

BenchCouncil Transactions

TBench

Volume 6, Issue 3

2026

on Benchmarks, Standards and Evaluations

Review Paper

- Charting the Benchmarking Landscape – A Survey of Benchmarks for Modern Memory Architectures

David Broneske, Christian Eichler, Hamid Farzaneh, Birte Friesel, Alexander Halbuer, Benedict Herzog, Muhammad Attahir Jibril, Sajad Karim, Sven Köhler and Manuel Vögele

Research Article

- Endogenous Punishment-Driven Distributed Computing Environment for Data Factor Property Right Protection

Chuan Fu and Jian Ye

- ContextFidelity-Bench: A Three-Paradigm Framework for Evaluating Multi-Turn Hallucination Behavior in Language Models

Parsa Bakhtary

ISSN: 2772-4859

© 2026 BenchCouncil Press on Behalf of International Open Benchmark Council

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

Charting the Benchmarking Landscape – A Survey of Benchmarks for Modern Memory Architectures

David Briones, Christian Eichler , Hamid Farzaneh, Birte Friesel, Alexander Halbuer, Benedict Herzog, Muhammad Attahir Jibril, Sajad Karim, Sven Köhler and Manuel Vögele01

Endogenous Punishment-Driven Distributed Computing Environment for Data Factor Property Right Protection

Chuan Fu and Jian Ye.....30

ContextFidelity-Bench: A Three-Paradigm Framework for Evaluating Multi-Turn Hallucination Behavior in Language Models

Parsa Bakhtary43

REVIEW PAPER

Charting the Benchmarking Landscape – A Survey of Benchmarks for Modern Memory Architectures

David Broneske^{1,*}, Christian Eichler², Hamid Farzaneh³, Birte Friesel⁴,
Alexander Halbuer⁵, Benedict Herzog², Muhammad Attahir Jibril⁶,
Sajad Karim⁷, Sven Köhler⁸ and Manuel Vögele²

¹Deutsches Zentrum für Hochschul- und Wissenschaftsforschung, Hannover, Germany, ²Ruhr University Bochum, Bochum, Germany, ³TU Dresden, Dresden, Germany, ⁴Osnabrück University, Osnabrück, Germany, ⁵Leibniz Universität Hannover, Hannover, Germany, ⁶TU Ilmenau, Ilmenau, Germany, ⁷Otto-von-Guericke University Magdeburg, Magdeburg, Germany and ⁸Hasso Plattner Institute, University of Potsdam, Potsdam, Germany

*Corresponding author. broneske@dzhw.eu. All authors contributed equally to the paper; therefore, they are listed in alphabetical order.

Received on 1 March 2026; Accepted on 3 September 2026

Abstract

Benchmarks play a crucial role in evaluating system components, comparing algorithmic advancements, and assessing software implementations. While well-established benchmark suites exist for various domains, modern memory technologies, such as high-bandwidth memory and near-memory computing, lack standardized benchmarking methodologies. Researchers often resort to custom benchmarks or adapt existing ones, making comparative analyses across different technologies challenging. This survey provides a structured overview of benchmarks used in modern memory architectures, catering to both newcomers and experienced researchers. It builds on top of a structured literature review of 1937 benchmark-related publications obtained via Google Scholar in 2024, out of which 940 were within the scope of this survey, covering a total of 834 unique benchmarks. Out of these, we categorize and analyze 17 proposed and 739 existing benchmarks by manual analysis, identify trends in their adoption, and highlight gaps in the benchmarking landscape. By offering insights into benchmarking methodologies and their evolution, this survey aims to guide researchers in selecting appropriate benchmarks for memory system evaluations, fostering consistency and comparability in future research.

Key words: Memory Benchmarks, Literature Survey, Memory Devices, Storage Devices

1. Introduction

Benchmarks are a key utility for characterizing system components, evaluating the benefits of algorithms or software implementations over the state of the art, and many additional tasks that require quantitative comparisons. Their use ranges from one-off microbenchmarks that focus on a single hardware or software attribute – where the benchmark itself is often merely a by-product of the evaluation – to widely adopted benchmark suites that allow for quantitative performance comparisons of different software or hardware stacks.

Modern memory technologies and architectures, such as high-bandwidth memory or near-memory computing, are no exception to this rule. However, considering their novelty, there are no established standard benchmarks for hardware characterization or comparative performance evaluation. Some authors choose to build custom benchmarks which are tailored specifically to the system components they work with, while others rely on existing benchmarks and apply them to these novel technologies, adjusting and extending them as needed. Under these conditions, comparing different disruptive memory technologies or corresponding software

stacks is challenging, as the corresponding publications are likely to use different (incompatible or incomparable) benchmarks or benchmark configurations.

This survey structures the field of benchmarks for modern memory architectures by providing an overview of both commonly used and newly developed benchmarks across a variety of memory systems. It is tailored towards newcomers and established memory/storage benchmark researchers alike. Newcomers can use it to get an overview of existing memory and storage technologies, including details as to how the research community typically benchmarks them and what common practices should be followed. Established researchers, on the other hand, can use it to identify trends in the usage of existing and proposed benchmarks, and identify gaps in the current benchmarking landscape. In both cases, the survey aims to aid researchers in selecting appropriate benchmarks for the evaluation of modern memory architectures, as well as algorithms and applications that work with those.

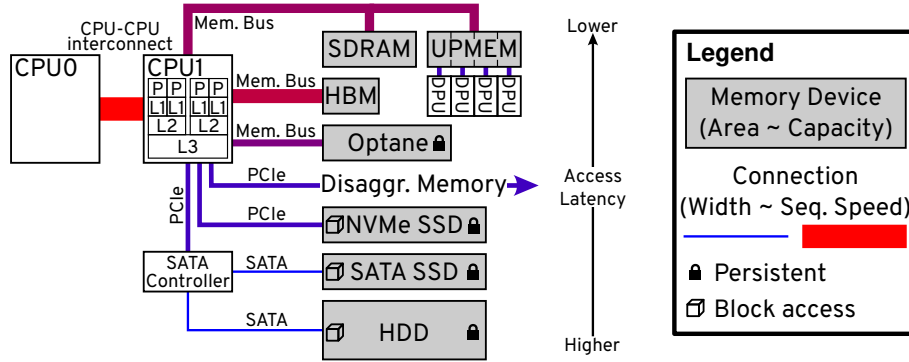


Fig. 1. Overview of different memory components in a modern computing system: The placement of the memory devices reflects their access latency, forming an arrangement inspired by the classical memory pyramid. Box size and line width (supported by color gradient) represent typical sizes and sequential speeds (not true to scale). Icons mark persistence and byte-/block-addressability. For clarity, we use a CPU-centric visualization, even if some technologies may be more common on other devices (e.g., HBM on GPUs). This does not affect the memory characteristics and, accordingly, the placement within this overview.

Our contribution in this article is threefold.

- We illustrate an *overview of existing and emerging memory technologies* and corresponding literature based on a structured literature review of high-ranked benchmarking-related publications for these technologies.
- We provide an extensive *characterization of novel benchmarks* that were originally proposed within our literature corpus, specifically targeting modern memory architectures. This includes guidelines for benchmark users and a discussion of research gaps that may be interesting for future work.
- We also give an *overview of existing benchmarks* that are being applied to modern memory architectures, even if they were originally not designed for them, and a thorough analysis of their use in practice. Our analyses of the evolution of benchmarking show that benchmark popularity follows general research trends and that a considerable amount of benchmark types and domains is already used for benchmarking. Hence, this sets standards for future benchmark papers.

We note that, throughout this article, we use *benchmark* as a general term that includes all types of data, applications, and similar that can be used for benchmarking. A detailed distinction between different types of benchmarks, ranging from algorithms over datasets to benchmark suites, follows in Section 4.4 as part of our methodology.

The remainder of the article is structured as follows. After giving an overview of the memory and storage technologies and vendor-specific implementations that we examine in Section 2, Section 3 lists the goals and research objectives that we address in this survey, and Section 4 describes our methodology. Section 5 follows up with an in-depth analysis of proposed benchmarks that we identified, and Section 6 shows existing benchmarks are applied to modern memory architectures. After examining related work in Section 8, we conclude in Section 9.

2. Modern Memory and Storage Devices

We start with an overview of the modern memory technologies covered in this survey and the established technologies they relate to. This technology overview serves as the foundation for our structured literature review. In addition, the relationship description provides context to how novel and emerging memory technologies differ from existing system components such as SDRAM or SSDs.

Fig. 1 provides an overview of the various memory components in modern computing systems. The figure shows a multi-socket system where all CPUs can access all memory, but access times might differ. For simplicity, memory attached to CPU0 is not shown. The key differentiators among these devices are speed, latency, size, access type, data persistence, and, to a wider extent, cost. Selecting and sizing memory involves balancing these attributes.

SRAM caches (L1-L3) within the CPU are at the top of the hierarchy, offering the fastest access. Main memory, typically built with SDRAM, is slower but cheaper, allowing for larger capacities. High-Bandwidth Memory (HBM) sacrifices latency for sequential speed, but has lower capacities due to its complex production process. Traditional block storage (e.g., HDDs and SSDs) provides data persistence in large capacities but with coarse access granularity. Persistent main memory like Intel Optane bridges this gap by offering byte-wise access to persistent data with near-DRAM performance.

Near-Memory Computing (NMC), exemplified by UPMEM, integrates computing power into memory modules for highly parallel access. The emerging disaggregated memory enables the expansion of main memory aside the conventional memory bus, allowing for a more flexible, efficient, and dynamic assignment of memory resources.

In the following, we comparatively review these memory and storage devices based on their capabilities and key performance numbers, summarized in Table 1. As a result, we do not only characterize the current and emerging memory and storage landscape, but also build the basis for our literature review on benchmarking modern memory.

2.1. Memory Technologies

While memory capacity and bandwidth have steadily improved over the past decades, access latency has remained largely constant [1]. Combined with the growing number of cores per CPU package, this further exacerbates the so-called *Memory Wall* [2]: no matter how efficiently a parallel algorithm has been implemented, CPU cores must wait for DRAM transfers to complete whenever working with data that is not available in their caches. This section gives a quick overview of SDRAM (the baseline for server memory) and more recent technologies that aim to improve upon it.

2.1.1. SDRAM

Synchronous Dynamic Random Access Memory (SDRAM) has been the go-to volatile storage medium for multi-megabyte embedded devices to multi-terabyte server systems for several decades. The vast majority of today’s general-purpose computers build upon DRAM that implements the *Double Data Rate* (DDR) standard, which has continuously evolved from DDR in 2001 to DDR5 in 2020 [3]. Capacity and bandwidth have steadily increased: nowadays, DDR4 and DDR5 memory modules provide tens to hundreds of GiB per memory module, with supported maximum data rates ranging from 12.8 GiB/s (DDR4-1600) to 64 GiB/s (DDR5-8000) [1]. Access latency has remained steady on the order of tens to hundreds of nanoseconds [1], and is influenced by pre-fetching on the CPU as well as implementation details: unbuffered, registered, and load-reduced memory modules behave differently [3].

SDRAM allows for building server systems with several TiB of total memory capacity and hundreds of GiB/s of combined memory bandwidth. While it is not classified as a disruptive memory technology, we include it as a baseline for evaluating novel and emerging memory technologies.

2.1.2. HBM

High Bandwidth Memory (HBM) is a stacked DRAM variant that offers higher bandwidths (up to 256 GB/s for HBM2 and 1229 GB/s for HBM3e) with lower power demand compared to DDR or GDDR (Graphics DDR) memory. The stacked DRAM dies are interconnected by through-silicon vias and connected to a memory controller via a wide memory interface (1024 bit, consisting of 8 to 16 channels). HBM is typically used in graphics cards, FPGAs, and as a CPU cache for high-bandwidth applications. However, due to the complex manufacturing process and signal routing on the mounting circuit board, the benefits of HBM come at a cost.

A single HBM stack can comprise up to eight 8 Gbit dies (HBM2) or twelve 24 Gbit dies (HBM3e), resulting in maximum capacities of 8 GiB and 36 GiB, respectively [4]. Multiple stacks can be connected to the memory controller, further widening the memory interface (e.g., 4096 bit). The high bandwidth is achieved through the wide memory interface, despite modest clock rates comparable to DDR. This results in a low power demand but potentially higher latencies [5].

2.1.3. PCM

Phase Change Memory (PCM) is a type of non-volatile memory that stores data by leveraging the reversible phase-changing properties of chalcogenide glass materials, such as germanium-antimony-tellurium (GST). The technology works by applying an electrical pulse to the material, which generates localized heat that alters the physical state of the material between amorphous (high resistivity) and crystalline (low resistivity) states. Their distinct resistivity states correspond to binary values; for example, the former state may represent binary 0, and the latter state binary 1, enabling data storage and retrieval. Research on PCM dates back to the 1960s, and its first patent was registered in September 1966. It attracted significant research interest, but subsequent advancements were impeded by intrinsic material limitations which contributed to operational inefficiencies, including high power consumption. Later, scaling issues and related challenges with flash and DRAM, along with ongoing improvements in PCM, have once again drawn the attention of researchers and the industry towards it [6].

Storage-class memory, embedded non-volatile memories, and in-memory computing are some applications for which PCM can be utilized. It is particularly considered a potential replacement for traditional NVM technologies, offering better latency, endurance,

and energy efficiency. However, PCM faces some challenges. For example, there are trade-offs involved with energy efficiency, performance, and scalability, as well as the need for error-correction mechanisms to address write disturbances. Moreover, when compared to other emerging NVM technologies, it offers limited write endurance (10^6 to 10^9) and high per-bit write energy (≈ 90 pJ to 1 nJ).

Intel Optane DCPMM: Optane DC Persistent Memory Modules (DCPMM) by Intel are the only commercially available NVM modules leveraging a PCM-based technology (3D-XPoint). The modules are available in DIMM form factor and different capacities (up to 512 GB) across two generations (Optane DCPMM 100 and 200 series). They are compatible with selected 2nd and 3rd Gen Intel Xeon Scalable processors, respectively. The modules connect to the memory bus alongside DRAM DIMMs. The CPU’s integrated memory controller (iMC) maintains a write-pending queue (WPQ), which, together with the DCPMM, constitutes the persistence domain, ensuring the data that reaches the WPQ is flushed to DCPMM even on power failure or shutdown. This is known as the asynchronous DRAM refresh (ADR) mode [7], while the enhanced ADR (eADR) extends the persistence domain to include CPU caches. The iMC communicates with the DCPMM via the DDR-T protocol, which operates at cache line granularity (typically 64 B). However, the 3D-XPoint media is accessed at 256 B granularity, leading to write amplification. Internally, DCPMM uses XPPrefetcher (a hardware prefetcher optimized for 3D-XPoint media) and XPBuffer (the media-specific buffer for writes) for prefetching and buffering reads and writes. It also uses logical addressing with an internal address indirection table for wear-leveling.

2.2. Storage Technologies

In the following, we also characterize HDD and SSD storage technologies that provide persistent, block-based data access. Just like SDRAM, these serve as baselines for analyzing novel and disruptive memory technologies and are shown for the sake of completeness. However, we will not consider these technologies as keywords for our literature review due to our main focus on memory benchmarks.

2.2.1. HDD

Hard Disk Drives (HDDs) have emerged as main mass storage in the past four decades [8]. They utilize ferromagnetism to store digital data on rotating disks, resulting in an economic balance of capacity and performance. Typical sizes are nowadays in the range of 500 GB for small 2.5” devices [9] up to about 26 TB for larger 3.5” NAS drives [10], organized into 4 KiB sectors [11]. While capacity has steadily increased [12], the performance could not keep up [13]. Current HDDs achieve a sequential read/write speed of approximately 100 MiB/s [14]. However, due to mechanical constraints, random accesses experience a notable performance decline as the actuator arm, carrying the read/write head, must physically traverse to the new location for each operation, resulting in an access latency of about 10 ms [14]. With these characteristics, HDDs are a cost-effective solution for storing large datasets with low to medium access activity, costing around 0.03 €/GB with a medium-sized drive (WD Red Pro, 4 to 8 TB)¹.

¹ Taken from the price comparison website [geizhals.de](https://www.geizhals.de) on 27/11/2024: 119.90 € (WD Red Pro 4 TB), 215.00 € (WD Red Pro 8 TB)

Properties	Device/Technology								
	SDRAM	HBM	RTM	NMC [20] PIM [21]	UPMEM	PCM	Optane DCPMM	SSD (NVMe)	HDD
Preferred Access Pattern	Random	Sequential [22]	N/A	NMC: $\approx$ DRAM PIM: $\approx$ PCIe MMIO	Sequential / 64 MiB [23]	Sequential [24, 25]	Sequential	Random	Sequential
Access Granularity	64 B	32 B [36]	Flexible (e.g., 100 bits)	64 B	512 B–2 KiB [23]	64 B (larger is better) [24]	256 B	4 KiB	4 KiB
Usual Sizes per Unit	8–256 GiB [27]	8–36 GiB [28, 4]	$\approx$ HDD [39]	NMC: 32x 128 MiB PIM: 4x 64 GiB	8 GiB [23]	1–32 GiB [30, 31]	128–512 GiB	250 GB–4 TB [32, 33]	0.5–26 TB [30, 3]
Retention Time	64 ms [34, 27]	$\approx$ DRAM [35, 36]	Years	$\approx$ DRAM	$\approx$ DRAM ^a	> 10 Years [37]	> 10 Years ^b	10 Years [37, 34]	> 10 Years [37, 34]
Write Endurance	> 10^{16} [38]	$\approx$ DRAM	∞	$\approx$ DRAM	$\approx$ DRAM	> 10^6 – 10^9 [37, 39, 40]	$\approx 10^6$ [41]	300–100K [42]	> 10^6 [24]
Bandwidth per Unit	Up to 64 GB/s (DDR5-8000) [1]	Up to 1.2 TB/s [28, 4]	N/A	NMC: Up to 1.875 GB/s PIM: Up to 15 GB/s (DDR4-1866)	CPU: up to 13 GB/s [22] DPU: $\approx$ 500 MB/s [23]	N/A	Up to 8.10 GB/s [41]	$\approx$ 7 GB/s [32, 33]	$\approx$ 100 MB/s [14]
Latency R(ead)/W(rite)	10–100 ns [34, 27, 43]	Slightly higher than DRAM	1–10 ns [37]	CPU: N/A (NMC, PIM) DPU: $\approx$ DRAM (PIM)	CPU: $\approx$ 100 μ s [22] DPU: $\approx$ 60 cycles [23]	R: 30–60 ns, W: 10–300 ns [40, 37, 39]	$\approx$ 400 ns [45]	$\approx$ 100 μ s [18]	$\approx$ 10 ms [14]
Price per GB	1–10 €	$\approx$ 3–5x (G)DDR ^{cd}	$\approx$ HDD [39]	N/A	$\approx$ 56 €	N/A	9.21–16.60 € [46]	$\approx$ 7 ct	$\approx$ 3 ct
Parallelism	Across DIMMs	Multiple dies per stack, multiple stacks	N/A	NMC: 32 (1 per vault) PIM: 4 (1 per DIMM)	One DPU with 24 tasklets per 64 MiB block [23]	N/A	Across DIMMs	256	1

^a <https://www.upmem.com/technology/>^b Retention period inferred based on underlying PCM technology due to lack of official data.^c <https://www.businesskorea.co.kr/news/articleView.html?idxno=225429>^d <https://dramexchange.com/>

Table 1. Key characteristics and features of various established, emerging, and upcoming memory and storage technologies. The given entries are typical values; optimized implementations (in any direction) may exist but do not reflect commonly used devices. The vertical double line separates byte- from block-addressable devices as use cases differ. Color scheme: best [■ ■ ■ ■ ■ ■ ■ ■ ■ ■] worst.

2.2.2. SSD

Unlike HDDs, Solid-State Drives (SSDs) rely on flash technology, eliminating mechanical components [15]. Available in common sizes ranging from 250 GB to 4 TB for 2.5" SATA and M.2 NVMe drives [16, 17], SSDs operate without mechanical components, resulting in a significantly lower access latency of about 0.1 ms [18]. They can handle multiple requests in parallel, achieving up to 1.5 million I/O operations per second (IOPS) and throughput of approximately 7 GB/s [17]. However, SSDs come with some drawbacks. They typically incur higher costs compared to HDDs and have limited write endurance. To mitigate per-gigabyte costs, many modern SSDs employ triple-level cell (TLC) flash technology, allowing for storing 3 bits per cell, thereby increasing density but impacting lifespan. The typical endurance of a TLC SSD is approx. 1000 write cycles [19] at costs of around 0.07 €/GB (Samsung 990 Pro, 2 TB)².

2.3. Emerging Technologies

After looking into baseline technologies and modern memory types that are already commercially available, we now examine emerging technologies and concepts. We cover both concepts that have not seen any use beyond individual prototypes or simulators yet and emerging technologies that are available, for instance, as research or engineering samples.

2.3.1. Racetrack Memory

Racetrack Memory (RTM) is a concept for a persistent memory device, and it is still in its design phase. The idea of RTM has been devised by Stuart Parkin [47] and led to innovative methods for operating RTM. The core concept is to store information on magnetic regions that encode individual bits – similar to HDDs. However, these regions can be freely arranged in a 2D or 3D structure and can be manipulated in a free fashion (allowing also a vertical shift or curved movement of data) [29]. Despite not having a market-ready prototype, RTM is said to deliver a higher storage density than any known device while providing high write endurance and low energy consumption.

2.3.2. Near-Memory Computing

Near-Memory Computing (NMC) is a memory-centric computing paradigm that provides computation capabilities to the usually passive memory subsystem. Thereby, it makes it an active processing component similar to the CPU. NMC or PIM (*Processing-in-Memory*) architectures are generally realized by bringing processing elements closer to the memory. Some architectures are based on 3D-stacked memory technologies [48], where different processing units are implemented in the logic layer of the 3D-stacked memory. However, this approach is limited by the area and heating constraints of the logic layer. Other architectures integrate processing cores into the memory chips of Dual Inline Memory Modules (DIMMs). Although this approach does not suffer from the limitations of the previous approach, there is a mismatch in fabricating memory and processing components on the same chip, leading to slower logic circuitry [23].

UPMEM PIM: UPMEM PIM is the first, and at the time

² Taken from the price comparison website [geizhals.de](https://www.geizhals.de) on 27/11/2024: 134.90 € (Samsung 990 Pro 2 TB)

of writing still only, commercially available implementation of NMC [49, 23]. It builds upon conventional DDR4 DRAM modules and extends those with 128 *DRAM Processing Units* (DPUs) per module. This way, it allows for compute tasks to be moved from the CPU to a set of DPUs, bypassing the memory controller and CPU cache bottlenecks.

UPMEM’s current implementation faces several limitations [23, 50]. Most prominently, DPUs do not behave like additional CPU cores that are simply located close to main memory. Instead, each 8 GiB UPMEM module is partitioned into 128 chunks of 64 MiB, with one DPU per chunk. DPUs cannot access memory outside this region: all inter-DPU synchronization must happen via the CPU.

