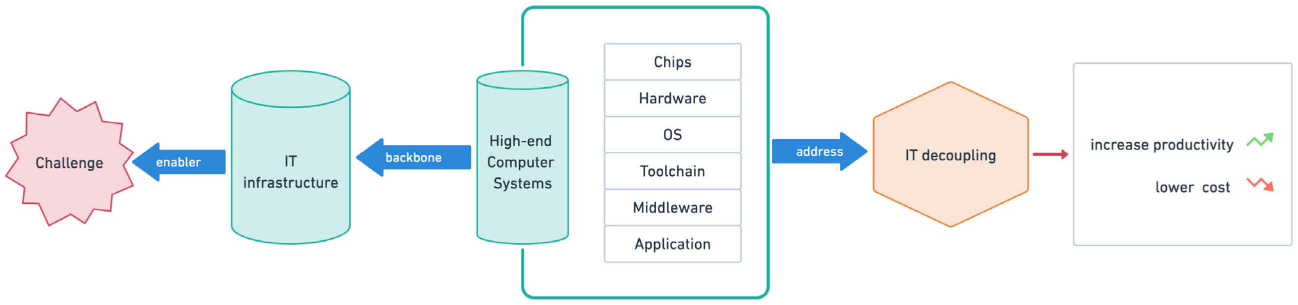

**Fig. 2.** The motivation for launching the open-source computer systems (OSCS) initiative.

**Table 2**
The explanations of symbols in Table 1 [8].

| Symbol | Explanation |
|---|---|
| $\Delta$ | Difference operator |
| $\propto$ | Proportional operator |
| $\Sigma$ | Summation operator |
| $U_t$ | User size varying with time |
| $U_i$ | Size of industry users |
| $F_t$ | Net technology change force varying with time |
| $F_{Emerging}$ | Change force acting on emerging technology |
| $F_{Existing}$ | Change force acting on existing technology |
| $F_{Create}$ | Change force resulted from creating a brand-new technology |
| $F_{Learn}$ | Change force resulted from learning cost |
| $F_{Ecosystem}$ | Change force resulted from ecosystem deviation |
| $F_{Experience}$ | Technology change force resulted from user experience |
| $F_{Cost}$ | Change force resulted from cost |
| $F_{Efficiency}$ | Change force resulted from efficiency |
| $F_{Other}$ | Change force resulted from other fundamental dimensions |
| $B$ | Technology breakthrough |
| $P$ | Gross productivity |
| $C$ | Total cost |
| $M$ | Number of decoupled supply chains |
| $N$ | Number of nations |
| $c$ | Cost |
| $p$ | Productivity of each contributor (industry user) |

increase costs and lower productivity. The dire IT decoupling will distract and weaken human beings' ability to handle those pressing challenges. This calls for our wisdom and actions to unify our science and technology community. The open science initiatives [9–12,12,13] partially responded to it.

### 3.2. Why launch the OSCS initiative?

The open-source software movement has become mainstream, like closed-source ones. The open-source software outspring includes Linux, Android, and many other software stacks. Opensource hardware is sporadic with a handy of hardware components that attain the performance and reliability that amount to the commodity components. However, it is far from ready to handle IT decoupling challenges.

First, as demonstrated in Table 4, IT infrastructure like high-end computer systems kept closed even open-source movement makes excellent progress. As shown in Fig. 2, high-end computer systems not only serve as the cornerstone of the IT infrastructure, but its components, like chips, hardware, OS, toolchain, and middleware, are also the basis for building IT infrastructure. So high-end computer systems are vital to addressing the challenges of IT decoupling.

The Open Compute Project Foundation (OCP) [30] was initiated in 2011 with a mission to open-source datacenter hardware (warehouse-scale computing) in mind; it still makes little progress in essential components. RISC-V [31] is an open standard instruction set architecture (ISA) following reduced instruction set computer (RISC) principles. The open-source chip project like RISC-V is promising as it is provided under open source licenses that do not require fees to use, unlike most other ISA designs [31]. For example, Institute of Computing Technology at Chinese Academy of Sciences has showcased progress on a fully open-source RISC-V processor, XiangShan, or "Fragrant Hills" [32], which is promising in competing Arm counterparts.

Second, fundamental changes in technology favor domain-specific hardware and software co-design, e.g. end of Dennard scaling, ending of Moore's Law, Amdahl's Law and its implications for ending 'easy' multicore era [33,34]. In the new golden age of computer architecture, it is essential to consider software and hardware together. So it is time for opensource software movements to converge with opensource hardware movements.

