BenchCouncil Transactions
on Benchmarks, Standards and Evaluations

TBench
Volume 2, Issue 2
2022

Editorial

A BenchCouncil view on benchmarking emerging and future computing
Jianfeng Zhan

Research Article

SAIBench: Benchmarking AI for Science
Yatao Li, Jianfeng Zhan

An efficient encrypted deduplication scheme with security-enhanced proof of ownership in edge computing
Yukun Zhou, Zhibin Yu, Liang Gu, Dan Feng

Short Communication

Asynchronous memory access unit for general purpose processors
Luming Wang, Xu Zhang, Tianyue Lu, Mingyu Chen

Survey Article

Performance and energy consumption tradeoff in server consolidation
Belen Bermejo, Carlos Juiz

ISSN: 2772-4859

BenchCouncil Transactions on Benchmarks, Standards and Evaluations (TBench) is an open-access multi-disciplinary journal dedicated to benchmarks, standards, evaluations, optimizations, and data sets. This journal is a peer-reviewed, subsidized open access journal where The International Open Benchmark Council pays the OA fee. Authors do not have to pay any open access publication fee. However, at least one of the authors must register BenchCouncil International Symposium on Benchmarking, Measuring and Optimizing (Bench) (https://www.benchcouncil.org/bench/) and present their work. It seeks a fast-track publication with an average turnaround time of one month.

Contents

A BenchCouncil view on benchmarking emerging and future computing 1
Jianfeng Zhan

SAIBench: Benchmarking AI for Science 10
Yatao Li, Jianfeng Zhan

An efficient encrypted deduplication scheme with security-enhanced proof of ownership in edge computing 20
Yukun Zhou, Zhibin Yu, Liang Gu, Dan Feng

Asynchronous memory access unit for general purpose processors 31
Luming Wang, Xu Zhang, Tianyue Lu, Mingyu Chen

Performance and energy consumption tradeoff in server consolidation 35
Belen Bermejo, Carlos Juiz

TBench Editorial Board 40

TBench Call For Paper 41

Bench 2022 Call For Paper 43

BenchCouncil Transactions on Benchmarks, Standards and Evaluations 2 (2022) 100064

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/

A BenchCouncil view on benchmarking emerging and future computing

Jianfeng Zhan

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

ARTICLE INFO

Keywords:
Benchmark science and engineering
Benchmarking challenges
Extrinsic property
Process entanglement
Instantiation bias
Unified benchmark definition
Conceptual framework
Benchmarking methodology
Traceability
Supervised learning
Emerging computing
Future computing
BenchCouncil Plan

ABSTRACT

The measurable properties of the artifacts or objects in the computer, management, or finance disciplines are extrinsic, not inherent — dependent on their problem definitions and solution instantiations. The processes of problem definition, solution instantiation, and measurement are entangled. Only after the instantiation can the solutions to the problem be measured. Definition, instantiation, and measurement have complex mutual influences. Meanwhile, the technology inertia brings instantiation bias — trapped into a subspace or even a point at a high-dimension solution space. These daunting challenges, which emerging computing aggravates, make metrology cannot work for benchmark communities. It is pressing to establish independent benchmark science and engineering.

This article presents a unifying benchmark definition, a conceptual framework, and a traceable and supervised learning-based benchmarking methodology, laying the foundation for benchmark science and engineering. I also discuss BenchCouncil's plans for emerging and future computing. The ongoing projects include defining the challenges of intelligence, instinct, quantum computers, Metaverse, planet-scale computers, and reformulating data centers, artificial intelligence for science, and CPU benchmark suites. Also, BenchCouncil will collaborate with ComputerCouncil on open-source computer systems for planet-scale computing, AI for science systems, and Metaverse.

1. Introduction

Benchmarking is widely practiced in different disciplines without a consensus on a consistent definition. For example, in the computer science discipline, the community uses a set of workload implementations to measure CPU (processor) performances [1,2]. In machine learning, standardized data sets labeled with ground truths are used to define a data science problem [3,4]. In the management discipline, the industry best practices are searched and compared against different products, services, and processes [5,6]. All are called benchmarks or benchmarking. In the previous work, I concluded five categories of benchmarks [6]: measurement standards, standardized data sets with defined properties, representative workloads, representative data sets, and industry best practices.

The inconsistency or chaos results from the following fact. Per JCGM 200 definition, metrology is the science of measurement and its application [7,8]. Metrology measures intrinsic properties independent of an observer, like length, time, and power. There is a true quantity for each inherent property, where a probability could state the coverage interval containing the true value [7,8]. However, the measurable properties of the artifacts or objects in the computer, management, or finance disciplines are extrinsic, not inherent — dependent on their problem definitions and solution instantiations. Unlike the processes in metrology that are linear and static, the processes of a benchmark have complex mutual influence. The problem definition, solution instantiation, and measurement processes are entangled and indivisible, and only after the instantiation can the solutions to the problem be measured, which I call process entanglement. Users adhere to existing products, tools, platforms, and services, called technology inertia [9]. The technology inertia traps the solution to a problem into a specific exploration path — a subspace or even a point at a high-dimension solution space. The instantiation bias impacts the measurement of the extrinsic properties.

Our society increasingly relies upon information infrastructure with daunting complexity that dwarfs the previous systems, making it difficult to trace the problem definition. Instead, the biased instantiation of solutions becomes the proxy of the problems, missing the forest for the trees. As shown in Fig. 1, these daunting challenges: extrinsic properties, process entanglement, and instantiation bias, result in the benchmark community's inability to reuse the metrology knowledge and the de facto isolation of benchmark communities, like computers, management, and finance, developing different methodologies, tools, and practices. It is pressing to establish independent benchmark science and engineering.

Echoing my past call [6], this article further builds up benchmark science and engineering. I define the benchmark from the perspective of problems and solutions. A benchmark is an explicit or implicit

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

https://doi.org/10.1016/j.tbench.2022.100064

Available online 25 May 2022

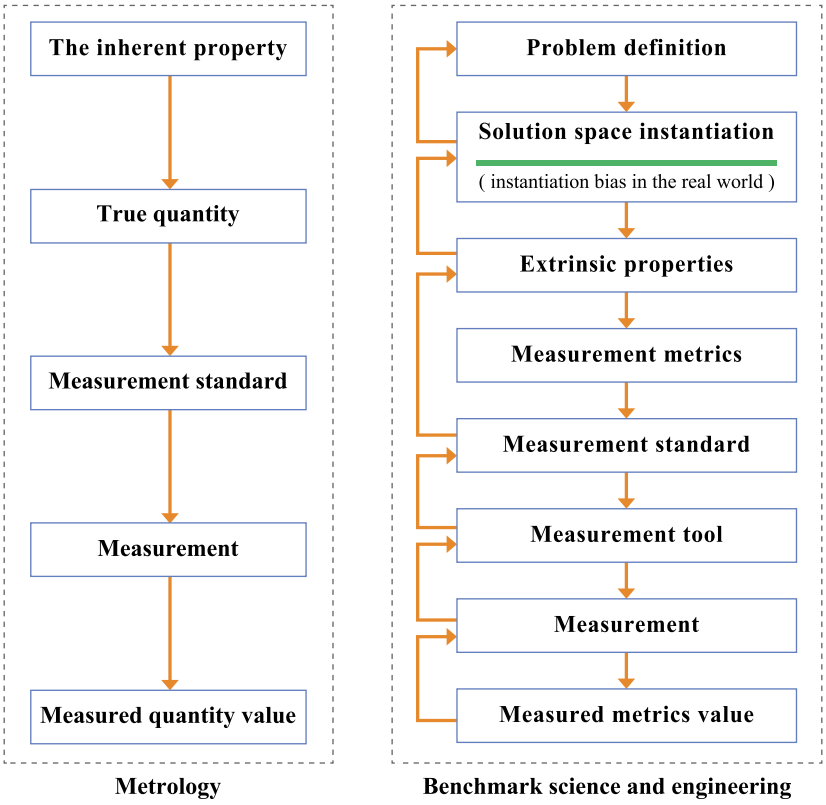


Fig. 1. With respect to metrology, the benchmarking challenges – the extrinsic properties, process entanglement, and instantiation bias – explain why metrology cannot work for the benchmark community. First, the property of a benchmark is not inherent but dependent on its problem definition and solution instantiation. Second, only after the instantiation can the solutions to the problem be measured. The processes of problem definition, solution instantiation, and measurement are entangled, and they have complex mutual influences. Third, instantiation introduces many biases.

definition of a problem, an instantiation of a problem, an instantiation of state-of-the-practice solutions as the proxy to the problem, or a measurement standard that quantitatively measures the solution space. I propose a concise conceptual framework for the benchmark science and engineering, at the core of which is the extrinsic properties. The extrinsic property is a benchmark property that is not inherent but dependent on a problem definition and solution instantiation. I propose a traceable and supervised learning-based methodology to tackle the challenges of extrinsic property, process entanglement, and instantiation bias. The essence of the methodology has two integrated parts: manage the traceability of the processes from the problem definition and solution instantiation to measurement; search for the best solution through supervised learning with reference to a thoroughly-understood process from the problem definition, solution instantiation to measurement.

Also, I discuss BenchCouncil's plan for emerging and future challenges. The ongoing projects include defining the challenges of intelligence, instinct, quantum computers, Metaverse, planet-scale computers, and reformulating data centers, artificial intelligence for science, and CPU benchmark suites. Also, BenchCouncil will collaborate with ComputerCouncil [10] on the open-source computer systems for Planet-scale computing [11], AI for science [12], and Metaverse [13].

The organization of this article is as follows. Section Two presents the background and challenges and explains why metrology cannot work for the benchmark community. Section Three describes why emerging computing aggravates the benchmark challenges. Section Four lays the foundation for benchmark science and engineering, including the unifying definition of benchmarks, the conceptual framework, and the benchmarking methodology. Section Five details BenchCouncil's plan. Section Six concludes.

2. Background and challenge: why metrology cannot be directly reused for benchmark science and engineering

In this section, I first introduce the metrology concepts as background and then present the benchmarking challenges and explain why metrology cannot work in the benchmark community.

2.1. Background: metrology concepts

As shown in Fig. 2, I provide a simple but systematic metrology concept framework to clarify why metrology cannot be directly reused for establishing benchmark science and engineering. I present and modify most of those concepts from [7,8]. But I define some concepts to emphasize why metrology cannot work for the benchmark community, e.g., inherent properties. To keep concise, I only stay with necessary metrology concepts.

The inherent property is a property of a phenomenon, body, or substance that is independent of an observer, e.g., length and energy [7]. The inherent property can have various magnitudes. True quantity is the magnitude of an inherent property of an individual phenomenon, body, or substance that is independent of an observer, e.g., the radius of a given circle, the kinetic energy of an identified particle in a given system [7,8]. Unit of measurement [8] is a definition and its physical realization, used as a reference to assign a value to a true

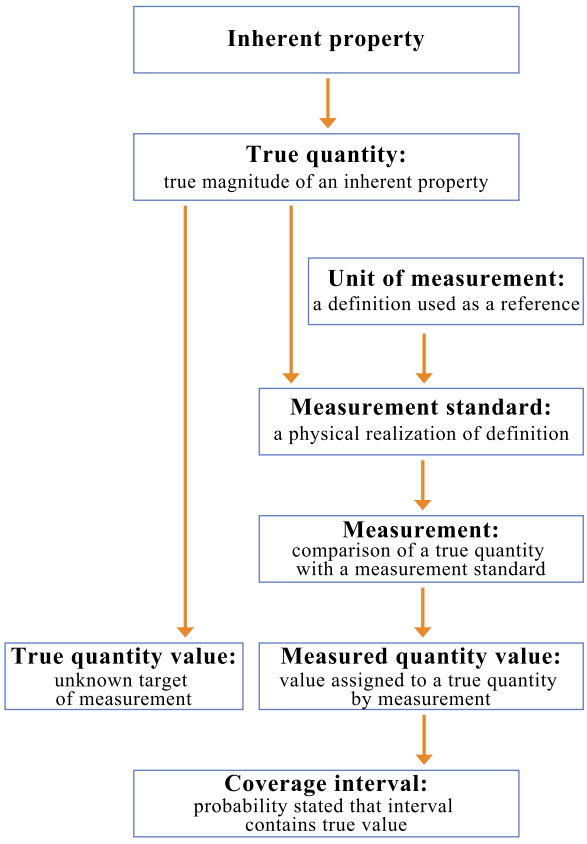


Fig. 2. A simple but systematic metrology conceptual framework is used to clarify why metrology cannot be directly reused for benchmark science and engineering. Some concepts are defined by myself, while the other concepts are reused or modified from [7,8]. Only necessary metrology concepts are reserved to keep concise.

quantity. Measurement standard [8] is a physical realization of a unit of measurement, with a stated quantity value and associated measurement uncertainty.

Measurement [8] is a process of comparing a true quantity with a measurement standard to assign the true quantity one or more quantity values that are traceable to a unit of measurement. A quantity value obtained by the measurement is referred to as a measured quantity value [7]. True quantity value [7] is a quantity value consistent with the definition of a quantity, which is an unknown measurement target [7]. A coverage probability [7] is a probability that a specified coverage interval contains the true quantity value.

2.2. The benchmarking challenges: extrinsic properties, process entanglement, and instantiation bias

In the previous work [6], I have noticed the differences in properties of the artifacts or objects in the computer, management, or finance disciplines from those classical ones, like length, time, and power. The properties of the artifacts or objects in the computer, management, or finance disciplines are extrinsic, dependent on their problem definitions and solution instantiations. Instead, the classical properties like time and length are inherent, independent of the observers. From a concept perspective, it is easy to say there are three essential processes: problem definition, solution instantiation, and measurement. However, a problem definition is abstract; only after the instantiation can the solutions to the problem be measured. Moreover, the problem definition, solution instantiation, and measurement processes are entangled and indivisible, which I call process entanglement. Only by fully understanding the side effect of the extrinsic properties and process entanglement can we avoid many traps. I elaborate on this viewpoint from different perspectives. Before proposing the conceptual framework for benchmark science and engineering (I defer it to Section 4.2), I stay with the metrology concepts in Section 2.1 to depict the challenges.

A subtle change in the definitions of a problem may lead to wildly varied solutions and significantly different measured quantity value. I take the classical matrix multiplication problem [14,15] as an example. Blalock et al. [15] reformulate the classical matrix multiplication problem as follows. The following reformulation is cited from [15]. A and B are two matrices. A is R^{NxD} and B is R^{DxM}, $N \gg D >= M$. Given a computation time budget τ, the task constructs three functions g(·), h(·), and f(·), along with constants α and β, such that

$$\|\alpha f(g(A), h(B)) + \beta - AB\|_F < \epsilon(\tau)\|AB\|_F \quad (1)$$

for as small an error $\epsilon(\tau)$ possible. For this reformulated problem, they introduce a learning-based algorithm that greatly outperforms existing methods [15]. This is a typical example of a subtle change in the definitions of a problem leading to wildly varied solutions and significantly different measured quantity values.

Furthermore, the solutions instantiations at different levels also interplay with each other and finally impact measured quantity values. The obvious example is deep learning. Algorithms and neural network architectures play a significant role. The hardware implementations, like different precision, e.g., single-precision, double-precision, or mixed precision, impact the learning dynamics. Even for the same system with different scales, the interactions among system size and mini-batch size significantly impact the measured quantity values like time-to-quality – the training time to achieve the state-of-the-art quality [16–20].

The processes in metrology are linear and static. However, for a benchmark, as shown in Fig. 1, the processes of problem definition, solution instantiation, and measurement are entangled, having complex mutual influences. The subtle difference in a problem definition will lead to a wildly varied solution, and its instantiation finally significantly impacts the measured quantity value. The solution instantiation provides the basis for measurement tools, and the latter uses state-of-the-practice instantiations that update frequently, which affects the measured quantity value. Also, the measured quantity values provide hints on searching for the best instantiation in the solution space.

Moreover, the instantiation introduces many biases, which I call instantiation bias. For example, in the computer system and architecture disciplines, Wang et al. [21,22] found that merely conducting microarchitecture-dependent or microarchitecture-independent, or ISA-independent (ISA is short for instruction set architecture) workload characterization (a form of measurement) will lead to misleading or erroneous conclusions. These significant differences in measured quantity values resulted from the solution instantiations at different levels themselves. Before performing microarchitecture-dependent or microarchitecture-independent, or ISA-independent workload characterization, the necessary step is instantiating a computer workload benchmark on a specific microarchitecture, a particular instruction set architecture, or an intermediate representation (very close to the source code), respectively. The community opts for the widely used ISA, IR (intermediate representation) for instantiation.

Matsuoka et al. also found the implementation of biases and complexity traps [23] in the instantiation process: on the one hand, any implementation of a computer workload benchmark entails multiple implicit biases towards algorithms, programming languages, data layouts, and parallelization approaches; on the other hand, the benchmarks, abstracted from large or legacy scientific codes and tuned for previous computer architectures, trap the co-design participants into considering only similar architectures.

Other observations are from the data sets, which many communities like machine learning use to explicitly or implicitly define a problem. It is prohibitively costly to build a representative and fidelity data set that can capture real-world characteristics. Hence, in reality, the goal often degrades to a workable data set. For example, for ImageNet [3],

it is easy to collect familiar animal and plant pictures, while the rare ones are difficult to obtain. Considering the data set is the cornerstone of many challenges like auto-driving and automatic medical diagnosis, this far-fetched methodology has many hidden flaws and risks.

3. Emerging computing aggravates the challenges[1]

Modern society is digitized, increasingly relying upon information infrastructure. The information infrastructure consists of massive Internet of Things (IoT), edge devices, data centers, and high-performance computers. Those systems collaborate to handle big data, train AI models and provide Internet services augmented by AI inference for huge end-users with guaranteed quality of services. From a benchmarking perspective, emerging computing like Big Data, AI, and Internet Services are significantly different from the traditional workloads characterized by SPECCPU (desktop workloads) [1], TPC-C [25], TPC-Web (Traditional web services) [26], and HPL (high-performance computing) [27] benchmarks, raising serious challenges.

The first challenge is fragmentation. There are substantial fragmented application scenarios, a marked departure from the past [24]. For example, hundreds or even thousands of ad-hoc big data solutions, termed NoSQL or NewSQL, are proposed to handle different application scenarios. For AI, the same observation holds. There are tens or even hundreds of organizations that are developing AI training or inference chips to tackle their challenges in different application scenarios, respectively [19,28].

The second challenge is de facto isolation. Internet service provider giants own and treat real-world data sets and workloads or even AI models as first-class confidential issues. The treasure is hidden in data centers and isolated between academia and industry, or even among different providers [29]. The dire situation poses a huge obstacle for our communities towards developing an open and mature research field [29].

The third challenge is the complexity of collaboration: HPC systems, data centers, edge, and IoT devices collaboratively handle the challenges; In the collaborations, different distributions of data sets, workloads, machine learning, or AI models may substantially affect the system's behaviors; the interaction patterns among IoT, edge, and data centers changes fast, embodying different architecture.

The fourth challenge originates from service-based architecture. On the one hand, the software-as-a-service (SaaS) development and deploy model makes the workloads change very fast (so-called workload churn) [30], and it is not scalable or even impossible to create a new benchmark or proxy for every possible workload [31]. On the other hand, modern Internet services adopt a microservice-based architecture, often consisting of various modules with long and complex execution paths across different data centers. As the worst-case performance (tail latency) [32] does matter, the micro-service-based architecture also poses a severe challenge to benchmarking [29,33].

The final but not least challenge is the stochastic nature of AI. AI techniques are widely used to augment modern products or Internet services. The nature of AI is stochastic, allowing multiple different but equally valid solutions [19]. Many factors manifest the uncertainties of AI, e.g., the adverse effect of lower-precision optimization on the quality of the final model, the impact of scaling training on time-to-quality, and run-to-run variation in terms of epochs-to-quality [19]. However, the measurement process mandates being repeatable (the same team) and reproducible (different teams). This conflict raises serious challenges.

Emerging computing aggravates the benchmarking challenges discussed in Section 2.2. First, it is difficult to trace the original problem definition, that is, the target to be achieved. Second, taking the instantiation of solutions as the proxy for the problem aggravates the instantiation bias and makes the community further trapped in the specific solutions.

[1] This section is written based on an unpublished technique report [24], of which I am the lead author.

4. Building up benchmark science and engineering

This section proposes the unifying definitions of benchmarks, the conceptual framework, and the benchmarking methodology, which lays the foundation for benchmark science and engineering.

4.1. The unifying definition of benchmarks

Previously, I concluded five categories of benchmarks [6]: measurement standards, standardized data sets with defined properties, representative workloads, representative data sets, and industry best practices. In this section, I give a simple and unifying definition to cover five categories of benchmarks and reveal their essence. A benchmark is an explicit or implicit definition of a problem, an instantiation of a problem, an instantiation of state-of-the-practice solutions as the proxy to the problem, or a measurement standard that quantitatively measures the solution space.

A benchmark has three essential processes, some of which often be omitted or implicitly stated in practice: definition, instantiation, and measurement. I explain the process of definition and instantiation from various perspectives in the rest paragraphs of Section 4.1. I leave the discussion of the process of measurement in Sections 4.2, 4.3.

4.1.1. Definition

The first process is the definition. Defining a problem explicitly or implicitly is the fundamental role that a benchmark could play in almost all disciplines. Only after clearly defining a problem can we figure out the solutions and compare them against the others. For example, Alan Turing, in 1950 [34] formulated the problem of what intelligence is as an imitation game: the game tests whether an interrogator can distinguish a machine's ability from a human. Turing's problem definition inspires several generations to explore the solutions to achieve intelligence.

There are many ways to define a problem, e.g., using a natural language or mathematics. From an accuracy perspective, mathematically defining the problem is a better choice. Unfortunately, many problems cannot be accurately depicted in this way.

The NAS parallel benchmarks [35] claimed that the common requirements should be specified in a paper-and-pencil approach [24]. A paper-and-pencil approach is a vague description — It can be mathematical, textual, or even visually. In the computer science discipline, this approach is well-practiced in the database community but not adopted in the computer architecture community.

Shun et al. [36] advocated a methodology to build benchmarks using problem definitions, and they created the problem-based benchmark suite (PBBS). PBBS is a set of benchmarks for comparing parallel algorithmic approaches, parallel programming language styles, and machine architectures across a broad set of problems. Specifically, a problem-based benchmark mandates a problem specification and a set of input distributions while not detailing the requirements in terms of algorithmic approach, programming language, or machine architecture [36].

4.1.2. Instantiation

The second process is instantiating a problem or instantiating state-of-the-practice solutions as the proxy to the problem or challenge. As a replacement or complement, these are two different ways. First, an instantiation of the problem is used. For example, a data set is often used to instantiate a problem in the machine learning community. Li et al. [12] further classified the problem definitions into problem class, problem settings, and problem cases. Second, an instantiation of state-of-the-practice solutions is used as the proxy for the problem. For example, the computer architecture community provides state-of-the-practice implementations of a group of computer workloads like SPECCPU [2,37]. SPECCPU is a proxy to the problems.

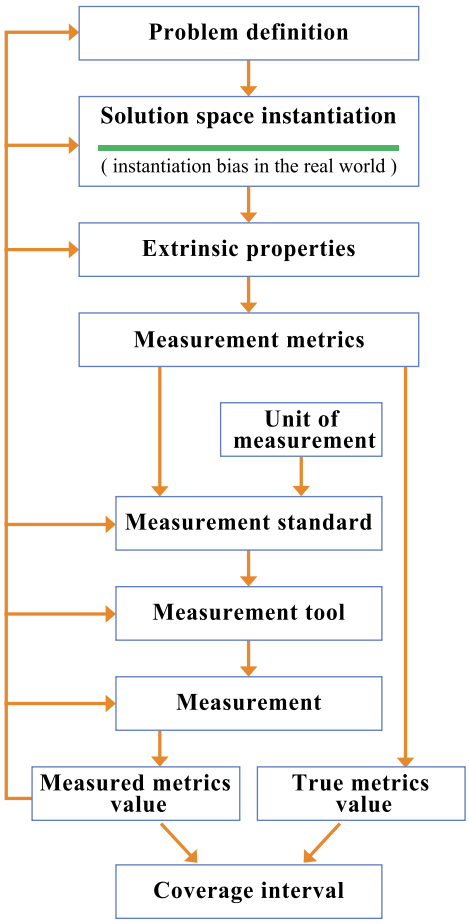


Fig. 3. The conceptual framework of benchmark science and engineering.

There are two reasons for this replacement or complement. First, as a replacement, it serves as the proxy for the problem that is too difficult to define. Second, as a complement, the instantiation brings enriched and necessary details that set more specific problem settings. Each instantiation is a subspace or point – which is often state-of-the-practice – in the solution space to the problem, e.g., using source code or binary code, which brings instantiation bias.

4.2. The conceptual framework of benchmark science and engineering

As shown in Fig. 3, I propose the conceptual framework of benchmark science and engineering. The extrinsic property is a benchmark property that depends on a problem definition and its solution instantiation. The extrinsic property can have various magnitudes. Measurement metrics are the magnitude of a benchmark's extrinsic property, which depends on a problem definition and solution instantiations. Unit of measurement [8] is a definition and its realization, used as a reference to assign a value to a measurement metric. Measurement standard [8] is a realization of a unit of measurement, with a stated metric value, associated measurement uncertainty, and a repeatable (the same team) and reproducible (different teams) measurement methodology. The measurement tool implements a measurement standard that can be calibrated and traceable. Traceability [7] is a property of a measurement result whereby the result can be related to a reference through a documented unbroken chain of calibrations, each contributing to the measurement uncertainty. The measurement tools should be open-sourced and can be replicated by different teams.

Measurement [8] is a process of comparing a measurement metric with a measurement standard to assign one or more measured values to a measurement metric that are traceable to a unit of measurement. A value obtained by the measurement is referred to as a measured metric value [7]. True metric value [7] is a value consistent with the definition of a measurement metric that is specific to the extrinsic properties of a concrete problem definition and solution instantiation. The true metric value is an unknown measurement target [7]. A coverage probability [7] is a probability that the true metric value is contained within a specified coverage interval.

4.3. The traceable and supervised-learning based benchmarking methodology

A benchmark has no inherent properties, and its extrinsic property is dependent on its problem definition and solution instantiation. Meanwhile, the processes of definition, instantiation, and measurement are entangled, and they have complex mutual influences.

I propose a traceable methodology to tackle the above challenge, at the core of which is to manage the traceability of the processes from the problem definition and solution instantiation to measurement. Fig. 4 shows that problem definition, solution instantiation, extrinsic properties, measurement standard, measurement tool, and measured metrics value have complex mutual influence. The problem definition is the origin of this relationship. No other below entities, like solution instantiation, can impact the problem definition directly. Still, solution instantiations may provide clues for the subtle change of the problem definition, affecting the other entities significantly. At the top level, I suggest a formal definition of problems and tracing the relationship among the different subtle definitions of problems. For many state-of-the-practice benchmarks, the definition process is omitted. It should regularly keep an eye on revisiting the process from the problem definition to solution instantiation, or else the outdated instantiation will be a trap. The solution instantiation provides the basis for measurement tools. It is mandatory to search for state-of-the-art or state-of-the-practice solutions and implement them in the measurement tool.

There is an explosion from the problem definition to solution instantiations. Put in other words, the lower level has more state space [6]. For example, there is increasing state space in a computer workload benchmark, from a mathematical problem definition, an algorithm, an intermediate representation, an ISA-specific representation, to a micro-architectural representation. The technology inertia traps the solution into a specific exploration path — a subspace or even a point at a high-dimension solution space, called instantiation bias. The instantiation bias impacts the measurement of the extrinsic properties. Also, an unguided exploration may drift away from optimized solutions.

I propose the supervised learning-based methodology to tackle the challenge of instantiation bias. Supervised learning is a machine learning branch that trains a predictive model using labeled data with known outcomes. Fig. 4 shows the thoroughly-understood process from the problem definition, solution space instantiation, extrinsic properties, measurement standard, measurement tool, to measured metrics value, standing as a ground truth. From this ground truth, it is easy to learn how the change of the top entities impacts the below. For example, if the problem is reformulated, the solution instantiation changes accordingly. Finally, the measured metrics values are significantly affected. The benchmark plays the role of connecting the problem with its solution space. By exploring the solution space and observing the effect of its change on the measured metrics values, it is possible to search for the best solution. This search process could leverage state-of-the-art deep learning techniques. Of course, this learning dynamic will be very complex. Fig. 5 shows an example on how to use this methodology in computer architecture.