Moreover, due to manufacturing and DDR interface limitations, data is interleaved between DPUs. A consecutive 64 MiB block of data (as viewed from the CPU perspective) does not end up in the memory region of a single DPU, but is scattered over several DPUs. Application developers have to rely on the UPMEM SDK or compatible implementations to explicitly transfer data between conventional DRAM and UPMEM PIM modules, and cannot easily access data in PIM RAM both from the CPU and from DPUs. Overall, this makes UPMEM PIM behave more like a computational offloading device than a true near-memory computing implementation [50].

Samsung’s HBM-PIM: Samsung introduced an NMC architecture referred to as Function-in-Memory DRAM (FIMDRAM) [51] or HBM-PIM [52]. In contrast to UPMEM, which integrates a RISC-V processor with each memory bank, HBM-PIM features 16 Single-Instruction-Multiple-Data (SIMD) Floating-Point Units (FPUs) per two banks. Each FPU includes a 16-bit floating-point adder and multiplier, making HBM-PIM particularly suited for machine learning applications, especially for efficiently executing *General Matrix-vector Multiplication* kernels. On the other hand, UPMEM’s processors are designed to be more general-purpose and are not tailored to a specific domain. While Samsung reports that their design is compatible with any DRAM family, they showcased its functionality using the 2.5D HBM DRAM.

2.3.3. In-Memory Computing

This section explains three primary in-memory architectures: crossbars, content-addressable memory (CAM), and boolean or arithmetic logic units.

Crossbar: Crossbar arrays are a fundamental circuit architecture for implementing high-density memory and in-memory computing capability. They consist of a grid of wordlines and bitlines with programmable or resistive elements (such as memristors, phase-change memory, or Resistive-RAM (ReRAM)) at their intersections. These structures enable efficient analog computation, particularly for matrix-vector multiplication, by exploiting Ohm’s Law and Kirchhoff’s Current Law [53, 54]. When storing matrix weights as conductance values in a crossbar, input vectors can be applied as voltage signals to the wordlines, generating output currents that inherently perform matrix-vector multiplication in a single operation.

CAM: Content-Addressable Memories (CAMs) are a class of memory systems designed for fast, parallel searching based on content rather than address [55, 56]. Unlike conventional RAM, which retrieves data by specifying a memory location, CAMs compare an input search key against all stored entries simultaneously, returning the address of a matching entry in a single clock cycle. This parallelism makes CAMs particularly useful for high-speed applications such as network routing tables, cache management, and associative computing. When extended to analog or resistive implementations, CAMs can be leveraged for in-memory

computing tasks, including matrix-matrix multiplication and dot-product computations essential for linear algebra operations.

Bitwise logic: Bitwise logic operations in memory have been implemented using various memory technologies to accelerate data-intensive computations by eliminating unnecessary data movement. In DRAM-based implementations, techniques like Ambit [57] leverage the charge-sharing property of DRAM cells to perform bitwise operations such as AND, OR, and NOT by simultaneously activating multiple rows and utilizing the row buffer to compute results in place [58]. Similarly, ReRAM and memristor-based designs use the conductance states of memory cells to implement NOR and NAND gates, enabling massively parallel logic operations that are particularly beneficial for machine learning and cryptographic applications [59].

2.3.4. Disaggregated Memory

Memory disaggregation addresses a key issue with traditional server architectures, where memory is statically bound to a single compute node, leading to stranded capacity when one node runs out of memory while another sits underutilized. Interconnect technologies like Compute Express Link (CXL) [60] enable memory disaggregation by decoupling compute and memory into independently scalable memory pools connected via a fabric that allows memory capacity to be allocated on demand [61]. This memory pooling can occur at different granularities, from rack-scale pools shared across multiple nodes to node-level memory expansion, and can be exposed to software either transparently as local memory or explicitly, requiring applications to manage placement across tiers [62, 63]. Other fabric-attached interconnects, namely Gen-Z for rack-scale memory fabric connectivity and CCIX and OpenCAPI for cache-coherent accelerator attachment, preceded CXL and are now part of the CXL Consortium since CXL’s specification expanded to cover their capabilities. Nevertheless, disaggregation introduces several inherent challenges. For instance, fabric traversal adds access latency and shared bandwidth bottlenecks [64]. Beyond performance, security and isolation is another challenge as sharing physical resources across a fabric complicates data protection and multi-tenant isolation [65]. Last but not least, reliability and fault tolerance pose another challenge, where fabric or memory-node failures can lead to unrecoverable host-side crashes [65]. These remain general limitations of memory disaggregation, however, compared to earlier RDMA-based disaggregation over Ethernet or InfiniBand (which utilizes existing network fabrics, treating remote memory access as a network operation with associated network-access overhead [66]), CXL achieves substantially lower access latencies by operating as a memory-semantic protocol over PCIe rather than a network-semantic protocol [61]. Moreover, it integrates hardware-level, end-to-end encryption and data integrity protocols, and incorporates different measures to manage errors and prevent host-side crashes during memory node failures [67, 68]. This makes CXL a preferred approach to memory disaggregation.

CXL utilizes the standard PCIe interface and has a mechanism to select either PCIe or CXL to communicate with devices. A variety of use cases are built upon three sub protocols (CXL.io, CXL.cache, and CXL.memory), whose combination results in three different device types: Type-1 devices (e.g., NIC or accelerators without memory), type-2 devices (e.g., GPUs or accelerators with memory), and type-3 devices (volatile and non-volatile memory expanders). While CXL.cache ensures coherent host memory access for type-1 and type-2 devices, CXL.mem enables device memory access from the host side for type-2 and type-3 devices. CXL.io is equivalent to the standard PCIe protocol, handles non-coherent load/store operations, and has to be implemented by all devices.

The integrators list on the CXL website³ provides an overview of available CXL-enabled memory expanders. Additional companies like Samsung have also announced their CXL-enabled memory expanders, but, due to the lack of detailed specifications, the following comparison focuses on the currently available devices listed in the integrators list.

CXL memory expanders can currently reach bandwidths of up to 64 GB/s (16x PCIe5), comparable to SDRAM and far higher than Optane DCPMM. They support configurations with up to 2 TB of memory, surpassing HBM. Additionally, they are compatible with DDR4 and DDR5 DIMMs, enhancing their flexibility and scalability. Moreover, similar to Intel’s Optane DCPMM, CXL memory expanders provide latency characteristics that are lower than those of NVMe SSDs but higher than DRAM. Evaluations presented in [69] and [70] across two prototypes indicate that the latencies are approximately 2.5 to 3 times greater than those of CPU-local DRAM.

2.4. Summary

Considering the reviewed advances in established and emerging memory and storage devices, a growing diversity and specialization of the devices becomes apparent. Memory is not inherently volatile, can be attached in a dedicated fashion, and now also provides processing capabilities. Despite these emerging characteristics, the fundamental benchmarking requirements still prevail and are extended by new use cases and choke points. Hence, mapping these characteristics with the investigated choke points of current and future benchmarks is an important contribution of our work.

3. Goal and Research Objectives

The field of benchmarking is huge, with diverse applications and domains. Hence, we focus our research on exploring the landscape of benchmarks for modern memory devices. For a systematic exploration of this landscape, we define four research objectives (RO) that our systematic literature survey will answer. Since this survey is intended to be of help for benchmark users as well as benchmark researchers in the area of modern memory, we first analyze a subset of benchmarks for which our data set contains the paper that originally proposed the benchmark (which we call *proposed benchmarks*) and analyze their characteristics (cf. RO₁) and the challenges they pose for the specific devices (cf. RO₂). Naturally, there are many benchmarks which are not being proposed by papers within our data set, that are nevertheless used in scenarios for modern memory (we call these *used benchmarks*). This can happen for benchmarks that are not tailored to benchmarking memory or for benchmarks for which no paper exists in the first place. To also analyze the landscape of used benchmarks, we characterize them as well (cf. RO₃) and show their usage patterns (cf. RO₄).

RO₁ Characterizing Benchmarks Proposed for Modern Memory Devices

This research objective aims to systematically analyze the characteristics and performance metrics of proposed benchmarks for modern memory devices. It reveals the focus areas of current research and helps indicate which new technologies are of interest, where existing evaluation methods are insufficient due to hardware differences, and which aspects of novel memory devices are considered important for characterization. Additionally, the objective is to assess the

strengths and limitations of the existing proposed benchmarks to uncover potential areas for optimizations or enhancements.

RO₂ Mapping Device Characteristics with Benchmark Design

Establishing the relation between the characteristics of memory devices and the design of relevant proposed benchmarks is crucial. Hence, we aim to map device attributes like random vs. sequential access patterns, access granularity, read & write performance, and write endurance to the proposed benchmarks. By utilizing these analyses, the objective is to validate the suitability of the proposed benchmarks with respect to the target hardware. Additionally, we propose recommendations for creating benchmarks that can accurately reflect the performance, efficiency, and limitations of various memory technologies.

RO₃ Characterization of Used Benchmarks

With our systematic survey, we do not only find benchmarks that were originally proposed by these papers but were used to benchmark modern memory. In order to complete the view on the benchmarking landscape of modern memory devices, we characterize these used benchmarks by their domain (i.e., use case) and type (ranging from simple algorithms to full-fledged benchmark suites). These two indicators give valuable insights into the maturity and spread of benchmarks from different areas of research.

RO₄ Analysis of Benchmark Usage

In order to further analyze the state of the art in benchmarking different memory devices, we analyze the usage of certain benchmarks in papers. This includes analyzing whether papers use different types or diverse domains for benchmarking. This gives useful insights into how extensively the different memory devices have been benchmarked. Our investigation also aims to include the time perspective, giving an in-depth view on adaptations of the benchmarking practices for modern memory to the evolution of the application landscape.

In order to achieve our four research objectives based on a solid foundation, we meticulously create a corpus of relevant literature. Our corresponding methodology is detailed in the following section. Afterward, we detail our contributions for RO₁ and RO₂ in Section 5 as well as RO₃ and RO₄ in Section 6.

4. Methodology

We, the 10 authors, performed a systematic literature review in order to first identify papers that use or propose memory benchmarks and then extract individual benchmarks for analysis and characterization from them. Our approach consists of four phases: (I) systematic literature review, (II) prefiltering, (III) normalization, and finally (IV) analysis and characterization. Fig. 2 gives a high-level overview, and the following sections describe the process in detail.

4.1. Phase I: Systematic Literature Survey

The initial phase consists of a systematic literature review [71, 72, 73]. The objective is to identify papers addressing modern memory technologies through a keyword-based search using a search engine for academic literature. The focus is particularly on papers that examine *non-functional* properties of these technologies. Non-functional properties refer to *how* a memory behaves (e.g., its workload-dependent latency, throughput, or retention time), and are typically captured by benchmarks. By contrast, *functional* properties describe *what* it provides (e.g., persistent data storage at 256 B granularity with up to 8 GB/s raw bandwidth); these attributes are typically specified as part of a datasheet and not

³ <https://computeexpresslink.org/integrators-list/>

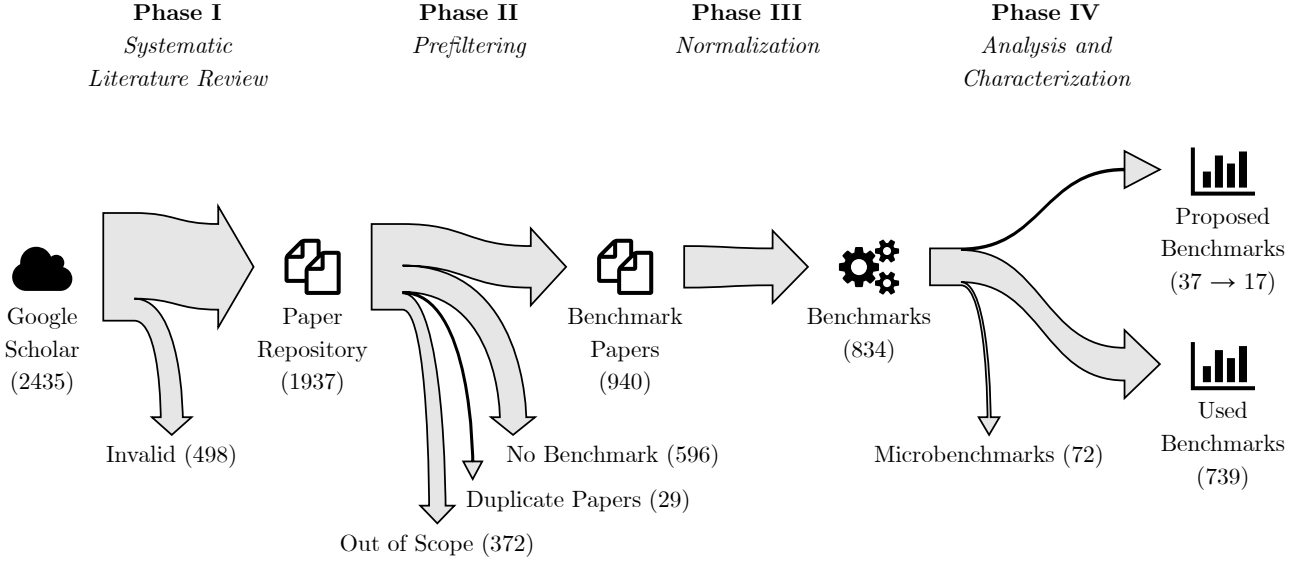


Fig. 2. High-level overview of the literature review workflow used to identify benchmarks for analysis and characterization. Arrow widths are proportional to paper/benchmark counts. Downwards arrows represent excluded papers/benchmarks. *Benchmark Papers* represent those papers that use or propose a benchmark, while *Benchmarks* are unique benchmarks that we identified. They have been categorized into *Proposed* and *Used Benchmarks* (note that these two groups overlap) for further analysis in this paper.

necessarily observable in all practical applications. As such, we are specifically interested in papers that benchmark these technologies or develop benchmarks for them, thus allowing reasoning about their non-functional properties in the context of practical applications. Additionally, papers exploring the impact of memory usage patterns also fall within the search domain.

We deliberately decided to use a single search engine (Google Scholar) for this phase in order to increase the fidelity of search results. Moreover, Google Scholar is well-accepted⁴, and hence the perfect candidate for our systematic literature review. Repeating the same search with identical keywords across multiple search engines would result in significant redundancy in the search results. Subtle differences in paper metadata between search engines (e.g., author names, paper titles) would make the identification and removal of these duplicates susceptible to false positive and false negative errors. Moreover, the performance (i.e., recall) of academic search engines has improved significantly over the years. Consequently, the search results of a single engine already contain most of the relevant papers, whereas additional search engines would mainly add noise and papers irrelevant to the topic.

Table 2 lists the search terms that form the foundation of our literature survey. Each term consists of a *domain keyword* and a *filter keyword*. The purpose of the domain keywords is to focus the results on modern memory technologies. We selected keywords from three domains:

- technology categories (e.g., Non-Volatile Memory),
- technology implementations (e.g., Race-Track Memory), and
- dominant brands (e.g., Optane).

We consider any paper that covers at least one of the specified memory technologies as potentially relevant. Since paper titles may refer to general technology categories, specific implementations, and occasionally even brand names (especially dominant ones like Optane or UPMEM), we include separate keywords for all these.

Domain Keywords	Filter Keywords
In-Memory Computing	Benchmark
Near-Memory Computing	
Non-Volatile Memory	
High-Bandwidth Memory	
Disaggregated Memory	
Heterogeneous Memory	
PIM	
UPMEM	Access Pattern
Optane	
MCDRAM	
3D XPoint	
PCM	
CAM	
Race Track Memory	

Table 2. Overview of the keywords used to form search terms for the literature survey. Each domain keyword is combined with each filter keyword. This yields 28 different search terms in total.

The second keyword category, that is, filter keywords, consists of keywords designed to limit results to papers that cover non-functional properties. As we are interested in benchmarks and their introduced choke points (especially access patterns), we use “Benchmark” and “Access Pattern” as filter keywords.

We formed search terms by combining all domain keywords with all filter keywords. When submitting queries to the search engine, we enclosed each keyword in quotation marks to ensure an exact match and exclude results containing only partial matches. Thus, each search term consists of exactly two keywords, which may themselves be multi-word phrases. In total, this process generated 28 search terms.

For each search term, we included the first 100 search results in our paper repository, or all results in case the search did not yield 100 results to begin with. The consistent metadata provided by using only a single search engine enabled us to reliably identify and remove duplicates from the start (i.e., identical publications

⁴ <https://paperpile.com/g/academic-search-engines/>

found through multiple search terms) based on the title and author list. Furthermore, there were search terms (especially UPMEM) that yielded less than 100 search results or led to inaccessible entries without a real paper source. In total, this search yielded 1937 valid papers, sorting out 498 invalid papers by a mixture of manual and supervised automatic analysis. Where possible, the papers were automatically downloaded; all papers were added to a shared paper repository.

4.2. Phase II: Prefiltering

The search terms in Phase I were intentionally kept broad and minimally restrictive to include as many relevant papers as possible into the paper repository. However, the permissive nature of the search terms resulted in a significant number of papers that match the keywords but do not align with the specific research areas of this survey. Examples include papers from unrelated fields like physics, chemistry, or medicine that use identical keywords with different meanings (e.g., CAM as a dash cam or the chemical compound), as well as papers that only mention storage technologies without further analysis.

As an automated identification and categorization of these unrelated papers was not feasible, we did so manually. Each survey author was assigned a subset of the paper repository for classification, and each classification decision was vetted by at least two different authors. We held frequent synchronization and discussion rounds in order to ensure a consistent classification scheme and deal with specialties in the corpus appropriately. We differentiate three categories: (1) out of scope, (2) no benchmark, and (3) benchmark.

4.2.1. Out of Scope

There are several reasons why a paper may be classified as *out of scope* (cf. Table 3). The most common reason is that the paper focuses on a different field of research, such as physics or chemistry. Additionally, non-academic papers and non-peer-reviewed papers are placed in this category. Examples include presentation slides, patent documents, non-English literature, books, and dissertations.

Dissertations are a special case because they may potentially address topics related to memory technologies and include benchmarks results or developed new benchmarks. However, we assume that the results from these dissertations have also been published in scientific papers, which would result in double-counting these occurrences. Furthermore, dissertations often combine results from several already-published papers into a single document, and exceed the length of regular papers. Therefore, dissertations were excluded from this analysis.

Papers that none of the eight participating institutions were able to access were also classified as *out of scope*. This occurred if the respective web server was unreachable or licenses for downloading the papers were not available, and the papers could not be obtained through other means.

Lastly, duplicates that the automatic duplication detection failed to identify were included in this category. This often happened when multiple versions of the same paper exist. A typical example involves papers that present preliminary findings at a workshop and are later extended into a full conference or journal publication. In these instances, only the most comprehensive version was considered, while all other versions were marked as *out of scope*. In total, 372 out of 1937 papers are categorized as *out of scope* and are not further considered.

4.2.2. No Benchmark

Papers that address topics of computer science but do not concern hardware attributes or papers which do not consider non-functional properties are categorized as *no benchmark*. This includes papers that describe novel concepts and approaches in theory but have not (yet) conducted benchmark tests or measurements. In total, 596 out of 1937 papers fall into this category and are not further analyzed.

4.2.3. Benchmark

Papers that explore modern memory technologies and investigate non-functional properties are categorized as *benchmark*. They can cover the analysis of non-functional properties in two different ways: 1) by using existing benchmarks or 2) by developing and presenting new benchmarks specifically designed to study the non-functional properties of the memory technology under test. We are interested in both types of papers to provide an overview of used benchmarks as well as a survey of modern proposed benchmarks.

Each paper is tagged based on whether it employs existing benchmarks or presents new benchmarks. Thereby, microbenchmarks are classified as existing benchmarks, not as new ones (see Phase III for a further discussion of microbenchmarks). In total, 940 papers fall into the *benchmark* category, with 903 using existing benchmarks, 33 introducing new benchmarks, and 4 utilizing both existing and new benchmarks. Additionally, for all of these 940 papers, we note which benchmarks are used or presented. This information is integrated into our paper repository and provides the basis for our normalization in Phase III.

To analyze and categorize all 1937 papers in this manner, the paper repository contents were evenly partitioned, distributed among all authors of this paper, and processed independently. In edge or unclear cases, the categorization was deferred and eventually decided upon collectively by all authors.

4.3. Phase III: Normalization

After the pre-filtering in Phase II, our paper repository contains 940 papers that assess modern memory technologies and consider non-functional properties. However, these papers do not reference benchmarks consistently. For instance, the SPEC CPU benchmark suite [74] is also referred to with identifiers such as “SPEC-CPU” and “SPEC CPU (2017)”. Since Phase II had to be conducted by different authors in parallel to handle the number of papers, these inconsistent references persisted in the paper repository. To ensure a consistent denomination of all benchmarks for further analysis, all identified benchmarks were normalized in this phase. A single denomination was established and defined in a post-processing script, which then mapped all alternative names to it. To limit the number of normalized benchmarks, different versions of a benchmark (or benchmark suite) were consolidated into one benchmark. For instance, the multitude of SPEC CPU benchmark suite releases that we encountered (e.g., SPEC CPU (2000), SPEC CPU (2006), and SPEC CPU (2017)) were consolidated into a single denomination: *SPEC CPU benchmark suite*.

4.4. Phase IV: Analysis and Characterization

The normalization phase yielded 834 unique benchmarks. These include custom microbenchmarks that only serve a single purpose related to a specific publication. Even if a paper designs a novel microbenchmark for its evaluation section, such a microbenchmark does not provide a complete picture for a given evaluation scenario, and is thus unsuitable for analysis as a proposed benchmark. However, the usage of microbenchmarks in papers still provides worthwhile information, just like the usage of proper benchmarks.

Inclusion	Exclusion
• Keyword match in Google Scholar • Paper is within top-100 results	• Unable to access the PDF • Paper comes from a different field • Paper is non-academic • Paper is a dissertation • Paper is a duplicate, e.g. work-in-progress report • Paper does not reference any benchmark(s)

Table 3. Inclusion and exclusion criteria for literature selection.

Hence, we subsume all 72 microbenchmarks in the repository under a single “microbenchmark” entry, and only consider it when examining used benchmarks.

Next, we examined each paper that was linked to the remaining 762 benchmarks to determine whether it introduces or simply uses the corresponding benchmark. In case a paper mentions the benchmark as a (novel) contribution and provides detailed information on it, we treat it as a *proposed* benchmark in the context of this specific paper. In case a paper focuses on using the benchmark for evaluation purposes and provides a reference to it (typically by means of a URL footnote or citation), we treat it as a *used* benchmark.

We found that these rules were sufficient to annotate each pair of publication and benchmark as either proposing or referencing it. Ultimately, we identified 23 benchmarks that were exclusively proposed as a novel benchmark in our corpus, 725 already-existing benchmarks that have been used in the papers mentioning them, and 14 benchmarks where our corpus contains both a paper proposing them as a novel benchmark and at least one paper referencing them as an existing benchmark.

Proposed and used benchmarks demand different treatment. Papers that propose novel benchmarks typically provide detailed information on benchmark structure, components, characteristics, and target platforms. In contrast, papers that merely use benchmarks are not required to specify such details; instead, they implicitly highlight the use cases of the benchmarks they employ across various scenarios and environments.