Third, merely repeating the methodology and process and replicating the closed-source counterpart will not succeed directly. According to the law of technology change force [8], the open-source initiative will make $F_{Cost} > 0$. Improving the closed-source counterpart in terms of user experience, costs, efficiency, or other fundamental dimensions, the open-source one can generate other positive components of the change force, e.g., $F_{Experience}$ – technology change force resulted from user experience, $F_{Efficiency}$ – change force resulted from efficiency, $F_{Other}$ – change force resulted from other fundamental dimensions.

**Table 3**
The analysis of thirteen successful IT using Zhan's three laws of technology [8]. The essential of using those laws is to measure the components of change force. The net change forces decide each technology's rise or fall. Contrasted with a closed ecosystem controlled by one entity, allowing the division of labor among contributors sharing an open technology ecosystem improves the gross productivity and lowers the costs amortized on each contributor [8]. Decoupling the supply chain will increase costs and lower productivity [8].

| Technology | Rivals | $F_{Learn}$ | $F_{Ecosystem}$ | $F_{Create}$ | $F_{Cost}$ | $F_{Efficiency}$ | $F_{Experience}$ | $F_{Other}$ | Closed/Open | Supply Chain |
|---|---|---|---|---|---|---|---|---|---|---|
| Deep learning | Shallow neutral networks | 0 | <0 | / | <0 | / | / | ≫0 (accuracy) | Open | Coupling |
| WWW | No | <0 | <0 | ≫0 | / | / | / | / | Open | Coupling |
| Google | No | <0 | <0 | ≫0 | / | / | / | / | Closed | Decoupling |
| Facebook | No | <0 | <0 | ≫0 | / | / | / | / | Closed | Decoupling |
| Internet | No | <0 | <0 | ≫0 | / | / | / | / | Open | Coupling |
| RAID | Single large expensive disk | 0 | 0 | / | >0 | >0 | / | / | Open | Coupling |
| Android | Windows Mobile, Symbian, iOS, Linux | <0 | 0 | / | >0 | / | >0 | >0 (Google ecosystem) | Open | Coupling |
| iOS | Windows Mobile, Symbian | <0 | <0 | / | <0 | / | ≫0 | >0 (AppStore) | Closed | Coupling |
| Windows | DOS | <0 | 0 | / | / | / | >0 | / | Closed | Coupling |
| Linux | UNIX | 0 | 0 | / | >0 | / | / | / | Open | Coupling |
| UNIX | Multics | <0 | <0 | / | / | >0 | / | >0 (standard) | Closed | Coupling |
| ARM | X86, RISC | 0 | 0 | / | / | >0 | / | >0 (energy efficiency) | Closed | Decoupling |
| RISC | CISC | 0 | 0 | / | / | >0 | / | / | Closed | Decoupling |

**Table 4**
Eight categories of high-end computer systems.

| Domain | Benchmark [14] | Metrics | Status | OSCS target |
|---|---|---|---|---|
| Planet-scale computers (Distributed IoTs, Edges, and datacenter systems) [20] | ScenarioBench [21] | Undefined | Not yet mature | Yes |
| AI for science | SAIBench [22] | Undefined | Not yet mature | Yes |
| Deep learning | AIBench [23] or MLPerf [24,25] | State-of-the-quality | Mature | No |
| Metaverse | MetaverseBench [26] | N/A | Not yet mature | Yes |
| High performance computing | HPCC [27] | FLOPS | Mature | No |
| Warehouse-scale computing | N/A | Throughput, Tail latency | Mature | No |
| Big Data | BigDataBench [28] or BigBench [29], | Throughput, Quality of services, turnaround | Mature | No |
| Cloud computing | N/A | System utilization, Quality of services | Mature | NO |

It is necessary to take other actions to generate other positive components of the change force. On the one hand, it is essential to leverage the inherent characteristics of a class of representative workloads (The second category of benchmarks in [14]) to co-explore the software and hardware architecture space [33,34]. On the other hand, it is necesary to develop new abstraction, methodology, and architecture in the OSCS initiative.

Fourth, being compatible with the ecosystem and users' learning habits is essential. Not only end-users but also industry users stick to the existing technology, which is consumer inertia and ecosystem inertia [8]. According to the corollary of measurement of technology change Force of Zhan's laws of technology, A new ecosystem or the deviation from existing technology ecosystems will generate a negative change force $F_{Ecosystem}$; Different use, which results in an end-user learning cost, will generate a negative change force $F_{Learn}$.

I conclude the essence of the OSCS initiative. The OSCS initiative has four implications. (1) It is an open-source movement where software converges with hardware. (2) It is to utilize the inherent characteristics of a class of representative workloads (The second category of benchmarks in [14]). (3) Instead of re-inventing the wheel, it is to propose innovative abstraction and methodology to co-explore the software and hardware design space to attain peak performance, security, and other fundamental dimensions. (4) it emphasizes compatibility with the ecosystem and users' learning habits. Fig. 1 reveals the essence of the OSCS initiative visually.