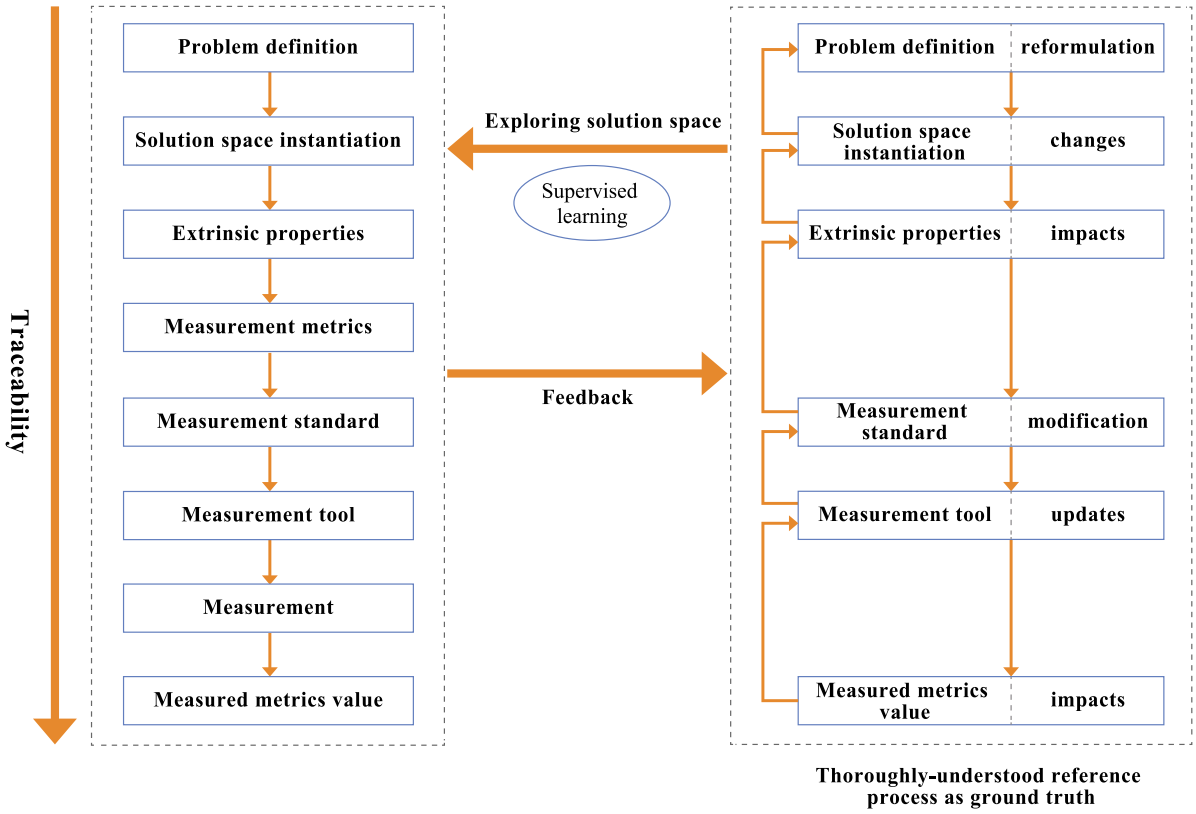


Fig. 4. A traceable and supervised learning-based benchmarking methodology to tackle the challenges of extrinsic property, process entanglement, and instantiation bias.

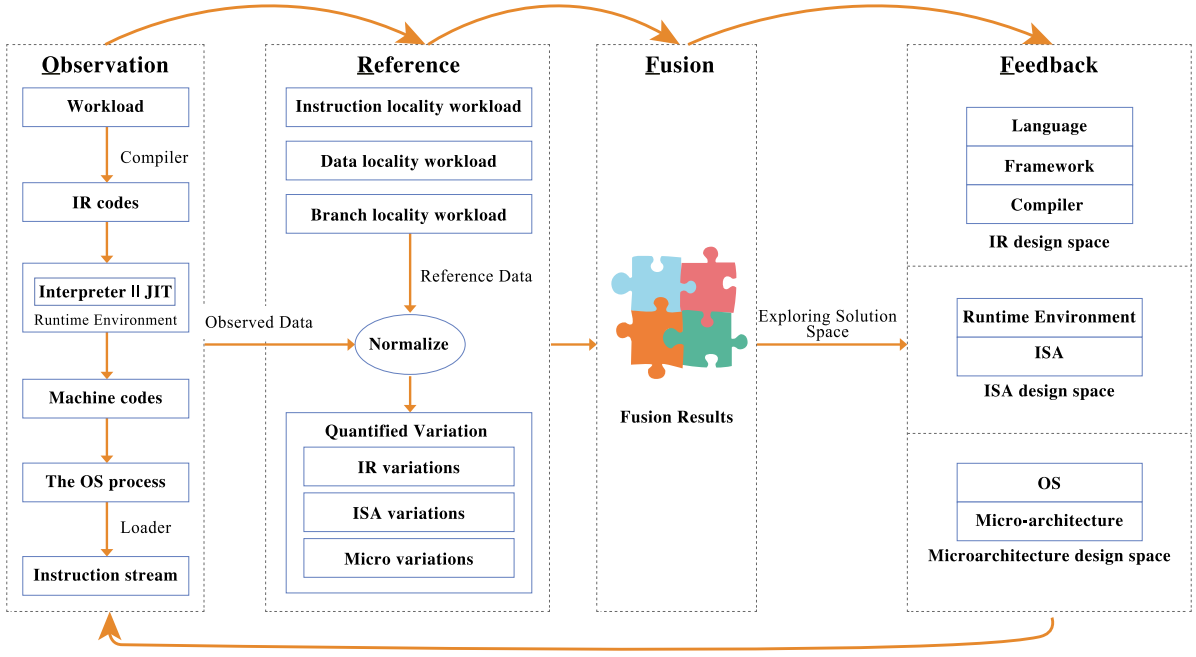


Fig. 5. The application of the traceable and supervision learning-based benchmarking methodology in computer architecture. This figure is cited from [22] with the permission of the authors.

4.4. Re-interpret five categories of benchmarks

I use the benchmark definition proposed in 4.1 to re-interpret five categories of benchmarks defined in [6].

The first category of the benchmark is a measurement standard used to measure the solution space to the problem. I use the Linpack benchmark – an example from high-performance computing – to explain this category of benchmarks. The Linpack benchmark [38] is widely used to report the performance of a high-performance computer. The problem definition of Linpack is a linear system of equations of an order n: $Ax = b$. The solution uses the LU factorization with partial pivoting. The measurement metrics are the floating-point operations count of the solving algorithm, which is $(2 * n^3/3 + 2 * n^2)$ operations, and the execution time of running the benchmark. HPL is one of the reference

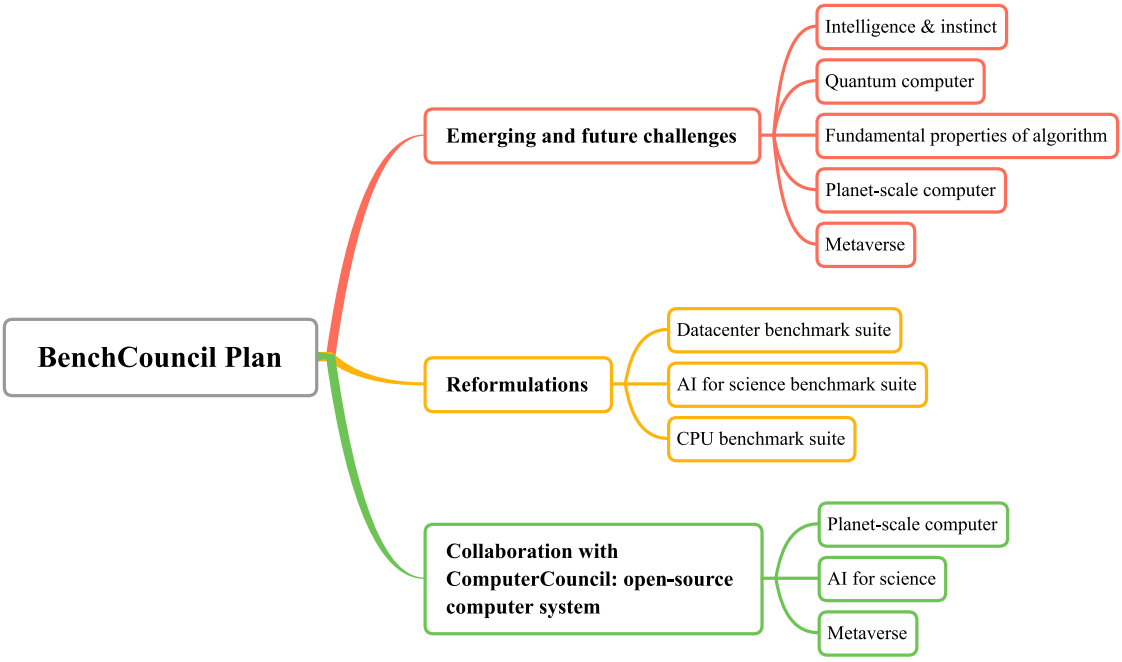


Fig. 6. BenchCouncil's plan on defining the challenges of emerging and future computing and the collaboration with ComputerCouncil on the open-source computer systems.

implementations of the measurement tool used to evaluate against different high-performance computer implementations (solutions). The measurement standard also details the reproducible and repeatable measurement methodology to compare against other solutions: users must report a residual for the accuracy of the solution with $\|Ax - b\|/(\|A\|\|x\|)$. The TOP500 list reports the measured metrics values. The measurement metrics highly depends on its problem definition and solution instantiations.

The second category of benchmarks is the representative workloads that run on the systems under measurement [6]. The representative workloads are the definition of the problem or an instantiation of state-of-the-practice solutions as the proxy to the problem. The problem-based benchmark suite (PBBS) [36], TPC-C [25], TPC-Web (Traditional web services) [26] are typical problem definition examples. They also provide the instantiation of state-of-the-practice solutions as the measurement tool. Without explicit problem definition, SPECCPU (desktop workloads) [1], BigDataBench [39], BigBench [40], AIBench [20,33] and MLPerf [19] are the instantiations of state-of-the-practice solutions and they serves as the proxy to the problem.

The third category of the benchmarks is the implicit definition of the problem using a standardized data set. The standardized data set represents a real-world data science problem with defined properties, some of which have ground truth [6,41]. ImageNet [3] (deep learning benchmark) and MIMIC-III [4] (critical care benchmark) are typical examples.

The fourth category of benchmarks is a representative data set, used as a Ref. [6]. This category of benchmarks is an instantiation of a problem. For example, an index (statistical measure) calculated from a representative set of underlying data and used as a reference for financial instruments or contracts [42] is a benchmark in finance. The London Interbank Offered Rate (Libor) and the Euro Interbank Offered Rate are well-known financial benchmarks [6,42].

The fifth category of benchmarks is the industry best practices in diverse domains [6]. Benchmarking is continuously searching the industry best practices with superior performance and measuring products, services, and processes against them [5,6]. The industry best practices are instantiations of the state-of-the-practice solutions to the problem or grand challenge.

5. BenchCouncil's plan on emerging and future computing

Fig. 6 presents BenchCouncil's plan for defining the challenges of emerging and future computing and collaborating with ComputerCouncil on the open-source computer systems. First, I introduce BenchCouncil's plan for emerging and future challenges.

First, what is intelligence? What is instinct? What is the distinction between intelligence and instinct? The pre-trained language models, such as BERT and GPT-3, seem to outperform the capability beyond the Turing test [43]. Many previous works have reformulated the problem of what intelligence is [44,45]. It is necessary to revisit the processes from the intelligence problem definition, solution instantiations to measurement. For example, there are many ways to solve these challenges to somewhat extent, including traditional machine learning, deep learning, and brain-inspired computing. Letting them compete in the same arena is essential. According to [46], instinct is an inborn impulse or motivation to action typically performed in response to specific external stimuli. But how do we distinguish intelligence from instinct? What are the differences between the octopus, birds, apes, and ants' behaviors? Are they intelligence or instinct?

Second, quantum computers emerge as a new computational paradigm with unprecedented capability [47]; what are the problems or grand challenges which the quantum computers do the best? How do state-of-the-practice computers compete against quantum computers in handling different or overlapping domains of problems or grand challenges? The community must ponder this fundamental issue before delving into different levels of instantiations of solutions.

Third, computer algorithms almost govern the running of our society. It is pressing to think, specify, verify and test what fundamental properties an algorithm must have before being embedded in our society. Think about Twitter and Facebook's impact on the election in many vote-based democratic societies. It is vital to propose benchmarks against those algorithms before putting them into practice.

Fourth, information infrastructure becomes the cornerstone of our society [10], and many fundamental applications like medical emergency management and smart cities applications rely upon planet-scale distributed systems consisting of massive Internet of Things (IoT) devices, edges and data centers, which I call planet-scale computers [11]. Different distributions of data sets, workloads, or AI models may substantially affect the system's behaviors, and the system architectures are

undergoing fast evolution regarding the interactions among IoT, edge, and data centers [24]. How can the community propose benchmarks for those ultra-scale emerging and future applications [33]?

Metaverse is an umbrella term. It is predicted to be a brand-new way for people to immersively access the Internet, interact with each other or digital avatars in the cyberworld, and manage digital assets. Though many industry giants are pushing towards those goals, the process itself is in a Cambrian explosion of different things in the forms of concepts, prototypes, products, or services. It is pressing to propose a benchmark suite to define the Metaverse problem or challenge, explore and evaluate state-of-the-art and state-of-the-practice solutions [13].

Many old problems need reformulation. For example, the Berkeley multidisciplinary groups proposed to use thirteen "Dwarfs" [48] – A dwarf is an algorithmic method that captures a pattern of computation and communication – to design and evaluate parallel programming models and architectures. When AI has seen as a new dawn in the traditional and emerging scientific area, how to reformulate those problems [12]?

Datacenters have become the fundamental infrastructure of modern society. There are substantial fragmented application scenarios in big data, AI, and Internet service areas, a marked departure from the past [24]. Virtualization technologies like containers are widely used as resource management and performance isolation facilities. However, the current BenchCouncil benchmark suites like BigDataBench [39] and AIBench [20,33] are fragmented without providing a full-picture definition of the problems or challenges in data centers. Moreover, a lack of simple but elegant abstractions prevents achieving both efficiency and general purpose [24]. For example, hundreds or even thousands of ad-hoc NoSQL or NewSQL solutions are proposed to handle different big data application scenarios [24]. Contrasted, the relation algebra is generalized for the database theory and practice, and any complex query can be written using five primitives like select, project, product, union, and difference [49]. Though domain-specific software and hardware co-design is promising [50], the lack of simple but unified abstractions has two side effects [24]: it is prohibitively costly to build an ad-hoc solution; single-purpose systems and architectures are structural obstacles to resource sharing. Proposing simple but elegant abstractions is an integrated part of managing the traceability of the process from the problem definition to solution instantiation.

The CPU benchmark suite like SPECCPU [2,37] advanced the evolution of different processor architectures. However, the SPEC CPU is an instantiation of state-of-the-practice solutions as the proxy to the problem, severely biased towards market-dominant CPU architecture, high-performance languages like C, and high-performance computing and desktop workloads. The BENCHCPU project [51] will propose a new CPU benchmark suite.

5.1. The collaboration with ComputerCouncil

As a non-profit international organization, the International Open-source Computer Council (ComputerCouncil) mission is to unite the science and technology community to tackle the challenges of information technology decoupling [52]. ComputerCouncil initiates the open-source computer system (OSCS) initiative to tackle the challenges of IT decoupling.

ComputerCouncil will choose three emerging areas: planet-scale computers — planet-scale distributed systems and applications built on IoTs, edges, and datacenters [11], AI for science [12], and Metaverse [13] as the initial targets of the OSCS initiative. BenchCouncil will cooperate with ComputerCouncil: the former focuses on the benchmarks, while the latter concentrates on the open-source computer systems for the three emerging areas.

6. Conclusion

This article concluded benchmarking challenges as extrinsic properties, process entanglement, and instantiation bias. The measurable properties of a benchmark are not inherent but dependent on their problem definitions and solution instantiations. The processes of problem definition, solution instantiation, and measurement are entangled and have complex mutual influences. The technology inertia leads to a specific exploration path –a subspace or even a point at a high-dimension design space. Those challenges make metrology cannot work for benchmark communities and call for independent benchmark science and engineering.

I proposed a unified benchmark definition, a conceptual framework, and a traceable and supervised learning-based benchmarking methodology, laying the foundation for benchmark science and engineering. A benchmark is an explicit or implicit definition of a problem, an instantiation of a problem, an instantiation of state-of-the-practice solutions as the proxy to the problem, or a measurement standard that quantitatively measures the solution space. At the core of the conceptual framework, the extrinsic property is a benchmark property that depends on a problem definition and its solution instantiation. The essence of the proposed benchmarking methodology has two integrated parts: manage the traceability of the processes from the problem definition and solution instantiation to measurement; search for the best solution through supervised learning with reference to the thoroughly-understood processes from the problem definitions and solution instantiations to measurements. Also, I elaborated BenchCouncil's plan to define emerging and future computing challenges and collaborate with ComputerCouncil on open-source computer systems.

Acknowledgments

I am very grateful to Mr. Shaopeng Dai for compiling the references, Mr. Shaopeng Dai and Mr. Qian He for drawing Figs. 1, 2, 3, 4, and 6, and Dr. Lei Wang for discussing and contributing significantly to the presentations of Figs. 1, 4, 6 and proofreading throughout this article. Fig. 5 is cited from [22] as an example with the permission of the authors. Section 3 and a part of Section 5 are based on the unpublished technical report [24], of which I am the lead author. The technical report [24] was based on my presentation at a BoF of SC 2019, and the web link is https://www.benchcouncil.org/file/BenchCouncil-SC-BoF.pdf. After the presentation, I drafted the technical report [24] as the first author. I am very grateful to the other authors for their discussions and contributions: Dr. Lei Wang, Dr. Wanling Gao, and Dr. Rui Ren.

References

[1] SPEC, SPEC CPU 2017 Benchmark, 2017, https://www.spec.org/cpu2017/.
[2] R. Panda, S. Song, J. Dean, L.K. John, Wait of a decade: Did SPEC CPU 2017 broaden the performance horizon? in: 2018 IEEE International Symposium on High Performance Computer Architecture, HPCA, IEEE, 2018, pp. 271–282.
[3] J. Deng, W. Dong, R. Socher, L.J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2009, pp. 248–255.
[4] A.E. Johnson, T.J. Pollard, L. Shen, L.w.H. Lehman, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. Anthony Celi, R.G. Mark, MIMIC-III, a freely accessible critical care database, Sci. Data 3 (1) (2016) 1–9.
[5] M. Zairi, P. Leonard, Origins of benchmarking and its meaning, in: Practical Benchmarking: The Complete Guide, Springer, 1996, pp. 22–27.
[6] J. Zhan, Call for establishing benchmark science and engineering, BenchCouncil Trans. Benchmarks Stand. Eval. 1 (1) (2021) 100012.
[7] I. BiPM, I. IFCC, I. IUPAC, O. ISO, The international vocabulary of metrology—basic and general concepts and associated terms (VIM), 2012, p. 2012, JCGM 200.
[8] R.N. Kacker, On quantity, value, unit, and other terms in the JCGM international vocabulary of metrology, Meas. Sci. Technol. 32 (12) (2021) 125015.
[9] J. Zhan, Three laws of technology rise or fall, BenchCouncil Trans. Benchmarks Stand. Eval. 2 (1) (2022) 100034.

[10] J. Zhan, Open-source computer systems initiative: The motivation, essence, challenges, and methodology, BenchCouncil Trans. Benchmarks Stand. Eval. 2 (1) (2022) 100038.
[11] 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.
[12] Y. Li, J. Zhan, SAIBench: Benchmarking AI for science, BenchCouncil Trans. Benchmarks Stand. Eval. (2022).
[13] ComputerCouncil, MetaverseBench: Instantiating and quantifying metaverse problems, benchmarks, and challenges, 2022, https://www.computercouncil.org/MetaverseBench.
[14] V.V. Williams, Multiplying matrices faster than coppersmith-winograd, in: Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing, 2012, pp. 887–898.
[15] D. Blalock, J. Guttag, Multiplying matrices without multiplying, in: International Conference on Machine Learning, PMLR, 2021, pp. 992–1004.
[16] Z. Jiang, L. Wang, X. Xiong, W. Gao, C. Luo, F. Tang, C. Lan, H. Li, J. Zhan, Hpc ai500: The methodology, tools, roofline performance models, and metrics for benchmarking hpc ai systems, 2020, arXiv preprint arXiv:2007.00279.
[17] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, K. He, Accurate, large minibatch sgd: Training imagenet in 1 hour, 2017, arXiv preprint arXiv:1706.02677.
[18] Y. You, Z. Zhang, C.J. Hsieh, J. Demmel, K. Keutzer, Imagenet training in minutes, in: Proceedings of the 47th International Conference on Parallel Processing, 2018, pp. 1–10.
[19] 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.
[20] 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.
[21] L. Wang, X. Xiong, J. Zhan, W. Gao, X. Wen, G. Kang, F. Tang, Wpc: Whole-picture workload characterization across intermediate representation, isa, and microarchitecture, IEEE Comput. Archit. Lett. 20 (2) (2021) 86–89.
[22] L. Wang, K. Yang, W. Gao, C. Luo, C. Wang, Z. Ge, L. Zhang, F. Zhang, J. Zhan, WPC: Whole-picture Workload Characterization, Technical Report, Institute of Computing Technology, Chinese Academy of Sciences, 2022.
[23] S. Matsuoka, J. Domke, M. Wahib, A. Drozd, R. Bair, A.A. Chien, J.S. Vetter, J. Shalf, Preparing for the future–rethinking proxy apps, 2022, arXiv preprint arXiv:2204.07336.
[24] J. Zhan, L. Wang, W. Gao, R. Ren, Benchcouncil's view on benchmarking ai and other emerging workloads, 2019, arXiv preprint arXiv:1912.00572.
[25] TPC, TPC-C Benchmarks, http://www.tpc.org/tpcc/.
[26] TPC, TPC-W Benchmarks, http://www.tpc.org/tpcw/.
[27] A. Petitet, HPL-a portable implementation of the high-performance linpack benchmark for distributed-memory computers, 2004, http://www.netlib.org/benchmark/hpl/.
[28] V.J. Reddi, C. Cheng, D. Kanter, P. Mattson, G. Schmuelling, C.J. Wu, B. Anderson, M. Breughe, M. Charlebois, W. Chou, et al., Mlperf inference benchmark, in: 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture, ISCA, IEEE, 2020, pp. 446–459.
[29] W. Gao, F. Tang, L. Wang, J. Zhan, C. Lan, C. Luo, Y. Huang, C. Zheng, J. Dai, Z. Cao, et al., AIBench: an industry standard internet service AI benchmark suite, 2019, arXiv preprint arXiv:1908.08998.
[30] L.A. Barroso, U. Hölzle, The datacenter as a computer: An introduction to the design of warehouse-scale machines, Synth. Lect. Comput. Archit. 4 (1) (2009) 1–108.
[31] 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.
[32] J. Dean, L.A. Barroso, The tail at scale, Commun. ACM 56 (2) (2013) 74–80.
[33] 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.
[34] A.M. TURING, I.—COMPUTING MACHINERY AND INTELLIGENCE, Mind LIX (236) (1950) 433–460.
[35] D.H. Bailey, Nas parallel benchmarks, in: Encyclopedia of Parallel Computing, 2011, pp. 1254–1259.
[36] J. Shun, G.E. Blelloch, J.T. Fineman, P.B. Gibbons, A. Kyrola, H.V. Simhadri, K. Tangwongsan, Brief announcement: the problem based benchmark suite, in: Proceedings of the Twenty-Fourth Annual ACM Symposium on Parallelism in Algorithms and Architectures, 2012, pp. 68–70.
[37] J.L. Henning, SPEC CPU2000: Measuring CPU performance in the new millennium, Computer 33 (7) (2000) 28–35.
[38] J.J. Dongarra, P. Luszczek, A. Petitet, The LINPACK benchmark: past, present and future, Concurr. Comput.: Pract. Exper. 15 (9) (2003) 803–820.
[39] 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.
[40] 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.
[41] MIT, AutoML benchmark datasets, 2021, [EB/OL]. https://openml.github.io/automlbenchmark/benchmark_datasets.html. (Accessed 2 December 2021).
[42] IOSCO, Financial Benchmarks, Technical Report, IOSCO, 2013.
[43] C.D. Manning, Human language understanding & reasoning, Daedalus 151 (2) (2022) 127–138.
[44] R.M. French, The Turing Test: the first 50 years, Trends Cogn. Sci. 4 (3) (2000) 115–122.
[45] C.H. Hoffmann, Is AI intelligent? An assessment of artificial intelligence, 70 years after turing, Technol. Soc. (2022) 101893.
[46] Britannica, instinct, 2022, https://www.britannica.com/topic/instinct.
[47] T. Tomesh, P. Gokhale, V. Omole, G.S. Ravi, K.N. Smith, J. Viszlai, X.C. Wu, N. Hardavellas, M.R. Martonosi, F.T. Chong, Supermarq: A scalable quantum benchmark suite, 2022, arXiv preprint arXiv:2202.11045.
[48] K. Asanovic, R. Bodik, B.C. Catanzaro, J.J. Gebis, P. Husbands, K. Keutzer, D.A. Patterson, W.L. Plishker, J. Shalf, S.W. Williams, et al., The landscape of parallel computing research: A view from berkeley, 2006.
[49] E.F. Codd, A relational model of data for large shared data banks, in: Software Pioneers, Springer, 2002, pp. 263–294.
[50] J. Hennessy, D. Patterson, A new golden age for computer architecture: Domain-specific hardware/software co-design, enhanced, in: ACM/IEEE 45th Annual International Symposium on Computer Architecture, ISCA, 2018.
[51] BenchCouncil, BENCHCPU, 2019, https://www.benchcouncil.org/benchcpu/index.html.
[52] ComputerCouncil, International opensource computer council, 2022, https://www.computercouncil.org/.

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.

BenchCouncil Transactions on Benchmarks, Standards and Evaluations 2 (2022) 100063

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/

SAIBench: Benchmarking AI for Science

Yatao Li [a,b,c,*], Jianfeng Zhan [a,b]

[a] Institute of Computing Technology Chinese Academy of Science, 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] Microsoft Research Asia, Building 2, No. 5 Dan Ling Street, Haidian District, 100080, Beijing, China

ARTICLE INFO

Keywords:
Scientific computing
Artificial intelligence
Benchmarking

ABSTRACT

Scientific research communities are embracing AI-based solutions to target tractable scientific tasks and improve research work flows. However, the development and evaluation of such solutions are scattered across multiple disciplines. We formalize the problem of scientific AI benchmarking, and propose a system called SAIBench in the hope of unifying the efforts and enabling low-friction on-boarding of new disciplines. The system approaches this goal with *SAIL*, a domain-specific language to decouple research problems, AI models, ranking criteria, and software/hardware configuration into reusable modules. We show that this approach is flexible and can adapt to problems, AI models, and evaluation methods defined in different perspectives. The project homepage is https://www.computercouncil.org/SAIBench.

1. Introduction

Artificial Intelligence has seen continuous and significant advancements over the past years, with Deep Learning methods being arguably the most representative and focused on. Blessed by the ever-increasing computation power in AI accelerators and general-purpose architectures alike, new AI paradigms and models are proposed which greatly improve the scalability, flexibility, and applicability of this data-driven approach. As a result, the IT industry is welcoming AI-powered solutions, integrating them into existing data processing pipelines that will otherwise require human intervention or prohibitive computation cost. This trend is also propagating into scientific research communities, as researchers are gaining interest in leveraging state-of-the-art AI solutions to tackle equally if not more difficult tasks, hence AI for Science.

From a bird's eye view, a scientific research activity can be mechanical or creative. A mechanical research activity can be algorithmically specified, with quantized or computationally verifiable input/output. On the other hand, a creative research activity breaks out of a mechanical system, for example, by defining a new problem or introducing ideas hard to quantify. In this work, we call a computationally verifiable research task a "tractable scientific task". That said, an AI for Science solution is introduced to bring improvements into the scientific research workflow and is usually targeting towards a tractable scientific task, such as:

- Mathematical Problem Solving — to solve mathematically well-defined problems.
- Pattern Matching — to classify, identify patterns, and detect region-of-interest in high volume scientific data.
- Prediction — to compute future world states, given an initial snapshot of the world state and evolving rules.
- Artifact enhancement — to improve the quality of data acquired from imperfect observations, e.g. incomplete, fragmented, noisy sensor data.
- Control — to use actuators to drive sensor readings into desired states, despite the imperfection of both.
- Hypothesis and Confirmation — to propose a theory (e.g. equations) that conforms with the observations.

Examples of these tasks are shown in Table 1. The term "AI for Science" is also conventionally deemed as an ensemble of vertical fields and tasks [1]. However, we argue that to fully realize the potential of AI for Science, it is not enough to cherry-pick an AI method, match it against a specific task, and heuristically compare it with existing methods. One strength of AI methods is that they abstract away the problem details and mathematical procedures, into generic functions that transform inputs into outputs — that is, every AI model possesses the potential to adapt to other tasks, some (for example, neural networks) even being universal approximators. Science is vast, and AI methods are many. A single effort to evaluate a taskmethod pair would leave other research communities unaware, of both the potential tasks that a model is capable of processing and potential models that can be applied to a task. This problem is exaggerated by the fact that the AI research is moving forward fast, that by the time a specific method is

* Corresponding author at: Institute of Computing Technology Chinese Academy of Science, No. 6 Kexueyuan South Road, Haidian District, 100190, Beijing, China.
E-mail addresses: yatli@microsoft.com (Y. Li), jianfengzhan@ict.ac.cn (J. Zhan).

https://doi.org/10.1016/j.tbench.2022.100063
Received 15 April 2022; Received in revised form 11 May 2022; Accepted 11 May 2022
Available online 24 May 2022

Table 1
Examples of tractable scientific tasks.

Mathematical Problem Solving	Partial derivative equations
	General matrix multiplication
	Matrix decomposition
	Integration
	Monte Carlo methods
Pattern Matching	Species Classification
	Event Identification [2]
	Climate Analysis [3]
	Anomaly Detection
Prediction	High-Energy Particle Simulation
	Molecular Dynamics
	Fluid Dynamics
	Protein Folding
Artifact Enhancement	Genome Sequence Alignment
	Astronomy Image Enhancement
	Medical Image Enhancement
	MRI Reconstruction
Control	Tokamak Plasma Control [4]
	Sensor Triggering
Hypothesis and Confirmation	Automatic Physics Laws Discovery
	Symbolic Regression

picked up by a scientific computing task, or a task is adapted to an AI method, it may be already bested by then state-of-the-art.