Hence, we divided work on the fourth phase into two groups that dealt with the 37 proposed benchmarks and 739 used benchmarks, respectively. Proposed benchmarks received an in-depth dissemination that includes characteristics such as access patterns, workload size and granularity, and evaluated performance metrics. Our analysis of used benchmarks instead focuses on insights that can be gained from relations between benchmark types, domains, publication date, and groups of benchmarks that are frequently used together (i.e., within a single paper).

In both cases, the detailed analysis included a separate filtering step, this time based on benchmarks rather than papers. Here, we removed all benchmarks that turned out to be unrelated to modern memory technologies, despite being part of a paper that fulfilled all of our structured literature review criteria. This includes benchmarks and datasets where the amount of publicly available information was insufficient for characterization, e.g., due to lack of source code or manuals.

4.4.1. Detailed Analysis of Proposed Benchmarks

We identified 20 items in the repository of proposed benchmarks that were unsuitable for a detailed analysis. Although some of these were also proposed benchmarks, they are not memory benchmarks in that they do not provide workloads or access patterns that are specifically testing the non-functional properties of an underlying memory technology. They include benchmarks that are made for testing runtime performance on processing devices

like modern CPUs and accelerators like GPUs, platform-specific implementations of existing benchmarks, applications that do not produce benchmarking results and may not even be targeting memory characteristics, dataset generators, and those that are out of our scope of research due to false positives in Google Scholar. Each decision on benchmark exclusion or inclusion was made by an online discussion and unanimous vote in a group of three authors. The same group-consensus procedure was used to extract the characteristics summarized in Table 5 where one (assigned) author filled in the values for each attribute (e.g., access pattern, workload granularity, target hardware) based on the benchmark’s publication and its source code, and at least one other author cross-checked the values against the same sources. The cases where the available documentation or sources did not provide enough information were marked “?”, and the cases where the benchmark’s configurability meant no single value could represent it were marked “*” rather than resolved by assumption.

In the end, this left us with 17 proposed memory benchmarks, which received further dissemination and discussion in Section 5. The goal of analyzing the proposed benchmarks is to gain insights into their planned use cases, characteristics, and the value they provide when used for evaluation. We first present a descriptive overview of each of the benchmarks, after which we extract attributes from the provided descriptions, such as target hardware, the language that they are programmed in, workload granularity, and workload size. Using this information, we perform a detailed analysis with regards to how each benchmark is tailored to the peculiarities of its target hardware, highlight the commonalities and differences between the benchmarks, outline guidelines for selecting an appropriate benchmark for a given memory system, and propose potential research directions for future benchmark design.

4.4.2. Analysis and Characterization of Used Benchmarks

Prior to the analysis, we characterize the used benchmarks. This allows us to provide insights into the context in which benchmarks are used, and assess, e.g., how the benchmarking landscape has changed over time and which types of benchmarks are most prominently used.

We first examined the suitability of existing computer science classification schemes for benchmark characterization. While this would have provided the benefit of an objective, standardized characterization scheme, we unfortunately found that they are not well-suited for our use case. Existing schemes focus on computer science research in general rather than the peculiarities of how benchmarks are used in the context of emerging memory technologies, and thus both provide a large amount of classes that are useless to us and too little classes to distinguish between actual benchmarks.

For example, the *ACM Computing Classification System* [75] offers a fine-grained separation of topics, consisting of 13 main groups, each further divided into hierarchical subgroups. While this detailed structure provides extensive coverage, it does not always align well with the nature and diversity of benchmarks

that we studied. For instance, among all the benchmarks, none fall into seven of the categories (General and Reference, Networks, Software and its Engineering, Human-Centered Computing, Applied Computing, and Social and Professional Topics). Additionally, multiple benchmarks may fall into the same category (e.g., Computing Methodologies), yet they can be very different, especially when considered in the specific context we aim to categorize them (e.g., machine learning versus parallel computing methodologies).

For this reason, we devised a classification scheme that better aligns with the nature of the benchmarks studied here, and contribute it as part of this survey: for each benchmark, we determine what *type* of benchmark it is and which *domains* it applies to. We define the following benchmark types.

- An *algorithm* is not a ready-to-use benchmark. It is a well-defined function (ranging from basic primitives such as GEMM to more complex operators such as k-means clustering) that is either provided in a library or implemented from scratch. In the latter case, it must be implemented on the evaluated hardware platform(s) first, allowing researchers to use architecture- and workload-specific optimizations.
- An *application* performs real-world tasks on real-world data; the use of application latency (or similar) as a benchmark is a side effect. Mostly, applications are shipped as a executable and treated as a black boxes. This type includes libraries and frameworks that are also used for real-world tasks (e.g., TensorFlow).
- An *application suite* consists of multiple applications.
- A *benchmark application* is a benchmark in the traditional sense: an executable script or binary that outputs at least one benchmark metric (like memory usage or execution time), and that has been designed with benchmarking in mind. Compared to applications (which only provide such metrics as a side effect), benchmarks typically feature a configurable workload.
- A *benchmark suite* consists of multiple benchmarks, typically within an over-arching domain or a special focus.
- *Datasets* serve as input (i.e., workload definition) to software components or applications, such as databases, neural networks, or graph processing applications. This also includes applications that generate datasets based on user-provided configurations.
- *Neural network architectures* are a special case, as machine learning papers treat them in different roles. Papers concerned with inference optimization take them as datasets, whereas for papers improving on architectures they serve more as an algorithm. Given their high prevalence in recent publications, we decided to treat them as a distinct kind.

Our domains are an orthogonal classification, which we define as follows. Note that some benchmarks, especially benchmark suites, can span multiple domains.

- The *databases (DB)* domain concerns storing, managing, and analyzing structured data. All algorithms (e.g., hash-join), applications (e.g., MySQL), benchmark applications (e.g., YCSB), datasets (e.g., TPC-H), etc. involved in these tasks are grouped in this domain.
- *Signal processing (SP)* concerns the transformation of data streams, especially to encrypt it (e.g., AES), compress or sort data (e.g., quick sort), or extract meaningful information (e.g., fast Fourier transformation).
- The *embedded systems (ES)* domain concentrates on performance evaluation (e.g., MLTinyPerf) and testing (e.g., revlib) of dedicated devices that perform specific tasks under resource constraints, such as microcontrollers.

- The *file systems (FS)* domain focuses on the functionality and performance of persisting and organizing data. Benchmarks originally targeted at block-oriented storage (e.g., Filebench) are likewise used in non-volatile memory system prototypes and products nowadays.
- *Graph processing (GP)* refers to benchmarks that provide graph-shaped data (e.g., Stanford Large Network Dataset Collection) or algorithms for analyzing graph data (e.g., PageRank).
- The *hardware characterization (HC)* domain concerns analyzing low-level aspects of hardware design, such as arithmetic throughput and basic read/write latency (e.g., STREAM).
- *High-performance computing (HPC)* benchmarks are usually from the scientific domain, evaluating the performance of large parallel computing systems (e.g., NPB). In this survey, we found many cases of reference implementations of particular algorithms being ported to and improved on new systems (e.g., LULESH), which we marked as *applications* within HPC.
- *Image processing (IP)* covers benchmarks that utilize and evaluate image analysis algorithms (e.g., Sobel Filter).
- The *machine learning and data mining (ML)* domain covers all benchmarks concerned with data preparation and statistical learning. It covers both data structures and algorithms for classification or clustering (e.g., k-means) and datasets for training or testing these (e.g., CIFAR).
- The *operating systems (OS)* domain covers benchmarks that evaluate aspects of system resource abstraction and organization like scheduling (e.g., sysbench) or across the entire software stack (e.g., CloudSuite).

For each benchmark, one of the authors initially proposed its type and domain. These assignments were then reviewed during an online meeting with three of the authors and approved by unanimous vote. The next section covers the novel benchmarks we identified, followed by used benchmarks in Section 6.

5. Proposed Benchmarks

For answering the research objectives RO₁ and RO₂, we analyze the 17 proposed benchmarks individually to give an in-depth view on their characteristics. To this end, we condense the descriptions of all the benchmarks into a single table (Table 5) listing their target hardware and the most important characteristics that are common across all benchmarks. Although the table summarizes all described benchmarks, thereby allowing for an easy comparison of common attributes, some of the proposed benchmarks are benchmark suites, and using a single table entry per benchmark suite would result in an incomplete representation. Therefore, we extend the characterization of benchmark suites into separate tables in Appendix A to get more profound insights into the different facets addressed within a suite. Based on all the above, we make two key contributions: firstly, we analyze the benchmarks' characteristics further, draw a comparison between them, and propose a guideline that benchmark users can use to decide which benchmark is most appropriate for which use case. Secondly, we identify research gaps in the design and scope of these benchmarks, with the aim of highlighting potential research directions for researchers toward future memory benchmark design.

5.1. Benchmark Descriptions

To address RO₁, we start with a brief description of each benchmark, focusing on its target aspects and highlighting special characteristics that cannot be conveniently covered in the format of Table 5.

Benchmarks in Table 5 with extended characterization tables in Appendix A are marked with ‡.

CircusTent [76] is a suite of microbenchmarks (summarized in Table 7) for evaluating the performance of atomic memory operations in shared and distributed memory systems. Designed to facilitate the development and prototyping of emerging computer architectures, *CircusTent* focuses on parallel processing and heterogeneous system designs. The suite supports multiple parallel programming models, including OpenMP, MPI, and OpenSHMEM, through a range of kernels. Its modular and extensible design enables researchers to easily adapt the benchmark to new system architectures.

The *DAMOV* suite [77] is a toolkit for profiling, evaluating, and optimizing data movement between main memory and CPU. It focuses on data movement bottlenecks such as DRAM bandwidth and latency issues, cache capacity limitations, and cache contention problems to identify potential candidates for near-data processing. The heart of the benchmark suite is a selection of 144 memory-bound functions filtered out of 77 000 functions from 345 applications across a broad range of domains. The *DAMOV* benchmark suite is quite diverse for a well-founded classification in our scheme⁵, which is why we decided against an in-depth analysis of the *DAMOV* benchmark suite in a separate table.

DLBENCH [78] is designed to evaluate different data organization schemes on high-performance computing (HPC) systems that feature multi-level and heterogeneous memory architectures. Its primary objective is to provide guidelines for optimizing data placement and layout in these complex systems. To achieve this, *DLBENCH* generates tailored versions of synthetic microbenchmarks, each addressing a specific organization scheme.

GORDON [79] is designed to study the performance and microarchitectural characteristics of Intel Optane PMem in combination with FPGAs. Its primary goal is to bridge the gap in traditional CPU-based profiling techniques, which are limited to certain aspects. The flexible FPGA-based approach offers a lower level of investigation, providing both statistical and trace modes for measuring throughput, latency, and queuing behaviors.

The *HMBench* suite [80] is designed to assess the performance and energy efficiency of heterogeneous memory systems with a range of microbenchmarks (see Table 8), bridging the gap between simulation-based studies and real-world hardware evaluations. Specifically, *HMBench* is built to target KeyStone II, a heterogeneous memory system that integrates diverse types of memory and processors. The benchmark bases on OpenMP 4.0 and tools like DataPlacer to analyze memory usage and computational behaviors of applications.

Hopscotch [81] is a microbenchmark suite (see Table 12) focusing on various memory characteristics. It supports sequential, random, and strided read and write operations and a tunable kernel allows further adjustment of access patterns to examine the effect of spatial and temporal locality. Its flexibility enables deep insights into the memory subsystem, and therefore it is an essential tool for the design and optimization of memory architectures.

KernelBenchmarks.jl [82] is a suite of synthetic microbenchmarks and real-world workloads like CNN training and graph analytics, that help study and analyze DRAM caches in NVRAM-based systems. The microbenchmarks focus on low-level behavior

and quantify cache miss rates and access amplification to guide the optimization of caching architectures. Moreover, the real-world workloads extend the low-level characteristics with practical results to provide a holistic view of the system under test.

The *LENS* [83] (Low-level profiler for Non-volatile memory Systems) benchmark addresses the properties and hardware characteristics of NVM, especially Intel Optane PMem. Unlike traditional benchmarks that evaluate performance at the system or software level, *LENS* emphasizes on the microarchitectural properties and design characteristics of NVM modules. Different microbenchmarks, including pointer chasing, overwrite, and stride tests, are designed to probe specific aspects of the system, such as DIMM-level buffers and caches, data migration policies, and key performance metrics like latency and bandwidth. Architectural insights from the benchmark can be used to optimize software as well as future NVM hardware design.

The *lmbench* suite [84] is a collection of microbenchmarks (detailed in Table 10) designed to investigate and diagnose performance issues across various system components such as processors, memory, and networks. It isolates performance bottlenecks from applications like databases and simulations into small reproducible microbenchmarks for simplified analysis while focusing on latency and bandwidth of data movement. A database of results allows for comparison across different operating systems and configurations.

PerMA-Bench [46] is a configurable benchmark framework that is designed to evaluate the performance of persistent memory. It can be used to measure performance metrics like bandwidth and latency under different configurable workloads. The provided in-depth performance analysis enables one to weigh up the cost-effectiveness of persistent memory systems.

pmemids_bench [85] is a benchmark suite (see Table 9 for a list of microbenchmarks) that aims for the analysis of data structures performance on Intel Optane PMem. While other existing benchmarks focus either on high-level system performance or low-level memory access, this suite addresses the mid-level with indexing data structures like **array lists**, **linked lists**, **hash tables** for storage applications. These data structures are tested under various conditions to draw conclusions for high-level applications build upon these.

PMidioBench [86] is a microbenchmark suite (see Table 11 for the full list) developed to study and analyze the low-level behavior of Intel Optane PMem. It focuses on various performance metrics such as latency, bandwidth, and the impact of different access patterns, with the goal of drawing a connection between low-level characteristics and high-level application behavior.

pmmeter [87] is a microbenchmark tool for Intel Optane PMem designed to analyze the synchronization costs associated with different memory access patterns and synchronization instructions such as barriers and cache flushes. It provides performance metrics like data transfer throughput, instruction throughput, and average latency, which can be compared to the DRAM performance as baseline in order to optimize applications for persistent memory.

The *PrIM* [88] Processing-In-Memory benchmark suite (see Table 13 for a list of microbenchmarks) focuses on the evaluation of memory-centric computing systems like processing-in-memory (PIM) architectures, especially on the first commercially available implementation – UPMEM. The suite consists of 8 microbenchmarks and 16 diverse memory-bound workloads from domains like linear algebra, databases, data analytics, and neural networks. The workloads assess various PIM capabilities (including computational throughput and memory bandwidth) to evaluate PIM architectures’ performance and scaling characteristics with the goal of identifying

⁵ *DAMOV* focuses on different kinds of memory-bound functions, where each function could be seen as an individual benchmark within this suite. Its high number of functions (144) and a shallow level of detail regarding the individual functions render an in-depth analysis impractical.

Benchmark	Venue	Ranking	Citations	Year	Cit./Year
CircusTent [76]	MEMSYS	—	9	2021	1.8
DAMOV [77]	IEEE Access	Q1	152	2021	30.4
DLBENCH [78]	CGO	A	16	2017	1.8
GORDON [79]	FCCM	B	15	2021	3.0
HMBench [80]	ISMM	A	39	2016	3.9
Hopscotch [81]	MEMSYS	—	30	2019	4.3
KernelBenchmarks.jl [82]	ISPASS	B	30	2021	6.0
LENS [83]	MICRO	A*	152	2020	25.3
Imbench [84]	ATC	A	1335	1996	44.5
PerMA-Bench [46]	PVLDB	A*	33	2022	8.2
pmemids_bench [85]	CCF	Q3	8	2022	2.0
PMIdioBench [86]	PVLDB	A*	99	2020	16.5
pmmeter [87]	BigComp	—	0	2023	0.0
PrIM [88]	IGSC	—	131	2021	26.2
Shuhai [5]	IEEE Computers	Q1	60	2022	15.0
SMOG [89]	CANDAR	—	0	2023	0.0
uFLIP-OC [90]	APSys	—	89	2017	9.9

Table 4. Overview of the publication venues for the 17 proposed memory benchmarks. The rankings are based on the SCImago Journal Rank (SJR) for journals and the CORE ranking for conferences. The table presents the first available venue classification following the publication of the corresponding paper. Citation counts are reported by Google Scholar as of July 2026.

suitable workloads, guiding software optimizations, and showing potential optimizations for future hardware designs.

Shuhai [5] is a tool analyzing the performance characteristics of High Bandwidth Memory (HBM) on Field-Programmable Gate Arrays (FPGAs) using a hardware-software co-design. The provided insights highlight different factors that affect HBM performance, such as address mapping policies and memory access patterns, allowing to build optimized FPGA designs. By comparing HBM of different generations and HBM with conventional DRAM, the benchmark results also show an evolution and differentiation of memory technologies.

SMOG [89] is a tool that mimics memory access patterns of real-world applications for analyzing heterogeneous memory systems, particularly NUMA and memory disaggregation. By supporting various combinations of access patterns, it enables the evaluation of their effect on data placement and migration strategies. During execution, the benchmark captures throughput as time-series data, enabling an in-depth evaluation of memory migration in a subsequent phase of analysis.

uFLIP-OC [90] is a benchmark to study I/O patterns for open-channel SSDs, which offer increased flexibility by exposing their internal architecture to the host system. This exposure enables direct management of I/O scheduling and data placement, allowing for fine-grained control over device performance. The benchmark helps in selecting an ideal point in the huge parameter space, to design more efficient data systems and applications.

5.2. Publication Venue Quality

Before comparing the benchmarks with respect to their technical characteristics, it is useful to assess the quality and diversity of the publication venues in which they were introduced. Since the proposed benchmark set is derived from a systematic literature review, the venue quality provides additional context regarding the scientific maturity and visibility of the underlying work. Table 4 therefore summarizes the publication venues of all 17 benchmarks and complements the benchmark descriptions from Section 5.1 with an assessment of the publication quality. For each benchmark, the ranking according to the SCImago Journal Rank (SJR) for journals and the CORE ranking for conferences is presented. The table presents the first available venue classification after publication

of the corresponding paper. For the two papers published in the proceedings of the VLDB Endowment (PVLDB), the corresponding CORE ranking of the International Conference on Very Large Data Bases (VLDB) is used.

As not all conferences have been classified by the CORE ranking, the number of citations reported by Google Scholar (July 2026) is additionally presented. To account for the systematic bias caused by older papers having had more time to accumulate citations, the citations per year are also presented.

In total, it can be seen that the publication venues are of relatively high quality on average. This can partly be explained by the fact that Google Scholar ranking mechanisms are directly influenced by venue ranks and indirectly by citation counts, which are often correlated with venue visibility and impact. However, publications from lower-ranked venues are also included. This indicates that our methodology successfully collected memory benchmarks from a broad range of academic venues and provides a comprehensive data basis for charting the benchmarking landscape.

5.3. Benchmark Characteristics

Table 5, further addressing RO₁, summarizes the common aspects of the previously presented benchmarks. For properties of suites or highly configurable benchmarks where a single value would not be sufficient for describing a characteristic, we use a “*”. Some papers do not mention every single aspect and do not provide source code for an easy retrieval of missing information, so we mark those entries with “?”.

5.3.1. Access Pattern

Memory benchmarks employ various access patterns (e.g., sequential, random, and strided reads and writes) or combinations thereof in order to capture the behavior of the target hardware under diverse application/workload scenarios. Some benchmarks, for instance Hopscotch [81], provide support for tunable access patterns. Access patterns differ fundamentally in terms of spatial and temporal locality and thus are important in analyzing caching and prefetching behavior with respect to the target hardware. Whereas sequential accesses mainly address the sustained bandwidth of a system, random access patterns highlight the latency. Strided access is not a complete scan over the memory, but its periodic pattern can

Proposed benchmark	Access Pattern									Language	Workload Granularity	Workload Size	Benchmark Type	Bandwidth	Latency	Energy Throughput	Runtime	CPU Bound	Mem Bound	Target Hardware	
	Seq.			Rand.			Strided														
	R	W	U	R	W	U	R	W	U												
CircusTent [‡]	-	-	*	-	-	*	-	-	*	C++	8B	GiB	BS	-	-	-	X	X	-	X	CPU+Memory/GPU
DAMOV	*	*	*	*	*	*	*	*	*	C/C++	?	?	BS	-	-	X	-	X	X	X	CPU+Memory/NMC/UPMEM
DLBENCH	X	X	X	-	-	-	X	X	X	?	B	KiB..GiB	BA	X	X	-	X	X	-	X	DDR/GDDR/HBM on GPU
GORDON	-	-	-	X	X	-	X	X	-	C++ API	64B..4KiB	KiB..GiB	BA	-	X	-	X	-	-	X	NVM
HMBench [‡]	X	X	X	X	X	X	X	X	X	C/C++	B	MiB	BS	-	X	X	-	X	-	X	Hybrid Memory Systems
Hopscotch [‡]	X	X	X	X	X	X	X	X	X	C/C++/ CUDA	B	KiB..GiB	BS	X	X	-	-	-	-	X	CPU+DRAM/MCDRAM
KernelBenchmarks.jl	X	X	X	X	X	X	-	-	-	Julia	64B..512B	GiB	BA	X	-	-	-	X	-	X	NVM+DRAM
LENS	X	X	-	X	X	-	X	X	-	Assy+C	B..KiB	B..MiB	BA	X	X	-	-	-	-	X	NVM (Storage)
lmbench [‡]	X	X	-	X	X	-	X	X	-	C	?	16MiB	BS	-	X	-	X	X	X	X	CPU+I/O+Memory
PerMA-Bench	X	X	X	X	X	X	-	-	-	C++	64B..4KiB	1GiB	BA	X	X	-	X	-	-	X	NVM/Optane
pmemids_bench [‡]	X	-	-	X	X	X	-	-	-	C/C++	B..KiB	MiB..GiB	BS	X	X	-	X	-	-	X	NVM/Optane
PMIdioBench [‡]	X	X	-	X	X	-	-	-	-	C/C++	B..KiB	?	BS	X	X	-	X	-	-	X	NVM/Optane
pmmeter	X	X	-	X	X	-	-	-	-	C	64B..256B	?	BA	X	X	-	-	X	-	X	NVM/Optane
PrIM [‡]	X	X	X	X	X	-	X	X	-	C	KiB..MiB	MiB..GiB	BS	X	X	X	X	X	X	X	NMC/UPMEM
Shuhai	-	-	-	-	-	-	X	X	-	SystemVerilog/ C/Tcl/C++/ Verilog/CUDA	32B..512B	KiB..MiB	BA	X	X	-	-	-	-	X	HBM on FPGA
SMOG	X	-	-	X	X	-	-	X	-	C++	Page	MiB..GiB	BA	X	X	-	X	-	-	X	Heterogeneous/Disaggr. Mem.
uFLIP-OC	X	X	-	-	-	-	-	-	-	C	KiB	?	BA	-	X	-	X	-	-	-	Open-Channel SSD

Table 5. Overview of proposed benchmarks: R(ead), W(rite), U(update); BA (Benchmark Application), BS (Benchmark Suite). Unknown entries are marked with “?”, ambiguous entries are marked with “*”, and benchmarks with additional tables in Appendix A are marked with “[‡]”.

typically be well predicted by prefetchers. The importance of the distinction between read/write/update lies in the fact that these cases have different caching effects, which is especially important for modern memory-coupled systems.