According to the Corollary of measurement of technology change force, the goal of the strategy is to maximize the positive value of the net change force. High-performance computing, cloud computing, and warehouse-scale computing are mature. Hence, generating a net change force that breaks through the technology inertia is much more challenging, i.e., improving an existing technology in terms of user experience, costs, efficiency, or other fundamental dimensions by several orders of magnitude. So I choose three emerging areas: planet-scale computers, which redesign the IoTs, edges, data centers and networks as a computer [20], AI for sciences, and Metaverse as the initial three targets of the OSCS initiative. I enforce the following strategies for each class of systems to maximize the net change force. (1) open-source initiative, which makes $F_{Cost} > 0$; (2) sharpen the edges like efficiency ($F_{Efficiency} > 0$, and the other fundamental dimension $F_{Other} > 0$; (3) provide the compatible ecosystem and lower the learning cost. ($F_{Learn} = 0$, $F_{Ecosystem} = 0$).

## 4. The challenges

This subsection will present the high-level challenges of the OSCS initiative. Low-level challenges are thoroughly discussed in [20,22,26].

(1) The challenge of system complexity.

I demonstrate the system complexity from two dimensions: the high-end computer systems and the processor ecosystem.

As shown in Fig. 2, high-end computer systems are the cornerstone of IT infrastructure with daunting complexity. As a case study, I review the state-of-the-art supercomputers on the Top 500 list [35]. Fugaku held the No. 1 position that it first earned in June 2020. Fugaku is based on Fujitsu's custom ARM A64FX processor, each with four NUMA nodes. With each NUMA node having 12 compute cores, each processor has 48 cores. Fugaku has 7,630,848 cores – using Fujitsu's Tofu-D interconnect to transfer data between nodes– achieving an HPL benchmark score of 442 Pflop/s [35]. High-end computer systems are a vivid demonstration of system engineering and art.

As I earlier analyzed in [8], an entire ecosystem of X86, ARM, or RISC-V processors consists of SoC (a system on a chip), ISA (Instruction Set Architecture), OS (operating system), toolchain, middleware, and applications. It is far beyond the reach of state-of-the-art and state-of-the-practice open-source projects. Even considering only the components, modern systems like Systems on Chips (SoCs) or Operating Systems have growing complexity that leads to a design productivity crisis [36]. For the SoC, the chipmaker shrinks different functions at each node and packs them onto a monolithic die, which becomes more complex and expensive at each node [37].

The high-end computer systems in Table 4 require aggressive, tactful, and coordinated plans for open-source computer systems.

(2) The tradeoff challenges between universal and ideal systems.

Turing [38] proposed the idea of a "universal" computing device – a single device that can implement any computable function. However, it will introduce "Turing Tax", or "Turing Tariffs" [15]: the performance, cost, or energy overhead of universality — the difference between a special-purpose device and a general-purpose one. Because industrial users adhere to existing products, tools, platforms, and services for investment protection — so-called ecosystem inertia [8], the user tends to opt for the universal systems or much narrow-scope general-purpose systems like GPGPUs. Modern systems, e.g., Systems on chip (SoCs) or Operating Systems, have growing complexity that leads to a design productivity crisis [36], and the communities cannot afford the cost of building new systems. That is another reason users prefer the universal or much narrower-scope general-purpose systems.

However, the new technology trends have gained momentum. The agile software and hardware process and methodology [36,39] enables small-team to build competitive high-performance microprocessors and software systems quickly. For each class of representative workloads, there should be an ideal system architecture instead of a universal system. Single-purpose system is not preferred; however, it is reasonable to tradeoff between the universal and ideal systems. Specifically, it motivates each class of workloads to explore the ideal architecture space.

(3) The challenges of guaranteeing the quality of computation results and performance under best-case, worst-case, or average-case.

Modern systems care about performance and value the quality of their computation results. For example, the deep learning community has come to terms with accepting the time of training an AI model to achieve a state-of-the-art quality as the primary metrics [23,25,40] in evaluating the system, which values both systems and algorithms.

I think the community should consider the quality of computation results and performance under different cases like best-case, worst-case, or average-case as first-class constraints in system design, implementation, verification,[1] and validation.[2] For example, how does a system achieve state-of-the-art quality with the 99th tail latency of fewer than 100 milliseconds? How to estimate or guarantee the performance and quality that satisfy a threshold in the best case (1 out of 1000) – consider the case of searching for alien civilizations? How about the average case? The design, implementation, verification, and validation costs vary wildly in different cases. The designers need to propose a new methodology to design, implement, verify and validate the systems with guaranteed confidence in quality and performance. Proposing the appropriate metrics is the first step in any-case system challenges. It is pressing to propose a new design, implementation, verification, and validation methodology and tool to address any-case system challenges.