To help the scientific research communities as a whole systematically absorb and integrate the advancements of AI research, and to avoid repeated efforts in development and evaluation, we propose *SAIBench*, a system that bridges scientific computing tasks and AI methods, and automatically benchmarks every sensible combination, collects performance metrics, and projects it back into rankings proper to each research community. Research groups of different backgrounds can focus on their needs while taking advantage of other benchmarking building blocks, without having to re-invent end-to-end evaluation processes.

The rest of this article is organized as follows. We first define the problem of scientific AI benchmarking in Section 2. In Section 3 we discuss the methodology, goal, and challenges. Section 4 elaborates our system design, including the details of each component. We showcase end-to-end scenarios involving multiple modules in Section 5.

2. Problem definition

Here we define the problem of scientific AI benchmarking. To begin with, we have a set of tractable scientific tasks as defined in the previous section, and an array of AI methods, each needs to be trained to solve a specific problem. To evaluate a method for such tasks, different scientific communities have different criteria. For example, instruments in High Energy Physics generate zettabytes of data, and the training data for AI models is virtually unlimited. An AI method could thus focus on throughput, time-to-solution, sample selection, etc. Meanwhile, for Biology and Life Sciences, sometimes there are just a few hundred data points, requiring high sample efficiency, and a strong ability to generalize and extrapolate onto unseen problem configurations.

Nonetheless, the qualification of a method can be categorized as follows:

- **Defined by Problem Class.** For purely computational tasks such as mathematical problem solving, it is preferable to target against classes of problems to see how the method performs under each set of mathematical constraints. For example in equation solving, it is desired to study how a method behaves for both stiff and non-stiff systems, where both types contain their problem class definitions.
- **Defined by Problem Setting.** Compared to purely mathematical problem classes, this type of problem definition usually embodies specific constraints under a class to match a physical setting. Scientific research communities have established well-respected problem settings that have been practiced and confirmed. This allows computational methods to interoperate with real-world experiments, as specific experimental settings can be virtually replicated.
- **Defined by Problem Cases.** For some tasks we are only interested in a narrow range within the whole problem space. Most data-driven tasks fall into this category, where the typical workloads of a task are defined by collected and/or labeled data. There are also "golden standards" defined in research fields, which are computational methods with superior accuracy and other desirable properties, at expensive computational costs. These methods are then used to collect data for very specific problem cases so that other faster but less accurate methods can be developed and evaluated.

Table 2
Examples of qualification criterion.

Problem class	Boundary Value Problem
	Stochastic Differential Equations
	Many-body Interactions
	Positive Definite Matrix Decomposition
Problem setting	Temperature and pressure dependence of alanine dipeptide
	Straight wire Magnetostatics
	Community Atmosphere Model (CAM5) [5] Simulation
Problem case	ANI-1x [6], GDB-17 [7]
	OASIS [8]

This categorization is not mutually exclusive though, as some tasks require more than one qualification criteria to properly define the problem. For example, a robotic control algorithm can be tested both in a simulated setting and on data points collected from real-world sensors. Nevertheless, the principle is that this categorization describes the hierarchy of problem definition — the more the definition leans towards the former (problem classes), the more computation is required; On the other hand, the more towards the latter (problem cases), the more data. Furthermore, the problem definition serves as a specification for the AI model behavior, for both training and testing.

Examples of these qualifications are shown in Table 2. However, AI-based methods likely require training, so the problem definition of all three types must be reduced to case-by-case training data points — for a problem class, the problem definition should generate independent problem instances that sufficiently cover the problem space. For a problem setting, the problem definition should generate state snapshots that conform to the constraints. For data-driven problem cases, the problem definition should simply enumerate from the dataset.

Furthermore, the evaluation of a method depends on the problem definition generating tests. For each test case, the performance is represented with a cost function. For a mathematical problem instance, the cost function can be the error against ground truth solution, or error against equality constraints [9,10]. For simulation settings, the cost function can be obtained by comparing the performance metrics derived from such experiments, as shown by previous works on specific tasks [11–13]. Lastly, for data-driven problem cases, the dataset can be split into training and test sets, and the cost function is the loss function applied to the test set.

Finally, it is crucial to realize that different benchmarking communities use the word "performance" to refer to different concepts. Scientific AI benchmarking concerns not only the accuracy of AI models but also the computation cost. The computation cost can be further broken down into two phases: (1) the cost for a model to reach certain accuracy, and (2) once the model is properly trained, the cost of using the model for inference tasks. For the first phase, the standard practice is to measure training time (wall clock or total CPU/GPU time) against the best/mean/worst accuracies, and for the second, the throughput/latency etc. for completing the inference tasks.

Moreover, for both phases, we can investigate the system performance with standard parallel computing benchmarking techniques [14] to expose different performance characteristics of a solution, for example, time-to-solution or cost-efficiency.

3. Methodology

The main goal of *SAIBench* is to build an inclusive and interconnecting environment for all the relevant research efforts, including problem definition, AI method, training algorithm, software and hardware environment, metric definition and ranking definition, and deliver benchmarking result efficiently with given computation resources. The desiderata brought by this goal is multifold.

We need to design the system with a modular paradigm and provide friendly programming interfaces for different modules. It should handle the impedance mismatches between different programming languages and environments while maintaining consistent standards. This is traditionally implemented with language bindings (for example, the computational chemistry package NWChem [15] can either execute its own scripting language, or be controlled by a Python language binding) or file-based inter-process communication, which is suboptimal because different programming environments may have incompatible constructs that cannot be bound into a single process, and distributed computing modules cannot be modeled easily.

A module should be self-descriptive so that the system can automatically discover benchmarking tasks it can participate in, so in addition to modular interfaces targeting benchmarking tasks, there should be a protocol for modules to exchange metadata and relate to each other. It is challenging to design such a protocol because it has to be generic, extensible, and yet carrying concrete meanings. For example, if we model the input/output of an AI model as tensors of required dimensions, it places strict constraints on what the AI model can solve, and the system will not be able to associate this AI model with even slightly incompatible tensors, not to mention non-tensor data that has to be converted to adapt. On the other hand, if we simply attach a textual description to each module, it would be too hard for machine-understanding, and require human in- tervention to develop the connections. To this end, machine-understandable flexibility and extensibility are needed, to enable modules to cooperate less rigidly. The previous example shows how a module for an AI model should describe itself. Similarly, for a problem definition, it should programmatically set up the training and test fixtures, and conduct the experiments. This way all the three types of problem definitions previously can be normalized and become accessible to AI models. In addition, it should expose metadata that allows the system to inspect the execution workflow, and identify tasks that can be completed by other modules. This type of meta-programming is practiced in programming language research and recently machine learning frameworks, implemented in declarative languages and domain-specific languages (DSLs) [16], yet largely unexplored in scientific computing, where most execution engines take a parse–interpret–execute approach [15,17].

As we discussed above, the system is not a single benchmark, but a collection of such, to be projected back to each research field and aggregated by a ranking criterion. Conflict of interests naturally arises, for example, to favor speed vs. to favor accuracy, first principle metrics vs. a particular set of derived properties. The system should be able to allow different perspectives of the same metrics and provide an interface for ranking modules to declare their preferences.

The performance of an end-to-end AI solution to a tractable scientific task depends on multiple aspects, including the AI model, the training algorithm, the computing software stack, the empowering hardware, and so on. These factors do not contribute to the final performance linearly, for example, a particular AI model may have the best work-precision properties under one hardware configuration but not the others. It is thus desirable to consider all these factors as benchmarking hyperparameters. There are several implications brought by this requirement. The AI module implementation should be declarative instead of being bound to a specific software/hardware stack; The software stack module should declare the capabilities (e.g. matrix multiplication and backward propagation) so that the system finds compatible model-software pairs; Also, the software stack module should describe the hardware compatibility and accept a standardized hardware configuration descriptor, so that the system can automatically schedule scalability tests.

With all the components modularized and parameterized, the whole benchmarking workflow can be formulated as follows. Each type of module introduces some *dimensions* to the benchmarking task, and the goal is to enumerate and test against the Cartesian product of all such dimensions, where each point in the problem space represent the combination of a specific task, solver, metrics, software and hardware configuration. This allows the modules to advertise themselves, discover the others, and therefore reuse data and interact with each other, without knowing them beforehand. This paradigm aligns well with the FAIR guiding principles for scientific data management [18], which suggests that scientific data should have findability, accessibility, interoperability, and reusability. This is the key difference between the methodology of this work and previous AI benchmarking and scientific benchmarking systems, where the benchmarked scenarios are pre-determined workload and model combinations, and the addition of a new AI model or dataset would not be automatically discovered and reused by existing modules in the system and has to be scripted by a programmer.

Last but not least, because the system automatically discovers potential benchmarking tasks, it is desired that the system can concurrently schedule computation resources to them. As different benchmarking tasks may require different computing environments, it is crucial that the system can elastically provision the environment for each task in a standardized manner, including the operating system, runtime libraries, setup scripts, and test fixture data. The challenge lies in how to design the system to efficiently support such needs and minimize the deployment overhead.

4. System design

In this section, we illustrate the overall design of the system and tap into each system component, and discuss how to address the aforementioned challenges. The architecture of the system is depicted in Fig. 1. The workflow is straightforward. The planner pulls all modules from the module repository and joins them into feasible tuples according to the metadata descriptors. The execution plan is then dispatched to the elastic

computing platform which provides storage, processors, and accelerators, where each benchmarking task tuple is executed in a "benchmarking pod". The purpose of the BenchPod is to provide task-level isolation to computation resources, a communication endpoint to interact with the planner, and experiment orchestration. A problem definition module either generates data on-the-fly or retrieves a well-known dataset into the BenchPod instance. The hardware definition module acquires hardware resources. The software definition module constructs a containerized environment, based on a standardized software package requirement descriptor. The entry point of the container is a shim program provided by the BenchPod instance that orchestrates the actual execution of the solver, metric collection, and aggregation.

4.1. SAIL: Scientific AI domain-specific language

Previous AI benchmarking systems either implicitly define a series of built-in modules [19,20], or expose a markup language schema to define modules [21]. For better programmability, discoverability, and user ergonomics, we propose to define modules with an embedded domain-specific language (eDSL) called SAIL. The eDSL is implemented as a Python package so that a module implementer can take advantage

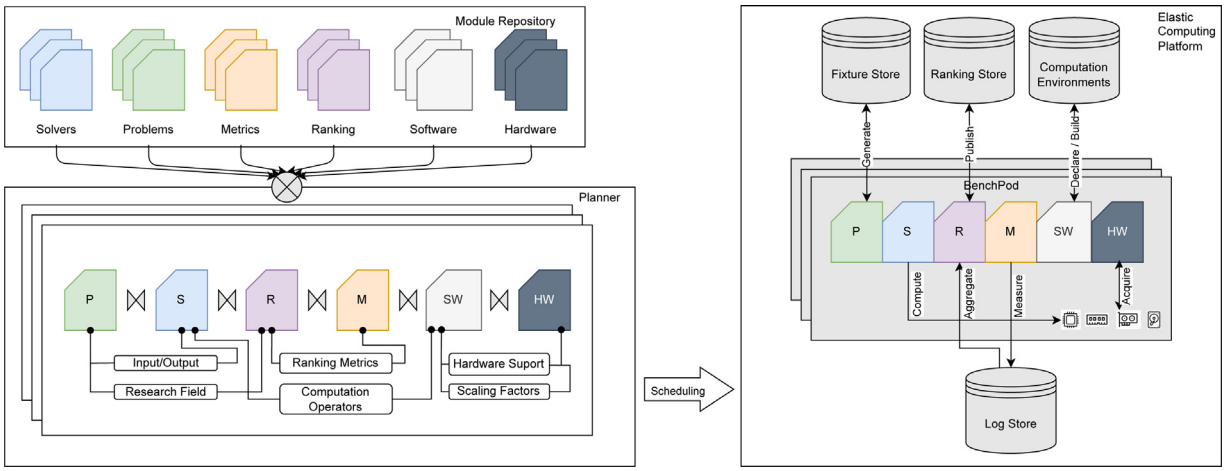


Fig. 1. System architecture.

```
@ProblemDefinition
def MNIST(epochs):
  data_type   = Tensor(28, 28)
  label_type  = Tensor(10)
  train_data  = Input("train-images.idx3-ubyte", data_type)
  train_label = Input("train-labels.idx1-ubyte", Scalar).OneHot(label_type)
  test_data   = Input("t10k-images.idx3-ubyte", data_type)
  test_label  = Input("t10k-labels.idx1-ubyte", Scalar).OneHot(label_type)
  for x,y in zip(train_data, train_label):
    Train.Classify(x,y)
  for x,y in zip(test_data, test_label):
    Test.Classify(x,y)
```

Fig. 2. MNIST Problem Definition.

of modern IDE features such as auto-completion and type checking while writing the module definition. To design an eDSL means that the desired features must be retrofitted into the target language. To achieve this, we take advantage of various Python language constructs that best fit the required features. Some features can be implemented with static analysis, for example, we use Python decorators to identify module entry points. This way we can easily scan for modules with reflection, and build our module repository. We use Python classes to represent type descriptors for our type system, which is a dual-role construct that both encodes the type information for static analysis, and dispatches code during benchmarking runtime. Benchmarking concepts are modeled as well-known global objects, and the methods attached to them represent benchmarking primitives. This gives a hint to the user that these concepts are stateful, and the primitives can function as both computation routine and data storage. Finally, we use declarative methods to construct the computation graph for AI models. Table 3 shows some language construct examples.

The module script, rather than directly executed in a Python interpreter, is first sent to the SAIL parser. The SAIL parser substitutes the actual execution logic with computation nodes and connects the nodes with computational dependencies to construct the computation graph, similar to the tape-recording technique in automatic differentiation frameworks [22]. The parser then analyzes the computation graph and synthesizes actual benchmarking code. The eDSL provides its own type system with both tensors and symbolic equations as first-class citizens, and helper functions to help connect different modules. In fact, with proper type inference, there is even no need to explicitly declare the input/output types of a module.

The flexibility of a scripting language also simplifies module definitions, for example, Fig. 2 illustrates a "hello world" problem definition module — the MNIST [23] image classification problem. This is a typical "defined by cases" problem as we illustrated in Section 2. The definition of this problem reads from four input files, joins them into pairs, and declares the data points and associated classification tasks into **Train** and **Test** collections. Note that the existence of both train and test collections is not necessary for some kinds of problems — for example, a PDE "problem class" definition may define a few equations in the test collection and expect a solver to accomplish the task without training or hints.

Note that the problem definition of MNIST resembles a machine learning training loop — but not entirely. The key point is that it only defines the problem, and does not try to solve or evaluate the results. This allows us to plug different evaluation metrics into the workflow. For example, the Machine Learning community traditionally focuses on the average performance over the whole dataset, while in a production critical environment, one may prefer to evaluate 99%

Table 3
Examples of SAIL language constructs.

Feature	Construct	Instances
Module entry points	Decorators	@ProblemDefinition @MetricDefiniton . . .
Type descriptors	Classes	**class** Tensor **class** Scalar . . .
Concepts and Primitives	Well-known Global Objects	**Train**. Classify **Model**. Predict **Test**. Compare . . .
AI models	Declarative methods	Pipeline Linear Relu Softmax . . .

```python
@MetricDefinition
def CriticalLoss():
    loss = []
    sz = len(Test.Classify)
    for (_, y0, y) in Test.Classify:
        loss.append(L2Error(y0, y))
    # critical 99% loss
    return loss.TopK(sz * 0.01)

@MetricDefinition
def WallTime():
    return ElapsedMilliseconds()
```

Fig. 3. Custom Evaluation Metric Definition.

percentile precision, or a hard fail condition, as shown in Fig. 3. Also shown in the code is a simple timer metric, and a task can be evaluated with multiple metrics. For example, the two in the code will combine into a work–critical loss 2D graph. For iterative tasks, a metric will also be evaluated iteratively, and a module can choose to keep states across multiple iterations, memorize the data points or obtain the average, etc.

Even for the same task, different research communities have different interests in performance evaluation. For example, scientific research groups focus on the quality of the end result, while computer system researchers focus on system performance metrics, such as throughput and latency [24]. This is why we further split the ranking module from problems and metrics. A ranking module can reference multiple metrics and aggregate them to obtain a total order, or implement a comparison between two instances to obtain partial order.

Another advantage of this approach is that the module definition can take input parameters and programmatically generate configurations. For example, in Fig. 4 we define how to pick the correct docker image tag for TensorFlow based on the hardware configuration, which is hard to model with a markup language. This also allows us to define generic AI modules that adapt to different input sizes and types and suggest hyper-parameter values. Fig. 5 shows the definition of a simple neural network, which not only defines the computation graph, but also the intended tasks, input/output type conversion, and layer width suggestions, so that the planner can grid search this hyperparameter. Also shown in the code snippet are two type converters, when combined, can automatically convert the input of an atom sequence into a single concatenated tensor.

4.2. Automatic benchmarking task discovery

As discussed before, the module definitions are not used for the actual execution of the benchmarking tasks. Rather, they are metaprogramming constructs that can be seen as a "dryrun" for the actual benchmarking. The system scans all python files and uses reflection to identify module entry points, and create records for them in the module repository. The system then enumerates all the modules from the repository and constructs candidate test fixtures, which are tuples of different kinds of modules. For each candidate tuple, the system executes the modules in it, providing input parameters, and extracting information such as the problems a model can solve, the research field of a problem, the suggested hyperparameters, and compatible metrics for a kind of task and so on. The execution order is determined by the type of modules and the implied dependencies — problem definitions execute first because they generally do not depend on other modules, and populate the metadata required to associate metrics and ranking. During execution, the system maintains the context for the current test fixture and accumulates the metadata from already executed modules in the candidate tuple, and later modules can either be filtered by metadata matching (for example, by matching data types) or actively reject the context. This is demonstrated in earlier examples, where a module can use the DSL primitive **Fail** to indicate that it does not know how to solve the problem, or the hardware does not support the current software configuration. Additionally, the system builds a graph where the nodes are data types and edges are converters, and employs breadth-first search to also allow type converter composition so that multiple converters can work together to relax type constraints and improve module compatibility. This process is akin to the inner join operation in relational databases, and the system builds complete tuples of the modules as test scenarios. Apart from automatic discovery, a module can also explicitly declare relationships with other modules so as to narrow down the search space. The logic is presented in algorithm 1.

Algorithm 1: Benchmarking Task Discovery.

Input: eDSL source files [*src*]
Output: Test scenarios [*test*]

```
repo ← ϕ;
// 1. Module discovery
foreach s : src do
    ast ← parse(s);
    foreach m : methods(ast) do
        if decorated(m, ProblemDefinition) then
            repo[0].append(m);
        if decorated(m, MetricDefinition) then
            repo[1].append(m);
        // Scan for other modules...
ctx ← ϕ; test ← ϕ; iters ← iterators(repo);
i ← 0;
// 2. Task discovery
while i ≥ 0 do
    if i = N then
        test.append(ctx);
        ctx.pop();
        i ← i − 1;
    else if next(iters[i]) then
        m ← get(iters[i]);
        metadata ← execute(m, ctx);
        if not Failed(metadata) then
            ctx.push(metadata);
            i ← i + 1;
    else
        ctx.pop();
        rewind(iters[i]);
        i ← i − 1;
return test
```

4.3. Experiment orchestration

When the planner is done generating benchmarking configuration tuples, it is necessary to prune unnecessary entries and make a schedule for the rest. There are multiple invariances in the benchmarking tasks to help pruning. For example, given the same AI model, software/hardware configuration, and similar problem size (of different problems), the throughput (in terms of FLOPS) can be comparable. Likewise, the precision evaluation should not be heavily impacted for the same model and problem on different software/hardware configurations. The executor should only pick significant tuples to maximize the diversity in measurements for all the metric dimensions, including model performance, scalability, generalization, and so on. Once the pruning is done, the scheduling problem concerns how to estimate the costs of each tuple, and efficiently pack them onto the hardware-task timeline.

```
@SoftwareDefinition
def TensorFlow(hw):
  tag = ':gpu'          if hw.UseCuda and hw.Arch == 'x64' else \
        ''              if not hw.UseCuda and hw.Arch == 'x64' else \
        ':gpu-ppc64le'  if hw.UseCuda and hw.Arch == 'ppc64le' else \
        ':ppc64le'      if not hw.UseCuda and hw.Arch == 'ppc64le' else \
        Fail('unsupported hardware configuration')
  Docker('tensorflow/tensorflow' + tag)
```

Fig. 4. TensorFlow Software Configuration.

```
@ConverterDefinition
def AtomEmbedding(a: Atom) -> Tensor:
    return atom_embeddings[a.name]
@ConverterDefinition
def Concat(a: list[T]) -> Tensor:
    return Concat(a.map(Convert(T,Tensor)))

@ModelDefinition
def MLP(task, inputType, outputType, hiddenSize):
  Suggest(hiddenSize, 128, 256, 512, 1024)
  if task == Classify:
    f = Softmax(outputType.dims)
  elif task == Enhance:
    f = id
  # ...
  else: Fail("don't know how to solve")
  return Pipeline([
    Convert(inputType, Tensor),
    Linear(hiddenSize), Relu(),
    Linear(outputType.dims), f,
    Convert(Tensor, outputType)
  ])
```

Fig. 5. AI Model Definition.

5. Case study

Now we discuss the details of a particular use case, Molecular Dynamics (MD). Given the initial states of the atoms (position and velocity vectors), the problem asks for a prediction of the movement of the atoms. In practice, the problem is decomposed into the problem of force prediction (Molecular Mechanics), and the integrated force over time to compute new states. In particular, force prediction is achieved in multiple ways developed by the Molecular Dynamics research community. "Classical MD" employs empirical models to compute pairwise forces between atoms, and first-principle methods (AIMD) employ quantum mechanic methods as DFT [25] and CCSD(t) [26] to first predict the potential energy of the system, and then obtain the forces by computing the partial derivatives of the energy over atom positions.

The problem definition module is shown in Fig. 6. It consists of two phases. First, an AI model is trained to predict the potential energy of a system, guided by a Molecular Dynamics software package, such as ORCA or Gaussian. Then, the performance of the model is evaluated on a different set of atom configurations. Unlike traditional AI benchmarks that merely evaluate the output of a model, here we provide multiple fixtures, including both the energy prediction, and the position and velocity updates computed from the prediction. It is the flexibility of problem scripting that gives us the ability to model additional fixtures other than the energy, which can be extended to benchmark fields other than Molecular Dynamics, for example, Raman Spectroscopy. This is a typical "defined by setting" problem as we illustrated in Section 2, because although it reads multiple data points from input files, each data point is not fed into the AI model, but rather into a simulation software to compute data points for the AI model.

There are multiple ways to specify an AI model for this problem — namely, given a set of atoms (atom types, positions, and velocities), predict a single scalar energy value. One way is to implement an end-to-end energy prediction model [27,28]. The other way aims to capture the essence of the end-to-end solutions and let the system synthesize the whole model. One key insight of the aforementioned energy prediction models is that the atom configuration is permutation invariant, which means that the input should be modeled as a set of atoms, not a list. Therefore, our goal here is to enable the system to compose an AI model to honor this property and take advantage of existing building blocks. A possible solution is shown in Fig. 7, where the input is typechecked to be a list, and the module requires a submodule that can complete the specified task (prediction in the Molecular Dynamics context) to map the element type to the output type. The element-wise results are then summed to combine a permutation invariant output. This way, the system is able to pick up the modules we defined earlier, such as the atom embedding converter, and the MLP model for conducting element-wise prediction.

Now we discuss another benchmarking scenario, deep-learning-based electron microscopy image segmentation, which is becoming a popular topic in Biological Chemistry [29–31]. One of the main challenges in this topic is the scarce of training data, due to complex and costly data acquisition process. Given limited data, supervised deep-learning methods require heavy human intervention and may fail to generalize to unseen data [32,33]. One way to circumvent the data problem is to introduce semi-supervised deep-learning techniques, such as pre-training with high volume unlabeled data [34]. To support pre-training in the benchmarking system means that a model under evaluation should be able to carry a part of its internal states (weights) from one task to another, and adjust its computation graph accordingly. The problem definition should also evaluate the performance of the model given different amounts of training data, to test its sample efficiency. The code for this scenario is shown in Fig. 8.

5.1. Comparison to other benchmarking systems

As mentioned above, previous systems focus on a fixed set of test scenarios [19–21]. Additionally, the lack of declarative modules means that it is hard to share data between the benchmarking suite and external scientific computing software packages, which is crucial in scientific AI benchmarking. For example, the **Gradient** primitive in SAIBench allows a training pipeline to extract gradients from an external package, which is usually not exposed programmatically. The differences are shown in Table 4.

6. Discussion

We have elaborated on the methodology and the overview of the system design, yet we look forward to further development in the

```
@ProblemDefinition
def MDSimulation():
    md = Require(MolecularDynamicsSoftware)
    # Training phase
    train_problems = [
        Input("ch2o.atoms").XYZ(),
        # ...
    ]
    for atoms in train_problems:
        for _ in range(200):
            e = md.PredictEnergy(atoms)
            Train.Predict(atoms, e)
            f = md.ComputeForces(atoms, e)
            atoms = md.ComputeNewStates(atoms, 1.0, f)
    # Testing phase
    test_problems = [
        Input("h2o.atoms").XYZ(),
        # ...
    ]
    for atoms in test_problems:
        atoms_golden = atoms
        atoms_pred = atoms
        for _ in range(200):
            e_g = md.PredictEnergy(atoms_golden)
            e_p = Model.Predict(atoms_pred)
            Test.Compare(e_g, e_p)
            f_g = md.ComputeForces(atoms_golden, e_g)
            pos = atoms_pred.map(lambda x: x.pos)
            f_p = Gradients(pos, e_p)
            atoms_golden = md.ComputeNewStates(atoms_golden, 1.0, f_g)
            atoms_pred = md.ComputeNewStates(atoms_pred, 1.0, f_p)
            Test.Compare(atoms_golden, atoms_pred)
```

Fig. 6. Molecular Dynamics Problem Definition.

```
@ModelDefinition
def PermutationInvariantModel(task, inputType, outputType):
    if not inputType.IsSet: Fail("Don't know how to solve")
    ety = inputType.ElementType
    m = Require(Model, task, ety, outputType)
    return Sum(Array(m, inputType.Size))
```

Fig. 7. Permutation Invariant Model Definition.

Table 4
Comparison to other benchmarking systems.

	SAIBench	MLPerf	MLHarness
Focus	Different scientific tasks/criterion	Accuracy, system throughput	Scalability, MLCommon coverage
Modules	Declarative	Hard-coded	Markup
Test scenarios	Automatic discovery	Fixed	Fixed

components. Brute-force enumeration of all possible test hyperparameters may not be feasible and while pruning can mechanically improve the situation, it is desirable that a particular problem module can suggest parameters suitable for a research field. More design work could be done to address model development and debugging needs, for example, to allow model validation in addition to training and testing. Python-based eDSL has its limitations, mostly due to the syntactic constraints of the language. To represent the modules more naturally, a programming language more geared towards scientific computing can be investigated [35].

Currently, SAIBench targets tractable scientific tasks, which are mechanical procedures that can be computed and measured. It is challenging to extend it to more creative scientific research activities because it would require the system to formally model the scientific concepts, and gain a deeper understanding of research topics, motivations, methodologies, and goals, and how various concepts interact with each other. Also, automated benchmarks require well-defined metrics, while open-ended scientific research ideas, in general, are hard to quantify.

Apart from type-based model composition, automatic AI model synthesizing given a particular problem definition is also a promising direction, given the advancement in AI-based code generation [36,37].

7. Conclusion

We have presented our definition of scientific AI benchmarking, which is an ensemble of scientific task definition, AI benchmarking, and system performance benchmarking. We have then presented our methodology for scientific AI benchmarking, with the key idea of decoupling and modularizing various components, automatically benchmarking sensible combinations. We have proposed a system design where the various modules are implemented with a domain-specific language for scientific AI computing. We have demonstrated that this design is flexible enough to support benchmarking different types of scientific tasks, defining AI models, deriving multiple metrics, combining metrics into ranking criteria, and configuring required hardware/software.

```
@ModelDefinition
def PretrainSegmentModel(task, inputType, outputType, embsize):
    embType = Tensor(embsize)
    # base model
    m_base = Require(Model, Embedding, inputType, embType)
    if task == Pretrain:
        # pretrain decoder
        m_pretrain = Require(Model, Segment, embType, inputType)
        return Pipeline([ m_base, m_pretrain ])
    elif task == Segment:
        # task decoder
        m_task = Require(Model, task, embType, outputType)
        return Pipeline([ m_base, m_task ])
    else:
        Fail("don't know how to solve.")

@ProblemDefinition
def ElectronMicroscopySegmentation():
    pretrain_data = Input("CEM500K").Image()
    labeled_data = [
        [Input("cell0_slice1.img").Image(),
         Input("cell0_slice1.msk").Image()],
        # ...
    ]
    for unlabled in pretrain_data:
        Train.Pretrain(unlabled)
    train_data, test_data = RandomSplit(labeld_data, 0.9)
    train_stages = Chunks(train_data, 10)
    for i in range(10):
        train_current = sum(train_stages[0..i])
        for x,y in train_current:
            Train.Segment(x,y)
        for x,y in test_data:
            Test.Segment(x,y)
```

Fig. 8. Electron Microscopy Image Segmentation.