5.3.2. Workload Granularity/Size

Workload size and workload granularity can be used to control the level of the memory/storage hierarchy where the data resides, which can be used to force cache evictions. Furthermore, workload size is useful in revealing scalability aspects such as bandwidth saturation as well as hardware-specific features, such as write amplification due to the write-combining buffer of Optane DCPMM. The granularity/size ratio also affects the probability of concurrent accesses in multithreaded scenarios (depending on the specific algorithm). A small granularity in combination with a large working set has a lower risk of access contention. Another aspect directly influenced by the granularity is alignment. Misalignment of data can drastically decrease application performance due to high pressure on the cache coherency protocols.

5.3.3. Metrics

Metrics are an important component of benchmark design. In order to compare different configurations, different hardware setups, or to evaluate optimization techniques, a deterministic and reproducible number is required that quantifies a benchmark run. Depending on the use case and the target hardware, different metrics could be used. Bandwidth and latency directly focus on the access characteristics of memory hardware. Throughput and runtime are further performance metrics, but with the focus shifted in the direction of application performance. Energy measurement adds another dimension and optimization target to the benchmark execution.

5.3.4. Hardware

The *Target Hardware* column in the table categorizes each benchmark by the specific memory system or technology it targets. This is necessary because modern computing systems encompass a wide range of memory types, technologies, interfaces, and hierarchies, each with unique characteristics that require tailored

evaluation methodologies (see Table 1). To address RO₂, we highlight the peculiarities of the memory technologies from a benchmarking perspective (see Section 2.1 for background details), linking them to how their respective benchmarks were specifically designed to target the memory technologies.

HBM. Since the design goal behind HBM is achieving very high parallelism at the expense of increased latency to serve many parallel computation units like GPUs, many-core CPUs or just simple tasks with very high sequential throughput, benchmarks aim to quantify such aspects of HBM or to compare HBM with DRAM in different workloads, not only for conventional CPUs [81], but also for accelerators like GPUs [78] and FPGAs [5]. Furthermore, benchmarks like DLBENCH [78] investigate which data to place in HBM/DRAM and when. Depending on the access pattern and layout, optimal access of data varies from one memory type to another. Similar is the strategy of moving data between different memory types. This is important, especially for emerging heterogeneous memory systems where data placement plays a crucial role in performance [80].

PCM/Optane. Apart from latency and bandwidth, peculiar properties of PCM/Optane that are targeted for benchmarking include asymmetric read/write performance [46, 85, 86]. Another important aspect is persistence. Depending on the implementation, different memory synchronization instructions and other mechanisms are used to ensure consistency and durability of persistent data. These mechanisms differ in complexity, performance, and hardware costs. Benchmarks can be used to quantify the performance and allow for a well-founded decision making. An example is pmmeter [87], which focuses on the performance impact of memory synchronization instructions on Optane. Moreover, benchmarks like GORDON [79] and LENS [83] investigate the microarchitectural features of Optane such as DDR-T queuing and scheduling; on-DIMM buffer capacity, organization, and entry size; data migration, data interleaving and block management.

PIM/NMC/UPMEM. There are a couple of aspects of PIM/NMC/UPMEM that are interesting for benchmarking. Firstly, benchmarks like DAMOV [77] investigate how the performance and energy consumption of numerous applications vary between memory-centric architectures (e.g., PIM systems) and traditional compute-centric architectures. That enables determining the

suitability of applications for NMC using metrics such as temporal locality, arithmetic intensity, last-level CPU cache misses per kilo-instruction, and the miss ratio between the LLC and L1 cache. Secondly, PIM increases parallelism since the numerous PIM cores that are placed close to the memory banks are capable of running computations concurrently. Hence, another interesting aspect to benchmark is the overall computation throughput achieved by PIM. Thirdly, PIM cores are close to the memory and offer high bandwidth, thereby bringing potential performance benefits. Benchmarking the aggregate bandwidth of PIM is thus important. Fourthly, as PIM helps in addressing the memory wall problem in that it also narrows the gap between compute instructions and memory access latency, benchmarks also target the evaluation of latency reduction achieved through PIM. Fifthly, the memory-centric paradigm of PIM mitigates data movement bottlenecks as data does not have to be transferred to the CPU. Consequently, this brings energy-saving benefits, especially for data-intensive workloads. The PrIM benchmark [88] targets the above, that is, benchmarking compute throughput, bandwidth, latency, and energy consumption. Moreover, it also compares application performance for different target platforms, like CPU, PIM, and GPU.

SSD. Benchmarks typically highlight the performance difference between HDDs and SSDs in practical applications, since SSDs conceptually service the same tasks as HDDs, but provide much better access latency and parallelism. SSDs often use different write modes internally to speed up sequential writes up to a capacity limit. Synthetic benchmarks can reveal those internals to weigh up the suitability of a device for a specific application. Benchmarks like uFLIP-OC [90] investigate the characteristics of open-channel SSDs from the angle of suitability of I/O patterns and their impact on performance.

5.4. Discussion

From our evaluation, and to further expand on addressing RO₁ and RO₂, we draw results with regard to two viewpoints. Firstly, from the perspective of a *benchmark user* (Section 5.4.1), the structured overview helps in selecting appropriate benchmarks for a purpose. Secondly, from the perspective of a *benchmark researcher* (Section 5.4.2), our study helps to determine which non-functional aspects of memory technologies are crucial for applications and, therefore, particularly relevant for benchmarking. In addition, it also helps in identifying missing aspects that provide further opportunities for future benchmark designs. Finally, we also discuss the effectiveness of the benchmarks regarding their analysis target (Section 5.4.3) and the applicability of benchmarks to upcoming technologies in an evolving system landscape (Section 5.4.4).

5.4.1. Perspective of a Benchmark User

Table 5 facilitates the selection of an appropriate benchmark for a memory system from a benchmark user’s perspective. Beyond that, on a conceptual level, several patterns emerge from the data. First, benchmark specialization correlates with technology availability: Optane, as the most mature emerging technology, has the most tailored benchmarks (7 of 17), while technologies still in early stages (RTM, disaggregated memory) have few or none. This suggests that benchmark development follows, rather than precedes, hardware prevalence, resulting in potential delays of systematic evaluation of new technologies.

Second, there is a clear split between interface-based and architecture-specific benchmarks: benchmarks targeting the CPU address space (e.g., LENS, PerMA-Bench) are transferable across memory technologies, while those exploiting PIM-specific features

(e.g., PrIM) are not. Users should be aware of this distinction when selecting benchmarks for evaluation.

Third, microbenchmarks dominate over workload-level benchmarks: most proposed benchmarks focus on low-level access patterns rather than end-to-end application behavior, reflecting the field’s emphasis on hardware characterization over system-level evaluation.

5.4.2. Perspective of a Benchmark Researcher

Benchmarking is essential for quantifying and improving the performance of hardware and software systems. From our analysis of the proposed benchmarks, several conclusions can be drawn for future benchmark design.

Most benchmarks are written in low-level languages such as C/C++. This choice is driven by the need for direct control over hardware, minimizing intermediate layers and avoiding excessive abstraction that could alter application behavior. Benchmarks targeting configurations beyond conventional CPU+DRAM setups, such as GPUs, use the provided platform-specific development kits like CUDA and OpenCL.

The analyzed benchmarks encompass the most common access patterns – sequential, random, and strided – across read, write, and update operations. They operate on memory sizes ranging from a few bytes to multiple terabytes in order to push the systems under test to their limits. The granularity of individual memory operations typically remains within a few bytes to precisely identify performance bottlenecks near the hardware level. Most benchmarks within our survey focus on raw performance using the metrics bandwidth, latency, throughput, and runtime. A few of them also include energy as an additional metric. However, with the growing attention on green computing, researchers need to give more relevance to the energy properties of emerging memory technologies, especially in benchmark design. Given that our focus is on memory benchmarks, most of the benchmarks are bound by memory performance, while a few benchmarks are also affected by the computing performance.

Our surveyed benchmarks cover a broad spectrum of hardware configurations, reflecting the diversity in memory research. They address conventional DRAM in today’s systems, tackle memory challenges associated with GPU offloading, and explore novel memory technologies such as non-volatile memory, near-memory computing, and hybrid-memory systems with heterogeneous or disaggregated memory architectures.

Our survey also reveals some aspects that are underrepresented or entirely missing in current benchmarks. No benchmark explicitly addresses the fairness of memory controllers in terms of request prioritization, a critical aspect for equitable resource allocation and usage. Few benchmarks consider the effects of multiple memory domains (multi-NUMA, heterogeneous memory, disaggregated memory). This is important for future research, as memory specialization and virtualization are necessary to meet the demands of big-data applications at moderate costs. The increasing complexity of CPUs and computing systems has led to a rise in potential security vulnerabilities and side-channel attacks. While most issues are addressed with software mitigations, their performance implications are not covered.

The gaps we identified can be classified into three categories of systematic limitations:

- *Limitations of scope:* What is not measured. Energy efficiency, fairness, and security-mitigation overhead are absent from nearly all benchmarks, reflecting a historical focus on raw performance rather than holistic system evaluation.

- *Limitations of methodology:* How measurements are taken. While developers already try to minimize interferences with the use of low-level languages, there is a lack of benchmarks that further isolate microarchitectural memory-controller, interconnect, and prefetcher effects mean that results often reflect platform-specific artifacts rather than intrinsic device properties.
- *Limitations of generalizability:* What conclusions can be drawn. The overfitting of benchmarks to specific technologies (especially Optane) and the rarity of multi-NUMA or disaggregated memory benchmarks limits the transferability of findings across architectures.

The proposed benchmarks represent the current state of the art in memory benchmarking and provide a solid foundation for evaluating modern memory architectures. However, addressing the underrepresented aspects in future benchmark designs will further enhance the benchmarking landscape, ensuring more complete and accurate performance evaluations.

5.4.3. The Effectiveness of Benchmarks

The proposed benchmarks are generally comprehensive and accurate in how they benchmark certain characteristics of target hardware. For instance, with regard to Optane PMem, PerMA-Bench [46] includes aspects of PMem that are often overlooked, e.g., DIMM size, number of DIMMs, DIMM power budget, memory bus speed, persist instructions and their effects in the presence of eADR, the variation of PMem characteristics under changing workloads and additional characteristics of the second-generation Optane. It is effective in capturing comprehensive behavior of PMem by providing reconfiguration of workloads, which influences the behavior. In addition to user-customized workloads, the workload space includes predefined workloads with fundamental access patterns like sequential, random and strided accesses, mixed read/write accesses, hybrid PMem-DRAM accesses. In contrast, PMidioBench [86] decouples PMem from Optane and identifies characteristics peculiar to PCM that do not generalize across PMem technologies, while also showing how some effects arise from system configuration rather than the PMem technology itself.

Although many benchmarks are skewed towards Optane NVM, PrIM [88] is comprehensive in its analysis of UPMEM PIM. It has microbenchmarks specific to UPMEM PIM’s architecture for measuring bandwidth, latency and compute throughput. It also contains a suite of memory-bound workloads from different application domains for benchmarking the performance, scalability and energy consumption of the UPMEM PIM. The microbenchmarks and workloads cover different memory levels in the PIM hierarchy, different access patterns, different compute operations, different data types and different communication patterns. Another example is DAMOV [77], which is based on a simulator-driven framework to explore data-movement bottlenecks under controlled variations of cache hierarchies, prefetchers, core counts, and near-data processing designs. While this enables high reproducibility and methodological isolation the abstraction of the memory controller, interconnect, and DRAM devices means that reported values are influenced by modeling assumptions rather than hardware-level behaviors.

Several benchmarks aim to isolate architectural and software-stack effects. CircusTent [76] provides a cross-platform view of atomic-operation behavior, but its measurements are influenced by CPU microarchitecture and runtime preferences/choices like cache coherence, scheduling, the OpenMP memory-allocation and atomic-primitive implementations. Therefore, the reported throughput often reflects cache and coherence-level effects rather than properties intrinsic to the underlying memory medium.

Similarly, DLBENCH [78] isolates layout-sensitive behavior using synthetic GPU kernels. However, the observed effects remain tightly coupled to compiler optimizations, coalescing behavior, thread/workgroup sizing, and host-device data movement that complicates the attribution to the memory itself.

With regards to accuracy of the benchmarks, Shuhai [5] benchmarks HBM on FPGAs and provides a more accurate benchmarking of HBM with custom hardware logic. This approach avoids interference effects typically present in CPU-based benchmarks, such as those caused by multi-level caches. GORDON [79] follows the same principle for Optane memory by employing custom hardware logic to profile Optane directly, thereby eliminating distortions introduced by CPU-based infrastructures, including processor microarchitecture and system software effects.

HMBench [80] and Hopscotch [81] both target end-to-end characterization on real platforms. HMBench captures system-level behavior on KeyStone II, However, the lack of hardware performance counters limits the visibility into cache, interconnect, and controller effects. Hopscotch controls for compiler and caching artifacts through pointer chasing and warm run exclusion; however, its results remain sensitive to write-allocation policies, prefetching, controller queue depth, and platform-specific features, which can cause reported traffic and latency to reflect controller and OS interactions rather than the memory device properties.

KernelBenchmarks.jl [82] and LENS [83] provide fine-grained visibility into Optane-equipped machines using hardware counters and carefully designed microbenchmarks. KernelBenchmarks.jl exposes effects such as access amplification, tag check, and DRAM cache interactions on Intel’s 2LM design, though the reported bandwidth and latency reflect the combined behavior of the controller and DRAM cache rather than the NVRAM media alone. It also remains sensitive to access size, alignment and interleaving. LENS reduces different CPU-related noises through kernel-mode probing and non-temporal access. Still, its measurements are influenced by controller buffering, request scheduling, and ordering costs.

Finally, lmbench [84] provides latency measurements through careful control of cache effects and repeated sampling, making it effective for identifying cache hierarchies and access plateaus. However, its focus on clean-read behavior and reliance on user-space and OS-level primitives may be affected by system-level factors such as caching and OS scheduling, which can influence the interpretation of intrinsic memory performance.

Synthesizing these insights into three trade-off axes reflect the different optimization directions in benchmark design:

- *Reproducibility vs. Accuracy:* Simulator-driven benchmarks (DAMOV) offer controlled, reproducible environments but abstract away hardware details. Hardware-based benchmarks (Shuhai, GORDON) provide higher fidelity but potentially introduce platform-specific effects. No single benchmark can achieve both. The choice depends on whether it targets isolated device properties or system-level behavior.
- *Specificity vs. Generality:* Benchmarks tightly coupled to a specific technology (e.g., PMidioBench for Optane, PrIM for UPMEM) capture its idiosyncrasies, but they cannot generally be applied to evaluate other architectures. More generalized approaches (e.g., LENS), target slightly higher application-level properties by being designed against a common interface (the virtual address space in the case of LENS), but trade specificity for broader applicability. The optimal balance depends on whether the goal is deep characterization of one technology or comparative evaluation across technologies.

Count	Benchmark	Type	Domain(s)
122	SPEC	Benchmark suite	HPC, FS, ML&DM, IP, SP
85	Microbenchmark	-	-
77	YCSB	Benchmark application	DB
62	STREAM	Benchmark application	HC
62	PARSEC	Benchmark suite	DB, ML&DM, HPC, SP, IP
59	TPC	Dataset	DB
50	NPB	Benchmark suite	HPC
33	Rodinia	Benchmark suite	HPC, GP, ML&DM, IP
31	GEMM	Algorithm	HPC
29	FIO	Benchmark application	FS, OS

Table 6. Top-10 used annotated benchmarks with their type and domain(s). Used abbreviations in the *Domain* column: **HPC**: High performance computing, **FS**: File Systems, **ML**: Machine learning and Data mining, **IP**: Image processing, **SP**: Signal processing, **DB**: Database, **GP**: Graph processing, **OS**: Operating systems

• *Isolation vs. Realism*: Microbenchmarks isolate individual hardware effects (e.g., write-combining buffer behavior) but may not reflect realistic behavior in a broader application context. Application-level benchmarks capture real-world interactions but are prone to multiple hardware and software effects. Comprehensive evaluation requires both, but few papers in our corpus achieve this balance.

Overall, it depends on the research question, which direction to follow. In most cases, a combination of different benchmarks achieves the best coverage of the sketched evaluation space.

5.4.4. Applicability of Benchmarks to an Evolving System Landscape

As discussed above, benchmarks target a specific purpose and a specific technology (e.g., the specialties of Intel Optane as the first commercially available incarnation of persistent memory). Nevertheless, the benchmarks can also be applied to all other memory technologies featuring the same *interface*. In this context, interface does not mean the electrical/logical connection on the hardware side, but the way software interacts with a memory technology. One common interface in this regard is the *physical address space* of the CPU. All memory that appears in this address space can be accessed with native load/store instructions. As a result, all benchmarks that have been designed against this interface can be applied to *all* technologies backing this interface. For example, all Optane benchmarks can also be used to test and compare the characteristics of DRAM, CPU-attached HBM, or (disaggregated) CXL-Memory, despite being originally designed for a specific memory type.

In contrast, some benchmarks have been explicitly designed to target the *combined* usage of different memory types in a heterogeneous/hybrid memory system (e.g., HMBench, SMOG). Those benchmarks do not actually have the memory as *device under test*, but the memory management layer that makes placement decisions to balance the characteristics of multiple memory types. New memory technologies – as long as they can be integrated into the same memory management layer – can be evaluated using the existing benchmarks.

Finally, a last category are benchmarks addressing a specific memory technology that fundamentally change the way how we use them (e.g., PIM). Those memories require tailored benchmarks to utilize their potential.

To turn these findings into actionable points, we recommend that benchmark developers clearly distinguish between (a) interface-level benchmarks that test the memory’s externally visible behavior and (b) architecture-specific benchmarks that probe internal implementation details. Both have value, but their distinct purposes

should be explicitly stated to allow for informed benchmark selection and adaptation of benchmarks to changing hardware specialties.

6. Used Benchmarks

Since benchmarking is the only way to compare approaches or systems, there is a zoo of benchmarks that are used for the quantitative comparison. In order to shed further light into how researchers benchmark modern memory and storage systems using existing benchmarks, we further characterized, based on our objective RO₃, the extracted benchmarks from our literature corpus with the *types* and *domains* described in Section 4.4. To exemplify what the characterization looks like, we show an excerpt of the Top-10 used benchmarks with their annotation in Table 6. For example, the Rodinia Benchmark Suite [91] was referenced 33 times in the works filtered in Phase II and is a benchmark suite covering benchmarks from the domains high-performance computing, graph processing, machine learning & data mining, and image processing.

While the characterization based on benchmark type and domain (i.e., our shared final annotated list will be shared upon publication) is a valuable contribution already (due to the high human effort in cleaning and annotating), we aim to give further insights into how researchers benchmark and how the usage of benchmarks evolved. To this end, we defined five major analysis questions (AQs) that we answer with our gathered data.

The following AQs range from simple and descriptive analyses (i.e., AQ₁ & AQ₂, which characterize the usage based on the abundance of types and domains) to more complex interpretive analyses (i.e., AQ_{3,1} & AQ_{3,2}, which extract patterns in our corpus). Moreover, AQ₁ addresses RO₃, while AQ₂ and AQ₃ address RO₄. Finally, AQ₄ describes the insights about the benchmarking domain that we gathered when annotating, cleaning, and consolidating the corpus and includes important limitations. Notably, our corpus, and hence the following investigation, is based on the seed of memory and storage systems and deliberately does not analyze the whole computer science research field.

AQ₁ *What is the most used benchmark type and domain?*

This analysis sheds light into how many benchmarks have been found for a specific benchmark type and domain. Our expectations are that there is an imbalance of types and domains, where the amount of some overshadows the amount of others. For instance, due to the focus of our literature corpus, the main focus will be on benchmarks applications and benchmark suites compared to datasets.

AQ₂ *Is there a prevalence regarding certain domains, especially over time?*

Domains’ popularity changes based on research community and

societal impact (e.g., crises, political and business decisions). Hence, it would be interesting to analyze how the popularity (i.e., usage of benchmarks from a certain domain) changes over time.

AQ₃ *How are modern memory and storage devices benchmarked?*

To answer this AQ, we focus on the comprehensiveness of used benchmark types and domains, from which we can derive the following two analysis questions.

AQ_{3.1} *Are benchmarks from different domains used?*

In order to have a versatile benchmarking, it is advisable to also use versatile benchmarks from different domains. This analysis presents how diverse the used benchmarks are in terms of their domain per paper.

AQ_{3.2} *Do papers benchmark their systems with different benchmark types?*

In a perfect paper, we would assume that researchers first benchmark their memory or storage systems with certain algorithms before going into deeper investigations of applications or benchmarks with diverse access patterns and access granularities. Hence, we investigate how many papers use different types of benchmarks to compare their contribution to the state of the art.

AQ₄ *What challenges did we identify when surveying benchmarks for memory and storage systems?*

The last analysis concerns a qualitative discussion about how researchers benchmark their systems and how careful they are in the documentation thereof.

6.1. AQ₁ – Most Used Benchmark Types and Domains

To investigate how many benchmarks can be found in our dataset for each type and domain, we created a histogram based on our annotated data entries. The histogram in Fig. 3 shows for each domain how many different benchmarks of a specific benchmark type have been found. Based on the data in our histogram, we draw the following conclusions from the three directions of benchmark types, benchmark domains, and type and domain correlation.

6.1.1. Benchmark Types

From the benchmark types, it is clearly visible that algorithms constitute the majority of benchmarks (55 out of 212) in the corpus, which shows that researchers still use many standalone algorithms in their tests since algorithms have a rather predictable access pattern and low complexity. Still, 48 different benchmark suites have been used, which is a good sign since it testifies that papers also care about a diverse set of benchmarks than a single benchmark to test their approaches in the memory and storage domain. Notably, there are only 3 application and algorithm suites in our corpus (i.e., AMD APP SDK, SuiteSparse⁶, and the NERSC application suite⁷), which are considerably often used (e.g., SuiteSparse being on place 31 with 14 usages).

6.1.2. Benchmark Domain

Considering the benchmark domains, the HPC domain is by far the leading domain from which benchmarks are drawn when benchmarking modern memory and storage devices. They are closely followed by the ML & Data Mining (46) and database (40) domain. A possible explanation for this leaderboard can be found when answering AQ₂. On the contrary, there are only a few benchmarks

from the embedded systems or hardware characterization domain. An explanation is that both domains rather focus on special hardware and resource-efficient processors, which does not meet our focus of the corpus based on our memory-focused search terms. Furthermore, due to the diversity of hardware and use cases, it is harder to architect common benchmarks.

6.1.3. Insights Between Type and Domain

Regarding the distribution of types across domains, there are four interesting insights that we gained from the investigation. First, especially in the HPC domain, benchmark suites are predominantly used, showing the versatility and maturity of this domain. Second, the majority of algorithms come from the signal processing, graph processing, and ML & data mining domain. While some of these found their way into suites (e.g., as part of AMD APP SDK), more efforts should be undertaken to transfer them into suites. Third, applications come mostly from the database domain or HPC domain (i.e., using database systems or HPC applications as benchmarking applications), showing that these domains are important for contributing complex workloads. Fourth, datasets and neural network architectures usually come from the domain of ML & data mining, which is an expected outcome.