I do not mean that these issues were totally overlooked in the past. HPC communities have much expertise in the tradeoff between performance and accuracy. Langou et al. [41] exploit single-precision operations whenever possible and resort to double-precision at critical stages while attempting to provide the full double precision results. The real-time system community [42] considered the hard and soft deadlines as first-class design constraints. In warehouse-scale computing [43], the tail latency of large-scale internet service – worst-case latency expressed in a percentile term, e.g., 98th, 99th – becomes the primary metric that outweighs the average latency. Lu et al. [44] claimed the modern data center operating system should gracefully achieve disparate performance goals in terms of both average and worst-case performance.

(4) The challenges of balancing legal, patent, and license issues.

In the past several decades, many different software models have become mainstream, such as the proprietary software product vendor model, the custom software developer model, advertising-supported software, and web-delivered software as a service [45]. How the OSCS initiative interacts and impacts those models? The license and patent policy may significantly impact that process. The situation becomes much more complex, especially considering the potential issues that may come to courts with the commercial project that hybridizes proprietary and open-source mechanisms [45]. Legal issues like long-arm jurisdiction in export control further complicate these situations.

The OSCS initiative is not a perfect plan. Also, it has several side effects. The most prominent one is how to prevent extremists or terrorism from leveraging open-source high-end computer systems.

## 5. The funclet methodology

This section presents the methodology for tackling the first challenge. I left the other challenges in the future work.

When the IT pioneer Gordon Moore [46] envisioned the future, he concluded "it might prove to be more economical to build large systems out of smaller functions, which are separately packaged and interconnected". Inspired by this philosophy of building large systems out of smaller functions, I propose the funclet methodology. First, I present the funclet abstraction, then the funclet architecture.

The funclet abstract represents the common proprieties of basic building blocks at different layers. Each funclet has the following characteristics. (1) each contains a well-defined and evolvable functionality with modest complexity (2) each can be reusable in different contexts. (3) each can be independently tested and verified before integrating. (4) each can be independently deployable. (5) each can interoperate with other funclets through a well-defined bus interface or interconnection.

As shown in Fig. 3, I propose a four-layer funclet architecture. As a start, I reuse two emerging concepts to describe funclet at the first and fourth layers, and then I elaborate on the other layers. The first-layer funclet is a chiplet, which is an integrated circuit (IC) with modest complexity, providing well-defined functionality [37,47]; it is designed to be susceptible to integration with other chiplets, connected with a die-to-die interconnect scheme [37,47]. A chiplet differs from the traditional, monolithic system on chip (SoC) in the following way [37,47]: a chipmaker can mix and match chiplets to reduce product development times and costs by integrating pre-developed die in an IC package [37, 47].

The fourth-layer funclet is a servlet, an independently deployable and evolvable component that severs users with a well-defined and modest-complexity functionality. A servlet supports interoperability through standardized software bus [48]. Microservice [48] is a form of a servlet. Another related concept is cloud functions [49]. Cloud functions package as Function as a Service (FaaS) offerings [49], which represents the core of serverless computing. Cloud functions are the general-purpose elements in serverless computing, leading to a simplified and general-purpose cloud programming model [49].

The second-layer funclet is an HWlet, an independently deployable, replaceable, and accessible hardware component, e.g., CPU, memory, storage. An HWlet could be aggregated into a resource pool with low latency and high bandwidth interconnection. I explain the HWlet concept in the context of an aggregated architecture [50–52]. For example, an HWlet could be a commodity memory module in a disaggregated memory design. Multiple compute blades can access an array of commodity memory modules that are encapsulated in a separate shared memory blade via a shared blade interconnect [52].

The third-layer funclet is an envlet, which is an independently deployable and evolvable environment component with well-defined functionality that supports the management of servlets. Envlets can interoperate through interconnections to form a management infrastructure. In the context of serverless computing, BaaS (Backend as a Service) [49] is a form of envlet. BasS provides the management services for cloud functions, responsible for the latter's automatic scaling with no need for explicit provisioning and usage-based billing [49].

[1] Verification determines whether each component and the assembly of components correctly meets its specification through testing("Are we building the thing right?") [36].

[2] Validation ensures that the product serves its intended purposes ("Are we building the right thing?") [36].