Declaration of competing interest

One or more of the authors of this paper have disclosed potential or pertinent conflicts of interest, which may include receipt of payment, either direct or indirect, institutional support, or association with an entity in the biomedical field which may be perceived to have potential conflict of interest with this work. For full disclosure statements refer to https://doi.org/10.1016/j.tbench.2022.100063. Yatao Li reports financial support was provided by Microsoft Research Asia. Yatao Li reports administrative support was provided by Institute of Computing Technology Chinese Academy of Sciences. Jianfeng Zhan reports financial support was provided by Institute of Computing Technology Chinese Academy of Sciences.

References

[1] A.N. Laboratory, AI for science report.URL https://publications.anl.gov/anlpubs/2020/03/158802.pdf.

[2] K. Albertsson, P. Altoe, D. Anderson, J. Anderson, M. Andrews, J.P.A. Espinosa, A. Aurisano, L. Basara, A. Bevan, W. Bhimji, D. Bona-corsi, B. Burkle, P. Calafiura, M. Campanelli, L. Capps, F. Carmi-nati, S. Carrazza, Y.-f. Chen, T. Childers, Y. Coadou, E. Coniavitis, K. Cranmer, C. David, D. Davis, A. De Simone, J. Duarte, M. Erd-mann, J. Eschle, A. Farbin, M. Feickert, N.F. Castro, C. Fitzpatrick, M. Floris, A. Forti, J. Garra-Tico, J. Gemmler, M. Girone, P. Glaysher, S. Gleyzer, V. Gligorov, T. Golling, J. Graw, L. Gray, D. Greenwood, T. Hacker, J. Harvey, B. Hegner, L. Heinrich, U. Heintz, B. Hoober-man, J. Junggeburth, M. Kagan, M. Kane, K. Kanishchev, P. Karpiński, Z. Kassabov, G. Kaul, D. Kcira, T. Keck, A. Klimentov, J. Kowalkowski, L. Kreczko, A. Kurepin, R. Kutschke, V. Kuznetsov, N. Köhler, I. Lako-mov, K. Lannon, M. Lassnig, A. Limosani, G. Louppe, A. Mangu, P. Mato, N. Meenakshi, H. Meinhard, D. Menasce, L. Moneta, S. Moort-gat, M. Neubauer, H. Newman, S. Otten, H. Pabst, M. Paganini, M. Paulini, G. Perdue, U. Perez, A. Picazio, J. Pivarski, H. Prosper, F. Psihas, A. Radovic, R. Reece, A. Rinkevicius, E. Rodrigues, J. Rorie, D. Rousseau, A. Sauers, S. Schramm, A. Schwartzman, H. Severini, P. Seyfert, F. Siroky, K. Skazytkin, M. Sokoloff, G. Stewart, B. Stienen, I. Stockdale, G. Strong, W. Sun, S. Thais, K. Tomko, E. Upfal, E. Usai, A. Ustyuzhanin, M. Vala, J. Vasel, S. Vallecorsa, M. Verzetti, X. Vilasís-Cardona, J.-R. Vlimant, I. Vukotic, S.-J. Wang, G. Watts, M. Williams, W. Wu, S. Wunsch, K. Yang, O. Zapata, Machine learning in high energy physics community white paper. URL http://arxiv.org/abs/1807.02876.

[3] T. Kurth, S. Treichler, J. Romero, M. Mudigonda, N. Luehr, E. Phillips, A. Mahesh, M. Matheson, J. Deslippe, M. Fatica, M. Houston Prabhat, Exascale deep learning for climate analytics, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis,

[4] J. Degrave, F. Felici, J. Buchli, M. Neunert, B. Tracey, F. Carpanese, T. Ewalds, R. Hafner, A. Abdolmaleki, D. de las Casas, C. Don-ner, L. Fritz, C. Galperti, A. Huber, J. Keeling, M. Tsimpoukelli, J. Kay, A. Merle, J.-M. Moret, S. Noury, F. Pesamosca, D. Pfau, O. Sauter, C. Sommariva, S. Coda, B. Duval, A. Fasoli, P. Kohli, K. Kavukcuoglu, D. Hassabis, M. Riedmiller, Magnetic control of toka- mak plasmas through deep reinforcement learning 602 (7897) 414–419. http://dx.doi.org/10.1038/s41586-021-04301-9. URL https://www.nature.com/articles/s41586-021-04301-9.

[5] R.B. Neale, A. Gettelman, S. Park, C.-C. Chen, P.H. Lauritzen, D.L. Williamson, A.J. Conley, D. Kinnison, D. Marsh, A.K. Smith, F. Vitt, R. Garcia, J.-F. Lamarque, M. Mills, S. Tilmes, H. Morrison, P. Cameron-Smith, W.D. Collins, M.J. Iacono, R.C. Easter, X. Liu, S.J. Ghan, P.J. Rasch, M.A. Taylor, Description of the NCAR community atmosphere model (CAM 5.0) 289.

[6] J.S. Smith, R. Zubatyuk, B. Nebgen, N. Lubbers, K. Barros, A.E. Roit-berg, O. Isayev, S. Tretiak, The ANI-1ccx and ANI-1x data sets, coupled-cluster and density functional theory properties for molecules, Sci. Data 7 (1) 134. http://dx.doi.org/10.1038/s41597-020-0473-z. URL http://www.nature.com/articles/s41597-020-0473-z.

[7] L. Ruddigkeit, R. van Deursen, L.C. Blum, J.-L. Reymond, Enumeration of 166 billion organic small molecules in the chemical universe database GDB-17, J. Chem. Inform. Model. 52 (11) 2864–2875. http://dx.doi.org/10.1021/ci300415d. URL https://pubs.acs.org/doi/10.1021/ci300415d.

[8] D.S. Marcus, T.H. Wang, J. Parker, J.G. Csernansky, J.C. Morris, R.L. Buckner, Open access series of imaging studies (OASIS): Cross-sectional MRI data in young, middle aged, nondemented, and demented older adults, J. Cogn. Neurosci. 19 (9) 1498–1507. http://dx.doi.org/10.1162/jocn.2007.19.9.1498. URL https://direct.mit.edu/jocn/article/19/9/1498/4427/Open-Access-Series-of-Imaging-Studies-OASIS-Cross.

[9] E. Weinan, J. Han, A. Jentzen, Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations, Commun. Math. Stat. 5 (4) 349–380. http://dx.doi.org/10.1007/s40304-017-0117-6. URL https://collaborate.princeton.edu/en/publications/deep-learning-based-numerical-methods-for-high-dimensional-parabo.

[10] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Computat. Phys. 378 686–707. http://dx.doi.org/10.1016/j.jcp.2018.10.045. URL https://www.sciencedirect.com/science/article/pii/S0021999118307125.

[11] F. Noé, Machine learning for molecular dynamics on long timescales, in: K.T. Schütt, S. Chmiela, O.A. von Lilienfeld, A. Tkatchenko, K. Tsuda, K.-R. Müller (Eds.), Machine Learning Meets Quantum Physics, Springer International Publishing, pp. 331–372, http://dx.doi.org/10.1007/978-3-030-40245-7_16.

[12] A. Mardt, L. Pasquali, H. Wu, F. Noé, VAMPnets for deep learning of molecular kinetics, Nature Commun. 9 (1) 5, http://dx.doi.org/10.1038/s41467-017-02388-1. URL https://www.nature.com/articles/s41467-017-02388-1.

[13] W. Jia, H. Wang, M. Chen, D. Lu, L. Lin, R. Car, W. E, L. Zhang, Pushing the limit of molecular dynamics with ab initio accuracy to 100 million atoms with machine learning Version: 1. arXiv:2005.00223. URL http://arxiv.org/abs/2005.00223.

[14] T. Hoefler, R. Belli, Scientific benchmarking of parallel computing sys- tems: Ttwelve ways to tell the masses when reporting performance re- sults, in: Proceedings of the International Conference for High Perfor- Mance Computing, Networking, Storage and Analysis, ACM, pp. 1–12, http://dx.doi.org/10.1145/2807591.2807644, URL https://dl.acm.org/doi/10.1145/2807591.2807644.

[15] E. Apra', E.J. Bylaska, W.A. de Jong, N. Govind, K. Kowalski, T.P. Straatsma, M. Valiev, H.J.J. van Dam, Y. Alexeev, J. Anchell, V. Anisi-mov, F.W. Aquino, R. Atta-Fynn, J. Autschbach, N.P. Bauman, J.C. Becca, D.E. Bernholdt, K. Bhaskaran-Nair, S. Bogatko, P. Borowski, J. Boschen, J. Brabec, A. Bruner, E. Cauët, Y. Chen, G.N. Chuev, C.J. Cramer, J. Daily, M.J.O. Deegan, T.H. Dunning, M. Dupuis, K.G. Dyall, G.I. Fann, S.A. Fischer, A. Fonari, H. Früchtl, L. Gagliardi, J. Garza, N. Gawande, S. Ghosh, K. Glaesemann, A.W. Götz, J. Ham-mond, V. Helms, E.D. Hermes, K. Hirao, S. Hirata, M. Jacquelin, L. Jensen, B.G. Johnson, H. Jónsson, R.A. Kendall, M. Klemm, R. Kobayashi, V. Konkov, S. Krishnamoorthy, M. Krishnan, Z. Lin, R.D. Lins, R.J. Littlefield, A.J. Logsdail, K. Lopata, W. Ma, A.V. Marenich, J. Martin del Campo, D. Mejia-Rodriguez, J.E. Moore, J.M. Mullin, T. Nakajima, D.R. Nascimento, J.A. Nichols, P.J. Nichols, J. Nieplocha, A. Otero-de-la Roza, B. Palmer, A. Panyala, T. Pirojsirikul, B. Peng, R. Peverati, J. Pittner, L. Pollack, R.M. Richard, P. Sadayappan, G.C. Schatz, W.A. Shelton, D.W. Silverstein, D.M.A. Smith, T.A. Soares, D. Song, M. Swart, H.L. Taylor, G.S. Thomas, V. Tipparaju, D.G. Truh-lar, K. Tsemekhman, T. Van Voorhis, Vázquez-Mayagoitia, P. Verma, O. Villa, A. Vishnu, K.D. Vogiatzis, D. Wang, J.H. Weare, M.J. Williamson, T.L. Windus, K. Woliński, A.T. Wong, Q. Wu, C. Yang, Q. Yu, M. Zacharias, Z. Zhang, Y. Zhao, R.J. Harrison, NWChem: Past, present, and future 152 (18) 184102. http://dx.doi.org/10.1063/5.0004997. URL http://aip.scitation.org/doi/10.1063/5.0004997.

[16] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, PyTorch: An imperative style, high- performance deep learning library, in: Advances in Neural Information Processing Systems, Vol. 32, Curran Associates, Inc., URL https://papers.nips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

[17] M. Brehm, SANscript – A scientific algorithm notation language. URL https://brehm-research.de/sanscript.php.

[18] M.D. Wilkinson, M. Dumontier, I.J. Aalbersberg, G. Appleton, M. Ax-ton, A. Baak, N. Blomberg, J.-W. Boiten, L.B. da Silva Santos, P.E. Bourne, J. Bouwman, A.J. Brookes, T. Clark, M. Crosas, I. Dillo, O. Dumon, S. Edmunds, C.T. Evelo, R. Finkers, A. Gonzalez-Beltran, A.J.G. Gray, P. Groth, C. Goble, J.S. Grethe, J. Heringa, P.A.C. 't Hoen, R. Hooft, T. Kuhn, R. Kok, J. Kok, S.J. Lusher, M.E. Martone, A. Mons, A.L. Packer, B. Persson, P. Rocca-Serra, M. Roos, R. van Schaik, S.-A. Sansone, E. Schultes, T. Sengstag, T. Slater, G. Strawn, M.A. Swertz, M. Thompson, J. van der Lei, E. van Mulligen, J. Vel-terop, A. Waagmeester, P. Wittenburg, K. Wolstencroft, J. Zhao, B. Mons, The FAIR guiding principles for scientific data management and stewardship, Sci. Data 3 (1) 160018. http://dx.doi.org/10.1038/sdata.2016.18. URL https://www.nature.com/articles/sdata201618.

[19] W. Gao, C. Luo, L. Wang, X. Xiong, J. Chen, T. Hao, Z. Jiang, F. Fan, M. Du, Y. Huang, F. Zhang, X. Wen, C. Zheng, X. He, J. Dai, H. Ye, Z. Cao, Z. Jia, K. Zhan, H. Tang, D. Zheng, B. Xie, W. Li, X. Wang, J. Zhan, Aibench: Towards scalable and comprehensive datacenter AI benchmarking, in: C. Zheng, J. Zhan (Eds.), Benchmarking, Measuring, and Optimizing, Vol. 11459, in: Lecture Notes in Computer Science, Springer International Publishing, pp. 3–9, http://dx.doi.org/10.1007/978-3-030-32813-9_1, URL http://link.springer.com/10.1007/978-3-030-32813-9 1.

[20] P. Mattson, C. Cheng, C. Coleman, G. Diamos, P. Micikevicius, D. Pat-terson, H. Tang, G.-Y. Wei, P. Bailis, V. Bittorf, D. Brooks, D. Chen, D. Dutta, U. Gupta, K. Hazelwood, A. Hock, X. Huang, A. Ike, B. Jia, D. Kang, D. Kanter, N. Kumar, J. Liao, G. Ma, D. Narayanan, T. Ogun-tebi, G. Pekhimenko, L. Pentecost, V.J. Reddi, T. Robie, T.S. John, T. Tabaru, C.-J. Wu, L. Xu, M. Yamazaki, C. Young, M. Zaharia, MLPerf training benchmark 14.

[21] Y.-H. Chang, J. Pu, W.-m. Hwu, J. Xiong, MLHarness: A scalable benchmarking system for ML Commons, BenchCouncil Trans. Benchmarks, Standards Eval. 1 (1) 100002. http://dx.doi.org/10.1016/j.tbench.2021.100002. URL https://www.sciencedirect.com/science/article/pii/S2772485921000028.

[22] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in PyTorch 4.

[23] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) 2278–2324. http://dx.doi.org/10.1109/5.726791. URL http://ieeexplore.ieee.org/document/726791/.

[24] J. Thiyagalingam, M. Shankar, G. Fox, T. Hey, Scientific machine learn- ing benchmarks.URL http://arxiv.org/abs/2110.12773.

[25] R. Haunschild, A. Barth, B. French, A comprehensive analysis of the history of DFT based on the bibliometric method RPYS, J. Cheminform. 11 (1) 72, http://dx.doi.org/10.1186/s13321-019-0395-y.

[26] H.G. Kümmel, A biography of the coupled cluster method 17 (28) 5311–5325, http://dx.doi.org/10.1142/S0217979203020442. URL https://www.worldscientific.com/doi/abs/10.1142/S0217979203020442.

[27] J. Han, L. Zhang, R. Car, W. E, Deep potential: A general representation of a many-body potential energy surface, Commun. Computat. Phys. 23 (3). arXiv:1707.01478, http://dx.doi.org/10.4208/cicp.OA-2017-0213. URL http://arxiv.org/abs/1707.01478.

[28] O.T. Unke, M. Meuwly, PhysNet: A neural network for predicting energies, forces, dipole moments and partial charges, J. Chem. Theory Computat. 15 (6) 3678–3693. arXiv:1902.08408, http://dx.doi.org/10.1021/acs.jctc.9b00181. URL http://arxiv.org/abs/1902.08408.

[29] E. Gómez-de Mariscal, M. Maška, A. Kotrbová, V. Pospíchalová, P. Mat-ula, A. Munõz-Barrutia, Deep-learning-based segmentation of small extracellular vesicles in transmission electron microscopy images, Sci. Rep. 9 (1) 13211. http://dx.doi.org/10.1038/s41598-019-49431-3. URL https://www.nature.com/articles/s41598-019-49431-3.

[30] L. von Chamier, R.F. Laine, J. Jukkala, C. Spahn, D. Krentzel, E. Nehme, M. Lerche, S. Hernández-Pérez, P.K. Mattila, E. Karinou, S. Holden, A.C. Solak, A. Krull, T.-O. Buchholz, M.L. Jones, L.A. Royer, C. Leterrier, Y. Shechtman, F. Jug, M. Heilemann, G. Jacquemet, R. Henriques, Democratising deep learning for microscopy with ZeroCostDL4Mic, Nature Commu. 12 (1) 2276. http://dx.doi.org/10.1038/s41467-021-22518-0. URL https://www.nature.com/articles/s41467-021-22518-0.

[31] J.M. Ede, Deep learning in electron microscopy, Mach. Learning: Sci. Technol. 2 (1) 011004. http://dx.doi.org/10.1088/2632-2153/abd614.

[32] S.M. Plaza, J. Funke, Analyzing image segmentation for connectomics, Front. Neural Circ. 12, 102. DOI: http://dx.doi.org/10.3389/fncir.2018.00102. URL https://www.frontiersin.org/article/10.3389/fncir.2018.00102/full.

[33] J.W. Lichtman, H. Pfister, N. Shavit, The big data challenges of connectomics, Nature Neurosci. 17 (11) 1448–1454. http://dx.doi.org/10.1038/nn.3837. URL http://www.nature.com/articles/nn.3837.

[34] R. Conrad, K. Narayan, CEM500K, a large-scale heterogeneous unlabeled cellular electron microscopy image dataset for deep learning, eLife 10 e65894, eLife Sciences Publications, Ltd. DOI: http://dx.doi.org/10.7554/eLife.65894.

[35] M. Innes, A. Edelman, K. Fischer, C. Rackauckas, E. Saba, V.B. Shah, W. Tebbutt, A differentiable programming system to bridge machine learning and scientific computing. URL http://arxiv.org/abs/1907.07587.

[36] S. Lu, D. Guo, S. Ren, J. Huang, A. Svyatkovskiy, A. Blanco, C. Clement, D. Drain, D. Jiang, D. Tang, G. Li, L. Zhou, L. Shou, L. Zhou, M. Tu-fano, M. Gong, M. Zhou, N. Duan, N. Sundaresan, S.K. Deng, S. Fu, S. Liu, CodeXGLUE: A machine learning benchmark dataset for code understanding and generation. URL http://arxiv.org/abs/2102.04664.

[37] D. Peng, S. Zheng, Y. Li, G. Ke, D. He, T.-Y. Liu, How could neural networks understand programs? in: Proceedings of the 38th International Conference on Machine Learning, PMLR, pp. 8476–8486, URL https://proceedings.mlr.press/v139/peng21b.html.

BenchCouncil Transactions on Benchmarks, Standards and Evaluations 2 (2022) 100062

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/

An efficient encrypted deduplication scheme with security-enhanced proof of ownership in edge computing

Yukun Zhou [a,b,c], Zhibin Yu [a,*], Liang Gu [b], Dan Feng [c]

[a] Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China
[b] Sangfor Technologies Inc, Shenzhen, China
[c] Wuhan National Laboratory for Optoelectronics, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China

ARTICLE INFO

Keywords:
Deduplication
Message-locked encryption
Proof of ownership
Edge computing

ABSTRACT

With the rapid expansion of Internet of Things (IoT), relevant files are stored and transmitted at the network edge by employing data deduplication to eliminate redundant data for the best accessibility. Although deduplication improves storage and network efficiency, it decreases security strength and performance. Existing schemes usually adopt message-locked encryption (MLE) to encrypt data, which is vulnerable to brute-force attacks. Meanwhile, these schemes utilize proof-of-ownership (PoW) to prevent duplicate-faking attacks, while they suffer from replay attacks or incur large computation overheads. This paper proposes SE-PoW, an efficient and location-aware hybrid encrypted deduplication scheme with a dual-level security-enhanced Proof-of-Ownership in edge computing. Specifically, SE-PoW firstly encrypts files with an inter-edge server-aided randomized convergent encryption (RCE) method and then protects blocks with an intra-edge edge-aided MLE method to balance security and system efficiency. To resist duplicate-faking attacks and replay attacks, SE-PoW performs the dual-level PoW algorithm. Then it combines the verification of a cuckoo filter and the homomorphism of algebraic signatures in sequence to enhance security and improve ownership checking efficiency. Security analysis demonstrates that SE-PoW ensures data security and resists the mentioned attacks. Evaluation results show that SE-PoW reduces up to 61.9% upload time overheads compared with the state-of-the-art schemes.

1. Introduction

With the high-speed development of 5G and edge computing, large amounts of data are collected in the core and edge devices, such as smartphones, wearables, automatic driving [1], and traffic flow detection [2]. In the big data era, IDC predicts that the digital universe will reach 175ZB in 2025 [3], and more than 44% of IoT-created data will be processed and analyzed at the network edge. Edge computing deploys computing and storage resources at the network edge to handle time-sensitive tasks while offering fast and convenient services to users [4]. Research analysis shows that there exist large amounts of redundant data (up to 76%) for workloads like VM images and car multimedia IoT data [5–7]. Data deduplication has been adopted in the modern central cloud (e.g., Dropbox [8], OneDrive [9]) and also pushed to the network edge for both optimized space and network efficiency. Fig. 1 describes a simple architecture of edge computing that deploys deduplication, for example, Ctera [10]. Edge computing can be seen as a three-tiered architecture. The central cloud stores and retrievals data from edge nodes and users. The edge nodes provide limited computing, indexing, storage, and other services [11,12]. Deduplication eliminates duplicate data on a file or block, which keeps only one physical copy and others refer to it. Deduplication can be classified into client-side or server-side approaches, while the former also saves network transmission. Edge computing deployed with deduplication has attracted lots of attention in both academia and industry [10–13], but it remains many security issues and potential threats [14,15].

Users usually encrypt data before outsourcing them to the edge and cloud for security and privacy concerns. However, encrypting the same data with different keys will result in different ciphertexts and makes deduplication impossible. Many researchers propose convergent encryption(CE) and message-locked encryption(MLE) [16–21] that adopt the hash value as the symmetric key to encrypt data, which users carry out deduplication over ciphertexts. Unfortunately, MLE suffers from resist brute-force attacks [18] that the attacker can recover the target file from a known set by offline encryption. To mitigate the attacks, researchers propose an oblivious pseudorandom function(OPRF) to generate MLE keys aided by secret messages of the server. However, client-side deduplication suffers from various attacks and privacy leakages, such as duplicate-faking attacks [15,22,23] and position attacks. That is, a malicious user can gain access to files belonging

* Corresponding author.
E-mail address: zb.yu@siat.ac.cn (Z. Yu).

https://doi.org/10.1016/j.tbench.2022.100062
Received 16 April 2022; Received in revised form 9 May 2022; Accepted 9 May 2022
Available online 24 May 2022

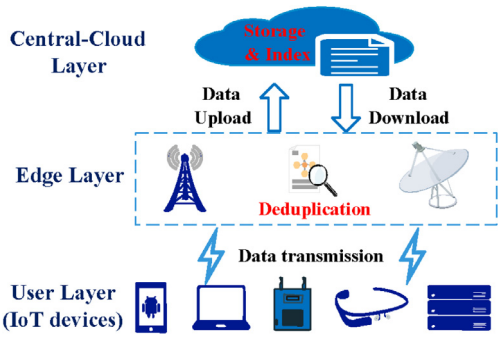


Fig. 1. An example of edge computing deployed with data deduplication.

to other users based on a hash value or upload corrupted data with valid hash values [24]. Some deduplication with Proof of ownership (PoW) schemes [22,25–27] are proposed to verify ownership of data users, such as MHT-PoW [22] or BF-PoW [21,26]. The user convinces the server that it owns the hash value and holds the file content. PoW is a protocol in which a server sends challenges, and the client returns the proofs as a response. Specifically, MHT-PoW encodes files into a fixed-size buffer and conducts a Merkle hash tree via a pairwise independent hash function [20,28]. BF-PoW divides files into fixed-size blocks, calculates the hash digests, and inserts them into a bloom filter [29]. However, existing encrypted deduplication schemes with PoW face new challenges.

First, existing schemes adopt MLE [17,30] or RCE [20,21] to protect data security, but they are vulnerable to brute-force attacks for the low-entropy files especially. Moreover, other encrypted deduplication schemes, such as OPRF [18], data re-encryption [20,21], and public encryption [31] bring a significant computational burden. They are not suitable for resource-constrained edge nodes and IoT devices.

Second, existing schemes introduce proof-of-ownership to resist duplicate-faking attacks. They nevertheless incur large computation overheads or suffer from replay attacks. Generally speaking, MHT-PoW brings a heavy computation burden because of the encoding of files and constructions of the Merkle hash tree. BF-PoW suffers from replay attacks and the privacy leakage of the false positive in a bloom filter. An attacker passes the verification of BF-PoW by generating valid proof from previous proofs without owning the original data. The replay attacks also have occurred in many scenes, such as provable data possession(PDP) and proof of retrievability (PoRs) [32].

To overcome these challenges, we propose an efficient encrypted deduplication with Security-Enhanced Proof-of-Ownership (SE-PoW). We observe that the capabilities and security risks for inter-/intra edge nodes are different [4], and duplicate files are mainly from multiple users [33,34]. The core idea behind SE-PoW is to employ different randomized MLE methods based on the location of deduplication. Specifically, SE-PoW first performs inter-edge encrypted deduplication for files via a server-aided RCE method. If the file is non-duplicate, SE-PoW further performs intra-edge encrypted deduplication for blocks via an edge-aided MLE method. Moreover, SE-PoW utilizes a dual-level proof-of-ownership to guarantee higher security. SE-PoW performs ownership checking based on a cuckoo filter to resist duplicate-faking attacks. SE-PoW adds unique labels and verifies the homomorphism of the algebraic signature [35] to resist replay attacks. Security analysis demonstrates that SE-PoW resists the above attacks from inside and outside attackers. Therefore, SE-PoW significantly reduces computation overheads compared with state-of-the-art schemes and ensures data security.

This paper makes the following contributions.

- We propose SE-PoW, a location-aware hybrid encrypted deduplication scheme in edge computing. SE-PoW performs inter-edge file-level and intra-edge block-level encrypted deduplication via server-aided RCE and edge-aided MLE algorithms, respectively. Thus SE-PoW balances data confidentiality and efficiency.
- SE-PoW proposes a dual-level security-enhanced proof-of-ownership by leveraging a cuckoo filter and algebraic signatures. SE-PoW achieves a higher security level and only increases little overheads, in which SE-PoW resists duplicate-faking attacks and replay attacks.
- We present a prototype of SE-PoW. Security analysis demonstrates that SE-PoW can ensure data confidentiality and resist duplicate-faking attacks and replay attacks under the proposed threat model. Experimental results based on real-world datasets show that SE-PoW reduces 21.9–61.9% upload time overheads compared with the state-of-the-art MHT-PoW.

The reset of our paper is organized as follows. Section 2 introduces the background and problems of SE-PoW in edge computing. In Section 3 the system model, threat model and security requirements are defined. Section 4 introduces the design and implementation details of SE-PoW. Section 5 discusses the security of SE-PoW. Section 6 presents the performance evaluation on real-world datasets. In Section 7, the related works on encrypted deduplication schemes are reviewed. Finally, Section 8 concludes this paper.

2. Background & problems

This section briefly introduces encrypted deduplication in edge computing and proof of ownership schemes. We further present the problems and motivation of SE-PoW.

2.1. Encrypted deduplication in edge computing

Many users store files at the network edge and respond to users' requests with low latency [4,34]. In Fig. 1, edge computing employs data deduplication at the network edge for space and network efficiency [11–13]. The user uploads/retrieves data and relevant information to the edge nodes. Then edge nodes could compute the tags of data via a hash function (i.e., SHA256) and encrypt data. Edge nodes maintain a deduplication index structure for local or cross-domain duplicate checking. Decentralized deduplication distributes data to multiple edge nodes for load balancing [13].

To protect data confidentiality, Douceur et al. [16] propose convergent encryption(CE) and Bellare et al. [17] propose Message-locked Encryption(MLE) and random to enable deduplication over ciphertexts. Specifically, the client derives a key $K \leftarrow H(P, M)$ from message M, and P is a public parameter and H is a cryptographic hash function. And it encrypts the message as $C \leftarrow \text{Encry}(K, M)$, where $Encry/Decry$ is a pair of encryption and decryption functions. The tag T derives $T \leftarrow H(P, C)$. In randomized convergent encryption(RCE), the client encrypts a message $C_1 \leftarrow Encry(L, M)$, where L is a randomly chosen key. Then it encrypts the key L and generate $C_2 \leftarrow L \oplus K$, where K is derived from the message $K \leftarrow H(P, M)$. The client generates tag $T \leftarrow H(P, K)$. When any owner receives $C_1 \| C_2 \| T$ from the server, he computes $L \leftarrow C_2 \oplus K$, and obtains M via $M \leftarrow Decry(L, C_1)$. However, MLE and RCE are vulnerable to brute-force attacks.

Bellare et al. [18] present server-aided MLE algorithms via an RSA-based oblivious PRF protocol to resist brute-force attacks. In edge computing, Ni et al. [36] put forward edge-based encrypted deduplication with BLS-OPRF and adopted proxy re-encryption on edge nodes. Yang et al. [15] propose a cross-domain deduplication scheme with server-aided MLE via HPS-OPRF in blockchain-enabled edge computing. In addition, Hur et al. [20] propose authorized encrypted deduplication with dynamic ownership management via proxy re-encryption [21]. In summary, encrypted deduplication has been widely used in edge computing.