6.2. AQ₂ – Evolution of Benchmarking Over Time

An important dimension of analyzing the benchmarking of modern memory is to investigate how benchmarking practices have evolved over time giving deeper insights into how specific types and domains were used for benchmarking. Hence, we used our annotated corpus to identify how papers that were published in a specific year of the last 20 years are using different domains or types of benchmarks.

6.2.1. Domains Over Time

Looking into the domains of benchmarks over time in Fig. 4a, there is a quite even distribution of domains visible. However, considering the last 7-9 years, especially HPC, ML & Data Mining, and Database are overruling the other domains. This is due to their recent popularity in research. For instance, due to the big data hype⁸, databases show an increasing usage since 2013, peaking in 2018 and 2019. This peak is associated with the announcement and introduction of Intel's Optane DCPMM, where database systems are striving for exploiting the persistence property (more than half of the papers of that time using database benchmarks also concern non-volatile memory). Furthermore, this figure illustrates that the use of real-world benchmarks – i.e., from the high-performance computing domain – has generally been more prevalent, as these benchmarks assess system performance using real-world applications. However, with the rise of artificial intelligence and machine learning, this trend is shifting, as a noticeable change since 2022 can be seen in the figure. Given the recent hype of artificial intelligence applications, this trend is also visible in the increased usage of those as benchmarks for modern memory systems in the past 5 years.

6.2.2. Types Over Time

Considering the benchmark types over time in Fig. 4b, there is a visible tendency toward using benchmark suites in most of the displayed years. This shows that most papers comprehensively benchmark their approaches using state-of-the-art benchmarks. However, benchmark applications and algorithms also show strong usage numbers, which can sometimes overrule benchmark suites. The reason behind this shift could be the recent surge in

⁶ <https://github.com/DrTimothyAldenDavis/SuiteSparse>

⁷ <https://docs.nersc.gov/applications/>

⁸ <https://www.gartner.com/en/documents/2571624>

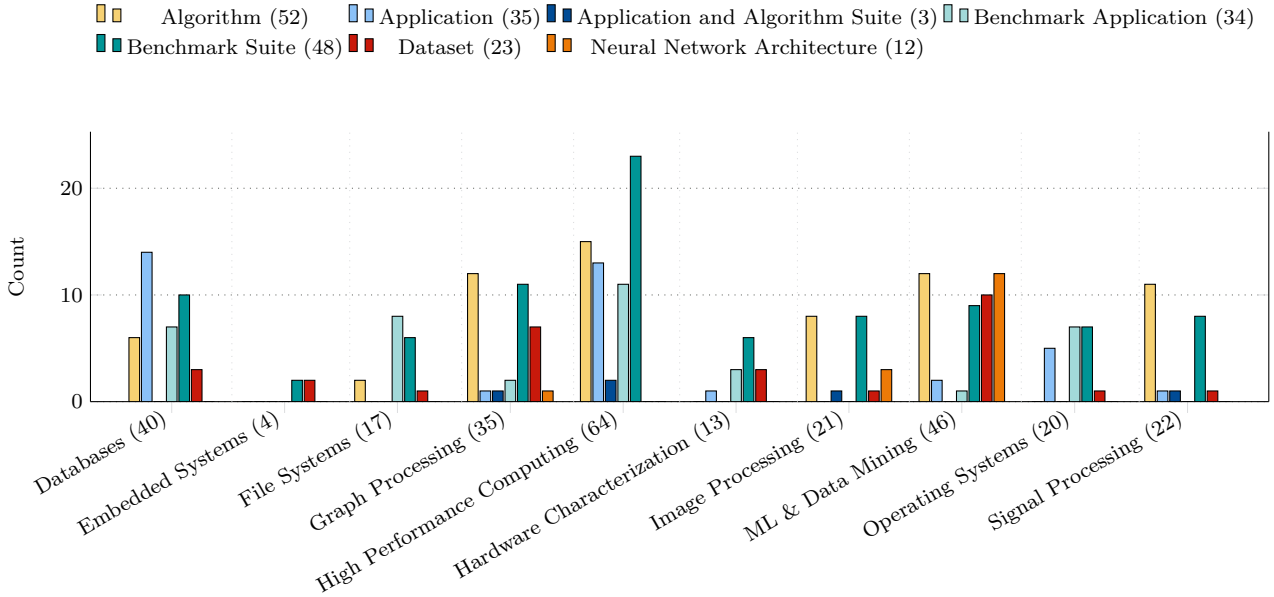


Fig. 3. Histogram showing for which domain which types of benchmarks have been found. Numbers in parentheses reflect the total group counts for each domain or type. Notably, each benchmark only has a single type but can belong to several domains (especially suites), which leads to a higher sum of benchmarks in all domains than actually present.

emerging architectures designed specifically to accelerate certain computational kernels [92, 53]. Interestingly, especially in recent years, neural network architectures are getting traction due to their current hype.

Notably, the distribution is highly impacted by the available HPC benchmarks because they constitute the highest share not only among the investigated benchmarks, but also their usage.

6.2.3. Insights for AQ_2

Overall, our investigation of AQ_2 has shown that the usage of certain benchmarks in our corpus is first and foremost focused on HPC workloads, which are apparently mostly used when benchmarking modern memory devices. Second, specific technical hypes influence what benchmark domains and types are prevalently used.

As a result of the investigation, it is evident that also the popularity of benchmarks is dependent on what domain is currently actively researched. Furthermore, these trends show that modern memory is frequently tested and revisited for testing the usability for a certain domain.

6.3. AQ_3 – Benchmarking Practices

Benchmarking practices vary between domains – what is typical in one domain could be an overly ambitious effort in another domain. Still, researchers (should) aim for the most comprehensive benchmarking. Hence, we investigate in how far these aims are also implemented based on the usage of benchmarks from different domains and of different types. Of course, the gold standard is to use benchmarks with different complexities (from simple algorithms to complex applications).

6.3.1. $AQ_{3.1}$ – Using Benchmarks from Different Domains

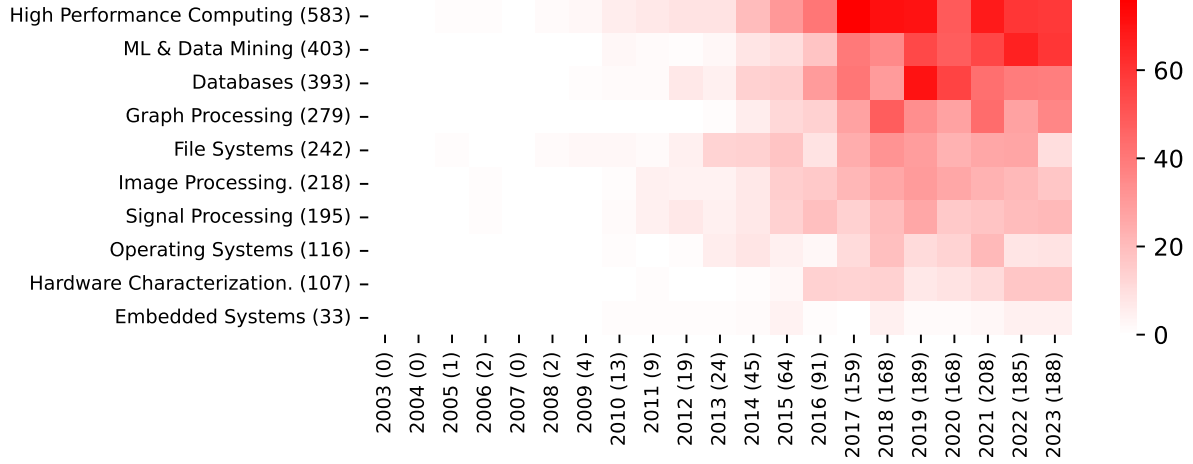
As a first indicator of how extensively the community benchmarks their systems, we investigate the number of domains that are used

inside each paper. We plot a bar chart in Fig. 4 that shows the number of papers that use between 1 and 9 (almost all) domains. Of course, there is a high number of papers using a single domain (roughly 35%). Although this is not a good practice, a single benchmark domain is mostly used because of the specific domain for which a system is designed. A positive finding is that the biggest share use two or more benchmark domains (roughly 65%) and almost the same amount of papers use benchmarks from three or more domains than those that use a single benchmark domain (275 papers using benchmarks of a domain compared to 272 using three or more benchmark domains). Interestingly, there are six papers with 8 or 9 benchmark domains used. Most of these papers reach this high number due to the usage of several benchmark suites incorporating benchmarks from several domains (e.g., Rodinia [91] with 4 domains, SPEC [74] with 2 domains, and PARSEC [93] with 5 domains), which shows that benchmark suites are ideal and well-renowned candidates for comprehensive benchmarking. Notably, a benchmark has 1.3 domains on average, while benchmark suites have 1.9 domains on average, showing their versatile use cases from numerous domains.

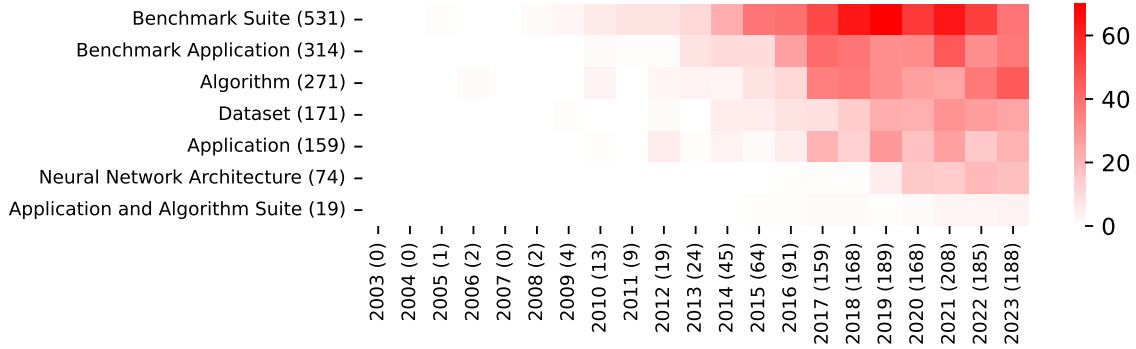
6.3.2. $AQ_{3.2}$ – Using Different Benchmarking Types

A second indicator for how researchers benchmark their systems is the types of benchmarks they are using. We argue that extensive benchmarking needs benchmarks of different kinds. While simple algorithms give in-depth insights for a small set of instructions, applications challenge the interplay of the hardware devices as a whole, which is why both have their merits and pitfalls and should be used in concert for a comprehensive benchmarking.

In Fig. 5, we review how many papers use which combination of benchmark types. Interestingly, the majority of papers use benchmark suites to benchmark their systems, closely followed by using benchmark applications, data sets, and algorithms. While using benchmark suites is a good practice, the usual



(a) Heatmap showing what domain of benchmarks have been prevalently used over time.



(b) Heatmap showing what types of benchmarks have been prevalently used over time.

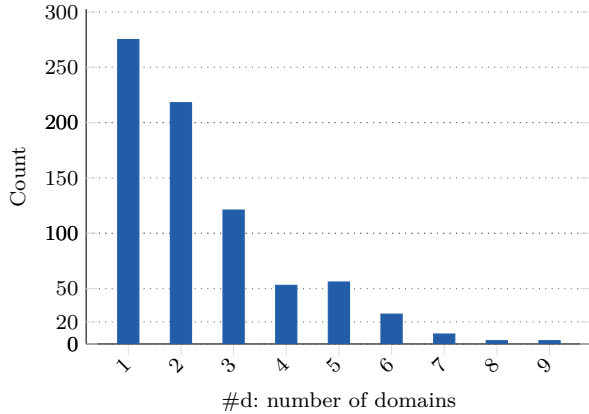


Fig. 4. Distribution of papers based on the number of domains used for evaluation. The figure illustrates the count of papers utilizing benchmarks from #d different domains.

benchmark suites (as also seen from those in Section 5) comprise benchmarks of the same granularity and complexity. That is why the

desired mixture of small and large-scale benchmarks (ranging from algorithms to applications) is often missing. There are numerous papers in our corpus that follow such a favored experimental design, which is visible in the long left tail, where several benchmark type combinations are prefixed with 'AL' and incorporating further benchmark types. Summing up the papers that use an algorithm plus a benchmark suite, benchmark application, neural network or application (i.e., evaluating the expression $AL(BS|BA|NN|AP)^+$) yields 62 papers. While this is a considerable fraction, of course there is room for improvement, especially given that 551 papers use a single benchmark type only.

6.3.3. Insights for AQ_3

Overall, this analysis has shown that researchers already use benchmarks from several domains and of diverse types, which is a very positive sign. Furthermore, especially due to the availability of respective benchmarks, there is no excuse for researchers using only a single benchmark type or domain. Hence, we demand that systems are extensively benchmarked in the future, which is an essential criterion for the reviewing process.

6.4. AQ_4 – Discussion of Challenges

During the process of analyzing the annotated benchmark list, we encountered several challenges. These stem, on the one hand, from

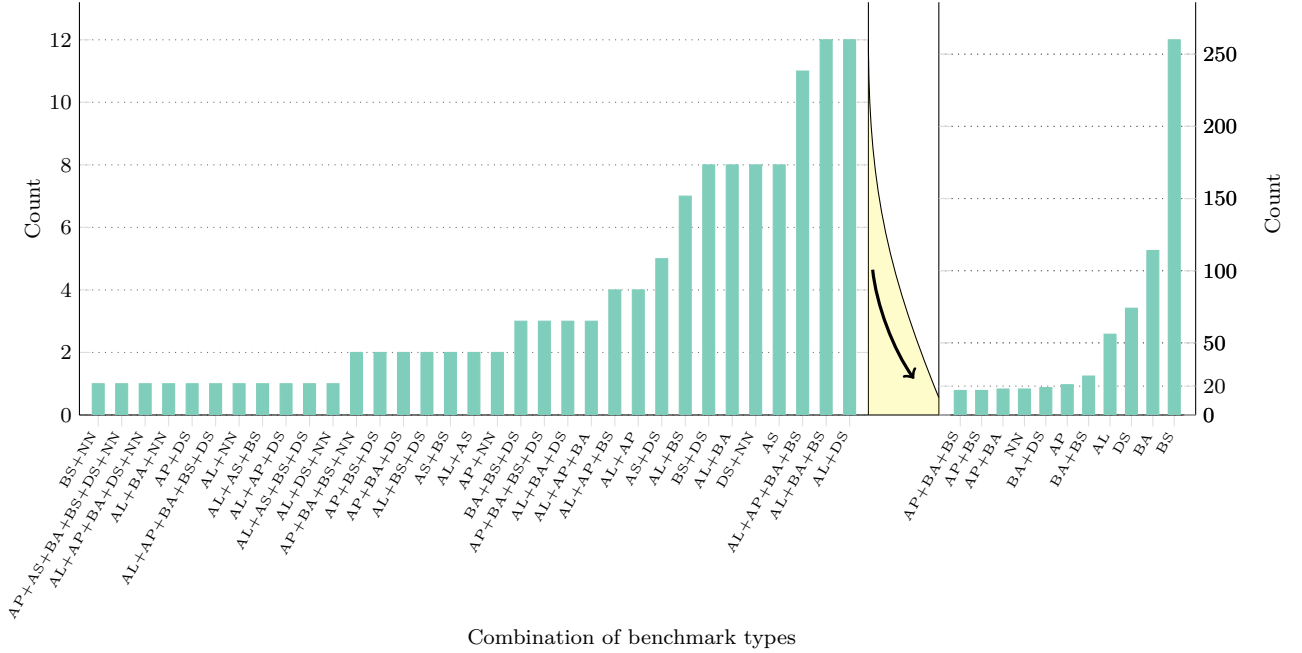


Fig. 5. Distribution of benchmark type combination across papers. The figure shows the number of benchmarks that utilize specific sets of benchmark types. Used abbreviations in the figure: **AL**: *Algorithm*, **AP**: *Application*, **AS**: *Application & Algorithm Suite*, **BA**: *Benchmark Application*, **DS**: *Dataset*, **NN**: *Neural Network Architecture*

the diverse origins of the data and also from the circumstance that multiple contributors collaboratively compiled the list, leading to inconsistencies due to human error and ambiguities inherent in the benchmarks themselves. In order to mitigate these problems, we annotated the type and domain with three persons and also meticulously revisited most of the corresponding papers, such that a solid judgment could be made.

Another main challenge was identifying a parent identifier for a set of benchmarks that have a lot of similarities. For instance, the SPEC benchmark suite exists in numerous variants, each updated over time and targeting distinct systems, such as CPUs, cloud environments, Java applications, and graphics workloads. While these sub-suites focus on specific aspects of performance, they often overlap significantly in workload type and domain. To address this, we created parent entries for closely related variants (i.e., the SPEC Benchmark Suite) while keeping dissimilar benchmarks, such as SPECstorage and SPECvirt, in separate entries.

Another challenge was the renaming of benchmarks over time, such as SuiteSparse, formerly known as the University of Florida Sparse Matrix Collection. Similarly, certain datasets or workloads (e.g., those in the SNAP collection) were sometimes part of larger collections but not explicitly referenced in their original descriptions.

Additionally, some benchmark suites, like HPCC, include widely used benchmarks such as STREAM and GUPS. When we identified that benchmarks were cited or used independently more than three times outside their parent suite, we treated them as separate entries rather than as part of the suite. These challenges required careful consideration and systematic adjustments to ensure accurate categorization and meaningful analysis of the benchmark data.

7. Discussion and Threats to Validity

The results presented in this survey should be interpreted in light of several methodological choices and inherent limitations. As with any

large-scale literature analysis, decisions regarding search strategy, screening procedures, normalization, and categorization introduce potential sources of bias and uncertainty. In this section, we discuss the main factors that may influence corpus composition, benchmark classification, and the interpretation of observed trends. We also outline the scope boundaries of our study, and assess the extent to which the findings are transferable beyond the analyzed dataset and to future memory technologies.

7.1. From Keywords to Corpus Coverage

The keyword list in Section 4 seeds the automated search for benchmark-related papers, was chosen to span a broad range of modern memory technologies and topics. It does not function as an exclusivity filter, nor does it imply equal depth of coverage for every technology in the resulting corpus. The search includes the full set of results for each term (not just a narrow first page), so papers can enter the corpus through related terminology even when a technology is not explicitly listed among the keywords. As a result, the depth of discussion reflects the literature that was retrieved and characterized rather than an a priori prioritization. For example, *Memristors* was not part of the keywords, yet papers [94] and [95] that exist in the search result are using such technology. Although we limited our study to papers from Google Scholar (as reasoned in Section 4.1), our approach could be applied to other scholarly search engines with similar coverage (e.g., Semantic Scholar, Scopus, or Web of Science), albeit with different indexing and metadata behavior.

7.2. Screening and Labeling Subjectivity

The screening and labeling steps required manual judgment. Automated analysis (e.g., using AI tools) is error-prone for this task because papers vary widely in format and terminology, and determining whether a paper truly targets the right context often

requires supervised checking. For this reason, we performed manual screening. In order to reduce human subjectivity, we set up strict guidelines for us authors as annotators for Phase II – Prefiltering. Furthermore, several authors characterized a single benchmark in the characterization in Phase IV, and a consensus had to be formed after a discussion of each benchmark. Hence, we argue for a strong objectivity in our results.

7.3. Normalization Effects on Trends

We consolidated benchmark variants (e.g., different releases of the same suite) and grouped microbenchmarks under a single entry to keep the analysis tractable. In some cases, the fine-grained distinctions between versions or microbenchmarks can be useful for pinpointing specific methodological shifts or niche evaluation practices. However, if we preserved every variant as a separate item, the resulting counts would be dominated by labeling noise and fragmentation, making cross-paper trends harder to interpret.

7.4. Taxonomy Constraints

Our type/domain taxonomy in Section 4.4.2 is a pragmatic abstraction tailored to benchmark analysis rather than a universal ontology. It trades the fine-grained ACM CCS for broader categories that better match benchmark artifacts, but this means some items span multiple domains or fit only imperfectly within a single type. For example, suites often span multiple domains, and neural network architectures are treated as a special case to align with how they are commonly reported in the literature. Consequently, counts by type or domain should be read as approximate rather than definitive classifications, and edge cases may shift across categories without changing the underlying benchmark practice.

7.5. External Validity and Transferability

As explained in Section 4, our corpus already covers a broad range of memory systems and technologies. If future hardware exposes performance or usage characteristics similar to those studied here, the benchmark insights and categorizations can transfer with minimal adaptation. However, predicting entirely new architectural behaviors or designing benchmarks for technologies that diverge substantially from current patterns is beyond the scope of this survey.

8. Related Work

To the best of our knowledge, there is no comprehensive survey on benchmarks for disruptive memory technologies. Previous studies have focused on other aspects, such as application domains like big data analysis or edge computing [96, 97, 98, 99, 100, 101]. In this section, we provide a comparison of our own methodology and findings with those presented in related works.

Ihde et al. analyze a set of big data, HPC, and ML benchmarks with a similar methodology to ours, despite not being focused on modern memory architectures [96]. First, they define benchmarking as the process of obtaining quantitative measures for performance comparisons between a set of target systems or components. In contrast, we also include measures that allow for standalone performance analysis, for instance, to determine performance bottlenecks.

Varghese et al. present a survey on performance benchmarks for edge computing platforms [97]. They distinguish between explicit benchmarking (papers that present a benchmark, benchmarking method, or toolchain) and implicit benchmarking (focused on performance evaluation, not benchmark development – similar to

our microbenchmark classification). The survey started with 3764 publications, screened down to 689, finally resulting in 21 explicit and 99 implicit benchmark publications for further analysis. As memory and storage technologies are part of the broad view on edge computing platforms, the survey accordingly has little overlap with our focus. Furthermore, their methodology shows some similarities: they present a timeline of notable benchmarks and related events, categorize benchmarks based on author knowledge and partially subjective classifiers, and distinguish micro- and macrobenchmarks. We note that our benchmark categories are more fine-grained and that we also provide a comparative analysis of benchmarking tools, allowing us to highlight evaluation gaps in the literature.

Bajaber et al. survey benchmarks for big data systems [98]. They follow a taxonomy-driven approach, classifying existing big data systems into categories such as NoSQL databases or BigSQL engines. The key contribution is the presentation of open challenges and missing requirements in current benchmarks. However, their approach is based on the manual selection of domain-specific benchmarks, therefore resulting in biased results. Our structured literature review aims to avoid such inaccuracies.

Han et al. examine big data benchmarks from a different perspective [99]. The study is solely based on their experience from developing the *BigDataBench* [102] benchmark suite and knowledge gained from several workshops, focusing on big data systems and other areas. The authors categorize benchmarks into microbenchmarks, end-to-end benchmarks, and benchmark suites, and also discuss workload generation techniques and execution modes. Notably, their classification aligns with our own benchmark types and domains (see Section 4.4.2).