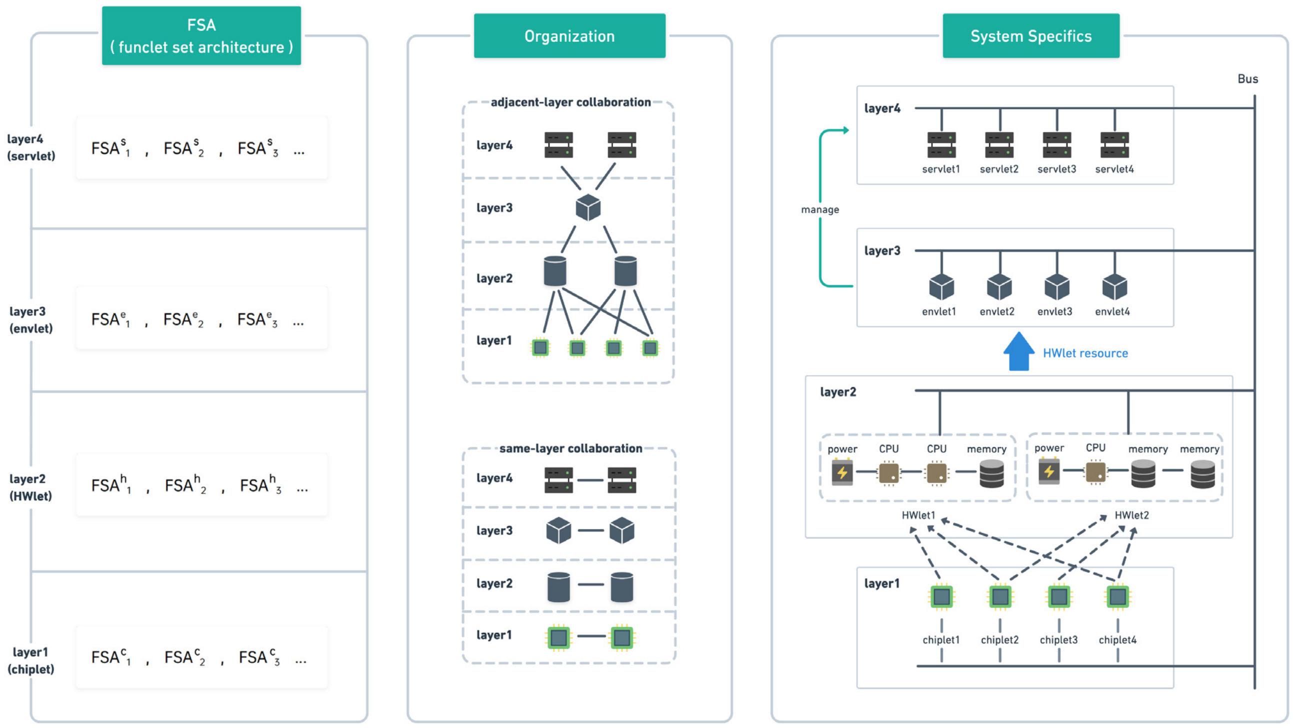

**Fig. 3.** An instance of the four-layer funclet architecture.

I use a methodology inspired by the computer architecture community to explicitly specify how funclets work together. I first present the methodology from the computer architecture community [16]. Then I elaborate on my proposed methodology.

In computer architecture, the terms instruction set architecture (ISA), organization or microarchitecture, and hardware are used to describe the architecture — the computer's design and implementation [16]. The ISA refers to the actual programmer-visible instruction set, serving as the boundary between the software and hardware [16]. The organization includes the high-level aspects of a computer's design, such as the memory system, the memory interconnects, and the design of the internal processor or CPU [16]. The hardware refers to the specifics of a computer, including the detailed logic design and the packaging technology of the computer [16].

I use a three-tuple {funclet set architecture (FSA), organization, system specifics} methodology to describe the funclet architecture, as shown in Fig. 3. The FSA refers to the actual programmer-visible function set [16], 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.

The funclet methodology specifies the architecture space in terms of (FSA, organization, system specifics). There is an explosive architecture space when searching for the optimal design. There is a pressing need for tools aiding the exploration of the funclet architecture. As each chiplet, HWlet, envlet, and servlet provide narrow-scoped functionality, it is essential to align the functions of the funclets that have similar resource-consumption characteristics for whole-stack optimization and resource management. In this context, the implication of the open-source computer systems initiative is to utilize the inherent characteristics of a class of representative workloads to co-explore the hardware and software design space in terms of (FSA, organization, system specifics) to attain peak performance and security.

I call the traditional system architecture, which consists of a monolithic chip, hardware, management environment, and application or services — the monolithic architecture. Meanwhile, some architectures have partially used the chiplet, microservice, or cloud functions, which I call hybrid architecture.