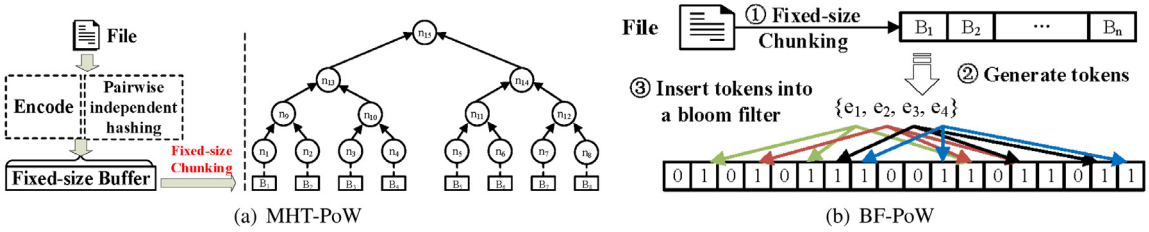


Fig. 2. We describe the procedure of MHT-PoW and BF-PoW. In figure (a), MHT-PoW encodes a file into a fixed-size buffer and constructs a Merkle hash tree. In figure (b), BF-PoW splits a file into blocks, generates tokens (e.g., e_i), and inserts them into a bloom filter. Finally, the verifier randomly selects N challenged blocks for ownership checking.

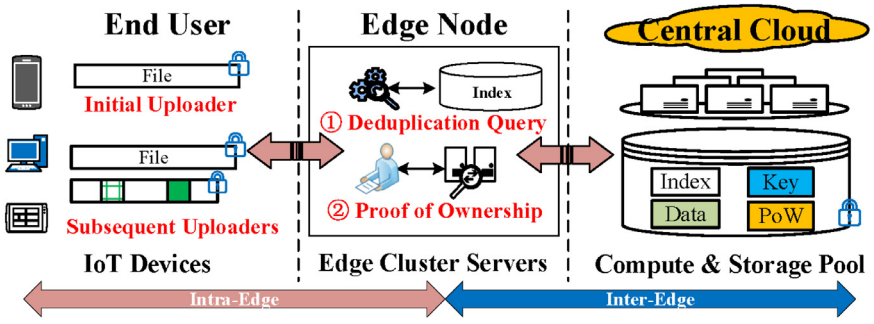


Fig. 3. The system model of SE-PoW in edge computing.

2.2. PoW Schemes & problems

Client-side encrypted deduplication schemes occur from the entities(i.e., IoT devices) and diminish bandwidth consumption significantly. However, the risks of privacy leakage arise in existing schemes, for example, duplicate-faking attacks [15,22]. In particular, an attacker uses a hash value to gain unauthorized access and download files in Dropbox [23]. Researchers propose proof-of-ownership to tackle the problem, which checks ownership and achieves authorized access. Existing schemes are classified into two categories: Merkle Hash Tree based PoW (MHT-PoW) [15,22,25] and Bloom Filter based PoW (BF-PoW) [21,37] in Fig. 2.

MHT-PoW. Halevi et al. [22] propose MHT-PoW to resist duplicate-faking attacks. In Fig. 2(a), the client and server simultaneously encode the file into a buffer via erasure coding and the pairwise independent hash function. The buffer is divided into fixed-size blocks as B_i ($0 < i < n$), and computes the hash value n_i for each data block B_i as the leaf node. And the parent node is to calculate the hash value of the two child nodes. Finally, they get the root node n_{15}. During the verification of MHT-PoW, the server randomly selected N leaf node indexes as the challenge information. The client returns the path information from the leaf node to the root node, and the server finally recalculates and compares the value of the root node. Similarly, ECC-based accumulators are adopted in [15]. Unfortunately, the encoding of files and the construction of structures in MHT-PoW will bring great computational and I/O overhead.

BF-PoW. To reduce the computation and I/O overheads, BF-PoW [21,26,30] uses a bloom filter to resist duplicate-faking attacks with a low error rate. In Fig. 2(b), BF-PoW divides the file into blocks and calculates the token e_i ($1 \le i < 5$) of the corresponding block with a pseudo-random function, and inserts it into the bloom filter. For example, the server selects data blocks 1 and 3 as challenge blocks. The client calculates tokens e_1 and e_3 and queries whether they exist in the bloom filter to check the ownership. When the false positive occurs in a bloom filter or the attackers utilize the previous valid proofs, BF-PoW leads to privacy leakage.

According to our analysis, existing schemes face security and performance challenges. First, encrypted deduplication schemes suffer from brute-force attacks or a heavy computational burden. Second, MHT-PoW incurs extensive time and I/O overheads. BF-PoW is subjected to replay attacks. We analyze the redundant distribution and architectural features in edge computing to solve these problems. From previous work in [4,33,34,38], more than 90.5%–99% redundant data remains in cross-domain duplicate files and duplicate blocks within users. In edge computing, performing deduplication at edge nodes is highly efficient and prevents privacy risks and information leakage. Meanwhile, client-side deduplication between edge nodes and the central saves network bandwidth and achieves security guarantees [14]. These motivate us to propose SE-PoW, a hybrid encrypted deduplication scheme for intra- and inter-edge with proof-of-ownership to achieve higher security in edge computing.

3. System model & threat model

This section firstly describes the system model and threat model of SE-PoW. Next, the security requirements and design goals of SE-PoW are listed as follows.

3.1. System model

Fig. 3 describes our system model that consists of three entities: Central Cloud(CC), Edge Node(EN), and End User (EU). The CC cannot offer high-quality services for large-scale data in a restricted network environment. Edge nodes locate on the user side and provide computing and storage services with limited resources. In the cloud offloading applications, deduplication will be done at edge nodes and the central cloud to save storage space and network bandwidth.

- **Central Cloud(CC).** The CC provides centralized storage /retrieval services. When a user is connected to the CC, CC will verify his password and credential. CC maintains file-level indices for inter-edge data deduplication. CC also stores ciphertexts of blocks, keys, information of PoW, and metadata. It assigns tasks to multiple edge nodes to handle a large amount of data.
- **Edge Node(EN).** The EN is an entity located at the network edge, which provides computing and storage services with limited resources. EN connects to CC via inter-network (e.g., Wide Area Network(WAN)) but communicates with users in a restricted domain (via intra-network). EN acts as a proxy between CC and the

user, supporting duplicate checking, encryption, and challenge-and-response of PoW. A trusted EN assists the user in generating random keys.

- **End User(EU).** The EU is a client or outsourcing entity (e.g., Mobile and IoT devices) consisting of initial and subsequent uploaders. EU uploads data to and retrievals data from the CC through the EN. The EU connects to edge nodes via intra-network (e.g., Local Area Network(LAN)). Moreover, EN can generate keys and encrypt/decrypt data with limited computation and storage resources. The initial uploader transmits data to CC and initializes the PoW. The subsequent uploaders with duplicate files need to verify the PoW protocol.

3.2. Threat model

We assume that the CC is "honest-but-curious" in edge computing. The CC will not maliciously delete or modify users' data, but the CC tries to learn the sensitive information as much as possible, such as data, keys, tags(i.e., hash value), and proofs PoW. Without loss of generality, we assume that the malicious CC may collude with other adversaries. The EN will perform our proposed protocol honestly. We assume that the EN is hard to be compromised in the intra-network [14] and is protected by firewalls and access control systems. A trusted EN helps users to generate secure keys. In our threat model, the adversaries can be classified into two types: outside adversaries and inside adversaries.

- Outside adversaries may be malicious users or hackers. They obtain some sensitive data (e.g., a hash value, proofs of PoW) via a public network, such as a web crawler and artificial intelligence. Outside adversaries aim to get target users' sensitive data content and keys from CC and EN. They may disguise themselves as a legitimate user to interact with the CC or EN.
- Inside adversaries follow the prescribed protocols but try to obtain users' information, such as plaintexts of data, tags, and proofs of a specific file. The inside adversaries try to cheat the EN and CC by using previous proofs and make the verification of PoW successful.

3.3. Security requirements & design goals

We aim to achieve the following security requirements and design goals based on the above threat model.

- **Data confidentiality:** We require that the encrypted data and keys will be achieved semantically secure and resist brute-force attacks [17].
- **Tag consistency:** The deduplication scheme should allow the users to verify data integrity. It can resist poison attacks, in which a malicious attacker cannot upload a valid hash value but replaces a file with a poisoned one.
- **Backward privacy:** When a user uploads a duplicate file that exists in the CC, CC will check the ownership. Unauthorized data owners who cannot pass the verification of the PoW would not access files.
- **Resistance to duplicate-faking attacks:** An attacker who only has the data tag cannot download the corresponding ciphertexts of files.
- **Resistance to replay attacks:** An attacker cannot pass the verification of PoW, even if it generates valid proofs from the previous message without owning files.

Design goals. Our scheme should achieve the following design goals. First, SE-PoW should meet the mentioned security requirements. SE-PoW also realizes upload and download protocols using encrypted deduplication, key generation, and proof-of-ownership among the EU, EN, and CC. Second, SE-PoW ensures system efficiency, which reduces the cost of computation, transmission, and storage. At last, other problems, such as data reliability [39], updating, and ownership management, are beyond the scope of this paper.

Table 1
Notations used in the proposed scheme.

Notation	Description
u_i	An end user
ID_{u_i}	The identity of u_i
F_i	A file
B_i	A block
n	Number of blocks
C	Ciphertext of a block/key
$OList_{F_i}$	An owner list of F_i
$K_{u_i}/K_{F_i}/K_{B_i}$	A user/file/block key
$CF_PoW[F_i]$	A cuckoo filter based PoW
$Sig_g(B_i)$	An algebraic signature
V_i	A tag to resist replay attacks

3.4. Preliminaries

Before introducing the design of SE-PoW, we describe two data structures: cuckoo filter and algebraic signature.

Cuckoo Filter. A cuckoo filter [40] is a data structure that is used to provide approximate set membership tests whether a given item is in a set or not. It is similar to Bloom filter [29]. A cuckoo filter is a compact variant of a cuckoo hash table that stores only fingerprints instead of key–value pairs. A set membership query for item x searches the hash table for the fingerprint of x and returns true if an identical fingerprint is found. A cuckoo filter can show false positives but not false negatives. It supports adding and removing items dynamically. It provides a higher lookup performance than Bloom filters. The cuckoo filter has various advantages over the Bloom filter. (1) It takes less time for lookups. (2) It has fewer false positives than the bloom filter for the same number of items stored. (3) It supports the deletion of items.

Algebraic Signature. Algebraic signature [35,41] is a hash function with homomorphic and algebraic properties. Algebraic signature has been widely used in remote data possession checking in distributed system [35] and cloud storage [42]. An algebraic signature consists of n symbols to verify the uniqueness of data content. The basic feature of the algebraic signature method is that the sum of the algebraic signature of data blocks is equal to the signature result of the corresponding sum of data blocks. Concretely speaking, let λ be a tuple in Galois Field, which $\lambda = (\lambda_1, \lambda_2, \cdots, \lambda_n)$ is a vector of distinct non-zero elements. The file F is divided into n blocks $f[1], f[2], \cdots, f[n]$, and the formula for calculating the algebraic signature of file F is

$$S_\lambda(F) = \sum_{i=1}^{n} f[i] \cdot \lambda^{i-1} \tag{1}$$

The properties of an algebraic signature are as follows:

Property 1. *Concatenating two data blocks $f[i]$ and $f[j]$ of length l and m, into a super block denoted $f[i] \parallel f[j]$. Then the signature $S_\lambda(f[i] \parallel f[j])$ is as follows.*

$$S_\lambda(f[i] \parallel f[j]) = S_\lambda(f[i]) + \lambda^l S_\lambda(f[j]) \tag{2}$$

Property 2. *The algebraic signature of the sum of all data blocks of file F equals the sum of the algebraic signatures of each data block.*

$$\begin{aligned} &S_\lambda(f[1]) + S_\lambda(f[2]) + \cdots + S_\lambda(f[n]) \\ &\quad = S_\lambda(f[1] + f[2] + \cdots + f[n]) \end{aligned} \tag{3}$$

4. Design and implementation of SE-PoW

In this section, we first describe the overview of SE-PoW. Then we present the design and proof of ownership algorithms used in SE-PoW. Table 1 describes the notations.

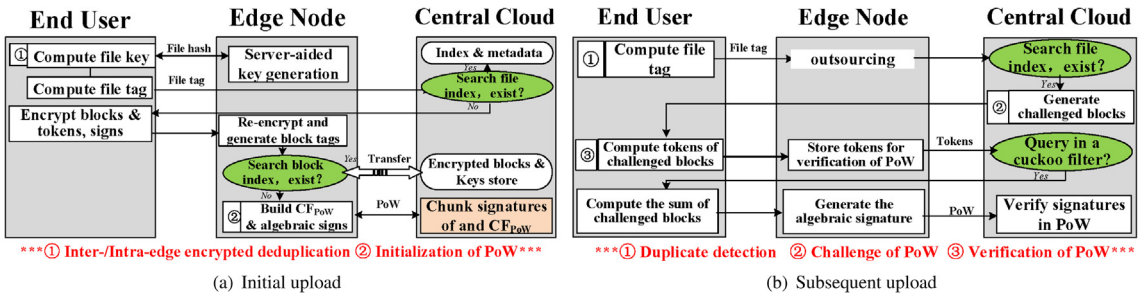


Fig. 4. The procedure of initial and subsequent data upload.

4.1. Overview of SE-PoW

We perform an inter-edge and intra-edge encrypted deduplication scheme for files and blocks in the upload phase. In Fig. 4, the EU is allowed to transfer files to the EN and CC and retrieves relevant files on demand. Data are encrypted via different MLE algorithms according to the location of deduplication to balance security and system efficiency. Specifically, the EU generates a file tag and encrypts the file before sending it to an EN. The EN outsources the file tag to CC for inter-edge(cross-domain) file-level deduplication via a server-aided RCE algorithm. If unique, EU encrypts blocks via an edge-aided MLE algorithm and initializes the tokens and algebraic signatures. EU outsources them to the EN for re-encryption and performs intra-edge block-level deduplication. Besides, the EN stores the data of SE-PoW based on a cuckoo filter and algebraic signatures for PoW verification. The EN transfers non-duplicated blocks and metadata to the CC and initializes a PoW protocol.

In the subsequent upload phase, the EU sends a tag of a duplicate file to the CC. Then the user performs a dual-level proof of ownership for duplicate files in edge computing to ensure data privacy. Concretely speaking, we first perform the challenge-and-response protocol over CF-PoW. If it passes, we will verify the homomorphism of algebraic signatures as the second-level PoW. Only verifying the ownership of SE-PoW, the end-user will send the file metadata without uploading data content.

4.2. Encrypted deduplication in SE-PoW

To resist brute-force attacks and minimize bandwidth overheads in edge computing, we proposed a location-aware hybrid encrypted deduplication in SE-PoW. SE-PoW combines server-aided RCE for inter-edge files and edge-aided MLE for intra-edge blocks. The encryption methods, such as MLE [17], RCE [17], and RSA-OPRF [18], are adopted from previous works. Details are shown as follows.

System setup. We choose two hash functions H_1 and $H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_p$. And the uploader adopts the AES APIs [43], such as Encry() and Decry(). The public parameters e and N of RSA are initialized and d is generated via $e \cdot d \equiv 1 mod \phi(N)$. Each edge node will initialize a master key K_{e_i}.

Data Upload. As show in Fig. 4(a), an user u_i uploads a file F_i to the central cloud. The CC will verify the identity ID_{u_i} and password. The following details are the file/block-level encrypted deduplication.

(1) u_i computes the tag $T_{F_i} \leftarrow H_1(H_1(F_i))$. Then u_i generates a server-aided RCE key K_{F_i} via oblivious pseudorandom protocol [18]. Specifically, for F_i, u_i chooses a random number $r \in N$, and sends $x = H_1(F_i) \cdot r^e \ mod \ N$ to a trusted edge node. The trusted edge node computes $y = x^d \ mod \ N$ and sends y back. u_i calculates $z = y \cdot r^{-1} \ mod \ N$. u_i could verify whether or not $H_1(F_i) \equiv z^e \ mod \ N$. Thus, u_i chooses a random key via $L_{F_i} \leftarrow \{0,1\}^{k(\lambda)}$, and denotes as $K_{F_i} = (L_{F_i}, z)$.

(2) u_i sends T_{F_i} to the edge nodes(EN) and forwards it to the central cloud(CC) for inter-edge file-level deduplication. The CC will check whether T_{F_i} exists in the inter-edge global file index. If no, u_i performs block-level deduplication and jumps to (3). Otherwise, the CC will check the ownership, and details are in Section 4.3.

(3) u_i performs intra-edge block-level encrypted deduplication via an edge-aided MLE. In particular, u_i divides F_i into n blocks via $\{B_i\} \leftarrow$ Chunking (F_i). For a block B_i $(0 \le i < n)$, u_i generates a MLE key K_{B_i} via $K_{B_i} \leftarrow H_1(B_i)$. Then u_i encrypts the block into ciphertexts $C_i || C_i^2 || C_i^3$ via $C_i \leftarrow Encry(K_{B_i}, B_i)$ and $C_i^2 \leftarrow Encry(L_{F_i}, K_{B_i})$, and $C_i^3 \leftarrow z \oplus L_{F_i}$.

(4) u_i transmits all block ciphertexts (C_i, C_i^2, C_i^3) to the EN. Then EN re-encrypts C_i via $C_i^1 \leftarrow$ Encry(K_{e_i}, C_i). The tag of block B_i is generated via $T_{B_i} \leftarrow H_1(C_i^1)$. The EN performs intra-edge block-level encrypted deduplication by checking T_{B_i} in the local block-level index. Then, the EN will upload all the ciphertext of non-duplicated blocks $C_i^1 \ || \ C_i^2 \ || \ C_i^3$ and metadata information $T_{B_i} \ || \ ID_{u_i}$ to the central cloud. The CC receives and stores ciphertexts and metadata. Then the central cloud adds the ID_{u_i} to the owner list $OList_{F_i}$.

(5) The EN has to generate tokens and algebraic signatures to initialize a dual-level PoW protocol. Details are present in the initialization of SE-PoW in Section 4.3.

Subsequent Upload. In Fig. 4(b), an subsequent uploader u_j sends the file F_i to the edge nodes and central cloud. First, u_j generates the file tag T_{F_i} as described in the upload phase. u_j outsources them to the central cloud. The central cloud finds that T_{F_i} exists in the file index via duplicate checking. Second, the central cloud performs a challenge-and-response phase to verify the ownership of u_i. (1) The central cloud randomly generates c challenged blocks and returns the position of blocks. (2) The edge node computes tokens and algebraic signatures of challenged blocks based on the position of blocks and transfers them to the central cloud. (3) The central cloud firstly checks the tokens whether or not they exist in the cuckoo filter. If it passes, the CC will verify the homomorphism of algebraic signatures. Otherwise, the central cloud returns failed results. Details are present in the Algorithm 2. Third, if u_j passes the verification of PoW, his identify ID_{u_j} will be added to the owner list $OList_{F_i}$. And the CC returns results to the EN and u_j.

Data Download. If a user u_i wants to download a file F_i, u_i will firstly send the identity ID_{u_i} and file tag T_{F_i} to the edge nodes and central cloud. The central cloud will firstly verify his identity ID_{u_i} whether or not in the owner list. If no, the download request will be rejected. Otherwise, the CC will read the metadata of F_i to return the ciphertexts of all blocks and keys $C_i^1 \ || \ C_i^2 \ || \ C_i^3$ $(0 \le i < n)$ to the EN. After receiving the ciphertexts, EN decrypts C_i^1 with K_{e_i} via $C_i \leftarrow$ Decry(K_{e_i}, C_i^1) And the EN forwards $C_i \ || \ C_i^2 \ || \ C_i^3$ $(0 \le i < n)$ to u_i. Next, u_i decrypts the block key via $L_{F_i} \leftarrow C_i^3 \oplus z$ and $K_{B_i} \leftarrow$ Decry(L_{F_i}, C_i^2). Thus, u_i decrypts the ciphertext of block to get B_i via $B_i \leftarrow$ Decry(K_{B_i}, C_i). At last, u_i creates a new file F_i and writes each block $B_i (0 \le i < n)$ sequentially to recover the file F_i.

Algorithm 1 The initialization of SE-PoW

Input: File F_i & parameter m of CF.
Output: $CF_{PoW}[F_i]$ & $Sig_g(B_i)$.
1: u_i divides F_i into blocks $B_i(0 \le i < n)$.
2: u_i generates tokens $t_{B_i} \leftarrow H_2(B_i)$.
3: u_i generates algebraic signatures of blocks $Sig_g(B_i) \leftarrow S_\lambda(B_i||ID_{F_i}||i)$.
4: u_i generates $V_i \leftarrow S_\lambda(ID_{F_i}||i)$
5: u_i outsources $t_{B_i}||Sig_g(B_i)||V_i$ to the EN and CC.
6: CC initializes CF_{PoW}=InitCF(m) and PRF.
7: **for** $i = 0 \rightarrow n-1$ **do**
8: $e_i \leftarrow$ PRF(t_{B_i}, i)
9: $CF_{PoW}[F_i] \leftarrow$ AddCF(e_i)
10: **end for**
11: CC stores all $Sig_g(B_i)||V_i$ and $CF_{PoW}[F_i]$.

4.3. Proof of ownership in SE-PoW

We propose a dual-level PoW algorithm over encrypted deduplication with a cuckoo filter and algebraic signatures to resist duplicate-faking attacks and replay attacks. The cuckoo filter [40] and algebraic signature have been used in storage systems [41]. SE-PoW is a challenge-and-response protocol between two entities on a file F: $\Pi = (P, V)$. P is the end-user and the edge node, and V indicates the central cloud. In addition, protocol Π consists of three phases: Initialization(Data upload), challenge, and verification (Subsequent upload). The details are present in Algorithm 1 and 2.

Initialization of SE-PoW. An initial uploader u_i generates tokens and algebraic signatures of each block in file F_i for the ownership verification. As shown in algorithm 1, u_i firstly divides F_i into blocks $B_i(0 \le i < n)$. u_i generates tokens via $t_{B_i} \leftarrow H_2(B_i)$ and algebraic signatures $Sig_g(B_i) \leftarrow S_\lambda(B_i \| ID_{F_i} \| i) \leftarrow \sum_{j=1}^{n}(B_{i,j} \| ID_{F_i} \| i) \cdot \lambda^{j-1}$. ID_{F_i} is the identity of file F_i and i is the index of block B_i. u_i also computes $V_i \leftarrow S_\lambda(ID_{F_i} \| i)$. Then SE-PoW resists replay attacks via unique labels $ID_{F_i} \| i$. u_i sends tokens t_{B_i} and algebraic signatures $Sig_g(B_i)$ to the EN. Next, the EN outsources tokens and signatures to the CC. Then the central cloud constructs a cuckoo filter $CF_{PoW}[F_i] \leftarrow$ InitCF(m) with the parameter of total items m. For each token $t_{B_i}(0 \leqslant i < n)$, the CC computes $e_i \leftarrow$ PRF(t_{B_i}, i) with a pseudorandom function PRF. The CC inserts all tokens into a cuckoo filter $CF_{PoW}[F_i] \leftarrow AddCF(e_i)$. Finally, the CC stores all the algebraic signatures $Sig_g(B_i) \,||\, V_i$ and $CF_{PoW}[F_i]$.

If a subsequent uploader u_j uploads a file F_i, u_j will perform the challenge and verification of SE-PoW to resist duplicate-faking attacks and replay attacks in Algorithm 2.

Challenge of SE-PoW. An subsequent uploader u_j generates the file tag and outsources it to the CC to perform inter-edge file-level deduplication. Specifically, u_j computes the file tag $T_{F_i} \leftarrow H_1(H_1(F_i))$ and outsources T_{F_i} to the central cloud. The CC searches T_{F_i} in the global file index. If it does not exist, the CC will return the result to u_j. Otherwise, CC randomly selects c indices of blocks $I[k]$ $(0 \leqslant k < c)$ as the challenge, and returns it to the edge node and u_j.

Verification of SE-PoW. Then, CC will perform verification of SE-PoW among CC, EN, and u_j via a dual-level PoW, including a cuckoo filter and algebraic signatures. Specifically, details are described in Algorithm 2. (1) u_j divides file F_i' into blocks and generates tokens of the challenged position belong to $I[k]$ $(0 \leqslant k < c)$ via $t_{B_k'} \leftarrow H_2(B_k')$. Then u_j sends $t_{B_k'}$ to EN and CC for the first-level verification of PoW. CC receives $t_{B_k'}$ and computes $e_k' \leftarrow$ PRF($t_{B_k'}$, k). Then CC executes $\eta =$ ContainCF($CF_{PoW}[F_i]$, e_k') for all tokens. If any token does not exist in CF_{PoW}, u_j does not pass the first-level verification of SE-PoW. (2) If u_j passes the first-level verification, CC will request u_j to verify the homomorphism of algebraic signatures. u_j reads the challenged blocks $B_k'(k \in I[k])$ and computes the sum of challenged blocks via $\gamma \leftarrow \sum_{k=0}^{c-1} B_k'$. Then, u_j sends γ to the EN. EN computes the algebraic signature $\sigma \leftarrow Sig_g(\gamma)$ and sends σ to the CC. (3) CC reads the block signature $Sig_g(B_k)$ of F_i and executes $\mu \leftarrow \sum_{k=0}^{c-1} Sig_g(B_k) \oplus V_k$ $(k \in I[k])$. Finally, CC verifies whether σ equals μ or not. If no, CC will return that u_j does not pass the verification. Otherwise, CC will add ID_{u_j} to the owner list $OList_{F_i}$. u_j just updates the metadata of F_i and does not upload the content of F_i.

Algorithm 2 The challenge and verification of SE-PoW

Input: $CF_{PoW}[F_i]$ & $Sig_g(B_k)$.
Output: The result of SE-PoW verification.
1: CC generates the index of challenged blocks $I[k]$ $(0 \leqslant k < c)$ and sends to u_j.
2: EN requests u_j to divide F_i' into B_i' $((0 \le i < n))$ and selects challenged blocks B_i' according to $I[k]$
3: **while** $k \in I[k]$ **do**
4: u_j executes $t_{B_k'} \leftarrow H_2(B_k')$ and sends to EN.
5: CC executes $e_k' \leftarrow$ PRF($t_{B_i'}$, k).
6: CC executes $\eta =$ ContainCF(e_k') (First-level PoW)
7: **if** $\eta = 0$ **then**
8: **return** $\perp$
9: **end if**
10: **end while**
11: CC verifies the second-level PoW.
12: **while** $k \in I[k]$ **do**
13: u_j reads the blocks B_k'.
14: u_j executes $\gamma \leftarrow \sum_{k=0}^{c-1} B_k'$.
15: **end while**
16: u_j and EN compute $\sigma \leftarrow Sig_g(\gamma)$ and send σ to CC.
17: **while** $k \in I[k]$ **do**
18: CC reads the signature $Sig_g(B_k)$ and V_k of F_i.
19: CC executes $\mu \leftarrow \sum_{k=0}^{c-1} Sig_g(B_k) \oplus V_k$
20: **end while**
21: **if** $\sigma = \mu$ **then**
22: **return** 1
23: **else**
24: **return** 0
25: **end if**

4.4. Implementation detail of SE-PoW

We propose a prototype based on the design of SE-PoW. To achieve the balance between security and efficiency, SE-PoW implements a dual-level hybrid encrypted deduplication in edge computing. Thus, SE-PoW lessens the pressure on network bandwidth and improves data security and privacy. Specifically, SE-PoW adopts a global file index in the central cloud and block indices in the edge nodes. The index is a key–value storage structure for tags and data storage locations, for example, hash tables. The key is the block's tag, and the value is the physical address of the data block (such as block offset and length). Furthermore, the hash and encryption function in SE-PoW is the CTR mode of SHA-256 and AES-256 [43], and the token calculation uses the SHA-1 function. The secure network transmission between the edge nodes and the central cloud uses SSL/TLS [43]. A trusted edge node is used to compute server-aided keys, and RSA-OPRF [18] is implemented for evaluation.

To realize the dual-level PoW, SE-PoW uses an efficient cuckoo filter [40] with better performance and lower false positive rate than a bloom filter. The cuckoo filter supports InitCF(), AddCF(), ContainCF() and DeleteCF(). In addition, the overall collision probability of an algebraic signature used in SE-PoW is very low [35].

5. Security analysis

SE-PoW is designed to ensure data confidentiality and backward privacy and resist attacks for encrypted deduplication in edge computing. We consider two types of adversaries: inside and outside adversaries. We assume that the following technologies are secure, such as symmetric encryption [43] and OPRF protocols [18]. In worst cases, the adversaries may compromise the CC and collude with users.

5.1. Data confidentiality

In this case, the adversary gets the ciphertexts of blocks by compromising the CC or EU. SE-PoW resists brute-force attacks in the hybrid deduplication scheme and ensures data confidentiality and tag consistency.

In general, the adversary obtains the ciphertext of target block $C_i^1 \| C_i^2 \| C_i^3$ $(0 \le i < n)$ from a specific file F_i. The adversary knows that the blocks $\{B_i'\}(0 \le i < n)$ are from a specific set $|S|$. For each block B_i', the adversary first gets the hash to get the key via K_{B_i}. The adversary gets the ciphertext via $C_i^{1'} \leftarrow Encry(K_{B_i'}, B_i')$ and compares it with C_i^1. However, C_i^1 is protected by the master key K_{e_i} of each EN. SE-PoW generates a random file key K_{F_i} via an oblivious pseudorandom function. All block keys K_{B_i} are protected securely by random key $L_i \,||\, K_{F_i}$. Thus the adversary cannot get the plaintext of file F_i. As a result, SE-PoW can resist brute-force attacks to ensure data confidentiality. In addition, the adversary compromises the data integrity by colluding with users. It uploads the valid tags but replaces the blocks with poisoned data. SE-PoW computes the hash value of C_i^1 via $T_{B_i'} \leftarrow H_1(C_i^1)$ and compares whether or not $T_{B_i'}$ equals T_{B_i}. Thus, SE-PoW ensures tag consistency.