Reniers et al. address benchmarks for NoSQL systems from the TPC and SPEC suites, such as key-value stores, document stores, column stores, and graph stores [100]. Although their focus is narrower than our survey and previous related work, it enables a taxonomy-driven approach tailored to NoSQL benchmarks. The study identifies gaps in benchmarking advanced workloads for NoSQL databases and discusses the strengths and weaknesses of different benchmark design approaches. The authors present arguments for working with benchmark suites that target specific families of NoSQL databases while still allowing overall comparisons.

Finally, Traeger et al. study benchmarking practices used in research papers related to file and storage systems [101]. Their survey aims to identify both good and poor practices, expose flaws, and provide guidelines for crafting robust benchmarks that yield accurate performance numbers. It follows a comprehensive domain-specific approach in that it analyzes 106 research papers. In contrast to our broad literature survey, the authors only consider full-length papers from specific conferences and further limit the analysis to publications that measure the latency or throughput. The authors categorize these benchmarks into macrobenchmarks, microbenchmarks, and trace replays and provide detailed evaluations of their use cases, limitations, and (mis-) applications.

In contrast to the other considered works, our survey differs in multiple dimensions. First, we focus on modern memory systems, which is a technology-driven classification, where other researchers approach their scope from an application domain. Additionally, our focus on *modern* memory technologies chronologically separates our work from others, as new hardware has emerged and some older technologies have disappeared. Second, we aim to provide an objective overview of the broad field of memory benchmarks by basing our study on an extensive literature search with strict classification rules instead of introducing too much subjective domain-specific knowledge of the individual participants.

9. Conclusion

The definition of standard benchmarks used in the whole community is still key to advancing the field and gradually setting up higher standards in system performance. Due to the abundance of available benchmarks, it is cumbersome to find the best benchmark for the given system that challenges its specific properties. This is especially true for benchmarking modern memory devices, which have diverse properties that need to be tested (e.g., asymmetric access latencies for different access patterns). To close this gap, we conducted an extensive literature review on the current state of the art in benchmarking memory devices and examined the benchmarks that were proposed and/or used in the respective papers.

Our **structured literature review** examined the Top-100 Google Scholar results for a total of 28 search terms that covered a variety of concrete memory technologies as well as emerging concepts. A prefiltering step, consisting of automated removal of duplicates as well as a manual analysis by domain experts in order to identify papers that are out of scope or not related to benchmarks, left us with a repository of 940 papers that propose or reference at least one dataset or memory benchmark. After extracting and normalizing those benchmarks, we obtained a total of 834 different benchmarks for our analysis. These, in turn, consisted of 37 proposed benchmarks, 725 benchmarks that were used for benchmarking but not proposed in the papers in our repository, and 72 microbenchmarks.

For the **proposed benchmarks**, we derived common characteristics in terms of tested access patterns, workload characteristics, language, and benchmarking target (e.g., bandwidth vs. latency or energy efficiency). We further interpreted this data with respect to the benchmarks' focus and target hardware.

Based on this characterization, we discussed the results from two different perspectives – from a user's and a researcher's perspective. From a user's perspective, our overview and systematic presentation of a variety of benchmarks can help to select a benchmark for a specific purpose. From a researcher's perspective, we showed which aspects of hardware are subjects of interest and which are underrepresented in current benchmarks.

From the **used benchmarks** we extracted, how the memory community benchmarks their systems. To this end, we first annotated the benchmarks that had been used by the papers in our corpus with their specific type and domain in order to characterize the benchmarking efforts. Since existing characterization schemes were not suitable for the peculiarities of memory benchmarks, we contribute a list of benchmark types and domains, and use this for characterization. Based on this, we defined several analysis questions that we answered using this annotated corpus. Our findings are fourfold. First, the community uses diverse types and domains of benchmarks, which is a good sign for comprehensive benchmark practices; however, it is still necessary to promote such practices. Second, the usage of domains is influenced by the developments and hypes in the application domain, as it shows fluctuations of domains' usage over the years. Third, it is evident that, while there is a considerable amount of simple evaluation setups with a single benchmark type or domain involved, numerous papers have a sophisticated benchmarking setup with several benchmarks of different domains and types. Finally, papers using diverse benchmarks should pay attention to a consistent mentioning of their used benchmarks to allow for robust meta-analyses.

Based on both studies, we **identified several limitations** in current benchmark design. First, benchmarks exhibit a narrow measurement focus: nearly all measure bandwidth, latency, and throughput, but only a handful include energy efficiency, and none systematically evaluates aspects beyond that, namely the

fairness of concurrent accesses (might be particularly relevant for shared hosts) or security related properties and costs of potential security mitigations. Second, there is a persistent trade-off between reproducibility, accuracy, and realism – simulator-based approaches offer isolated inspection of properties but abstract away real hardware behaviors, while hardware-based benchmarks are more accurate but introduce platform-specific side-effects from CPU microarchitecture, caching, and OS scheduling. Third, many benchmarks are overfitted to specific technologies (particularly Intel Optane), limiting cross-technology comparability and transferability to emerging architectures such as CXL-attached memory.

To address these limitations, we recommend several **actionable guidelines** for developers and users. Benchmark designers should adopt multi-metric evaluation (including energy), validate across both simulation and hardware, and design for cross-technology transferability by targeting common software interfaces rather than specific implementations. Researchers should embrace multi-domain and multi-type benchmarking – our analysis shows that benchmark suites are the most effective tool for this. Finally, the community should standardize benchmark documentation (persistent identifiers, explicit measurement methodology, and public source code) to enable robust meta-analyses, comparison, and reproducibility.

We finally **conclude** that benchmarking practices must evolve with the application landscape. As new applications and memory technologies emerge, the community should periodically reassess their repertory of benchmarks. New benchmarks should be developed as hybrid evaluation frameworks that combine the reproducibility of simulation with the accuracy of hardware measurement. Mixing benchmarks from diverse domains and types, which we identified as a common and reasonable practice, ensures a holistic view on future system's properties.

A. Appendix

Microbenchmarks	Access Pattern								
	Sequential			Random			Strided		
	R	W	U	R	W	U	R	W	U
Rand	-	-	-	-	-	X	-	-	-
Stride1	-	-	-	-	-	-	-	-	X
StrideN	-	-	-	-	-	-	-	-	X
PtrChase	-	-	-	-	-	X	-	-	-
Central	-	-	X	-	-	-	-	-	-
S/G	-	-	X	-	-	X	-	-	-
Scatter	-	-	X	-	-	X	-	-	-
Gather	-	-	X	-	-	X	-	-	-

Table 7. An overview of CircusTent microbenchmarks.

Microbenchmarks	Application Domain
mtrans	linear algebra
mmulti	linear algebra
bfs	graph algorithm
cfid	fluid dynamics
hotspot	physics simulation
kmeans	data mining
lavaMD	molecular dynamics
lud	linear algebra
nn	data mining
nw	bioinformatics
particlefilter	medical imaging
pathfinder	grid traversal
srad	image processing

Table 8. An overview of HMBench microbenchmarks.

Data Structures	Persistent Modes	Parallel Modes	Workloads
• Arraylist • Linkedlist • Hashtable • Skiplist • B/B+-Tree • RB-Tree • Compressed Sparse Row • Blocked Adjacency List	• DRAM • PMem-Volatile • PMem-Persist • PMem-Trans	• Single-threaded • Parallel-Saturated • Concurrent-Contention • NUMA	• YCSB • SNAP

Table 9. An overview of pmemids_bench configurations.

Microbenchmarks	Component Tested	Metric	Usage
System Call Latency	CPU/OS	Latency	Measuring OS efficiency in handling system calls
Context Switch	CPU	Latency	Assessing the time taken for task switching
Memory Bandwidth	Memory	Bandwidth	Evaluating the memory transfer rate
File System Latency	File System	Latency	Measuring file access time
TCP Latency	Network	Latency	Measuring network communication delay
TCP Bandwidth	Network	Bandwidth	Assessing the data transfer rate over TCP
Disk Read/Write	Disk	Latency/Bandwidth	Measuring disk I/O performance
Pipe Latency	CPU/OS	Latency	Assessing IPC efficiency
Memory Latency	Memory	Latency	Measuring access time for memory reads
Network Latency	Network	Latency	Measuring delay in network packet transmission

Table 10. An overview of lmbench microbenchmarks.

Microbenchmarks	Observations	Root Causes
MB1	Asymmetry in load & p-store latencies	CPU cache flush latency
MB2	Asymmetry in load & p-store bandwidths	Contention at iMC & 3D-XPoint latency
MB3	Poor bandwidth for small-sized random IO	Mismatch between access granularities $< 256\text{B}$ and $\geq 256\text{B}$
MB4	Poor p-store bandwidth for PMem on remote NUMA	Limited UPI/inter-socket bandwidth & XPoint prefetcher
MB5	Regular store bandwidth lower than p-store bandwidth	XPoint prefetcher
MB6	Sequential IO faster than random IO	CPU & XPoint prefetchers
MB7	P-stores are read-modify-write transactions	XPoint buffer design

Table 11. An overview of PMIdioBench microbenchmarks.

Microbenchmarks	r_seq_ind	r_seq_reduce	r_rand_ind	r_rand_pchase	r_stride_<k>	r_tile	w_seq_memset	w_seq_fill	w_rand_ind	w_stride_<k>	w_tile	rw_seq_inc	rw_gather	rw_scatter	rw_scatter_gather	rw_tile
Read Access	seq	seq	rand	rand	stride	tile	-	-	-	-	-	seq	seq + rand	seq	seq + rand	tile
Write Access	-	-	-	-	-	-	seq	seq	rand	stride	tile	seq	seq	rand	rand	tile

Table 12. An overview of Hopscotch microbenchmarks with sequential/random/strided/tunable read/write access patterns.

Microbenchmarks	Access Pattern									Workload Granularity	Workload Size	Bandwidth	Latency	Energy	Throughput	Runtime	Bound	
	Sequential			Random			Strided										CPU	Memory
	R	W	U	R	W	U	R	W	U									
Arithmetic-Throughput	-	-	X	-	-	-	-	-	-	4B..8B	B..KiB	-	-	-	-	X	X	-
CPU-DPU	X	X	-	-	-	-	-	-	-	8B	MiB..GiB	X	-	-	-	-	-	X
MRAM-Latency	X	X	-	-	-	-	-	-	-	8B	KiB..MiB	-	X	-	-	-	-	-
Operational-Intensity	-	-	X	-	-	-	-	-	X	4B	KiB..MiB	-	-	-	X	-	X	-
Random-GUPS	-	-	-	X	X	-	-	-	-	8B	MiB	X	-	-	-	-	-	X
STREAM	X	X	-	-	-	-	-	-	-	8B	MiB	X	-	-	-	X	X	X
STRIDED	-	-	-	-	-	-	X	X	-	8B	KiB..MiB	X	-	-	-	-	-	X
MRAM	X	X	-	-	-	-	X	X	-	4B	KiB..MiB	X	-	-	-	-	-	X

Table 13. An overview of PrIM microbenchmarks: R(ead), W(rite), U(pdate).

10. Acknowledgments

This work was partially funded by the following projects from the DFG-SPP 2377 on Disruptive Memory Technologies: 502268500, 502565817, 501887536, 465958100, 502228341, 502388442, 502352642. In accordance to this, we thank the corresponding PIs of the projects: Jeronimo Castrillon, Daniel Lohmann, Michael Kuhn, Timo Hönig, Gunter Saake, Kai-Uwe Sattler, Olaf Spinczyk. We thank Lars Wrenger, Fia Wünsche, and Lennart Schmidt, who helped in the early stage of collecting benchmarks from Google Scholar.

Ethical Statement

No ethical approval was required for this study, as it did not involve human or animal subjects.

Funding

This work was partially funded by the following projects from the DFG-SPP 2377 on Disruptive Memory Technologies: 502268500, 502565817, 501887536, 465958100, 502228341, 502388442, 502352642.

Declaration of competing interests

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.

Data Availability Statements

The data supporting the findings of this study are openly available in <https://zenodo.org/records/15098479>.

Credit authorship contribution statement

David Broneske: Methodology, Conceptualization, Data Curation, Visualization, Writing - Original Draft Preparation, Writing - Review & Editing, Supervision.

Christian Eichler: Methodology, Conceptualization, Data Curation, Visualization, Writing - Original Draft Preparation, Writing - Review & Editing.

Hamid Farzaneh: Methodology, Conceptualization, Data Curation, Visualization, Writing - Original Draft Preparation, Writing - Review & Editing, Software.

Birte Friesel: Methodology, Conceptualization, Data Curation, Visualization, Writing - Original Draft Preparation, Writing - Review & Editing, Software.

Alexander Halbuer: Methodology, Conceptualization, Data Curation, Visualization, Writing - Original Draft Preparation, Writing - Review & Editing.

Benedict Herzog: Methodology, Conceptualization, Data Curation, Visualization, Writing - Original Draft Preparation, Writing - Review & Editing.

Muhammad Attahir Jibril: Methodology, Conceptualization, Data Curation, Visualization, Writing - Original Draft Preparation, Writing - Review & Editing.

Sajad Karim: Methodology, Conceptualization, Data Curation, Visualization, Writing - Original Draft Preparation, Writing - Review & Editing, Software.

Sven Köhler: Methodology, Conceptualization, Data Curation, Visualization, Writing - Original Draft Preparation, Writing -

Review & Editing, Software.

Manuel Vögele: Methodology, Conceptualization, Data Curation, Visualization, Writing - Original Draft Preparation, Writing - Review & Editing.

References

1. S. Li, D. Reddy, and B. Jacob, “A performance & power comparison of modern high-speed DRAM architectures,” in *Proceedings of the International Symposium on Memory Systems*, ser. MEMSYS '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 341–353, doi: <https://doi.org/10.1145/3240302.3240315>.
2. S. A. McKee, “Reflections on the Memory Wall,” in *Proceedings of the 1st Conference on Computing Frontiers*, ser. CF '04. New York, NY, USA: Association for Computing Machinery, 2004, pp. 162–167, doi: <https://doi.org/10.1145/977091.977115>.
3. T. M. Hollis *et al.*, “Recent Evolution in the DRAM Interface: Mile-Markers Along Memory Lane,” *IEEE Solid-State Circuits Magazine*, vol. 11, no. 2, pp. 14–30, 2019, doi: <https://doi.org/10.1109/MSSC.2019.2910617>.
4. Micron Technology, Inc. (2023) Product Brief – Micron HBM3E. Micron Technology, Inc. [Online]. Available: <https://www.micron.com/content/dam/micron/global/public/documents/products/product-flyer/hbm3e-product-brief.pdf>
5. H. Huang *et al.*, “Shuhai: A Tool for Benchmarking High Bandwidth Memory on FPGAs,” *IEEE Transactions on Computers*, vol. 71, no. 5, pp. 1133–1144, 2022, doi: <https://doi.org/10.1109/TC.2021.3075765>.
6. C. H. Lam, *History of Phase Change Memories*. Boston, MA, USA: Springer US, 2009, pp. 1–14, doi: https://doi.org/10.1007/978-0-387-84874-7_1.
7. A. M. Rudoff. (2016) Deprecating the PCOMMIT Instruction. Intel Corporation. [Online]. Available: <https://www.intel.com/content/www/us/en/developer/articles/technical/deprecate-pcommit-instruction.html>
8. T. Alsop. (2023, Apr) Hard disk drive (HDD) unit shipments worldwide from 1976 to 2022. Statista. [Online]. Available: <https://www.statista.com/statistics/398951/global-shipment-figures-for-hard-disk-drives/>
9. Western Digital. (2025) WD Blue 2.5in PC Mobile Hard Drive. Western Digital. [Online]. Available: <https://www.westerndigital.com/en-us/products/internal-drives/wd-blue-mobile-sata-hdd?sku=WD5000LPZX>
10. —. (2025) WD Red Pro 3.5in NAS Hard Drive. Western Digital. [Online]. Available: <https://www.westerndigital.com/en-us/products/internal-drives/wd-red-pro-sata-hdd?sku=WD2002FFSX>
11. Seagate. (2024) Transition to Advanced Format 4K Sector Hard Drives. Seagate. [Online]. Available: <https://www.seagate.com/blog/advanced-format-4k-sector-hard-drives-master-ti/>
12. Y. Shiroishi *et al.*, “Future Options for HDD Storage,” *IEEE Transactions on Magnetism*, vol. 45, no. 10, pp. 3816–3822, 2009, doi: <https://doi.org/10.1109/TMAG.2009.2024879>.
13. L. Gough. (2019, Dec) Hard Drive Performance Over the Years. Gough’s Tech Zone. [Online]. Available: <https://goughlui.com/the-hard-disk-corner/hard-drive-performance-over-the-years/>
14. S. Nedev and V. Kamenov, “HDD performance research,” in *Proceedings of the International Scientific Conference Computer Science*, ser. CS '18. Berlin, Germany: ResearchGate, 01 2018, pp. 106–111.

15. F. Chen, D. A. Koufaty, and X. Zhang, "Understanding Intrinsic Characteristics and System Implications of Flash Memory based Solid State Drives," in *Proceedings of the 11th International Joint Conference on Measurement and Modeling of Computer Systems*, ser. SIGMETRICS '09. New York, NY, USA: Association for Computing Machinery, 2009, pp. 181–192, doi: <https://doi.org/10.1145/1555349.1555371>.
16. Samsung. (2021) SSD 870 EVO SATA III 2.5in – 250 GB. Samsung. [Online]. Available: <https://www.samsung.com/de/memory-storage/sata-ssd/870-evo-250gb-sata-3-2-5-ssd-mz-77e250b-eu/>
17. —. (2023) 990 PRO NVMe M.2 SSD – 4 TB. Samsung. [Online]. Available: <https://www.samsung.com/de/memory-storage/nvme-ssd/990-pro-4tb-nvme-pcie-gen-4-mz-v9p4t0bw/>
18. L. Smith. (2021) Samsung 870 EVO SSD Review. Storage Review. [Online]. Available: <https://www.storagereview.com/review/samsung-870-evo-ssd-review>
19. S. Downing. (2023) Samsung 990 Pro 4TB Review: The Best Gets Bigger. Tom's Hardware. [Online]. Available: <https://www.tomshardware.com/reviews/samsung-990-pro-4tb-ssd-review>
20. S. Corda, G. Singh, A. J. Awan, R. Jordans, and H. Corporaal, "Platform Independent Software Analysis for Near Memory Computing," in *Proceedings of the 22nd Euromicro Conference on Digital System Design*, ser. DSD '19. New York, NY, USA: IEEE, 2019, pp. 606–609, doi: <https://doi.org/10.1109/DSD.2019.00093>.
21. D. Lee et al., "Optimizing Data Movement with Near-Memory Acceleration of In-memory DBMS," in *Proceedings of the 22nd International Conference on Extending Database Technology*, ser. EDBT '20. Konstanz, Germany: Open Proceedings, 2020, pp. 371–374, doi: <https://doi.org/10.5441/002/edbt.2020.35>.
22. B. Friesel, M. Lütke Dreimann, and O. Spinczyk, "Performance Models for Task-based Scheduling with Disruptive Memory Technologies," in *Proceedings of the 2nd Workshop on Disruptive Memory Systems*, ser. DIMES '24. New York, NY, USA: Association for Computing Machinery, 11 2024, pp. 1–8, doi: <https://doi.org/10.1145/3698783.3699376>.
23. J. Gómez-Luna, I. E. Hajj, I. Fernandez, C. Giannoula, G. F. Oliveira, and O. Mutlu, "Benchmarking a New Paradigm: Experimental Analysis and Characterization of a Real Processing-in-Memory System," *IEEE Access*, vol. 10, pp. 52 565–52 608, 2022, doi: <https://doi.org/10.1109/ACCESS.2022.3174101>.
24. F. Ferdaus and M. T. Rahman, *Security of Emerging Memory Chips*. Cham: Springer International Publishing, 2021, pp. 357–390, doi: https://doi.org/10.1007/978-3-030-64448-2_14.
25. J. Yang, J. Kim, M. Hoseinzadeh, J. Izraelevitz, and S. Swanson, "An Empirical Guide to the Behavior and Use of Scalable Persistent Memory," in *Proceedings of the 18th USENIX Conference on File and Storage Technologies*, ser. FAST '20. Santa Clara, CA: USENIX Association, Feb. 2020, pp. 169–182. [Online]. Available: <https://www.usenix.org/conference/fast20/presentation/yang>
26. R. Bharath and K. S. Pande, "HBM3 Architectural Specific Checks and Timing Closure," in *Proceedings of the 5th International Conference on Smart Electronics and Communication (ICOSEC)*, ser. ICOSEC '24. New York, NY, USA: IEEE, 2024, pp. 239–244, doi: <https://doi.org/10.1109/ICOSEC61587.2024.10722076>.
27. JC-42.3 Subcommittee on DRAM Memories, "JEDEC Standard 79-5: DDR5 SDRAM," JEDEC Solid State Technology Association, Standard, Apr. 2024.
28. K. Kim and M.-j. Park, "Present and Future, Challenges of High Bandwidth Memory (HBM)," in *Proceedings of the International Memory Workshop*, ser. IMW '24. New York, NY, USA: IEEE, 2024, pp. 1–4, doi: <https://doi.org/10.1109/IMW59701.2024.10536972>.
29. R. Blasing et al., "Magnetic Racetrack Memory: From Physics to the Cusp of Applications Within a Decade," *Proceedings of the IEEE*, vol. 108, no. 8, pp. 1303–1321, 2020, doi: <https://doi.org/10.1109/JPROC.2020.2975719>.
30. A. Fazio, "Advanced Technology and Systems of Cross Point Memory," in *Proceedings of the International Electron Devices Meeting*, ser. IEDM '20. New York, NY, USA: IEEE, 2020, pp. 24.1.1–24.1.4, doi: <https://doi.org/10.1109/IEDM13553.2020.9371976>.
31. Z. Song et al., "From octahedral structure motif to sub-nanosecond phase transitions in phase change materials for data storage," *Science China Information Sciences*, vol. 61, 2018, doi: <https://doi.org/10.1007/s11432-018-9404-2>.
32. Samsung. (2025) SSDs (Solid State Drive). Samsung. [Online]. Available: <https://www.samsung.com/de/memory-storage/?nvme-ssd>
33. Kingston. (2024) Solid State Drives (SSDs) for laptops, desktop PCs and servers. Kingston Technology. [Online]. Available: <https://www.kingston.com/en/ssd?interface=nvme>
34. JC-42.3 Subcommittee on DRAM Memories, "JEDEC Standard 79-4: DDR4 SDRAM," JEDEC Solid State Technology Association, Standard, Sep. 2012.
35. —, "JEDEC Standard 238A: High Bandwidth Memory DRAM (HBM3)," JEDEC Solid State Technology Association, Standard, Jan. 2023.
36. —, "JEDEC Standard 235D: High Bandwidth Memory DRAM (HBM1, HBM2)," JEDEC Solid State Technology Association, Standard, Mar. 2021.
37. S. Bertolazzi et al., "Nonvolatile Memories Based on Graphene and Related 2D Materials," *Advanced Materials*, vol. 31, no. 10, p. 1806663, 2019, doi: <https://doi.org/10.1002/adma.201806663>.
38. J. J. Kan et al., "A Study on Practically Unlimited Endurance of STT-MRAM," *IEEE Transactions on Electron Devices*, vol. 64, no. 9, pp. 3639–3646, 2017, doi: <https://doi.org/10.1109/TED.2017.2731959>.
39. S. Munjal and N. Khare, "Advances in resistive switching based memory devices," *Journal of Physics D: Applied Physics*, vol. 52, no. 43, p. 433002, 2019, doi: <https://doi.org/10.1088/1361-6463/ab2e9e>.
40. S. Kargar and F. Nawab, "Challenges and future directions for energy, latency, and lifetime improvements in NVMs," *Distrib. Parallel Databases*, vol. 41, no. 3, pp. 163–189, Sep. 2022, doi: <https://doi.org/10.1007/s10619-022-07421-x>.
41. Intel Corporation. (2023) Intel Optane Persistent Memory 200 Series Brief. Intel Corporation. [Online]. Available: <https://www.intel.com/content/www/us/en/products/docs/memory-storage/optane-persistent-memory/optane-persistent-memory-200-series-brief.html>
42. A. I. Alsalihi, S. Mittal, M. A. Al-Betar, and P. B. Sumari, "A survey of techniques for architecting SLC/MLC/TLC hybrid Flash memory-based SSDs," *Concurrency and Computation: Practice and Experience*, vol. 30, no. 13, p. 4420, 2018, doi: <https://doi.org/10.1002/cpe.4420>.
43. S.-I. Jang, S.-K. Yoon, K. Park, G.-H. Park, and S.-D. Kim, "Data Classification Management with its Interfacing Structure for Hybrid SLC/MLC PRAM Main Memory," *The Computer Journal*, vol. 58, no. 11, pp. 2852–2863, 2015, doi: <https://doi.org/10.1093/comjnl/bxu133>.