The advantage of the funclet architecture is four-fold. (1) it increases technology openness, improves productivity, and lowers cost. The funclet philosophy and methodology facilitate the contributors to focus on each funclet, allowing the efficient division of labor among contributors sharing an open technology ecosystem [8]. It finally improves the gross productivity and lowers the cost amortized on each contributor [8]. (2) it can help tackle the challenge of the complexity of computer systems from the horizontal and vertical dimensions. The system is divided among chiplet, HWlet, Envlet, and servlet from the vertical dimension. The system is described from FSA, organization, and system specifics from the horizontal and vertical dimensions. (3) it can help optimize the whole-stack system. With the close alignment of the functions of a servlet, Envlet, HWlet, and chiplet, the system can be optimized across the whole stack for narrow-scoped workloads with similar characteristics, e.g., resource requirements. (4) it can improve the reusability. Each funclet can be independently designed, implemented, and tested. Finally, the users can assemble the funclets into a complex system. (5) it can improve reliability. Before assembling, each funclet can be independently deployed and tested, enhancing the whole system's reliability.

Here, I emphasize the goal of the funclet architecture is not to replace all the monolithic or hybrid architectures. Instead, the former complements the latter. Similarly, Zhan's three laws of technology [8] will govern how they compete in the future.

## 6. Conclusion

This article initiates an open-source computer system (OSCS) movement. The OSCS initiative is an open-source movement where software converges with hardware. It is to utilize the inherent characteristics of a class of representative workloads and propose innovative abstraction and methodology to co-explore the software and hardware design space to attain peak performance, security, and other fundamental dimensions. I discuss four OSCS challenges: the daunting system complexity, the tradeoff between universal and ideal systems, guaranteeing best-case, worst-case, or average-case quality and performance, and balancing legal, patent, and license issues.

I propose the funclet abstraction and architecture to tackle the system complexity challenges. The funclet abstraction is a well-defined, evolvable, reusable, independently deployable, and testable functionality with modest complexity. Each funclet interoperates with other funclets through standard bus interfaces or interconnections. Four funclet building blocks: chiplet, HWlet, envlet, and servlet at the chip, hardware, environment management, and service layers form the four-layer funclet architecture.

## Acknowledgments

I am very grateful to Mr. Shaopeng Dai and Mr. Qian He for drawing Figs. 1, 2, and 3, Dr. Lei Wang for contributing to Fig. 3 and proofreading the article, Dr. Wanling Gao for proofreading this article.