We discuss the security of SE-PoW under different situations. In the best case, the adversary compromises the CC but cannot access the EN. All data and metadata stored in the CC are encrypted with random keys. The adversary cannot obtain the plaintext of files even if it performs brute-force attacks. In the worst case, the adversary may get the master key of a specific EN and collude with malicious users. SE-PoW can still ensure security for unpredictable data that are not falling into a known set. The users access the EN through an intra-network, which naturally faces fewer security threats than inter-network. SE-PoW makes the worst-case rarely occur by further protecting the EN and file metadata with access control policies.

5.2. Security of proof of ownership

For a file F_i, the adversary's goal is to pass the verification of SE-PoW by leveraging replay attacks or the false positive in a cuckoo filter. The adversary knows parts of the file, but he does not own the entire content of the file.

We define that the event v_i is the adversary could pass the verification of SE-PoW when he gets a *token*. It happens in the following two cases: (1) The adversary receives the correct proof. (2) When the cuckoo filter checks the element, a false positive occurs. We define the false positive of the CF as p_f. According to the above analysis, the probability of event v_i can be described as:

$$\begin{aligned} P(v_i) &= P(v_i \cap (token_i \cup \overline{token_i})) \\ &= P(v_i|token_i)P(token_i) + P(v_i|\overline{token_i})P(\overline{token_i}) \\ &= P(token_i) + p_f P(\overline{token_i}) \end{aligned} \tag{4}$$

The adversary performs replay attacks by leveraging the previous proofs and the false positive of the cuckoo filter. After receiving the proofs, the CC will verify the ownership. To resist these attacks, SE-PoW adopts algebraic signatures as the second verification of PoW. It satisfies the property that the sum of algebraic signatures of challenged blocks equals the signature of the sum of challenged blocks. That is whether or not $\sigma = \mu$.

$$\begin{aligned} \sigma &= Sig_g(\gamma) \\ &= S_\lambda(\sum_{k=0}^{c-1} B_k') \\ &= S_\lambda(B_0' + B_1' + \cdots + B_{c-1}') \\ &= S_\lambda(B_0') + S_\lambda(B_1') + S_\lambda(B_{c-1}') \\ &= \sum_{k=0}^{c-1} S_\lambda(B_k') \end{aligned} \tag{5}$$

After receiving the proofs from the EN and end user, the CC could verify the ownership.

$$\begin{aligned} \mu &= \sum_{k=0}^{c-1} Sig_g(B_k) \oplus V_i \\ &= \sum_{k=0}^{c-1} S_\lambda(B_k \| ID_{F_i} \| i) \oplus S_\lambda(ID_{F_i} \parallel i) \\ &= \sum_{k=0}^{c-1} S_\lambda(B_k) \oplus \lambda^l S_\lambda(ID_{F_i} \parallel i) \oplus S_\lambda(ID_{F_i} \parallel i) \\ &= \sum_{k=0}^{c-1} S_\lambda(B_k) \\ &= \sigma \ (B_k' = B_k) \end{aligned} \tag{6}$$

Then, SE-PoW prevents the attacks of the false positive of CF via a dual-level PoW. Moreover, SE-PoW can resist replay attacks because the adversary does not know V_i. Thus we denote:

$$p_f P(\overline{token_i}) = 0, \quad and \quad P(v_i) = P(token_i) \tag{7}$$

We define event g_i, the adversary gets tokens of the challenged block B_i, and the probability is p. The token is the output of the hash function of H_2 with the length l. Based on the random oracle model, the probability of guessing the correct token is 2^{-l}. Thus, the probability of event $token_i$ is:

$$\begin{aligned} P(token_i) &= P(token_i \cap (g_i \cup \overline{g_i})) \\ &= P(token_i|g_i)P(g_i) + P(token_i|\overline{g_i})P(\overline{g_i}) \\ &= p + (1-p) \cdot 2^{-l} \end{aligned} \tag{8}$$

The adversary needs to get at least c tokens of challenged blocks. Thus, the probability $P(succ)$ is defined as the adversary can pass the verification of SE-PoW.

$$P(succ) = (p + (1-p) \cdot 2^{-l})^c \tag{9}$$

We set up a security parameter k to derive a lower bound for c, that is $P(succ) \leqslant 2^k$. To ensure the security of SE-PoW, the number of challenged blocks is:

$$c \ge \frac{k \ln 2}{p + (1-p) \cdot 2^{-l}} \tag{10}$$

The probability of running a successful SE-PoW should be negligible under the security parameter k and the number of tokens c. SE-PoW can resist duplicate-faking attacks, and it also prevents replay attacks and the false positive of CF.

5.3. Security discussion of SE-PoW

Table 2 shows the comparison results of encrypted deduplication schemes. Halevi [22] and Xu [25] refer to the encrypted deduplication schemes that implement with MHT-PoW and the variants of CE. Yang [15] encrypts data with server-aided MLE and achieves MHT-PoW via ECC-based accumulators. In addition, Lorena [37] and Jiang [21] realize an encrypted deduplication via BF-PoW. The difference is that

Table 2
Comparison of encrypted schemes with PoW.

Scheme	Brute-force attack	Duplicate-faking attack	Replay attack	Perf.[a]
Halevi [22]/ Xu [25]	×	√	×	*L*
Yang [15]	√	√	×	*L*
Lorena [37]	×	√	×	*H*
Jiang [21]	×	√	×	*H*
SE-PoW	√	√	√	*H*

[a]"L" means Low and "H" refers to High.

Table 3
Description of three real-world datasets.

Name	Size (GB)	Num.	Description
LNX-set	111.32	258	258 tar files of linux source code
VMA-set	58.67	135	Virtual machine images, including Fedroa & Ubuntu.
WEB-set	43.31	16	16 days of snapshot files, retrieval depth is 3 by wget

they use CE and RCE, respectively. We discuss them regarding resistance to brute-force attacks, duplicate-faking attacks and replay attacks, and performance.

Since all the schemes allow users to encrypt data and realize deduplication over ciphertexts, they can guarantee data confidentiality. On the one hand, the method of Halevi et al. suffers from brute-force attacks because of the utilization of CE. The scheme of Halevi et al. [22] and Yang et al. [15] are both vulnerable to replay attacks and incur large time overheads due to the Merkle Hash Tree. On the other hand, Lorena et al. [37], and Jiang et al. [21] cannot prevent brute-force attacks and replay attacks, but they achieve high performance. As mentioned above, SE-PoW can resist brute-force attacks and ensure data confidentiality and tag consistency. Furthermore, SE-PoW also resists duplicate-faking attacks and replay attacks to ensure backward privacy, only adding little overheads compared with the scheme of Jiang [21].

6. Performance evaluation

6.1. Experimental setup

Platform: We conduct experiments to evaluate the performance of SE-PoW. These machines are equipped with an Intel(R) Core(TM) i7-4770@3.40 GHZ 8-core CPU, 96 GB memory and 2 TB hard disk. They are installed with an Ubuntu 20.04 LTS 64-bit operation system. These machines are connected with 100 Mbps and 1000 Mbps ethernet network.

Methodology: To evaluate the performance of SE-PoW, we implement a research prototype to compare the related schemes, including MHT-PoW [22,25] and BF-PoW [21,26,30,37]. MHT-PoW is a file-level encrypted deduplication scheme based on the MHT and variants of CE [22,25]. BF-PoW refers to Jiang et al. [21] scheme that is a block-level encrypted deduplication scheme with BF-PoW and hybrid RCE algorithms. Meanwhile, the encryption schemes consist of convergent encryption(CE), server-aided Message-locked Encryption(MLE), and SE-PoW. We mainly use quantitative metrics for encryption time, the cumulative time of SE-PoW, initial and subsequent upload time, metadata, and storage overheads. The time of SE-PoW consists of phases: initialization, challenge, and verification. We also observe the impacts of varying block size, number of tokens, and file size.

Finally, the security parameters are set according to MHT-PoW [22] and BF-PoW [21,37]. Where security parameters, the number k is 66, and the token length is set to 16 bytes. According to formula (10) in the security analysis, the number of challenge blocks is set {102, 204, 509, 1017}. Note that our evaluation results should be interpreted as an approximate assessment of other schemes.

Datasets: There are two types of datasets used in SE-PoW for performance evaluation, including synthetic datasets and real-world datasets. (1) Synthetic datasets: artificial files with random content of different sizes or different average block size, and each file is divided into fixed-size blocks. (2) Table 3 describes the real-world datasets, which contain three different types, namely LINUX-set, VMA-set and WEB-set. Linux-set contains the tar package file of the 258 version of the Linux source code. VMA-set [44] is collected images of different operating systems of virtual machines, including Fedora and Ubuntu. WEB-set is a snapshot of 15-day web pages downloaded from *news.sina.com* using the tool *wget*, and the maximum retrieval depth is 3.

6.2. A sensitivity study on encryption & PoW

This subsection evaluates the performance of encryption and proof-of-ownership algorithms varying different block sizes and file sizes with synthetic datasets. First, to assess time overheads of encryption, we upload a 1024 MB unique file repeatedly with varying block sizes, i.e., 2 kB, 4 kB, and 8 kB. Second, to evaluate time overheads of related pow schemes, we use files that are generated with random contents of size 2^i kB for $i \in \{5, \ldots, 21\}$, which ranging from 16 kB to 2048 MB. Third, to evaluate the performance of SE-PoW, we use a 2 GB file varying different average block sizes, file sizes, and the number of tokens.

Fig. 5(a) shows that the location-aware hybrid encryption scheme used in SE-PoW reduces more time overheads than server-aided MLE, and it is similar to CE and RCE as discussed in Section 4.2. In addition, the larger the average block size, the shorter time overhead. It is because the OPRF protocol costs a lot and the time of key generation decreases, as discussed in Section 2. As described in other papers [30], encrypted deduplication schemes based on proxy re-encryption [21] also incur high computation cost.

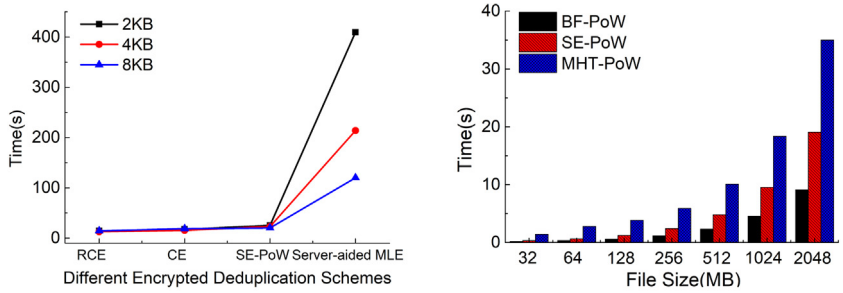
We also evaluate the overheads on the edge side of SE-PoW. SE-PoW mainly adds the time overheads of data re-encryption, tag generation, and duplication checking in the block-level index. For example, we use a 2 GB unique file with random content, and the average block size is 8 kB. We evaluate the server-side overhead of SE-PoW. The re-encryption time is 10.639 s. The time of tag generation and duplication checking are 0.841 s and 8.576 s, respectively.

Fig. 5(b) shows the results that SE-PoW significantly reduces the cumulative time compared with MHT-PoW and only increases little overheads than BF-PoW as discussed in Section 4.3. Specifically, for an individual file of 1 GB, SE-PoW reduces 70–86.7% time overheads relative to MHT-PoW. The erasure coding and construction of the Merkle hash tree used in MHT-PoW incur large time overheads. Compared with BF-PoW, SE-PoW only increases 13.2–14.9% the cumulative time overheads because of the calculation of algebraic signatures.

Fig. 6(a), (b) and (c) evaluate the time overheads of the proof of ownership protocol in SE-PoW, including initialization, challenge and verification phases. Fig. 6(a) shows that the cumulative time increases with the file size, and the initialization phase accounts for more than 60%. The time overhead of PoW is low. For example, SE-PoW costs 0.88 s for a file with 2 GB. Fig. 6(b) evaluates the performance of varying different average block sizes. The result shows that the cumulative time of SE-PoW decreases with the increase of the average block size. Fig. 6(c) shows that only the verification phase costs more time for a larger number of tokens, as discussed in Section 4.3.

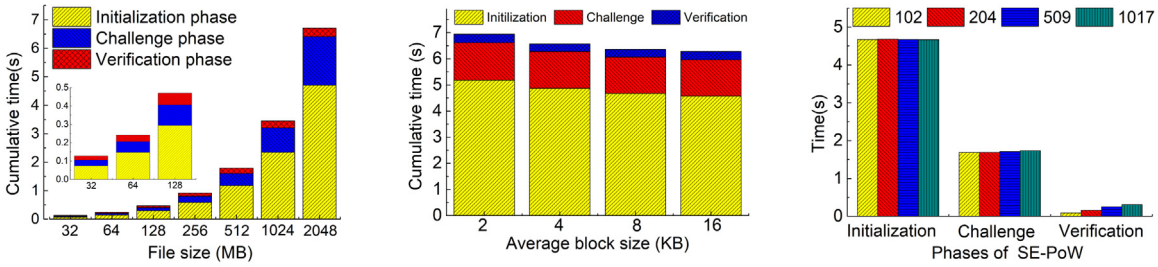
6.3. Evaluating SE-PoW on real-world datasets

In this subsection, we evaluate the overall performance of SE-PoW compared with MHT-PoW and BF-PoW on three real-world datasets. First, we assess the storage and metadata overheads. Second, the user



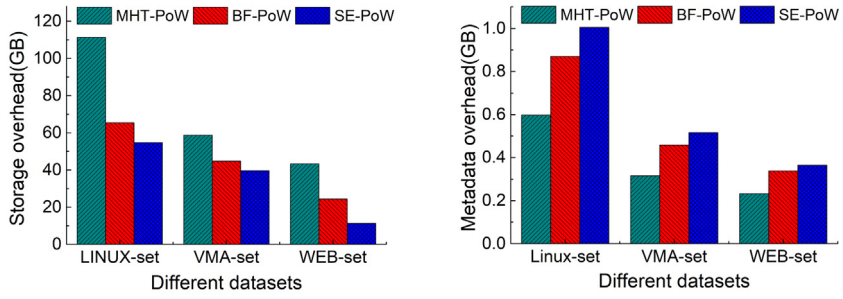
(a) Time of encrypted deduplication schemes (b) The cumulative time of three PoWs

Fig. 5. The encryption time on different average block size and the cumulative time of three PoW methods.



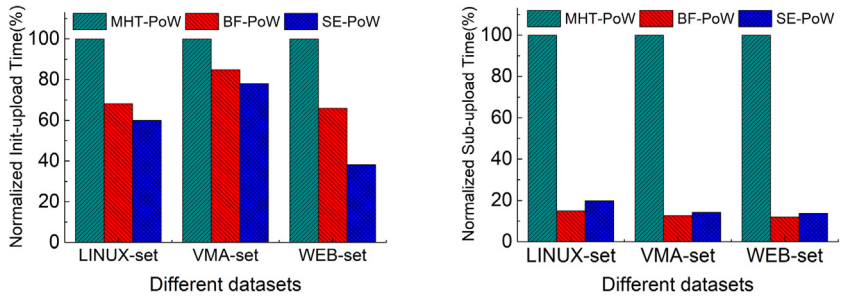
(a) File Size (b) Average block size (c) Number of tokens

Fig. 6. He cumulative time overheadT of SE-PoW varying on file size, average block length and the number of tokens.



(a) Storage overhead (b) Metadata overhead

Fig. 7. Comparison of storage overhead and metadata overhead under datasets of MHT-PoW, BF-PoW and SE-PoW.



(a) Initial upload time of the file (b) Subsequent upload time of the file

Fig. 8. The relative time of the initial and subsequent uploads of files under datasets of MHT-PoW, BF-PoW and SE-PoW.

performs file-level and block-level deduplication in the first upload, uploads non-duplicated data blocks, and initializes the PoW protocol.

Fig. 7(a), compared with MHT-PoW, SE-PoW reduces storage overhead by 31.7–73.3%. SE-PoW reduces storage overhead by 10.8–52.9% relative to BF-PoW. In Fig. 7(b), the growth trend of metadata overhead is exactly the opposite of storage overhead. SE-PoW increases 56.6–68.2% and 7.7–15.6% metadata overheads compared with MHT-PoW and BF-PoW. MHT-PoW has less metadata. BF-PoW needs to store tokens of blocks, while SE-PoW stores extra algebraic signatures of all blocks. Compared to MHT-PoW and BF-PoW, SE-PoW reduces the overall data and metadata storage overhead by 31.7–73.3% and 10.8–52.9%, respectively. It is because that SE-PoW combines them for inter-edge and intra-edge and utilizes a content-defined chunking algorithm to balance efficiency and storage overheads.

As shown in Fig. 8(a), in the initial upload, SE-PoW reduces 21.9–61.9% and 6.8–27.7% upload time compared with MHT-PoW and BF-PoW under the real-world datasets. The encoding and construction of the Merkle Hash tree bring large computation overheads, as

discussed in Section 2.2. Fig. 8(b) shows that SE-PoW reduces the subsequent upload time by over 80% compared with MHT-PoW. And SE-PoW increases about 14.4% subsequent upload time related to BF-PoW. Compared with BF-PoW, SE-PoW adds the calculation overheads of algebraic signatures (See Section 4.3).

7. Related work

Edge computing has been gaining much popularity in recent years. Data deduplication at the network edge can exploit the geographic distribution and low latency to achieve high performance and optimized storage cost. Li et al. [11] partition the resource-constrained edge nodes into disjoint clusters. They perform decentralized deduplication within these clusters to improve the deduplication ratio. They also present HotDedup [13] to maximize edge service rate and storage efficiency with deduplication at the network edge by exploiting data popularity and similarity. Cheng et al. [12] proposed LOFS, a file storage strategy via a three-layer hash mapping scheme to allocate files to the proper edge servers for data deduplication. However, they do not solve the problem of data confidentiality and proof-of-ownership.

Encrypted Deduplication. To protect data confidentiality of deduplication, randomized convergent encryption(CE) and message-locked encryption(MLE) and their variants have been proposed in [16,17]. To resist brute-force attacks, DupLESS [18] and ClearBox [45] leverage server-aided MLE via an oblivious pseudorandom protocol (e.g., RSA-OPRF, BLS-OPRF). Liu et al. [46] present a secure deduplication scheme without additional independent servers by using a PAKE protocol. The convergent key management [38,47] are studied to ensure key reliability and reduce space overheads. Moreover, encrypted deduplication has gained much attention in fog and edge computing. Koo et al. [14] combine server-side deduplication and client-side deduplication in fog computing. Fo-SSD [48] leverages BLS-OPRF to support encrypted deduplication and enables fog nodes to remove replicate data. Yang et al. [15] use a hash proof system-based OPRF to resist brute-force attacks and provide dynamic cross-domain deduplication in blockchain-enabled edge computing. However, they do not address the problem of PoW or suffer from potential attacks and time overheads.

Proof-of-Ownership. To solve the problem that attackers can access files with a small hash value, Halevi et al. [22] present MHT-PoW, using erasure coding to build a Merkle Hash Tree(MHT) for ownership verification. Ng et al. [49] proposed a private PoW scheme over encrypted data. Xu et al. [25] firstly encrypt data and generate a hash digest to construct a Merkle Hash Tree, which enhances data security of client-side deduplication under a bounded leakage setting. Yang et al. [15] adopt ECC-based accumulators for MHT-PoW and achieve better performance. However, these schemes require high computation and I/O overheads, which are not suitable for edge computing. Pietro et al. [50] propose s-PoW to reduce computation and I/O overheads, which outputs a proof with each bit that is selected at a random position of the file. BF-PoW [21,26,30,37] generates a token for each block and inserts tokens into a bloom filter for ownership checking under the bounded leakage setting. In addition, access control and user revocation have been studied. REED [39] encrypts data with a deterministic version of the all-or-nothing transform. It achieves deduplication with dynamic access control. Nevertheless, they suffer from privacy leakage of false positives in a bloom filter and replay attacks.

8. Conclusion

Nowadays, edge computing employs data deduplication to reduce storage and computation overheads. However, the state-of-the-art schemes face some security and performance problems, including data confidentiality and security of proof-of-ownership. We design SE-PoW, which employs a location-aware hybrid encrypted deduplication method and a dual-level security-enhanced proof-of-ownership algorithm.

To resist brute-force attacks, SE-PoW exploits server-aided RCE for inter-edge file-level encrypted deduplication. For non-duplicate files, SE-PoW utilizes edge-aided MLE for intra-edge block-level encrypted deduplication. To resist duplicate-faking attacks, we further exploit a cuckoo filter as the first-level PoW to verify the ownership. Then we prove the homomorphism of algebraic signatures to enhance the security of SE-PoW and resist replay attacks. Finally, the security analysis demonstrates that SE-PoW achieves higher security. And the performance evaluation makes it clear that SE-PoW is efficient compared with the state-of-the-art schemes. The problems of data reliability, updating, and dynamic user management are our future 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

We want to thank the reviewers and editors for their constructive comments and suggestions. This research is partly supported by ZDSY20200811143600002. We also thank anyone who helped us improve this work.

References

[1] Fengmin Tang, Feng Gao, Zilong Wang, Driving capability-based transition strategy for cooperative driving: From manual to automatic, IEEE Access 8 (2020) 139013–139022.
[2] Wang Xiaoyang, Ma Yao, Wang Yiqi, Jin Wei, Wang Xin, Tang Jiliang, Jia Caiyan, Jian Yu, Traffic flow prediction via spatial temporal graph neural network, in: Proceedings of the Web Conference 2020(WWW'20), 2020, pp. 1082–1092.
[3] The future of data: Data age 2025, 2019, https://www.seagate.com/files/www-content/our-story/trends/files/Seagate-WP-DataAge2025-March-2017.pdf.
[4] Rethik data: Put more of your business data to work - from edge to cloud, 2021, https://www.seagate.com/files/www-content/our-story/rethink-data/files/Rethink_Data_Report_2020.pdf.
[5] Tim Süß, Tunahan Kaya, Markus Mäsker, Andre Brinkmann, Deduplication analyses of multimedia system images, in: USENIX Workshop on Hot Topics in Edge Computing (HotEdge'18), Boston, MA, 2018.
[6] Jianbing Ni, Kuan Zhang, Yong Yu, Xiaodong Lin, Xuemin Sherman Shen, Providing task allocation and secure deduplication for mobile crowdsensing via fog computing, IEEE Transactions on Dependable and Secure Computing(TDSC) 17 (3) (2020) 581–594.
[7] Yu Wang Hongyang Yan, Chunfu Jia, Centralized duplicate removal video storage system with privacy preservation in IoT, Sensors 18 (6) (2018) 1814.
[8] Dropbox, 2022, https://www.dropbox.com/.
[9] Microsoft OneDrive, 2022, https://drive.google.com/.
[10] Ctera edge X series, 2022, https://www.ctera.com/x-series/.
[11] Shijing Li, Tian Lan, Bharath Balasubramanian, Moo-Ryong Ra, Hee Won Lee, Rajesh Panta, EF-Dedup: Enabling collaborative data deduplication at the network edge, in: 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), 2019, pp. 986–996.
[12] Geyao Cheng, Deke Guo, Lailong Luo, Junxu Xia, Siyuan Gu, LOFS: A Lightweight online file storage strategy for effective data deduplication at network edge, IEEE Trans. Parallel Distrib. Syst. (TPDS) (01) (2021) 1.
[13] Shijing Li, Tian Lan, Hotdedup: managing hot data storage at network edge through optimal distributed deduplication, in: IEEE INFOCOM 2020-IEEE Conference on Computer Communications, IEEE, 2020, pp. 247–256.
[14] Dongyoung Koo, Youngjoo Shin, Joobeom Yun, Junbeom Hur, A hybrid deduplication for secure and efficient data outsourcing in fog computing, in: IEEE CloudCom'2016, 2016, pp. 285–293.
[15] Yang Ming, Chenhao Wang, Hang Liu, Yi Zhao, Jie Feng, Ning Zhang, Weisong Shi, Blockchain-enabled efficient dynamic cross-domain deduplication in edge computing, IEEE Internet Things J. (2022) 1.
[16] J. Douceur, A. Adya, W.J Bolosky, et al., Reclaiming space from duplicate files in a serverless distributed file system, in: Proceedings of IEEE ICDCS, 2002, pp. 617–624.
[17] M. Bellare, S. Keelveedhi, T. Ristenpart, Message-locked encryption and secure deduplication, in: Proceedings of EUROCRYPT, 2013, pp. 296–312.
[18] S. Keelveedhi, M. Bellare, T. Ristenpart, DupLESS: server-aided encryption for deduplicated storage, in: Proceedings of Usenix Security, 2013, pp. 1–16.

[19] Jingwei Li, Zuoru Yang, Yanjing Ren, Patrick P.C. Lee, Xiaosong Zhang, Balancing Storage Efficiency and Data Confidentiality with Tunable Encrypted Deduplication, Association for Computing Machinery, New York, NY, USA, 2020.
[20] Junbeom Hur, Dongyoung Koo, Youngjoo Shin, Kyungtae Kang, Secure data deduplication with dynamic ownership management in cloud storage, IEEE Transactions on Knowledge and Data Engineering(TKDE) 28 (11) (2016) 3113–3125.
[21] Shunrong Jiang, Tao Jiang, Liangmin Wang, Secure and efficient cloud data deduplication with ownership management, IEEE Transactions on Services Computing (TSC) 13 (6) (2018) 1152–1165.
[22] Shai Halevi, Danny Harnik, Benny Pinkas, Alexandra Shulman-Peleg, Proofs of ownership in remote storage systems, in: Proceedings of ACM CCS, 2011.
[23] Dropship: dropbox aip utilities, 2012, https://github.com/driverdan/dropship.
[24] Martin Mulazzani, Sebastian Schrittwieser, Manuel Leithner, Markus Huber, Edgar Weippl, Dark clouds on the horizon: Using cloud storage as attack vector and online slack space, in: The 20th USENIX Security Symposium (Security'11), 2011, pp. 363–370.
[25] Jia Xu, Ee-Chien Chang, Jianying Zhou, Weak leakage-resilient client-side deduplication of encrypted data in cloud storage, in: Proceedings of ACM AsiaCCS, 2013, pp. 195–206.
[26] J. Blasco, R. DiPietro, A. Orfila, A. Sorniotti, A tunable proof of ownership scheme for deduplication using Bloom filter, in: Proceedings of IEEE CNS, 2014, pp. 481–489.
[27] Dongyoung Koo, Junbeom Hur, Privacy-preserving deduplication of encrypted data with dynamic ownership management in fog computing, Future Generation Computer Systems(FGCS) 78 (2018) 739–752.
[28] Haoran Yuan, Xiaofeng Chen, Jianfeng Wang, Jiaming Yuan, Hongyang Yan, Willy Susilo, Blockchain-based public auditing and secure deduplication with fair arbitration, Inform. Sci. 541 (2020) 409–425.
[29] B.H. Bloom, Spacetime trade-offs in hash coding with allowable errors, Commun. ACM 13 (7) (1970) 422–426.
[30] Jinbo Xiong, Yuanyuan Zhang, Shaohua Tang, Ximeng Liu, Zhiqiang Yao, Secure encrypted data with authorized deduplication in cloud, IEEE Access 7 (2019) 75090–75104.
[31] Xue Yang, Rongxing Lu, Kim Kwang Raymond Choo, Fan Yin, Xiaohu Tang, Achieving efficient and privacy-preserving cross-domain big data deduplication in cloud, IEEE Trans. Big Data 8 (1) (2022) 73–84.
[32] Yong Yu, Yafang Zhang, Jianbing Ni, Man Ho Au, Lanxiang Chen, Hongyu Liu, Remote data possession checking with enhanced security for cloud storage, 52 (C), November 2015.
[33] Dutch T. Meyer, William J. Bolosky, A study of practical deduplication, in: The 9th USENIX Conference on File and Storage Technologies (FAST'11), USENIX Association, San Jose, CA, USA, 2011, pp. 229–241.
[34] The digitization of the world - from edge to core, 2019, https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf.
[35] Thomas J.E. Schwarz, Ethan L. Miller, Store, forget and check: Using algebraic signatures to check remotely administered storage, in: Proceedings of IEEE ICDCS, 2006, pp. 1–12.
[36] Jianbing Ni, Xiaodong Lin, Kuan Zhang, Yong Yu, Secure and deduplicated spatial crowdsourcing: A fog-based approach, in: IEEE GLOBECOM, 2016, pp. 1–6.
[37] Lorena Gonz'alez-Manzano, Agusti'n Orfila, An efficient confidentiality-preserving proof of ownership for deduplication, J. Netw. Comput. Appl. 50 (2015) 49–59.
[38] Yukun Zhou, Dan Feng, Wen Xia, Min Fu, Fangting Huang, Yucheng Zhang, Chunguang Li, SecDep: A user-aware efficient fine-grained secure deduplication scheme with multi-level key management, in: Proceedings of IEEE MSST, 2015, pp. 1–14.
[39] Jingwei Li Chuan Qin, Patrick P.C. Lee, The design and implementation of a rekeying-aware encrypted deduplication storage system, ACM Trans. Storage (TOS) 13 (1) (2017) 9:1–9:30.
[40] B. Fan, D. G. Andersen, M. Kaminsky, M. D. Mitzenmacher, Cuckoo filter: Practically better than bloom, in: Proceedings of ACM CoNEXT, 2014, pp. 75–88.
[41] W. Litwin, T. Schwarz, Algebraic signatures for scalable distributed data structures, in: Proceedings. 20th IEEE ICDE, 2004, pp. 412–423.
[42] Jian Shen, Dengzhi Liu, Debiao He, Xinyi Huang, Yang Xiang, Algebraic signatures-based data integrity auditing for efficient data dynamics in cloud computing, IEEE Trans. Sustain. Comput. (ISSN: 2377-3782) 5 (02) (2020) 161–173.
[43] OpenSSL Project, 2022, https://www.openssl.org/.
[44] W. Xia, Y. Zhou, H. Jiang, D. Feng, Y. Hua, Y. Hu, Q. Liu, Y. Zhang, FastCDC: A fast and efficient content-defined chunking approach for data deduplication, in: Proceedings of USENIX ATC'16, 2016, pp. 101–114.
[45] A. Frederik, B. Jens-Matthias, K. Ghassan O., Y. Franck, Transparent data deduplication in the cloud, in: Proceedings of ACM CCS'15, 2015, pp. 886–900.
[46] Jian Liu, N. Asokan, Benny Pinkas, Secure deduplication of encrypted data without additional independent servers, in: Proceedings of ACM CCS, 2015, pp. 874–885.
[47] Jin Li, Xiaofeng Chen, Mingqiang Li, Jingwei Li, Patrick PC Lee, Wenjing Lou, Secure deduplication with efficient and reliable convergent key management, IEEE TPDS 25 (6) (2014) 1615–1625.
[48] Jianbing Ni, Kuan Zhang, Yong Yu, Xiaodong Lin, Xuemin Sherman Shen, Providing task allocation and secure deduplication for mobile crowdsensing via fog computing, IEEE Transactions on Dependable and Secure Computing (TDSC) (2018).
[49] Wee Keong Ng, Yonggang Wen, Huafei Zhu, Private data deduplication protocols in cloud storage, in: Proceedings of ACM SAC, 2012, pp. 441–446.
[50] Roberto Di Pietro, Alessandro Sorniotti, Boosting efficiency and security in proof of ownership for deduplication, in: Proceedings of the 7th ACM AsiaCCS, 2012, pp. 81–82.