44. S. Lubner. (2024) What is Racetrack Memory? Vogel IT-Medien GmbH. [Online]. Available: <https://www.storage-insider.de/was-ist-racetrack-memory-racetrack-speicher-a-88d68615005e07c329bc4debe394921a/>
45. A. van Renen, L. Vogel, V. Leis, T. Neumann, and A. Kemper, "Persistent Memory I/O Primitives," in *Proceedings of the 15th International Workshop on Data Management on New Hardware*, ser. DaMoN'19. New York, NY, USA: Association for Computing Machinery, 2019, doi: <https://doi.org/10.1145/3329785.3329930>.
46. L. Benson, L. Papke, and T. Rabl, "PerMA-bench: benchmarking persistent memory access," *Proceedings of the VLDB Endowment*, vol. 15, no. 11, p. 2463–2476, Jul. 2022, doi: <https://doi.org/10.14778/3551793.3551807>.
47. S. S. Parkin, "Shiftable magnetic shift register and method of using the same," dec 2004, uS Patent No. 6,834,005, Filed June 10th, 2003, Issued Dec. 21st, 2004.
48. D. Zhang, N. Jayasena, A. Lyashevsky, J. L. Greathouse, L. Xu, and M. Ignatowski, "TOP-PIM: throughput-oriented programmable processing in memory," in *Proceedings of the 23rd International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 85–98, doi: <https://doi.org/10.1145/2600212.2600213>.
49. F. Devaux, "The true Processing In Memory accelerator," in *Proceedings of the 31st IEEE Hot Chips Symposium*, ser. HCS '19. New York, NY, USA: IEEE, 2019, pp. 1–24, doi: <https://doi.org/10.1109/HOTCHIPS.2019.8875680>.
50. B. Friesel, M. Lütke Dreimann, and O. Spinczyk, "A Full-System Perspective on UPMEM Performance," in *Proceedings of the 1st Workshop on Disruptive Memory Systems*, ser. DIMES '23. New York, NY, USA: Association for Computing Machinery, 2023, pp. 1–7, doi: <https://doi.org/10.1145/3609308.3625266>.
51. Y.-C. Kwon *et al.*, "A 20nm 6GB Function-In-Memory DRAM, Based on HBM2 with a 1.2TFLOPS Programmable Computing Unit Using Bank-Level Parallelism, for Machine Learning Applications," in *Proceedings of the IEEE International Solid-State Circuits Conference*, ser. ISSCC '21, vol. 64. New York, NY, USA: IEEE, 2021, pp. 350–352, doi: <https://doi.org/10.1109/ISSCC42613.2021.9365862>.
52. S. Lee *et al.*, "Hardware Architecture and Software Stack for PIM Based on Commercial DRAM Technology," in *Proceedings of the 48th Annual International Symposium on Computer Architecture*, ser. ISCA '21. New York, NY, USA: IEEE, 2021, pp. 43–56, doi: <https://doi.org/10.1109/ISCA52012.2021.00013>.
53. P. Gu *et al.*, "Technological Exploration of RRAM Crossbar Array for Matrix-Vector Multiplication," in *Proceedings of the 20th Asia and South Pacific Design Automation Conference*, ser. ASP-DAC '20. New York, NY, USA: IEEE, 2015, pp. 106–111, doi: <https://doi.org/10.1109/ASPDAC.2015.7058989>.
54. A. Siemieniuk *et al.*, "OCC: An Automated End-to-End Machine Learning Optimizing Compiler for Computing-In-Memory," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 6, pp. 1674–1686, 2022, doi: <https://doi.org/10.1109/TCAD.2021.3101464>.
55. V. R. Datti and P. Sridevi, "Performance Evaluation of Content Addressable Memories," in *Proceedings of the 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)*, ser. ICRITO '18. New York, NY, USA: IEEE, 2018, pp. 596–598, doi: <https://doi.org/10.1109/ICRITO.2018.8748808>.
56. H. Farzaneh, J. P. C. De Lima, M. Li, A. A. Khan, X. S. Hu, and J. Castrillon, "C4CAM: A Compiler for CAM-based In-memory Accelerators," in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ser. ASPLOS '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 164–177, doi: <https://doi.org/10.1145/3620666.3651386>.
57. V. Seshadri *et al.*, "Ambit: in-memory accelerator for bulk bitwise operations using commodity DRAM technology," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 273–287, doi: <https://doi.org/10.1145/3123939.3124544>.
58. H. Farzaneh *et al.*, "SHERLOCK: Scheduling Efficient and Reliable Bulk Bitwise Operations in NVMs," in *Proceedings of the 61st ACM/IEEE Design Automation Conference*, ser. DAC '24. New York, NY, USA: Association for Computing Machinery, 2024, doi: <https://doi.org/10.1145/3649329.3658485>.
59. M. Moreau *et al.*, "Reliable ReRAM-based Logic Operations for Computing in Memory," in *Proceedings of the International Conference on Very Large Scale Integration*, ser. VLSI-SoC '18. New York, NY, USA: IEEE, 2018, pp. 192–195, doi: <https://doi.org/10.1109/VLSI-SoC.2018.8644780>.
60. D. Das Sharma, R. Blankenship, and D. Berger, "An Introduction to the Compute Express Link (CXL) Interconnect," *ACM Computing Surveys*, vol. 56, no. 11, Jul. 2024, doi: <https://doi.org/10.1145/3669900>.
61. D. Gouk, M. Kwon, H. Bae, S. Lee, and M. Jung, "Memory Pooling With CXL," *IEEE Micro*, vol. 43, no. 2, pp. 48–57, 2023, doi: <https://doi.org/10.1109/MM.2023.3237491>.
62. C. Guo, J. Xu, and M. Zukerman, "Disaggregated Architectures and the Redesign of Data Center Ecosystems: Scheduling, Pooling, and Infrastructure Trade-Offs," *IEEE Communications Magazine*, vol. 64, no. 6, pp. 111–117, 2026, doi: <https://doi.org/10.1109/MCOM.001.2500230>.
63. A. Puri, J. Jose, and T. Venkatesh, "Design and Evaluation of a Rack-Scale Disaggregated Memory Architecture For Data Centers," in *2022 IEEE 24th Int Conf on High Performance Computing & Communications; 8th Int Conf on Data Science & Systems; 20th Int Conf on Smart City; 8th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*, 2022, pp. 212–217, doi: <https://doi.org/10.1109/HPCC-DSS-SmartCity-DependSys57074.2022.00060>.
64. A. Puri, K. Bellamkonda, K. Narreddy, J. Jose, V. Tamarapalli, and V. Narayanan, "DRackSim: Simulating CXL-enabled Large-Scale Disaggregated Memory Systems," in *Proceedings of the 38th ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, ser. SIGSIM-PADS '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 3–14, doi: <https://doi.org/10.1145/3615979.3656059>. [Online]. Available: <https://doi.org/10.1145/3615979.3656059>
65. S. Karim, J. Wünsche, M. Kuhn, G. Saake, and D. Brönske, "NVM in Data Storage: A Post-Optane Future," *ACM Trans. Storage*, vol. 21, no. 3, Jul. 2025, doi: <https://doi.org/10.1145/3731454>. [Online]. Available: <https://doi.org/10.1145/3731454>
66. A. Geyer *et al.*, "Near to Far: An Evaluation of Disaggregated Memory for In-Memory Data Processing," in *Proceedings of the 1st Workshop on Disruptive Memory Systems*, ser. DIMES '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 16–22, doi: <https://doi.org/10.1145/3609308.3625271>.

- [Online]. Available: <https://doi.org/10.1145/3609308.3625271>
67. C. Li, J. Zhao, and Y. Xu, "Efficient Security Support for CXL Memory through Adaptive Incremental Offloaded (Re-)Encryption," in *Proceedings of the 58th IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '25. New York, NY, USA: Association for Computing Machinery, 2025, p. 1102–1116, doi: <https://doi.org/10.1145/3725843.3756119>. [Online]. Available: <https://doi.org/10.1145/3725843.3756119>
 68. A. Psistakis, B. Ocalan, C. Alverti, F. Chaix, R. Alagappan, and J. Torrellas, "Towards cxl resilience to cpu failures," 2026. [Online]. Available: <https://arxiv.org/abs/2602.08271>
 69. D. Habicht, Y. Khalil, L. Werling, T. Gröninger, and F. Bellosa, "Fundamental OS Design Considerations for CXL-based Hybrid SSDs," in *Proceedings of the 2nd Workshop on Disruptive Memory Systems*, ser. DIMES '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 51–59, doi: <https://doi.org/10.1145/3698783.3699380>.
 70. Z. Zhu et al., "Lupin: Tolerating Partial Failures in a CXL Pod," in *Proceedings of the 2nd Workshop on Disruptive Memory Systems*, ser. DIMES '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 41–50, doi: <https://doi.org/10.1145/3698783.3699377>.
 71. B. A. Kitchenham and S. Charters, "Guidelines for Performing Systematic Literature Reviews in Software Engineering," Keele University, Tech. Rep. EBSE-2007-01, 2007.
 72. B. A. Kitchenham, D. Budgen, and P. Brereton, *Evidence-Based Software Engineering and Systematic Reviews*. New York, NY, USA: Chapman and Hall/CRC, 2015, doi: <https://doi.org/10.1201/b19467>.
 73. Y. Shakeel et al., "(Automated) Literature Analysis - Threats and Experiences," in *Proceedings of the International Workshop on Software Engineering for Science*, ser. SE4Science '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 20–27, doi: <https://doi.org/10.1145/3194747.3194748>.
 74. Standard Performance Evaluation Corporation. (2025) SPEC Benchmarks and Tools. Standard Performance Evaluation Corporation. [Online]. Available: <https://www.spec.org/products/>
 75. Association for Computing Machinery. (2024) ACM Computing Classification System. Association for Computing Machinery. [Online]. Available: <https://dl.acm.org/ccs>
 76. B. Williams, J. Leidel, X. Wang, D. Donofrio, and Y. Chen, "CircusTent: A Benchmark Suite for Atomic Memory Operations," in *Proceedings of the International Symposium on Memory Systems*, ser. MEMSYS '20. New York, NY, USA: Association for Computing Machinery, 2021, pp. 144–157, doi: <https://doi.org/10.1145/3422575.3422789>.
 77. G. F. Oliveira et al., "DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks," *IEEE Access*, vol. 9, pp. 134457–134502, 2021, doi: <https://doi.org/10.1109/ACCESS.2021.3110993>.
 78. A. Qasem, A. M. Aji, and G. Rodgers, "Characterizing data organization effects on heterogeneous memory architectures," in *Proceedings of the International Symposium on Code Generation and Optimization*, ser. CGO '17. New York, NY, USA: IEEE Press, 2017, p. 160–170, doi: <https://doi.org/10.1109/CGO.2017.7863737>.
 79. J. Zhang, N. Beckwith, and J. J. Li, "GORDON: Benchmarking Optane DC Persistent Memory Modules on FPGAs," in *Proceedings of the 29th Annual International Symposium on Field-Programmable Custom Computing Machines*, ser. FCCM '21. New York, NY, USA: IEEE, 2021, pp. 97–105, doi: <https://doi.org/10.1109/FCCM51124.2021.00019>.
 80. D. Shen, X. Liu, and F. X. Lin, "Characterizing Emerging Heterogeneous Memory," in *Proceedings of the International Symposium on Memory Management*, ser. ISMM '16. New York, NY, USA: Association for Computing Machinery, 2016, pp. 13–23, doi: <https://doi.org/10.1145/2926697.2926702>.
 81. A. Ahmed and K. Skadron, "Hopscotch: A Micro-benchmark Suite for Memory Performance Evaluation," in *Proceedings of the International Symposium on Memory Systems*, ser. MEMSYS '19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 167–172, doi: <https://doi.org/10.1145/3357526.3357574>.
 82. M. Hildebrand, J. T. Angeles, J. Lowe-Power, and V. Akella, "A Case Against Hardware Managed DRAM Caches for NVRAM Based Systems," in *Proceedings of the International Symposium on Performance Analysis of Systems and Software*, ser. ISPASS '21. New York, NY, USA: IEEE, 2021, pp. 194–204, doi: <https://doi.org/10.1109/ISPASS51385.2021.00036>.
 83. Z. Wang, X. Liu, J. Yang, T. Michailidis, S. Swanson, and J. Zhao, "Characterizing and Modeling Non-Volatile Memory Systems," in *Proceedings of the 53rd Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '20. New York, NY, USA: IEEE, 2020, pp. 496–508, doi: <https://doi.org/10.1109/MICRO50266.2020.00049>.
 84. L. McVoy and C. Staelin, "lmbench: Portable Tools for Performance Analysis," in *Proceedings of the USENIX Annual Technical Conference*, ser. ATC '96. Santa Clara, CA: USENIX Association, 1996, pp. 279–294.
 85. A. A. R. Islam, C. York, and D. Dai, "A performance study of optane persistent memory: from storage data structures' perspective," *CCF Transactions on High Performance Computing*, vol. 4, no. 4, pp. 370–393, 2022, doi: <https://doi.org/10.1007/s42514-022-00123-x>.
 86. S. Gugnani, A. Kashyap, and X. Lu, "Understanding the Idiosyncrasies of Real Persistent Memory," *Proceedings of the VLDB Endowment*, vol. 14, no. 4, p. 626–639, Dec. 2020, doi: <https://doi.org/10.14778/3436905.3436921>.
 87. H. Yoshioka, Y. Hayamizu, K. Goda, and M. Kitsuregawa, "pmmeter: A Microbenchmark for Understanding Synchronization Cost on Persistent Memory," in *Proceedings of the International Conference on Big Data and Smart Computing*, ser. BigComp '23. New York, NY, USA: IEEE, 2023, pp. 326–327, doi: <https://doi.org/10.1109/BigComp57234.2023.00069>.
 88. J. Gómez-Luna, I. El Hajj, I. Fernandez, C. Giannoula, G. F. Oliveira, and O. Mutlu, "Benchmarking Memory-Centric Computing Systems: Analysis of Real Processing-in-Memory Hardware," in *Proceedings of the 12th International Green and Sustainable Computing Conference*, ser. IGSC '21. New York, NY, USA: IEEE, 2021, pp. 1–7, doi: <https://doi.org/10.1109/IGSC54211.2021.9651614>.
 89. A. Grapentin, F. Eberhardt, and A. Polze, "SMOG – An Explicitly Composable Memory Benchmark Suite for Heterogeneous Memory," in *Proceedings of the International Symposium on Computing and Networking*, ser. CANDAR '23. New York, NY, USA: IEEE, 2023, pp. 113–119, doi: <https://doi.org/10.1109/CANDAR60563.2023.00022>.
 90. I. L. Picoli, C. V. Pasco, B. T. Jónsson, L. Bouganim, and P. Bonnet, "uFLIP-OC: Understanding Flash I/O Patterns on Open-Channel Solid-State Drives," in *Proceedings of the 8th Asia-Pacific Workshop on Systems*, ser. APSys '17. New York, NY, USA: Association for Computing Machinery, 2017, doi: <https://doi.org/10.1145/3124680.3124741>.

91. S. Che *et al.*, “Rodinia: A Benchmark Suite for Heterogeneous Computing,” in *Proceedings of the International Symposium on Workload Characterization*, ser. IISWC '09. New York, NY, USA: IEEE, 2009, pp. 44–54, doi: <https://doi.org/10.1109/IISWC.2009.5306797>.
92. R. Shinde, A. Goel, P. Gupta, and D. Dutta, “Similarity search and locality sensitive hashing using ternary content addressable memories,” in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 375–386, doi: <https://doi.org/10.1145/1807167.1807209>.
93. C. Bienia, S. Kumar, J. P. Singh, and K. Li, “The PARSEC benchmark suite: characterization and architectural implications,” in *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques*, ser. PACT '08. New York, NY, USA: Association for Computing Machinery, 2008, pp. 72–81, doi: <https://doi.org/10.1145/1454115.1454128>.
94. F. Lalchhandama, K. Datta, S. Chakraborty, R. Drechsler, and I. Sengupta, “CoMIC: Complementary Memristor based in-memory computing in 3D architecture,” *Journal of Systems Architecture*, vol. 126, p. 102480, 2022, doi: <https://doi.org/10.1016/j.sysarc.2022.102480>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1383762122000613>
95. Y. Li *et al.*, “In-Memory Computing using Memristor Arrays with Ultrathin 2D PdSeOx/PdSe2 Heterostructure,” *Advanced Materials*, vol. 34, no. 26, p. 2201488, 2022, doi: <https://doi.org/10.1002/adma.202201488>. [Online]. Available: <https://advanced.onlinelibrary.wiley.com/doi/abs/10.1002/adma.202201488>
96. N. Ihde *et al.*, “A Survey of Big Data, High Performance Computing, and Machine Learning Benchmarks,” in *Proceedings of the 13th TPC Technology Conference on Performance Evaluation and Benchmarking*, ser. TPCTC '21. Cham: Springer International Publishing, 2021, pp. 98–118, doi: https://doi.org/10.1007/978-3-030-94437-7_7.
97. B. Varghese *et al.*, “A Survey on Edge Performance Benchmarking,” *ACM Computing Surveys*, vol. 54, no. 3, Apr. 2021, doi: <https://doi.org/10.1145/3444692>.
98. F. Bajaber, S. Sakr, O. Batarfi, A. Altalhi, and A. Barnawi, “Benchmarking big data systems: A survey,” *Computer Communications*, vol. 149, pp. 241–251, 2020, doi: <https://doi.org/10.1016/j.comcom.2019.10.002>.
99. R. Han, L. K. John, and J. Zhan, “Benchmarking Big Data Systems: A Review,” *IEEE Transactions on Services Computing*, vol. 11, no. 3, pp. 580–597, 2018, doi: <https://doi.org/10.1109/TSC.2017.2730882>.
100. V. Reniers, D. Van Landuyt, A. Rafique, and W. Joosen, “On the State of NoSQL Benchmarks,” in *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion*, ser. ICPE '17 Companion. New York, NY, USA: Association for Computing Machinery, 2017, pp. 107–112, doi: <https://doi.org/10.1145/3053600.3053622>.
101. A. Traeger, E. Zadok, N. Joukov, and C. P. Wright, “A Nine Year Study of File System and Storage Benchmarking,” *ACM Transactions on Storage*, vol. 4, no. 2, 2008, doi: <https://doi.org/10.1145/1367829.1367831>.
102. L. Wang *et al.*, “BigDataBench: a Big Data Benchmark Suite from Internet Services,” in *Proceedings of the 20th International Symposium on High Performance Computer Architecture*, ser. HPCA '14. New York, NY, USA: IEEE, 2014, pp. 488–499, doi: <https://doi.org/10.1109/HPCA.2014.6835958>.

RESEARCH ARTICLE

Endogenous Punishment-Driven Distributed Computing Environment for Data Factor Property Right Protection

Chuan Fu^{1,*} and Jian Ye¹¹Institute of Computing Technology, Chinese Academy of Sciences, 100190, Beijing, China*Corresponding author. chuanfu@ict.ac.cn

Received on 21 December 2025; Accepted on 23 September 2026

Abstract

The non-excludability and non-rivalry characteristics of data pose challenges to Coase's property rights theory. Based on Ostrom's Common Pool Resources (CPR) Governance Theory, this study proposes defining the data factor as a six-tuple and designing a computational environment that supports the protection of data factor property rights. This definition integrates the original dataset, transaction entities, commitments, supervision, and punishment into a unified whole, specifically referred to as data factor. Similar to the intellectual property protection system, the six-tuple first endows data factors with excludability in a contractual sense. Secondly, to effectively support such exclusivity, this study proposes a distributed property rights protection computing environment with the capabilities of "supervision-punishment-commitment", adopting the "terminal-pipe-cloud" architectural template. This computing environment better matches the self-organizing nature of data trading activities. We use evolutionary game theory to verify how the new data factor definition promotes the development of trading markets. To provide practical reference, the simulation focuses on two key aspects: market stability under the coexistence of multiple strategies, and the impact of the initial conditions of the institutional environment on participants' strategy choices.

Key words: Data Factor Property Rights Protection, Common Pool Resources Governance, Evolutionary Game, Endogenous Punishment, Network Address Translation(NAT)

1. Introduction

The wealth value of data does not depend on "how much" data one possesses, but on "how to utilize" it. However, the non-excludability and non-rivalry of data[1],[2] pose challenges to the property rights theory advocated by Coase. For instance, how to address the issue that an additional copy halves the data's price, and how to ensure that data is reliable, trustworthy, and usable. The property rights economic theory, which is based on the excludability and rivalry of factors, holds that clear property rights constitute the core prerequisite for ensuring market collaboration and orderly transactions[3]. The ongoing debate over the property rights of data factors precisely indicates that data has not yet fully shed its "pure resource" attribute.

Unlike general factors of production, whose excludability enables owners to sustain prices through use control, data factors are non-excludable. Anyone who gains access can replicate them at zero cost. This drives supply to infinity, prices toward zero, and ultimately severs the link between price and the intrinsic value of the data.

Data open-sourcing is a typical approach to demonstrating the value of data factors and exploring avenues for multi-party collaboration. Among its key components, open-source test datasets[4] serve as critical benchmarks for algorithm validation and model training, furnishing a unified evaluation baseline for technological iteration across diverse domains including artificial intelligence and computer vision[5]. To ensure stable and sustainable supply capacity, we need to prioritize the development dilemmas of open-source datasets[6] as well as the potential incentive mechanisms for them. The Sponsors program rolled out by GitHub, a world-renowned open-source platform, offers valuable implications for the development of the open-source ecosystem. Relevant study[7] have demonstrated that even when a large number of developers do not receive direct sponsorship after joining the program, their enthusiasm for engaging in contributory activities such as code optimization and vulnerability remediation is significantly boosted—and this constitutes the core foundation for them to build reputation and garner trust within the open-source community.