## References

[1] C.-.E.M. Collaborators, Estimating excess mortality due to the COVID-19 pandemic: a systematic analysis of COVID-19-related mortality, 2020–21, Lancet 21 (6) (2022) 691–708.
[2] D. Adam, Covid's true death toll: much higher than official records, 2022, https://www.nature.com/articles/d41586-022-00708-0.
[3] WWF, IMPACTS of Global Climate Change.
[4] Engineering, and Medicine and others National Academies of Sciences, Information Technology Innovation: Resurgence, Confluence, and Continuing Impact, National Academies Press, 2020.
[5] National Research Council and others, Continuing Innovation in Information Technology, The National Academies Press, 2012.
[6] National Research Council, Innovation in Information Technology, The National Academies Press, 2003.
[7] National Research Council and others, Funding a Revolution: Government Support for Computing Research, National Academies Press, 1999.
[8] J. Zhan, Three laws of technology rise or fall, BenchCouncil Trans. Benchmarks Stand. Eval. (2022) 100034.
[9] S. Friesike, T. Schildhauer, Open science: many good resolutions, very few incentives, yet, in: Incentives and Performance, Springer, 2015, pp. 277–289.
[10] A.B. Powell, Open culture and innovation: integrating knowledge across boundaries, Media, Culture Soc. 37 (3) (2015) 376–393.
[11] J.M. Pearce, Open-Source Lab: How To Build Your Own Hardware and Reduce Research Costs, Newnes, 2013.
[12] A. Katz, Towards a functional license for open hardware, IFOSS L. Rev. 4 (2012) 41.
[13] P.A. David, Towards a cyberinfrastructure for enhanced scientific collaboration: providing its' soft'foundations may be the hardest part, 2004.
[14] J. Zhan, Call for establishing benchmark science and engineering, BenchCouncil Trans. Benchmarks Stand. Eval. 1 (1) (2021) 100012.
[15] P.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.
[16] J.L. Hennessy, D.A. Patterson, Computer Architecture: A Quantitative Approach, Elsevier, 2019.
[17] Z. Shervani, I. Khan, T. Khan, U.Y. Qazi, et al., World's fastest supercomputer picks COVID-19 drug, Adv. Infect. Dis. 10 (03) (2020) 211.
[18] D. Adam, Simulating the pandemic: What COVID forecasters can learn from climate models, Nature 587 (7835) (2020) 533–535.
[19] H. Fu, J. Liao, J. Yang, L. Wang, Z. Song, X. Huang, C. Yang, W. Xue, F. Liu, F. Qiao, et al., The sunway TaihuLight supercomputer: system and applications, Sci. China Inf. Sci. 59 (7) (2016) 1–16.
[20] ComputerCouncil, The IoTs, edges, datacenter and networks as a computer: Building open-source planet-scale computers (PSC) for emerging and future computing, 2022, https://www.computercouncil.org/PSC.
[21] W. Gao, F. Tang, J. Zhan, X. Wen, L. Wang, Z. Cao, C. Lan, C. Luo, X. Liu, Z. Jiang, Aibench scenario: Scenario-distilling ai benchmarking, in: 2021 30th International Conference on Parallel Architectures and Compilation Techniques (PACT), IEEE, 2021, pp. 142–158.
[22] Y. Li, J. Zhan, SAIBench: Benchmarking AI for science, BenchCouncil Trans. Benchmarks Stand. Eval. (2022) 100034.
[23] F. Tang, W. Gao, J. Zhan, C. Lan, X. Wen, L. Wang, C. Luo, Z. Cao, X. Xiong, Z. Jiang, et al., Aibench training: balanced industry-standard ai training benchmarking, in: 2021 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), IEEE, 2021, pp. 24–35.
[24] P. Mattson, C. Cheng, G. Diamos, C. Coleman, P. Micikevicius, D. Patterson, H. Tang, G.-Y. Wei, P. Bailis, V. Bittorf, et al., Mlperf training benchmark, Proc. Mach. Learn. Syst. 2 (2020) 336–349.
[25] Y.-H. Chang, J. Pu, W.-m. Hwu, J. Xiong, Mlharness: A scalable benchmarking system for mlcommons, BenchCouncil Trans. Benchmarks, Stand. Eval. 1 (1) (2021) 100002.
[26] ComputerCouncil, Metaversebench: Instantiating and quantifying metaverse problems, benchmarks, and challenges, 2022, https://www.computercouncil.org/MetaverseBench.
[27] P.R. Luszczek, D.H. Bailey, J.J. Dongarra, J. Kepner, R.F. Lucas, R. Rabenseifner, D. Takahashi, The HPC Challenge (HPCC) benchmark suite, in: Proceedings of the 2006 ACM/IEEE Conference on Supercomputing, Vol. 213, 10.1145, pp. 1188455–1188677.
[28] L. Wang, J. Zhan, C. Luo, Y. Zhu, Q. Yang, Y. He, W. Gao, Z. Jia, Y. Shi, S. Zhang, et al., Bigdatabench: A big data benchmark suite from internet services, in: 2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA), IEEE, 2014, pp. 488–499.
[29] A. Ghazal, T. Rabl, M. Hu, F. Raab, M. Poess, A. Crolotte, H.-A. Jacobsen, Bigbench: Towards an industry standard benchmark for big data analytics, in: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, 2013, pp. 1197–1208.
[30] The Open Compute Project Foundation, The open compute project, 2022, https://www.opencompute.org/.
[31] A. Waterman, Y. Lee, D.A. Patterson, K. Asanovi, The risc-v instruction set manual. volume 1: User-level isa, version 2.0, Technical Report, California Univ Berkeley Dept of Electrical Engineering and Computer Sciences, 2014.
[32] Gareth Halfacree, Chinese chip designers hope to topple arm's cortex-a76 with XiangShan RISC-v design, 2021, https://www.theregister.com/2021/07/06/xiangshan_risc_v/.
[33] J.L. Hennessy, D.A. Patterson, A new golden age for computer architecture, Commun. ACM 62 (2) (2019) 48–60.
[34] W. Gao, J. Zhan, L. Wang, C. Luo, D. Zheng, F. Tang, B. Xie, C. Zheng, X. Wen, X. He, et al., Data motifs: A lens towards fully understanding big data and ai workloads, in: Proceedings of the 27th International Conference on Parallel Architectures and Compilation Techniques, 2018, pp. 1–14.
[35] P.H.J. Kelly, Still waiting for exascale: Japan's fugaku outperforms all competition once again, 2021, https://www.top500.org/news/still-waiting-exascale-japans-fugaku-outperforms-all-competition-once-again/.
[36] Y. Lee, A. Waterman, H. Cook, B. Zimmer, B. Keller, A. Puggelli, J. Kwak, R. Jevtic, S. Bailey, M. Blagojevic, et al., An agile approach to building RISC-V microprocessors, Ieee Micro 36 (2) (2016) 8–20.
[37] Mark Lapedus, The good and bad of chiplets, 2020, https://semiengineering.com/the-good-and-bad-of-chiplets/.
[38] R. Herken, The Universal Turing Machine a Half-Century Survey, Springer-Verlag, 1995.
[39] M. Fowler, J. Highsmith, et al., The agile manifesto, Softw. Develop. 9 (8) (2001) 28–35.
[40] C. Coleman, D. Narayanan, D. Kang, T. Zhao, J. Zhang, L. Nardi, P. Bailis, K. Olukotun, C. Ré, M. Zaharia, Dawnbench: An end-to-end deep learning benchmark and competition, Training 100 (101) (2017) 102.
[41] J. Langou, J. Langou, P. Luszczek, J. Kurzak, A. Buttari, J. Dongarra, Exploiting the performance of 32 bit floating point arithmetic in obtaining 64 bit accuracy (revisiting iterative refinement for linear systems), in: SC'06: Proceedings of the 2006 ACM/IEEE Conference on Supercomputing, IEEE, 2006, p. 50.
[42] P.A. Laplante, et al., Real-Time Systems Design and Analysis, Wiley New York, 2004.
[43] L.A. Barroso, U. Hölzle, The datacenter as a computer: An introduction to the design of warehouse-scale machines, Synthesis Lect. Comput. Arch. 4 (1) (2009) 1–108.
[44] G. Lu, J. Zhan, C. Tan, X. Lin, D. Kong, C. Zheng, F. Tang, C. Huang, L. Wang, T. Hao, Isolate first, then share: a new os architecture for datacenter computing, 2016, arXiv preprint arXiv:1604.01378.
[45] G.R. Vetter, Commercial free and open source software: knowledge production, hybrid appropriability, and patents, Fordham L. Rev. 77 (2008) 2087.
[46] G.E. Moore, et al., Cramming more components onto integrated circuits, 1965.
[47] T. Li, J. Hou, J. Yan, R. Liu, H. Yang, Z. Sun, Chiplet heterogeneous integration technology—Status and challenges, Electronics 9 (4) (2020) 670.
[48] I. Nadareishvili, R. Mitra, M. McLarty, M. Amundsen, Microservice architecture: aligning principles, practices, and culture, " O'Reilly Media, Inc.", 2016.
[49] E. Jonas, J. Schleier-Smith, V. Sreekanti, C.-C. Tsai, A. Khandelwal, Q. Pu, V. Shankar, J. Carreira, K. Krauth, N. Yadwadkar, et al., Cloud programming simplified: A berkeley view on serverless computing, 2019, arXiv preprint arXiv:1902.03383.
[50] J. Fan, M. Chen, Dynamic self-organized computer architecture based on grid-components(DSAG), J. Comput. Res. Develop. 40 (12) (2003) 1737–1742.
[51] K. Lim, J. Chang, T. Mudge, P. Ranganathan, S.K. Reinhardt, T.F. Wenisch, Disaggregated memory for expansion and sharing in blade servers, ACM SIGARCH Comput. Archit. News 37 (3) (2009) 267–278.
[52] K. Lim, Y. Turner, J.R. Santos, A. AuYoung, J. Chang, P. Ranganathan, T.F. Wenisch, System-level implications of disaggregated memory, in: IEEE International Symposium on High-Performance Comp Architecture, IEEE, 2012, pp. 1–12.