Yukun Zhou received the B.E. and Ph.D. degrees in computer science and technology from Huazhong University of Science and Technology in 2013, and 2019, respectively. He is an expert at Sangfor Inc. His research interests include storage security and edge computing. His research works have been published in Usenix ATC, IEEE TC, TPDS, INFOCOM, MSST, FGCS, etc.

Zhibin Yu received his Ph.D. degree in computer science from Huazhong University of Science and Technology (HUST) in 2008. He visited the Laboratory of Computer Architecture (LCA) of ECE of the University of Texas at Austin for one year and he worked in Ghent University as a postdoctoral researcher for half of a year. Now he is a professor in SIAT. His research interests are micro-architecture simulation, computer architecture, workload characterization and generation, performance evaluation, multi-core architecture, GPGPU architecture, virtualization technologies, big data processing and so forth. He won the outstanding technical talent program of Chinese Academy of Science (CAS) in 2014 and the 'peacock talent' program of Shenzhen City in 2013. He is a member of IEEE and ACM. He serves for ISCA 2013, 2015, 2020, 2021, 2022, MICRO 2014, HPCA 2015, 2018, 2020, PACT 2016, and ICS2018.

Liang Gu received the Ph.D. degree in Computer Software and Theory from Peking University in 2010. He worked as an associate research fellow at Yale University from 2010 to 2015. He is currently the chief scientist and the director of Sangfor Research Institute at Sangfor Technology Inc. As the person in charge of R&D technology at Sangfor, he is responsible for the technical framework improvement of a series of core products, including NGAF, AC, a Cloud HCI, aSAN, etc.

Dan Feng received the B.E., M.E., and Ph.D. degrees in computer science and technology from Huazhong University of Science and Technology (HUST), Wuhan, China, in 1991, 1994, and 1997, respectively. She is a Professor and the Dean of the School of Computer Science and Technology, HUST. Her research interests include computer architecture, non-volatile memory technology, distributed and parallel file system, and massive storage system. She published more than 100 papers in IEEE TC, TPDS, TCAD, ACM TOS, FAST, USENIX ATC, EuroSys, ICDCS, SC, ICS, IPDPS, and DAC. She is a member of IEEE and ACM, chair of Information Storage Technology Committee of Chinese Computer Academy.

BenchCouncil Transactions on Benchmarks, Standards and Evaluations 2 (2022) 100061

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/

Asynchronous memory access unit for general purpose processors

Luming Wang, Xu Zhang, Tianyue Lu, Mingyu Chen *

Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100190, China
School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing, 100049, China

ARTICLE INFO

Keywords:
Asynchronous memory access
Far memory
Micro-architecture

ABSTRACT

In future data centers, applications will make heavy use of far memory (including disaggregated memory pools and NVM). The access latency of far memory is more widely distributed than that of local memory accesses. This makes the efficiency of traditional out-of-order load/store mechanism in most general-purpose processors decrease in this scenario. Therefore, this work proposes an in-core asynchronous memory access unit to fully utilize the far memory resources.

1. Introduction

In recent years, to improve the utilization of resources in cloud data centers, more and more resources are organized into resources pools. Memory resources trends to be the next resources organized as a pool. However, nowadays remote memory pools usually use software interfaces (such as key-value, RDMA, files, etc.) rather than direct load/store [1]. Recently, new interconnect technologies and protocols (such as OpenCAPI [2], Gen-Z [3] and CXL [4]) enable the construction of load/store interface based disaggregated memory pool that contains multiple nodes. Prototypes of such systems have already been constructed by researchers [5]. It is foreseeable that complex disaggregated memory pools using direct load/store interface will emerge soon.

On the other hand, Non-volatile Main Memory (NVMM) start to emerge, offering higher memory density and lower standby power consumption. However, it faces similar challenges as remote memory systems. Compared to traditional DRAM, NVMM has higher latency and a wide range of latency variation (6x-30x higher write latency and 5x-10x higher read latency) [6]. Currently, there is no efficient and standard interface for accessing NVMM. Commercial products, such as Intel's Optane DC [7], still use a modified synchronous DRAM interface.

Both remote memory and non-volatile main memory are sometimes called "far memory" as their access latency is larger than local DRAM [8]. There are two main differences in accessing far memory compared to accessing traditional DRAM-based local memory.

Widely distributed latency The memory allocated from a disaggregated memory pool can locate on some faraway remote nodes. Furthermore, the memory device can be DRAM, NVM, or other emerging memory devices. As a result, memory access latency becomes uncertain. Latency may distribute over a wide range.

Potential large aggregated bandwidth As memory resources may come from multiple different machines, the maximum aggregated access bandwidth may increase significantly compared to local memory which is limited by physical channels, making it a challenge to make use of the abundant bandwidth.

Access latency in traditional memory systems is also uncertain due to the multi-level cache hierarchy. However, the distribution of latency is relatively narrow. The latency of a single access memory request is around 1 ns (when L1 hits) to 100 ns (when accesses local DRAM). Modern processors can tolerate this difference in latency by out-of-order execution and non-blocking cache. Fig. 1 shows the limitations of current Out-of-Order processors in far memory scenarios. The range of latency they can tolerate is limited by the number of entries in the instruction queue, ROB, MSHRs, etc. Once one of these resources is exhausted, the OoO processor cannot issue more memory access requests any longer. The resource insufficiency even occurs in the local memory scenario. For example, due to the limited number of MSHRs, a single core of Intel Skylake processor can only reach a memory bandwidth of about 15 GB/S, which is even lower than that of a single DDR4-2400 DIMM. It is difficult for modern processors to tolerate the access latency fluctuations (300 ns-10 μs) of far memory.

Although improving the out-of-order execution capability of traditional general-purpose processor cores [9–12] (e.g., increasing the number of entries of MSHRs and ROB, using multi-level MSHRs and ROB, etc.) can also improve the performance of load/store in this scenario. But such an improvement, even if it is feasible, requires significant hardware resources. The key issue is that traditional load/store instructions are synchronous, every outstanding memory access needs to hold at least one hardware resource until the operation is completed. The more parallelism the more hardware resources will be needed.

One approach to address this problem is asynchronous memory access. A similar predicament had already existed in network programming, where applications call blocking socket interfaces made program

* Corresponding author at: Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100190, China.
E-mail addresses: wangluming@ict.ac.cn (L. Wang), zhangxu19s@ict.ac.cn (X. Zhang), lutianyue@ict.ac.cn (T. Lu), cmy@ict.ac.cn (M. Chen).

https://doi.org/10.1016/j.tbench.2022.100061
Received 21 March 2022; Received in revised form 7 May 2022; Accepted 7 May 2022
Available online 24 May 2022

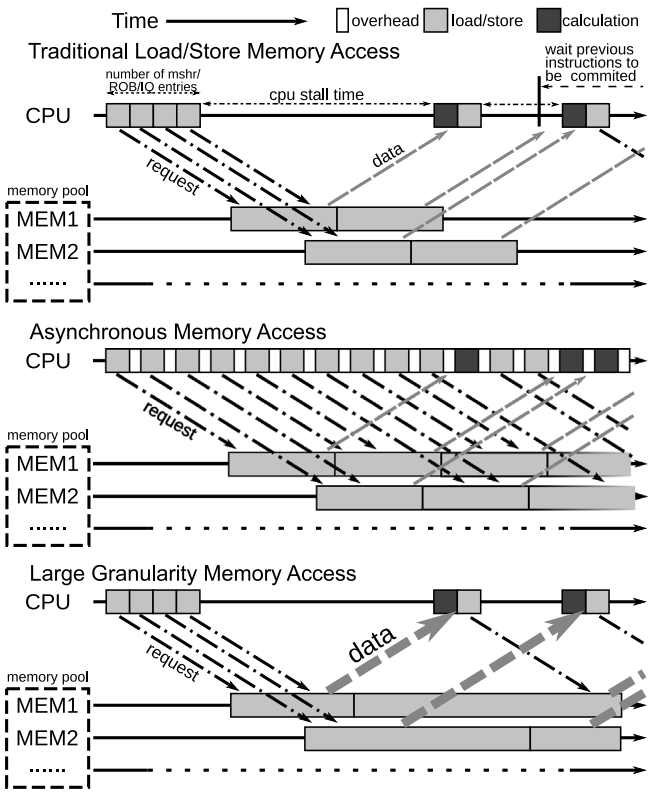


Fig. 1. Ways to improve memory bandwidth utilization.

performance suffer from network latency. To solve this problem, asynchronous non-blocking interfaces such as select() and epoll() had been built. Just as network programming has evolved from the early synchronous blocking model to today's asynchronous non-blocking model, we suggest that an asynchronous non-blocking model for memory access is also needed. Therefore, there should be some mechanisms for invoking asynchronous memory access efficiently in a general purpose processor.

Another approach is supporting memory access with variable granularity. As shown in Fig. 1, to improve performance, applications can initiate memory requests with a large granularity to hide the latency and fully utilize the bandwidth. Furthermore, applications can adjust the granularity based on data semantics to access memory flexibly and efficiently.

Concerning the above approaches, we propose an in-core Asynchronous Memory access Unit (AMU) to support asynchronous access in a general purpose processor. AMU enables applications to asynchronously initiate many variable granularity memory access requests by simple instructions, such as asynchronous load/store. AMU also enables applications to start various complex memory access requests with additional configuration registers. Processors can still use traditional synchronous load/store instructions for compatibility while the data from both sources can be consumed by computation instructions transparently. We argue that this is a more practical and efficient way than designing an un-core or off-chip asynchronous memory accelerator.

The following chapters outline the main features of the asynchronous memory access unit.

2. Asynchronous memory access unit

Traditionally, mostly used load/store instructions are implemented as synchronous mode. Each pending memory operation will hold certain hardware resources such as GPR, ROB and MSHR entry. The hardware resources will not be released until the memory operations have finished.

We propose a new class of asynchronous memory access instructions that will not hold hardware resources during the operation. An asynchronous instruction that invokes a memory access request will be committed immediately once it is accepted by functional unit and sent out. Thus, applications can continue to execute other operations rather than waiting for the memory access operation to finish. Then, applications can poll whether there is a completed request later. Moreover, even if an outstanding request did not finish for a long time, there is no pipeline stall due to the shortage of ROB or other hardware resources.

Asynchronous memory access instructions are decoded and issued as normal instructions. The functional unit to process asynchronous instructions is called Asynchronous Memory access Unit (AMU). AMU is inspired by the Vector Processing Unit (VPU) of many modern processors. The VPU is a separate functional unit in the CPU. Applications use the VPU through a standalone instruction set (i.e., vector instruction set), which contains a set of extra registers (i.e., vector registers) to hold the wide data to be processed. Besides, vector instructions are scheduled together with scalar instructions. Vector registers and scalar registers can exchange data efficiently. Just as VPU, AMU can coexist with synchronous load/store Unit and can be ignored when compatibility comes first.

AMU is responsible for processing asynchronous instructions. However, it cannot depend on internal hardware registers or queues to keep the status of outstanding memory operations, which will become another potential bottleneck for parallelism. In fact, the status of pending requests are stored in SPM. Each processor core is equipped with a ScratchPad Memory (SPM), which acts as vector registers in VPU but has a larger capacity and flexible data structure. Data is moved asynchronously and automatically between SPM and main memory by AMU. To initiate asynchronous memory access requests, applications can prepare data in SPM and then execute asynchronous memory access instructions. After receiving the request, AMU will move the data between memory and SPM in background.

From the view of an application, the SPM is a stand-alone memory space. Applications can use synchronous load/store instructions to access the data in the SPM and process them with other regular instructions. In addition, applications can copy data from main memory to the SPM and vice versa. The SPM is fully compatible with the processor's original data access and processing mechanisms.

By asynchronous access, AMU can support as many requests as the capacity of SPM can support in theory. However, AMU does not assure the consistency among all outgoing memory operations. The overhead of hardware consistency checking is one of the reasons that limit the capacity of traditional load/store queue and MSHRs. In the AMU design, we leave the consistency issue to software. We argue that software and hardware cooperation is the right way to exploit the memory parallelism over large latency.

2.1. Instructions

There are three core instructions of AMU. These instructions enable the most basic asynchronous memory access.

Asynchronous load/store instructions In AMU, *aload/astore* instruction invokes a data movement request between SPM and memory. An SPM address and a memory address are passed to AMU by registers. AMU will move data between the provided SPM address and off-core memory address. Then a request id, which is used for tracking the request, is stored in the destination register.

Instruction for getting an id of finished request We propose *getfin* instruction for getting an id of any completed request. If there is no finished request, the instruction returns a failure code. This instruction does not block execution regardless of whether there is a completed request or not.

2.2. Registers

Due to the limited field space of instructions, some complex memory access settings cannot be encoded in a single instruction. To solve this problem, we designed several configuration registers, which contain advanced parameters.

Memory Access Configuration Registers These registers contain advanced memory access configurations, including address format, granularity, priority, etc. Settings in the configuration will be combined with each access instruction to form a rich semantic memory request. Application can keep several different configuration registers for different data regions.

Default Configuration Register Due to the limited encoding space of some instructions, it is even not possible to specify all configuration registers. For such case, the system automatically chooses the configuration register specified in this register.

Access Pattern Registers The access pattern registers are used to initiate complex asynchronous memory access. They contain the access pattern(such as stride, neighbor, stream, etc.) of a class of complex memory access requests.

2.3. Programming model

Listing 1 shows a basic example of asynchronous memory accessing. The code initiates an asynchronous memory access request with the *aload* instruction. The code then keeps retrying the *getfin* instruction to get the id of a completed request and can do other work while the request is still pending. After the request completes, the code then reads the data from the SPM with the *load* instruction.

Listing 1: Asynchronous Memory Access Basic Example

```
int memory_need_to_be_accessed;
int *spm_space = (int*) A_SPM_ADDR;
// Invoke an asynchronous memory access
// requests. The request's id is ignored.
aload(spm_space,
  &memory_need_to_be_accessed);
while ((rd = getfin()) != 0) {
  // Do something else
}
// Access data from SPM via load/store
printf("%d\n", *spm_space);
```

AMU instructions can support a variety of programming paradigms.

- **Vector Model** Vector instructions and vector processors are mature techniques for exploiting data-level parallelism. As a technique to improve data-level parallelism, AMU instructions have many similarities to vector instructions. Thus, it is possible to combine AMU instructions with vector instructions efficiently.
- **Event-Driven Model** The event-driven model is a common paradigm in single-thread non-blocking network programming. Furthermore, the *aload/astore* instructions are like non-blocking socket *read()/write()*. *getfin* instructions are like the *select()* in network programming. Thus, the event-driven model can be naturally applied to asynchronous memory accesses especially for out of order scenarios.
- **Coroutine Model** For asynchronous access requests with complex access patterns, coroutines or lightweight software threads are more suitable programming paradigms. Coroutines can easily work with high performance concurrent data structures and enable more interactions between software and AMU.

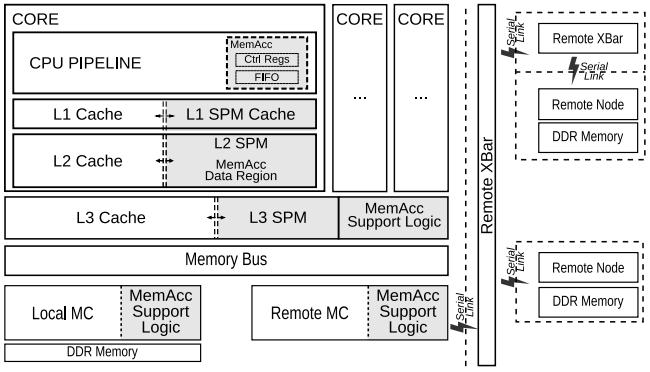


Fig. 2. Architecture design.

3. Architecture design

Fig. 2 shows the architecture design of AMU. There are three key design choices:

CPU Pipeline Integration To support efficient asynchronous memory access instructions, many state control registers are integrated into CPU core pipeline. Some of these registers indicate the AMU's status, which allows programs to rapidly get the status of outstanding instructions. In addition, speculative execution of asynchronous memory access instructions brings new challenges since some states are stored in SPM. This requires the pipeline of the processor core to be designed carefully.

Re-configurable Cache/SPM Space We propose to dynamically configure part of the CPU Cache as SPM. So there are no proprietary SPM resources and interface needed. This design also allows more flexibility for the software to decide how to use the SPM. Applications can adjust the size of Cache and SPM themselves based on the workload. For example, Random-Access benchmark needs only about 12 KB to support up to 512 in-flight 8B memory requests.

Integration with L2 Controller We propose to integrate the AMU logic with the L2 cache controller. Since the size of L2 is large enough compared to register file and reserve part of L2 will not affect much performance as L1. The logic implements the management and execution of asynchronous access requests and the engine to move data between SPM and memory controller.

Because the main metadata of memory requests maintained by AMU are stored in the SPM, the extra storage overhead is only about a few KB and does not vary when the required MLP increases.

AMU can work with a standard memory controller for local memory or far memory access. However to better support various asynchronous memory instructions with rich semantics and various far memory resources, there should also be newly designed memory controllers that can do the transformation between local bus requests and network packets, such as packing, unpacking, filtering, compressing, routing, etc.

We do not specify the server-side of memory resources. For asynchronous memory access supported by AMU, there are no latency or granularity limitations for memory servers. Different kinds of memory resources can be accessed through a unified interface. That is the separation of memory organization and memory access.

4. Early evaluation

To demonstrate the concepts of AMU, an early simulator prototype has been built. We modified GEM5 to implement a cycle-accurate model of AMU and evaluated it by running Random-Access benchmark

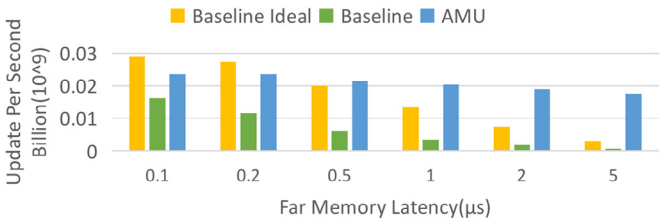


Fig. 3. Performance of random access benchmark.

from HPCC (shown in Fig. 3). The far memory latency is set to several different values (from 0.1 μs to 5 μs) respectively. 64 KB of the 256 KB Cache Capacity are reserved for SPM. AMU performs similarly across different configurations, while the performance of baseline drops rapidly when access latency increases. The results show that AMU can better tolerate a large range of access latency.

5. Discussion

5.1. Efficient ID management

When initiating an asynchronous access request, an id needs to be assigned to the request and be released after finishing. Since the id is needed every time an asynchronous access is initiated, the overhead of id management must be as low as possible. For this goal, we chose to design an efficient hardware id management mechanism instead of leaving id management to software. Also, this mechanism should be integrated with out-of-order and speculation mechanisms smoothly.

5.2. Consistency

As mentioned in Section 2, this work relies on software to handle the consistency. For many data-parallel programs (such Key-Value databases and graph processing [13], etc.), they easily apply coroutine model mentioned above. As each interleaved coroutines processes independent data, thus naturally avoiding data consistency problems.

For other programs that need strong consistency, it is possible to use a combination of hardware and software solution to handle the consistency problem of asynchronous access to far memory. For example, consistency checking can be implemented in local memory or local cache. Applications can check the consistency of requests locally before invoking asynchronous accesses to far memory. In addition, some explicit and efficient locking mechanisms might also be provided by hardware to ensure consistency. Furthermore, software can deal with locking asynchronously to avoid blocking. Although these approaches introduce extra complexity, we argue that the overhead is acceptable comparing with the benefit.

5.3. Comparison with prefetching

There are three major differences between asynchronous memory access and prefetching. First, prefetching mechanisms do not provide any method to query whether prefetch requests have been completed. Thus, applications cannot know if the data has been transferred to local cache. This may impact the efficiency of the application as the access latency is distributed in a wide range. Second, modern processors' prefetching mechanisms are also limited by the number of MSHR entries, while the proposed AMU fully bypass the MSHRs. Third, the memory space for prefetching results is not reserved so prefetched data might be lost before access.

6. Future work

In this paper, only the basic instructions and structures of AMU are presented. Further design and evaluation are undertaking. The AMU design can be easily extended support more memory access protocols. For example, we can add configuration registers and instructions for issuing processing-in-memory related requests. The AMU will enable more programming flexibility if the underlying memory system support more richer semantics, such as message interface based memory systems [14].

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

This work is partially supported by Strategic Priority Research Program of Chinese Academy of Sciences (Grant No. XDC05030400), and Huawei Technologies Company, Ltd.

References

[1] M.K. Aguilera, N. Amit, I. Calciu, X. Deguillard, J. Gandhi, P. Subrahmanyam, L. Suresh, K. Tati, R. Venkatasubramanian, M. Wei, Remote memory in the age of fast networks, in: Proceedings of the 2017 Symposium on Cloud Computing, in: SoCC '17, Association for Computing Machinery, New York, NY, USA, 2017, pp. 121–127, http://dx.doi.org/10.1145/3127479.3131612.

[2] openCAPI specification, 2017, Online URL http://opencapi.org. (Accessed Febrary 2022).

[3] Gen-Z specification, 2018, Online URL https://genzconsortium.org/specifications. (Accessed Febrary 2022).

[4] Compute express link, 2022, Online URL https://www.computeexpresslink.org/. (Accessed Febrary 2022).

[5] C. Pinto, D. Syrivelis, M. Gazzetti, P. Koutsovasilis, A. Reale, K. Katrinis, H.P. Hofstee, ThymesisFlow: A software-defined, HW/SW co-designed interconnect stack for rack-scale memory disaggregation, in: 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO, 2020, pp. 868–880, http://dx.doi.org/10.1109/MICRO50266.2020.00075.

[6] H.-K. Liu, D. Chen, H. Jin, X.-F. Liao, B. He, K. Hu, Y. Zhang, A survey of non-volatile main memory technologies: State-of-the-arts, practices, and future directions, J. Comput. Sci. Tech. 36 (1) (2021) 4–32, http://dx.doi.org/10.1007/s11390-020-0780-z.

[7] Intel optane persistent memory, 2022, Online URL https://www.intel.com/content/www/us/en/architecture-and-technology/optane-dc-persistent-memory.html. (Accessed Febrary 2022).

[8] A. Lagar-Cavilla, J. Ahn, S. Souhlal, N. Agarwal, R. Burny, S. Butt, J. Chang, A. Chaugule, N. Deng, J. Shahid, G. Thelen, K.A. Yurtsever, Y. Zhao, P. Ranganathan, Software-defined far memory in warehouse-scale computers, in: Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, in: ASPLOS '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 317–330, http://dx.doi.org/10.1145/3297858.3304053.

[9] J. Tuck, L. Ceze, J. Torrellas, Scalable cache miss handling for high memory-level parallelism, in: 2006 39th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO'06, IEEE, 2006, pp. 409–422.

[10] M. Asiatici, P. Ienne, Stop crying over your cache miss rate: Handling efficiently thousands of outstanding misses in fpgas, in: Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 2019, pp. 310–319.

[11] S.T. Srinivasan, R. Rajwar, H. Akkary, A. Gandhi, M. Upton, Continual flow pipelines: achieving resource-efficient latency tolerance, IEEE Micro 24 (6) (2004) 62–73.

[12] A. Hilton, S. Nagarakatte, A. Roth, iCFP: Tolerating all-level cache misses in in-order processors, in: 2009 IEEE 15th International Symposium on High Performance Computer Architecture, IEEE, 2009, pp. 431–442.

[13] T.J. Ham, L. Wu, N. Sundaram, N. Satish, M. Martonosi, Graphicionado: A high-performance and energy-efficient accelerator for graph analytics, in: 2016 49th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO, IEEE, 2016, pp. 1–13.

[14] L.-C. Chen, M.-Y. Chen, Y. Ruan, Y.-B. Huang, Z.-H. Cui, T.-Y. Lu, Y.-G. Bao, MIMS: Towards a message interface based memory system, J. Comput. Sci. Tech. 29 (2) (2014) 255–272, http://dx.doi.org/10.1007/s11390-014-1428-7.

BenchCouncil Transactions on Benchmarks, Standards and Evaluations 2 (2022) 100060

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/

Performance and energy consumption tradeoff in server consolidation

Belen Bermejo *, Carlos Juiz

Computer Science Department, Universitat de les Illes Balears, Spain

ARTICLE INFO

Keywords:
Metrics
Consolidation
Performance
Energy
Servers
Benchmarking

ABSTRACT

Server consolidation is one of the techniques used to increase energy efficiency in datacentres. Nevertheless, the server consolidation has an inherent trade-off between performance degradation and energy consumption which has to be quantified to be managed. In this paper, the CiS^2 index is proposed to quantify the mentioned trade-off. We validated de use of the CiS^2 index through real experimentation. Also, these observations lead us to propose the second contribution, which focuses on the consolidation overhead. We proposed a general method to quantify this overhead and be able to manage its effect on performance degradation. To sum up, this paper improved the management of energy efficiency in datacentres' servers through the CiS^2 index and the server consolidation determination method.

1. Introduction

In the last years, organizations started to be concerned about the impact of information technology (IT) on energy consumption. For this reason, the Green IT initiatives appeared to make companies more environment-friendly [1,2].

The datacentres consume a huge amount of power and emit greenhouse gases in the form of CO_2. In a current datacentre, 30% of servers either are even not used or their utilization is very low, around 5%–10% [3,4]. Also, servers are the most power-demand device of a datacentre [5].

During last years, Green IT was used as an umbrella covering overlapping concepts like server consolidation and power management, among many others. Then, the aspiration of Green IT is achieving higher energy efficiency in the use of the IT devices and to increase the utilization of already installed devices in datacentres using the virtualization technology, specifically the server consolidation technique [1].

The server consolidation technique is based on the reallocation of virtual servers (could be virtual machines) among different physical servers using machine migration (see Fig. 1). As a consequence, the utilization of physical servers increases and the number of switched-on physical servers can be reduced.

Therefore, server consolidation increases the utilization of physical servers. However, due to the possibility of switch-off some physical servers, the power consumption is reduced. Nevertheless, as [6] states, the energy consumption depends on the overhead inherent to virtualization. The virtualization overhead is the extra workload that the physical server has to perform due to being virtualized, that is, tasks of managing virtual machines and coordinating the access to physical resources. As a consequence, the larger the number of consolidated virtual machines is, the higher the overhead is because of the coordination of simultaneously demanding resources access [5].

In certain cases, the energy-saving is not compensated with the performance degradation, which will be very high. It could also be the opposite case, that is, high performance of the datacentre may not be able to compensate servers to reduce it [5]. The current challenge in server consolidation is how to determine if a consolidated server is efficient or not in terms of energy consumption and performance degradation.

Therefore, the research question we attempted to solve in this work is: **could the performance-energy trade-off of physical servers when consolidating virtual machines be quantified?**

2. State of the art

In this work, we are interested in the server consolidation point of view tracking the management of these issues proposing several metrics to quantify the performance and the efficiency of a datacentre and servers.

The main developed work [7] explores the diverse metrics that are currently available to measure numerous datacentre infrastructure components behaviour. Also, they proposed a taxonomy of metrics based on datacentre dimensions. In addition, authors argue for the design of new metrics considering factors such as locations and resource co-locations, to assist in the strategic datacentre design and operations processes. One of the challenges authors announced is that it is hard to know the energy consumption due to datacentre sub-components, as operating systems and virtual machines. Due to that, in this work, we

* Corresponding author.
E-mail address: belen.bermejo@uib.es (B. Bermejo).

https://doi.org/10.1016/j.tbench.2022.100060
Received 26 April 2022; Received in revised form 10 May 2022; Accepted 10 May 2022
Available online 25 May 2022

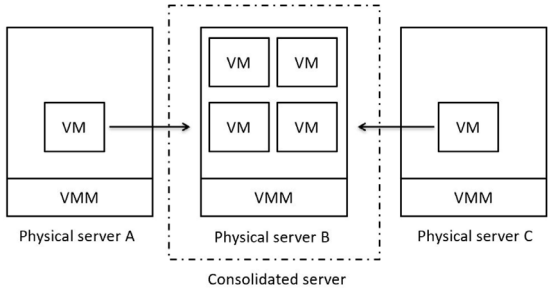


Fig. 1. Virtual machine consolidation, from [9].

proposed a metric for performance-energy trade-off, which took into account the number of allocated virtual servers.