Dr. **Jianfeng Zhan** is a Full Professor at Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), and University of Chinese Academy of Sciences (UCAS), the director of Research Center for Advanced Computer Systems, ICT, CAS. He received his B.E. in Civil Engineering and MSc in Solid Mechanics from Southwest Jiaotong University in 1996 and 1999, and his Ph.D. in Computer Science from Institute of Software, CAS, and UCAS in 2002. His research areas span from Chips, Systems to Benchmarks. A common thread is benchmarking, designing, implementing, and optimizing a diversity of systems. He has made substantial and effective efforts to transfer his academic research into advanced technology to impact general-purpose production systems. Several technical innovations and research results, including 35 patents, from his team, have been adopted in benchmarks, operating systems, and cluster and cloud system software with direct contributions to advancing the parallel and distributed systems in China or even in the world. He has supervised over ninety graduate students, post-doctors, and engineers in the past two decades. Dr. Jianfeng Zhan founds and chairs BenchCouncil and serves as the Co-EIC of TBench with Prof. Tony Hey. He has served as IEEE TPDS Associate Editor since 2018. He received the second-class Chinese National Technology Promotion Prize in 2006, the Distinguished Achievement Award of the Chinese Academy of Sciences in 2005, and the IISWC Best paper award in 2013, respectively.