On the one hand, the performance metrics attempt to quantify the suitability of the amount of work accomplished by a server or a datacentre. The values can be directly monitored from the system or inferred [8]. In the same manner, the energy and power metrics quantify the consumption of power or energy of a datacentre and/or a physical server. To take into account both previous aspects simultaneously, it is necessary to measure the relationship between performance and energy. These metrics relate to the performance of servers or datacentres with the power or energy consumption.

Power management metrics and techniques at different levels in datacentres are shown in [7]. System's administrators may measure information from software and hardware optimization. In this work, we are focused on metrics regarding software-oriented optimizations, specifically virtual machine consolidation, and, hardware-oriented optimizations, focused on the power and energy reduction in physical servers.

As we can observe from the previous works, all the used metrics are focused on the performance degradation and energy consumption, but, they are not considering the fact of having virtual servers consolidated. Then, with the current metrics it is not possible to know the efficiency of a consolidated server, or which number of virtual servers is more efficient in a specific scenario.

As a result, to the best of our knowledge, this is the first attempt to define metrics to quantify the performance and energy trade-off in server consolidation.

3. The CiS^2 index

In this section, we presented a new metric: Consolidated index for CPU- Server Saturation (CiS^2) which attempts to quantify the performance and energy trade-off taking into account the number of consolidated virtual machines (or containers) per server [10].

The CiS^2 index is defined as the product of the speed-up of the performance and the ratio of the consumed energy (see Eq. (1)). The speed-up of the performance is calculated as the ratio between the mean response time of the consolidation scenario and the physical server execution ($SP_p = R^c/R^p$). In the same manner, the ratio of the consumed energy is the division between the energy consumed in the consolidated scenario and the physical one ($SP_e = E^c/E^p$) [10].

Therefore, it is a simple quadratic efficiency and also the name of CiS^2. Also, in Fig. 2 the desirable index in function of the possible number of consolidated virtual machines (or containers) is represented. In the vertical axis, the combined performance and energy efficiency values are represented. In the horizontal axis the number of machines to be consolidated, either they are available in the datacentre or they are only considered for future capacity planning and forecasting bottlenecks.

$$CiS^2 = SP_p \cdot SP_e \tag{1}$$

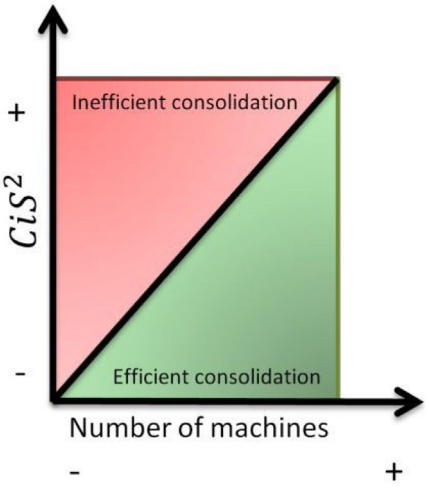


Fig. 2. CiS^2 index values and reference diagonal.

3.1. Graphical representation and interpretation

The representation of the CiS^2 index values, concerning the incremental number of consolidated machines, would be a square where the reference diagonal separated the scenario configurations of high performance and low energy from those that degrade and/or consume excessive energy about the following rules of thumb:

- N virtual machines consolidated in one physical machine should be N times slower than N physical machines (linear performance degradation).
- N virtual machines in one physical machine should consume as energy as N physical machines (energy conservation).

One the hand, the more consolidated virtual servers hosted in physical servers, the more performance degradation and consequently, the energy consumption would increase due to the increment of the mean response time. On the other hand, the more physical machines used, the more power is consumed, and consequently, the energy is also increased. Thus, we argue that it is possible to measure the balance between both situations. Due to that, the CiS^2 index compares different configurations in the performance-energy trade-off between different server consolidation scenarios.

From the energy efficiency point of view, the balanced efficiency metric shown through CiS^2 should be the one in which the average energy of a number of consolidated physical machines in a number of a virtual servers is exactly the same of using corresponding physical machines, i.e. the energy ratio is equal to one ($SP_e = 1$). However, from the performance speed-up point of view, the balanced efficiency shown through CiS^2 should be the one in which performance degradation is linear, that is, the slowdown is the same of the number of consolidated virtual servers per physical machine, i.e. N is the number of machines ($SP_p = N$).

3.2. Desirable values

Being CiS^2 values the result of the product of performance and energy speed-up, CiS^2 index acts as a qualifier of the consolidation of several virtual machines in comparison with this balanced (and pessimistic) reference diagonal described by the application of the rules of thumb.

Therefore, we also defined the CiS^2 reference diagonal as the imaginary border separating the" inefficient" CiS2 values (above the line) from the "efficient" CiS^2 values (below the lines) as we depicted in Fig. 3. This reference diagonal represents the linearity of the

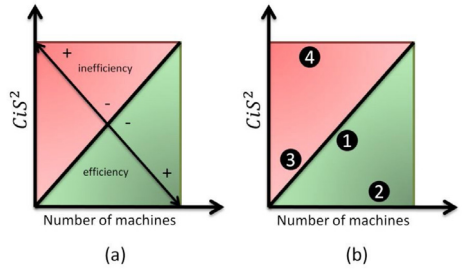


Fig. 3. Desirable values o CiS^2 index.

Table 1
Physical servers of the experimentation.

Server	Number of CPUs	RAM size (GB)
Fujitsu RX600S5–1	48	1024
Dell PowerEdge T330	16	16
Dell PowerEdge T430	16	8
Dell PowerEdge R310	4	4
Dell PowerEdge T3400	2	8

consolidation in terms of performance and energy so that increasing consolidation would mean lowering proportionally the performance and also it means exchanging power by energy [11].

Having two different areas to distinguish among different consolidations one server or to compare different servers' consolidation spending on the area position of values, another interesting feature of the CiS^2 index for system administrators. Any consolidation configuration is more efficient or more inefficient, depending on the Euclidean distance of the index to the reference diagonal, above or below the reference diagonal, respectively as it is shown in Fig. 3a.

For example, the point 2 represented in 3b, which is on the green area, is more efficient than the point 1 because it is far from the diagonal. On the contrary, the point 3 represented in 3b, which is on the red area, is more efficient than the point 4 because it is closer to the diagonal than the point 4.

3.3. CiS^2 index evaluation

In the real experimentation, several factors should be considered, such as the hypervisor type, the benchmark or workload kind and the server hardware features. To simplify these factors, we represented the system using a black-box model. The workload is submitted in the system (consolidated servers) and we monitor the system behaviour (mean response time and power consumption) until the workload is completed [12,13].

The experimental set-up is composed of a set of different physical servers, which the number of CPUs and the RAM size are described in Table 1. Besides, the power consumption was measured by the Chroma 66200 power meter, the used hypervisors to deploy the consolidation are KVM, Virtual Box and Docker. In addition, it is important to note that the physical CPU executes the workload under the saturation condition, that is, the % of utilization is around 100%. The selected workloads are the Sysbench-CPU and the Stress-ng, which are intensive-based CPU workloads [14].

3.4. CiS^2 evaluation's results

In Fig. 4 the CiS^2 values for each physical server in the function of the number of consolidated virtual machines can be shown. The first that can observe is that the CiS^2 values have the same shape, that is, it starts increment and then it goes down when the physical machine has a certain number of allocated virtual machines. Therefore, there is an inflexion point that determines the number of minimum consolidated

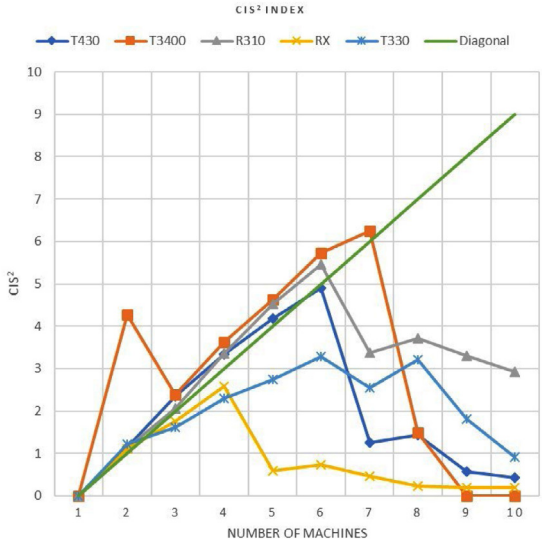


Fig. 4. CiS^2 index values.

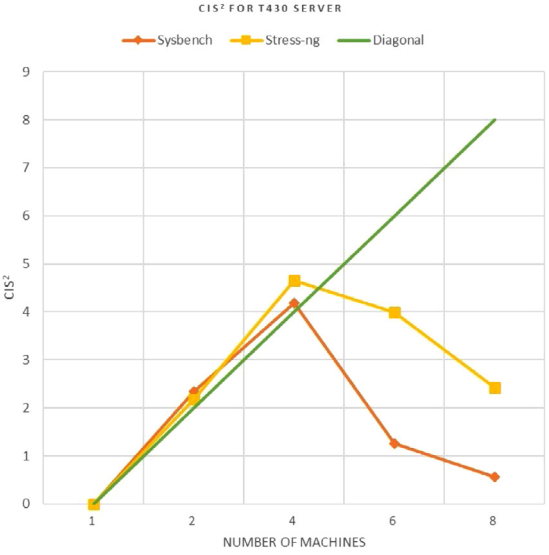


Fig. 5. CiS^2 index values for different workload.

virtual machines the server needs to have a good CiS^2 value. The inflexion point depends on the physical server, and, as a consequence, it depends on the physical resources the server has.

Besides, the graphical representation of the CiS^2 index allows us to distinguish between the efficient and non-efficient consolidation configurations. For example, taking the T430 server from Fig. 4, it can be observed that $N = 6$ is more efficient than $N = 3$ because it is under the diagonal. In addition, for the HPI server, $N = 4$ is more efficient than $N = 3$ because it is far away from the diagonal.

Moreover, in Fig. 5 the CiS^2 index for the Sysbench workload in comparison with the Stress-ng workload for the T430 server is shown. It can be observed that for different nature of CPU workload, the behaviour of the CiS^2 index is the same. It starts growing, and it goes down after the inflexion point. Also, the inflexion point is the same for both workloads.

In previous sections, we stated that the CiS^2 index can be used for server's benchmarking and comparison. In Fig. 6 the physical servers by their consolidation efficiency considering the CiS^2 value is shown. It can be observed that the RX server is the most efficient because its CiS^2 value at the inflexion point is the lowest one.

Server	Scalability vector size (N)	Number of machines at the inflexion point	Inflexion point (h, CiS^2)	Mean CiS^2 value	Selection order by CiS^2
RX	$n \geq 9$	h = 3	(3, 2.587)	0.875	First
T330	$n \geq 9$	h = 5	(5, 3.281)	2.182	Second
T430	$n = 9$	h = 5	(5, 4.902)	2.181	Third
R310	$n = 9$	h = 6	(6, 6.254)	3.317	Fourth
T3400	$n = 6$	h = 6	(6, 6.254)	4.055	Fifth

Fig. 6. Server selection depending on the values of parameters obtained from benchmarking (sorted by server efficiency).

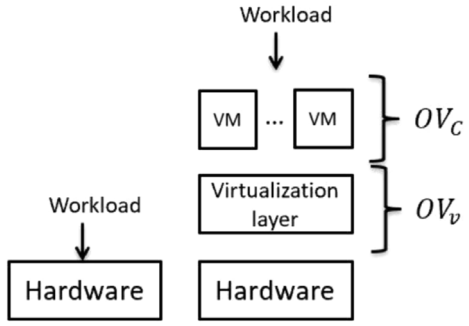


Fig. 7. Server consolidation overhead sources.

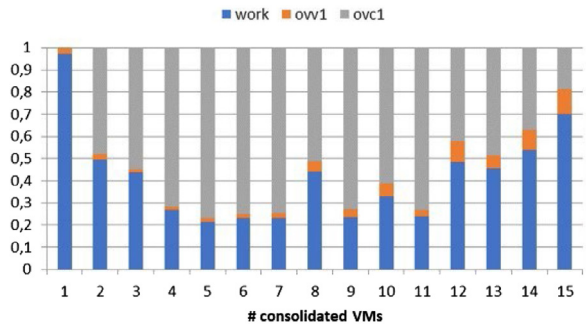


Fig. 8. Consolidation overhead representation for T430 server and KVM hypervisor.

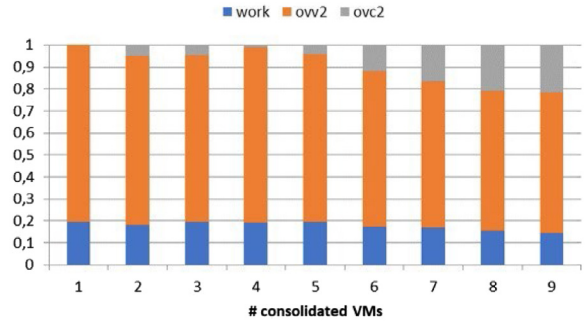


Fig. 9. Consolidation overhead representation for T430 server and Virtual-Box hypervisor.

4. Consolidation overhead quantification method

As we observed in previous results, the value of the CiS^2 index depends on the hardware features, the number of allocated virtual machines and the workload nature, together with the performance degradation inherent to the server consolidation. Therefore, the second contribution of this work regards the server consolidation overhead [9].

The server consolidation overhead is defined as the extra workload that the system has to perform to manage the consolidation. This extra workload comes from the fact of having a hypervisor and the current access to physical resources from several consolidated virtual machines (or containers). Therefore, there are two sources of overhead (see Fig. 7) [15]:

- OV_v: overhead of virtualization.
- OV_c: overhead of consolidation.

Regarding the server consolidation overhead, the aim is to provide a general method for quantifying OV_v and OV_c. Let us define R^C and the mean response of the consolidated server, R^V as the mean response time of the physical server with a single consolidated virtual machine, and R^{PM} as the mean response time of the physical server. The OV_v can be defined as the difference between R^V and R^{PM} (see Eq. (2)). In the same manner, OV_c can be defined as the difference between R^C and R^V (see Eq. (3)).

$$OV_v = R^V - R^{PM} \tag{2}$$

$$OV_c = R^C - R^V \tag{3}$$

The main advantage of the proposed method is that it can be applied to any consolidation scenario considering any physical server, hypervisor and workload type.

The evaluation of the proposed method was performed using the previous experimental set-up, monitoring the mean response time of the required scenarios. We represented for each consolidation configuration the value of OV_v, OV_c and the useful work in percentage. The useful work represents the portion of the time that the system is executing just the workload, in this case, the CPU operations.

In Figs. 8 and 9 the values of OV_v (blue), OV_c (orange) and %work (grey) for the T430 are represented, in function of the number of consolidated virtual machines. It can be observed, that for the KVM hypervisor the values of OV_v and OV_c are smaller than the values for the Virtual-Box hypervisor. Also, it can be seen that these values depend on the number of consolidated machines and the hypervisor. However, in any case, the consolidation is not for free, being more than 50% for Virtual-Box hypervisor configurations.

5. Conclusion and future work

This paper aims to measure the performance-energy tradeoff in server consolidation. Since there are no metrics to capture how server consolidation is managed considering the relationship between performance and energy, the CiS^2 index is proposed to achieve this aim. As the results show, this index can be applied to any type of server, under any virtualization platform and any level of use of its resources, in this case, the CPU. In addition, it enables the datacentre administrator to make better consolidation decisions thanks to the proposed graphical representation.

Also, the proposed index reflects a set of behaviours inherent to consolidated servers. The second contribution of this paper consists of the classification and quantification of the factors that affect the behaviour of server consolidation, in this case, two types of overhead (OV_v and OV_c). By the application of a simple method, these overheads can be quantified through the proposed method, which is also independent of the type of server, the executed workload, the virtualization and the percentage of CPU utilization.

Therefore, through this work, a step has been made towards a more efficient management of virtualized servers, and the datacentres. Now, the performance and energy balance of servers can be measured through the CiS^2 index and graphically analysed with a general

method. Besides, system's administrators dispose of a method to go in-depth the overhead caused by the consolidation of servers and thus be able to make better decisions regarding the improvement of these systems.

As future work, the CiS^2 index can be extended to multiple devices. Also, it can be extended for scales workload and considering different workload distributions. Moreover, system properties could be described by the CiS^2 index. Regarding the overhead quantification method, it could be extended considering the power and energy consumption.

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.

References

[1] M. Uddin, A.A. Rahman, Energy efficiency and low carbon enabler green IT framework for data centers considering green metrics, Renew. Sustain. Energy Rev. 16 (6) (2012) 4078–4094.
[2] C.-J. Tang, M.-R. Dai, H.-C. He, C.-C. Chuang, Evaluating energy efficiency of data centers with generating cost and service demand, Bull. Netw. Comput. Syst. Softw. 1 (1) (2012) pp–16.
[3] L.A. Barroso, U. Hölzle, The case for energy-proportional computing, Computer (12) (2007) 33–37.
[4] L. Minas, B. Ellison, Energy Efficiency for Information Technology: How to Reduce Power Consumption in Servers and Data Centers, Intel Press, 2009.
[5] F. Abaunza, A.-P. Hameri, T. Niemi, EEUI: A new measure to monitor and manage energy efficiency in data centers, Int. J. Prod. Perform. Manag. 67 (1) (2018) 111–127.
[6] G. Lovász, F. Niedermeier, H. De Meer, Performance tradeoffs of energy-aware virtual machine consolidation, Cluster Comput. 16 (3) (2013) 481–496.
[7] A.M. Ferreira, B. Pernici, Managing the complex data center environment: An integrated energy-aware framework, Computing 98 (7) (2016) 709–749.
[8] X. Molero, C. Juiz, M. Rodeño, Evaluación Y Modelado Del Rendimiento De Los Sistemas Informáticos, Pearson Educación London, 2004.
[9] B. Bermejo, C. Juiz, C. Guerrero, Virtualization and consolidation: A systematic review of the past 10 years of research on energy and performance, J. Supercomput. (2018) 1–29.
[10] C. Juiz, B. Bermejo, The CiS2: A new metric for performance and energy trade-off in consolidated servers, Cluster Comput. 23 (4) (2020) 2769–2788.
[11] B. Bermejo, C. Juiz, C. Guerrero, On the linearity of performance and energy at VMC: the 2 index for CPU workload in server saturation, in: IEEE High Performance Computing and Communications, HPCC-2018, 2018.
[12] B. Bermejo, C. Juiz, N. Thomas, On the virtualization overhead and energy consumption in consolidated servers, in: UK- Performance Engineering Workshop, UKPEW, 2018.
[13] S.K. Panda, P.K. Jana, An energy-efficient task scheduling algorithm for heterogeneous cloud computing systems, Cluster Comput. (2018) 1–19.
[14] E. Casalicchio, A study on performance measures for auto-scaling CPU-intensive containerized applications, Cluster Comput. (2019) 1–12.
[15] B. Bermejo, C. Juiz, On the classification and quantification of server consolidation overheads, J. Supercomput. 77 (1) (2021).

TBench Editorial Board

More information about this series at https://www.benchcouncil.org/tbench/

TBench Call For Papers

BenchCouncil Transactions on Benchmarks, Standards and Evaluations (TBench)
ISSN:2772-4859

Aims and Scopes

BenchCouncil Transactions on Benchmarks, Standards, and Evaluations (TBench) publishes position articles that open new research areas, research articles that address new problems, methodologies, tools, survey articles that build up comprehensive knowledge, and comments articles that argue the published articles. The submissions should deal with the benchmarks, standards, and evaluation research areas. Particular areas of interest include, but are not limited to:

- 1. Generalized benchmark science and engineering (see https://www.sciencedirect.com/science/article/pii/S2772485921000120), including but not limited to
  - measurement standards
  - standardized data sets with defined properties
  - representative workloads
  - representative data sets
  - best practices
- 2. Benchmark and standard specifications, implementations, and validations of:
  - Big Data
  - AI
  - HPC
  - Machine learning
  - Big scientific data
  - Datacenter
  - Cloud
  - Warehouse-scale computing
  - Mobile robotics
  - Edge and fog computing
  - IoT
  - Chain block
  - Data management and storage
  - Financial domains
  - Education domains
  - Medical domains
  - Other application domains
- 3. Data sets
  - Detailed descriptions of research or industry datasets, including the methods used to collect the data and technical analyses supporting the quality of the measurements.
  - Analyses or meta-analyses of existing data and original articles on systems, technologies, and techniques that advance data sharing and reuse to support reproducible research.
  - Evaluating the rigor and quality of the experiments used to generate the data and the completeness of the data description.
  - Tools generating large-scale data while preserving their original characteristics.
- 4. Workload characterization, quantitative measurement, design, and evaluation studies of:
  - Computer and communication networks, protocols, and algorithms
  - Wireless, mobile, ad-hoc and sensor networks, IoT applications
  - Computer architectures, hardware accelerators, multi-core processors, memory systems, and storage networks
  - High-Performance Computing
  - Operating systems, file systems, and databases

- Virtualization, data centers, distributed and cloud computing, fog, and edge computing
- Mobile and personal computing systems
- Energy-efficient computing systems
- Real-time and fault-tolerant systems
- Security and privacy of computing and networked systems
- Software systems and services, and enterprise applications
- Social networks, multimedia systems, Web services
- Cyber-physical systems, including the smart grid

- 5. Methodologies, metrics, abstractions, algorithms, and tools for:
  - Analytical modeling techniques and model validation
  - Workload characterization and benchmarking
  - Performance, scalability, power, and reliability analysis
  - Sustainability analysis and power management
  - System measurement, performance monitoring, and forecasting
  - Anomaly detection, problem diagnosis, and troubleshooting
  - Capacity planning, resource allocation, run time management, and scheduling
  - Experimental design, statistical analysis, simulation
- 6. Measurement and evaluation
  - Evaluation methodology and metric
  - Testbed methodologies and systems
  - Instrumentation, sampling, tracing, and profiling of Large-scale real-world applications and systems
  - Collection and analysis of measurement data that yield new insights
  - Measurement-based modeling (e.g., workloads, scaling behavior, assessment of performance bottlenecks)
  - Methods and tools to monitor and visualize measurement and evaluation data
  - Systems and algorithms that build on measurement-based findings
  - Advances in data collection, analysis, and storage (e.g., anonymization, querying, sharing)
  - Reappraisal of previous empirical measurements and measurement-based conclusions
  - Descriptions of challenges and future directions the measurement and evaluation community should pursue

Bench 2022 Call For Papers

2022 BenchCouncil International Symposium on Benchmarking, Measuring and Optimizing (Bench 2022) Calls For Papers

https://www.benchcouncil.org/bench22/index.html

Full Papers deadline: July 28, 2022, 23:59:59 AoE
Notification: September 6, 2022, 23:59:59 AoE
Final Papers due: October 11, 2022, 23:59:59 AoE
Conference date: Nov. 7th - Nov. 9th, 2022 (Virtual)
Submission site: https://bench2022.hotcrp.com/

Introduction

Benchmarks, Data, Standards, Measurements, and Optimizations are fundamental human activities and assets. The Bench conference has two essential duties: promote data or benchmark-based quantitative approaches to tackle multidisciplinary and interdisciplinary challenges; connect architecture, system, data management, algorithm, and application communities to better co-design for the inherent workload characterizations.

The Bench conference provides a high-quality, single-track forum for presenting results and discussing ideas that further the knowledge and understanding of the benchmarks, data, standards, measurements, and optimizations community as a whole. It is a multidisciplinary and interdisciplinary conference. The past meetings attracted researchers and practitioners from the architecture, system, algorithm, and application communities. It includes both invited sessions and contributed sessions.

Regularly, the Bench conference will present the BenchCouncil Achievement Award ($3000), the BenchCouncil Rising Star Award ($1000), the BenchCouncil Best Paper Award ($1000), and the BenchCouncil Distinguished Doctoral Dissertation Awards in Computer Architecture ($1000) and in other areas ($1000). This year, the BenchCouncil Distinguished Doctoral Dissertation Award includes two tracks: computer architecture and other areas. Among the submissions of each track, four candidates will be selected as finalists. They will be invited to give a 30-minute presentation at the Bench’22 Conference and contribute research articles to BenchCouncil Transactions on Benchmarks, Standards and Evaluation. Finally, for each track, one among the four will receive the award for each track, which carries a $1,000 honorarium.

Organization

General Co-Chairs

Emmanuel Jeannot, INRIA, France
Peter Mattson, Google, USA
Wanling Gao, University of Chinese Academy of Sciences, China

Program Co-Chairs

Chunjie Luo, ICT, Chinese Academy of Sciences, China
Ce Zhang, ETH Zurich, Switzerland
Ana Gainaru, Oak Ridge National Laboratory, USA

Publicity Co-Chairs

David Kanter, MLCommons
Rui Ren, Beijing Institute of Open Source Chip
Zhen Jia, Amazon

Web Co-Chairs

Jiahui Dai, BenchCouncil

Jiahui Dai, BenchCouncil
Qian He, Beijing Institute of Open Source Chip

Award Committees

BenchCouncil Distinguished Doctoral Dissertation Award Committee in Other Areas:
Jack Dongarra, University of Tennessee
Xiaoyi Lu, The University of California, Merced
Jeyan Thiyagalingam, STFC-RAL
Lei Wang, ICT, Chinese Academy of Sciences
Spyros Blanas, The Ohio State University

BenchCouncil Distinguished Doctoral Dissertation Award Committee in Computer Architecture:
Peter Mattson, Google
Vijay Janapa Reddi, Harvard University
Wanling Gao, Chinese Academy of Sciences

Bench Steering Committees

Jack Dongarra, University of Tennessee
Geoffrey Fox, Indiana University
D. K. Panda, The Ohio State University
Felix, Wolf, TU Darmstadt
Xiaoyi Lu, University of California, Merced
Resit Sendag, University of Rhode Island, USA
Wanling Gao, ICT, Chinese Academy of Sciences & UCAS
Jianfeng Zhan, ICT, Chinese Academy of Sciences &BenchCouncil

Call For Papers

The Bench conference encompasses a wide range of areas and topics in benchmarking, measurement, evaluation methods and tools. We solicit papers describing original and previously unpublished work. The areas and topics of interest include, but are not limited to the following.

- 1. Areas:
  - Architecture
  - Data Management
  - Algorithm
  - Datasets
  - System
  - Network
  - Reliability and Security
  - Application
- 2. Topics:
  - Benchmark and standard specifications, implementations, and validations
  - Dataset Generation and Analysis
  - Workload characterization, quantitative measurement, design and evaluation studies
  - Methodologies, abstractions, metrics, algorithms and tools
  - Measurement and evaluation

Paper Submission

Papers must be submitted in PDF. For a full paper, the page limit is 15 pages in the LNCS format, not including references. For a short paper, the page limit is 8 pages in the LNCS format, not including references. The submissions will be judged based on the merit of the ideas rather than the length. The reviewing process is double-blind. Upon acceptance, the proceeding will be published by Springer LNCS (Indexed by EI). Please note that the LNCS format is the final one for publishing. Distinguished papers will be recommended to and published by the BenchCouncil Transactions on Benchmarks, Standards and Evaluation (TBench).
At least one author must pre-register for the symposium, and at least one author must attend the symposium to present the paper. Papers for which no author is pre-registered will be removed from the proceedings.

Submission site:https://bench2022.hotcrp.com/
LNCS Latex template:https://www.benchcouncil.org/file/llncs2e.zip

Awards

* BenchCouncil Achievement Award ($3,000)
- This award recognizes a senior member who has made long-term contributions to benchmarking, measuring, and optimizing. The winner is eligible for the status of a BenchCouncil Fellow.

* BenchCouncil Rising Star Award ($1,000)
- This award recognizes a junior member who demonstrates outstanding potential for research and practice in benchmarking, measuring, and optimizing.

* BenchCouncil Best Paper Award ($1,000)
- This award recognizes a paper presented at the Bench conferences, which demonstrates potential impact on research and practice in benchmarking, measuring, and optimizing.

* BenchCouncil Distinguished Doctoral Dissertation Award ($2000)
- This award recognizes and encourages superior research and writing by doctoral candidates in the broad field of benchmarks, data, standards, evaluations, and optimizations community. This year, the award includes two tracks, including the BenchCouncil Distinguished Doctoral Dissertation Award in Computer Architecture ($1000) and BenchCouncil Distinguished Doctoral Dissertation Award in other areas ($1000).

Technical Program Committee

Murali Krishna Emani, ANL
Shin-ying Lee, AMD
Steve Farrell, NERSC
Krishnakumar Nair, Meta
Greg Diamos, Landing.AI
Fei Sun, Alibaba
Narayanan Sundaram, Facebook
Zhen Jia, Amazon
Shengen Yan, SenseTime
Gang Lu, Tencent
Rui Ren, Beijing Open-Source IC Academy
Bin Hu, ICT, CAS
Khaled Ibrahim, Lawrence Berkeley National Laboratory
Sascha Hunold, TU Wien
Woongki Baek, UNIST
Mario Marino, Leeds Beckett University
Bin Ren, William & Mary

Gwangsun Kim, POSTECH
Vladimir Getov, University of Westminster
Guangli Li, ICT, CAS
Biwei Xie, ICT, CAS
Nicolas Rougier, INRIA