Exploring and refining incentive mechanisms for open data sharing not only serves to fully mobilize the enthusiasm of data contributors, but also accelerates the transformation of data from a general-purpose resource into an economically valuable production factor. In the context of data factors, we argue that—much like peer-to-peer (P2P)[8] systems—distributed environments not only offer the dual advantages of minimizing operational costs and streamlining data access, management, and maintenance for providers, but also, by supporting necessary game-theoretic behaviors, can endow data factors with the core attributes of exclusivity and competitiveness, thereby building a robust computing environment that underpins the protection of data factor property rights.

Elinor Ostrom, the 2009 Nobel laureate in Economics, probed into that whether a common resource (such as a pond) can possess “excludability” depends not only on its inherent natural attributes but also on the costs incurred by the appropriator(s) in attempting to prevent others from using it[9],[10]. To align the computing environment for data factor property rights protection with the requirements of Ostrom’s theory of common-pool resource governance, this study defines data factors in the form of a six-tuple denoted as $\langle P, Q, D, C, \text{punish}, \text{supervisor} \rangle$. Among them: P and Q represent the objectified descriptions of transaction entities; D denotes the transaction data; C stands for the contract signed by P and Q regarding D , which is also an objectified description before the entities of P and Q are determined; “punish” refers to the endogenous punishment or sanction function between entities (This punishment does not require third-party consent or implementation); “supervise” represents the supervision function between entities.

This 6-tuple indicates that when any party involved in the transaction detects a violation related to D , the transaction entities (i.e., P or Q) may impose punishment on the violator in accordance with C , and such punishment or sanctions occur directly between P and Q . The form of the 6-tuple clearly shows that the punitive operations connoted in data factors are intended to protect the interests of specific groups, namely second-tier excludability[10].

Private goods achieve “excludability” in the first-tier sense, whereas club goods and common-pool resources achieve “exclusivity” in the second-tier sense.[10]. Second-tier excludability refers to exclusivity that is realized within a certain scope and based on specific rules or conditions. From the perspective of excludability, the expectation of endogenous punishment can motivate entities involved in sharing to proactively avoid risks and reduce the possibility of data dissemination—similar to how patents are brought under legal protection. On the other hand, the convenience of imposing punishments during transactions destabilizes illegal trading markets[11], which can minimize the number of participants to the greatest extent.

Through the collaboration of “punish” and “supervise”, a mechanism is provided for groups that have already secured access to shared data factors, enabling them to prevent others from using the factors at low cost. The computational environment defined by this six-tuple supports the protection of data factor property rights and is better aligned with the self-organizing nature of data trading activities.

Based on the six-tuple of data factors, this study proposes a distributed property rights protection computing environment that can supervise data factor transaction behaviors, enhance the cooperation level of the transaction market, and features automatic data leakage detection and autonomous contract

breach penalty imposition. The main innovations are as follows:

1. General factors of production possess the attribute of excludability, which enables them to unify both value and price within a single transaction framework. Data factors, however, lack this attribute, causing their price to be gradually eroded during the process of value realization. Addressing the non-excludability characteristic of data—and shifting the analytical object accordingly—we propose a framework that bridges economic theory with technical implementation. By leveraging the incentives embedded in the technical implementation to influence transaction entities, the framework achieves the governance effects prescribed by economic theory, thereby resolving the fundamental contradiction inherent in treating data as a tradable factor. Resolving the non-excludability problem would facilitate the widespread development of data transactions.
2. Based on the CPR Governance Theory, this study proposes a six-tuple framework for describing data factors. Relying on the endogenous punishment mechanisms and customized contracts within the six-tuple, it realizes the excludability(second-tier excludability) and rivalry required for factors, thereby providing support for clarifying the boundaries of property rights.
3. According to the six-tuple of data factors, this study proposes a distributed data factor transaction and supervision system composed of “cloud”, “pipe”, and “terminal”. Specifically: the blockchain serves as the “cloud” to record transaction contracts; a virtual machine-based sandbox is used as the execution environment for data users, allowing data owners to monitor whether the execution process of training programs complies with specifications through the sandbox; by referring to telephone call blocking and utilizing signaling technology[12], we have realized the capability to autonomously punish untrustworthy entities (in a NAT (Network Address Translation) network environment) by blocking and restricting data transmission paths.
Blocking access paths[13],[14] and isolating violators from business cooperative relationships[15],[16] not only reduce their actual operational capabilities but also harm their business reputation through the industry credit transmission mechanism.
4. To verify the role of autonomous punishment in improving the cooperation level of the data transaction market, this study models the autonomous punishment behavior of traders using game theory, based on the prisoner’s dilemma principle. On this basis, we simulate four transaction strategy combinations using the Moran process. The simulation results show that under the constraint of the autonomous punishment mechanism, the probability of individuals choosing the cooperative strategy is much higher than that of choosing other strategies, and the group can benefit from cooperation more quickly. Such simulations can also help policymakers understand the role and value of each external condition in designing the initial system of the trading market.

In terms of the service goal, this system is designed around the theme that data trading requires promoting cooperation among subjects, controlling the dissemination of data among subjects, and supporting subjects in autonomously punishing

each other within a specific relationship, which is essentially different from systems that encourage information sharing. To the best of our knowledge, this is the first paper to define data factors through an extension based on datasets; it is also the first paper to discuss how to construct the excludability attribute of data factors from the perspective of punishment; and it provides, in conjunction with the definition of data factors, the implementation principles of a computing environment that specifically supports factor transactions.

The remainder of this article is structured as follows: First, it introduces the current contradictions in data factor tradings, proposes an extended approach to defining data factors, and briefly describes the innovations of this paper. Second, it introduces the CPR Governance Theory, and based on the six-tuple of data factors, proposes a computing environment supporting the trading market and its implementation mechanism. Third, drawing on simulations of four transaction strategy combinations, it uses evolutionary game theory to explain the important role of the mutual restriction mechanism (supporting autonomous punishment) in enhancing governance capabilities, maintaining the market's cooperation level, and maximizing cooperative benefits. Fourth, it reviews the current research status of data factors and discusses the property rights dilemmas of various schemes. Finally, it presents the conclusions and future work.

2. Data Factor and Property Rights Protection Computing Environment

2.1. Theoretical Basis for Data factor Definition

Ostrom's[9] research focuses on the governance of public goods (e.g., forests, pastures, roads, parking lots)—which also possess non-rivalry and non-excludability. She found that when tackling issues such as free-riding, shirking of responsibilities, and opportunistic behaviors, integrating mechanisms including community rules, supervision, and hierarchical governance can resolve the collective action dilemma, enable effective resource allocation, and achieve economic goals. The development of data factor trading markets also faces collective action dilemmas arising from opportunistic behaviors. Beyond government regulation and privatization models, Ostrom's theory offers a new perspective for defining data factors and constructing relevant trading systems.

Different from property rights economic theory, the governance of common pool resources (CPR)[9] is based on the clear definition of boundaries. The core purpose of clearly defining boundaries is to clarify responsibilities, as well as to fulfill obligations and obtain returns in accordance with these responsibilities. In practice, clearly defining boundaries means that each participant makes explicit commitments to themselves and others. Such commitments do not stem from external imposition but must be self-implemented by the involved parties. The governance solution proposed by Ostrom aims to enable interdependent principals to organize themselves autonomously and achieve sustained collective benefits—even when faced with temptations such as free-riding.

Ostrom argued that commitments and supervision must be strategically nested: effective supervision cannot be achieved solely by relying on external forces; only by integrating supervision and hierarchical sanctions into a complete set of institutional principles and operating them in coordination can strong public institutions be established or reshaped. Supervision is crucial, yet third-party supervision is not the

only approach. When individuals commit to a contingency strategy for abiding by rules—i.e., "I act as others act"—they will be motivated to supervise others to ensure that most people comply with the rules. We argue that supervision and sanctions (i.e., punishment) together constitute effective oversight.

Based on the CPR Governance Theory, this study defines data factors using a six-tuple framework: $\langle P, Q, D, C, \text{punish}, \text{supervise} \rangle$, which essentially serves to define boundaries. The components are defined as follows:

- **P and Q:** Objectified descriptions of transaction entities, i.e., abstract descriptions of the entities' characteristics. All entities that meet the requirements are potential counterparts to these entities (for convenience, P refers to the data owner, and Q refers to the data user).
- **D:** Denotes the transaction data.
- **C:** Represents the contract (i.e., commitment) signed by P and Q regarding D. Before the entities of P and Q are determined, this contract also functions as an objectified description.
- **punish (C, punisher, defaulter):** Represents the punishment or sanction function between entities. The return value of this function is either 0 or 1, where 0 indicates no punishment and 1 indicates the imposition of punishment. The specific form of punishment or sanction shall be determined in accordance with Contract C.
- **supervise (D, C, proctor):** Refers to the inter-entity supervision function. The return value of this function is either 0 or 1, where 0 indicates compliance with relevant agreements and 1 indicates a violation of such agreements. The specific supervision actions and rules are determined in accordance with Data D and Contract C.

This six-tuple implies that when any party involved in the transaction detects a violation related to D, the transaction entity (either P or Q) can impose punishment on the violator in accordance with C, and the punishment or sanction occurs directly between P and Q. The values of "punisher", "defaulter", and "proctor" correspond to the entities represented by P or Q.

When understanding the definition of data factors from the perspective of excludability in property rights economic theory, the expectation of endogenous punishment can motivate entities to proactively avoid risks and abandon the use of relevant datasets. Endogenous punishment first directly leads to losses in expected benefits and prior costs; second, it indirectly affects the entity's business reputation and other ongoing transactions involving the entity. When the total cost of infringement exceeds the benefits of infringement, punishment measures achieve excludability by increasing various costs[17]. Data factors described by the six-tuple can exhibit the excludability required of production factors. On the other hand, the convenience of imposing punishments during transactions destabilizes illegal trading markets[11], which can minimize the number of participants to the greatest extent.

When the dataset is determined, the expression $\langle P, \emptyset, D, C(\emptyset), \text{punish}, \text{supervise} \rangle$ represents the basic data factor released by the data owner. The customized contracts signed by different entities, along with the connections between contracts formed around the same dataset, collectively endow the six-tuple-described data factors with the characteristic of rivalry. Based on the CPR Governance Theory, the six-tuple data factors defined in this study can externally exhibit the characteristics of production factors that are adapted to market transactions.

2.2. Property Rights Protection Computing Environment

To support the trading of six-tuple data factors, this study proposes a distributed property rights protection computing environment for data factor trading, which consists of "cloud", "pipe", and "terminal".

2.2.1. System Structure of the Computing Environment

The six-tuple-based computing environment consists of three components: "cloud", "pipe", and "terminal", as illustrated in Figure 1.

- **Cloud:** The blockchain serves as the "cloud" to record transaction contracts (i.e., C in the six-tuple), which represent commitments.
- **Pipe:** By referring to telephone call blocking and utilizing signaling technology[12], we have realized the capability to independently punish untrustworthy entities (i.e., "punish" in the six-tuple) in a NAT (Network Address Translation) network environment by blocking and restricting data transmission paths.
- **Terminal:** This component is divided into two categories:
 - Terminal systems for data usage (i.e., Q in the six-tuple) are virtual machine-based sandboxes serving as execution environments for data users. Data owners monitor whether the execution process of training programs complies with specifications through the sandbox (i.e., "supervise" in the six-tuple).
 - Terminal systems for data owners (i.e., P in the six-tuple) provide data access environments for data owners and collect information about data users' training data usage.

Both types of terminals access the internet through NAT devices, with data transmission paths between NAT devices established using the SIP protocol.

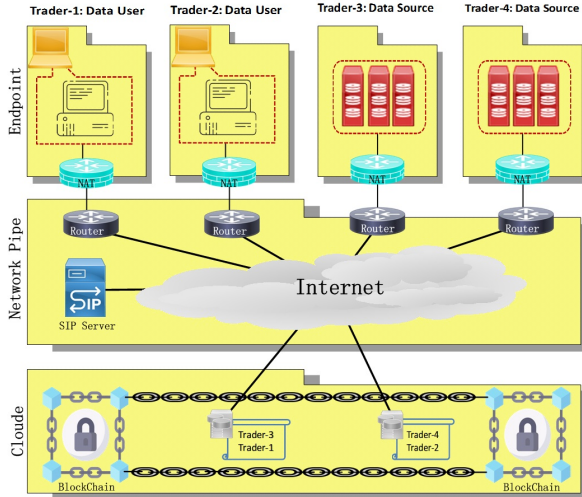


Fig. 1. A computing environment supporting property rights protection.

2.2.2. Commitment and Monitoring

A distributed file system similar to IPFS is used to store contract texts[18],[19]. When a contract text is uploaded, it is

first hashed, and the resulting file hash value is then uploaded to the blockchain for storage. The integration of distributed storage and blockchain technology ensures the tamper-proof nature of the commitments recorded in the contract, providing traceable and reliable guarantees for enforcing punishment in the event of infringement.

The sandbox of the first type of terminal system (for data usage) is capable of tracking the information flow during the training process and restricting the scope of information diffusion. This type of terminal system downloads data from the second type of terminal system (for data owners) and stores the data in a virtual space. Drawing on the privacy-preserving data protection methods for Android systems proposed in References[20],[21],[22], we adopt data contamination marking and attribute-based access control as the core mechanisms for sandbox monitoring of data leakage. These mechanisms can track the original data and the model parameters derived from the original data, and restrict their diffusion scope.

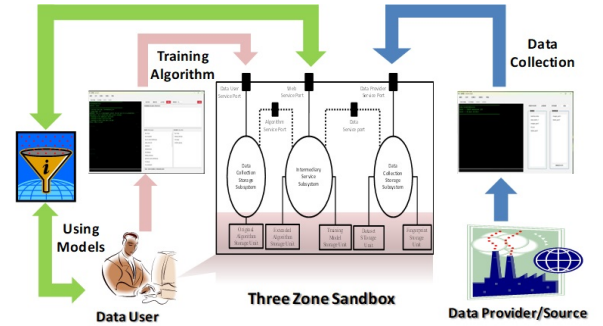


Fig. 2. Three-space sandbox and its interaction process.

Based on the confidentiality and integrity requirements of the security access mode[23], we partition the continuous model training environment into three mutually independent subspaces(zone): the data provider subspace, the data user subspace, and the intermediary service subspace—forming the "three-space sandbox", as illustrated in Figure 2. The data provider subspace stores datasets, the data user subspace stores algorithms, and the intermediary service subspace executes model training and provides services to the outside.

The access relationships among the three subspaces are determined based on the dataset. The dataset is confirmed as complete by both parties prior to the transaction, and this integrity serves as the foundation for subsequent cooperation. Based on confidentiality requirements, the original data has a higher protection level than the algorithm code; therefore, the data provider subspace is assigned a higher level than the data user subspace. The intermediary service subspace, which performs auditing, has the highest level of privilege.

AI model training follows a fixed data flow: sample reading, preprocessing (normalization/tensor conversion), forward propagation, loss computation, and backpropagation. The training programs provided by data users typically rely on existing third-party libraries. We decompose this workflow across the three subspaces and insert checkpoints at key stages. When the data provider uploads data, the data provider subspace generates a sample fingerprint for each record in the original dataset and stores these fingerprints in a fingerprint table[24]. Upon receiving the algorithm code submitted by the data user, the intermediary service subspace scans the

code using syntactic analysis tools and inserts integrity-checking code before the preprocessing (normalization/tensor conversion) stage. During execution, the integrity-checking code computes the fingerprint of the sample being processed and compares it against the fingerprint table. If no matching record is found, the checking code triggers an alarm. The relative privilege levels among the subspaces ensure that the data user cannot detect the intermediary service's inspection of the algorithm code.

To broaden the coverage of parties participating in data trading, the core purpose of the sandbox is supervision, not absolute protection, and it is designed to simulate the canary in a coal mine. In normal operation, the sandbox achieves mutual isolation between different subjects, as well as mutual isolation between data and the original execution program. When a violation occurs, the data integrity and confidentiality defined by the sandbox will be breached, thereby triggering an alarm to the data provider. This will lead the data provider to initiate punishment. Once a violation is detected, the deterrence comes from the endogenous punishment mechanism—the blocking of signaling and data paths—rather than from the sandbox itself.

2.2.3. Network Environment Supporting Endogenous Punishment

Endogenous punishment serves as the foundation for data factors to possess excludability. Existing studies have applied the PageRank algorithm[15],[16] to evaluate the reputation of nodes in supply networks—nodes in an isolated state within the network fail to obtain favorable evaluations. Therefore, we consider restricting the punished entity's access to the network as a punishment measure[13],[14]. When the data transmission path is blocked, this is equivalent to isolating the defaulter from business cooperative relationships, which directly diminishes their operational capacity and business reputation.

Telecom network operators can compel users with unpaid fees to settle their payments promptly by blocking their calls. As illustrated in Figure 3, based on the SIP protocol and NAT devices, we propose a network environment that not only supports data exchange between transacting parties but also enables the autonomous implementation of punishment. Using NAT (Network Address Translation) devices to provide internet access services to users is a common solution for addressing IP address shortages, and the NAT network environment lays the foundation for us to implement punishment. When a transaction entity accesses the public network via a NAT device using a private address, it requires the assistance of a third-party service deployed on the public network (i.e., a transaction matcher) to communicate with another transaction entity located behind a NAT device. The technical solution for establishing data transmission channels between NAT devices based on the SIP (Session Initiation Protocol) has been widely adopted in the internet. Once the tunnel is established, the two parties can perform data transmission and status synchronization.

As illustrated in Figure 4, when services are provided to transaction entities based on the SIP protocol, the transaction matcher controls the mutual access capabilities of all entities. In the aforementioned network environment, taking the scenario where the first type of terminal detects abnormal behavior as an example, the punishment implementation process is as follows:

1. In accordance with the punishment clauses specified in the contract between the two transaction parties,

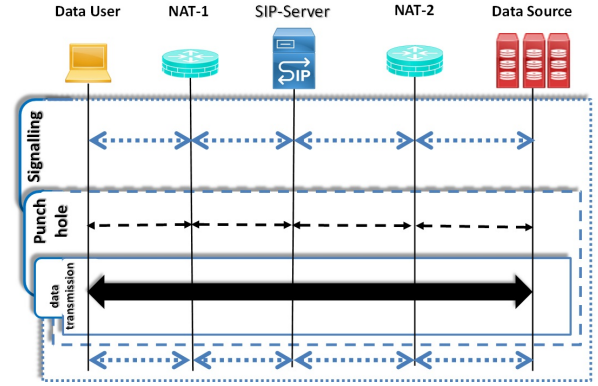


Fig. 3. Process of Establishing Data Paths Between NAT Devices

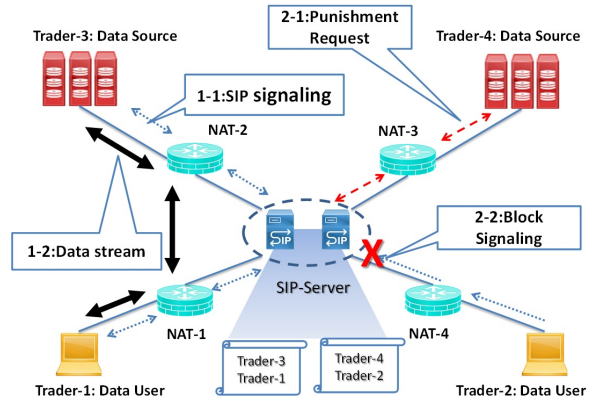


Fig. 4. Process of Implementing Punishment

the transaction matcher can configure an automatic punishment implementation service for the transaction entities.

2. When an entity submits a punishment request, the automatic punishment implementation service is triggered to restrict or block the punished entity from establishing data transmission paths with other entities. The punished entity may also file a reverse punishment request against the initiator of the original punishment request. The transaction matcher will similarly restrict the punishment initiator in accordance with the provisions of the punishment clauses. At this point, the ability of both parties to access other entities in the system will be frozen until the dispute is resolved.
3. During the freezing period, the two parties negotiate to reach a resolution—similar to how PPP (Public-Private-Partnership) models[25] adjust project outputs, modify cost-sharing ratios, or even terminate cooperation through negotiation.
4. After the two transaction parties resolve the dispute, the initiator of the original punishment request submits a revocation request. The transaction matcher then automatically restores both parties' ability to access other entities in the network.

Under normal circumstances, a trader will not initiate a punishment request without cause. From a game-theoretic perspective, this is a lose-lose state and attracts attention from multiple parties. However, it cannot be ruled out that someone

may raise a false alarm or deliberately initiate punishment. Therefore, before implementation, it is necessary to confirm that the sandbox alarm actually occurred. SIP is a protocol with rich tags and complete call-sequence records. Because systems behind different NAT devices need to communicate via the SIP protocol, and SIP has the capability to record sessions in the call server, when a subject submits a punishment request, the call service will check its local records to determine whether the cooperative session exists—that is, whether the alarm message was actually issued by the sandbox. In future work, we will treat security issues in data trading as an independent research problem, and analyze and study security threats from multiple aspects, ranging from system-level attacks to transaction-level attacks.

3. From Endogenous Punishment to Cooperation

To analyze the role of punishment-based restraint mechanisms in enhancing entities' willingness to cooperate, we categorize the strategies available to entities participating in data transactions into four basic types: cooperation, defection, loner, and punishment. In each round of the game, transaction entities can independently choose a strategy based on their own objectives.

- **Cooperation Strategy:** Refers to a strategy where an entity is willing to invest resources (i.e., incur costs) in cooperation and share the resulting benefits.
- **Defection Strategy:** Refers to a strategy where an entity refuses to invest resources (i.e., avoids incurring costs) in cooperation but still seeks to share the benefits.
- **Punishment Strategy:** Refers to a strategy where an entity is willing to invest resources (i.e., incur costs) in cooperation, share the resulting benefits, and proactively punish defectors upon detecting defection—while also bearing the costs associated with the punishment itself.
- **Loner Strategy:** Refers to a strategy where an entity generates value independently using its data, without engaging in cooperation with other entities.

Based on the above strategies, we define four strategy combinations: Cooperation-Defection, Cooperation-Defection-Loner, Cooperation-Defection-Punishment, and Cooperation-Defection-Loner-Punishment. Using the social prisoner's dilemma model as a foundation, we discuss the cooperation tendencies of entities under various combinations by analyzing the fixation probability in evolutionary games. The so-called fixation probability refers to the probability that a strategy spreads from a single mutant individual within a population to eventually dominate the entire population[26],[27].

3.1. Profit Analysis of Transaction Strategies in the Game

In the data transaction market, some entities are responsible for data collection, while others are tasked with data processing. The benefits generated from data are the result of collective efforts. Prior to cooperation, participants cannot observe the strategies adopted by other entities in the market. For the sake of simplicity, we assume that cooperative benefits are evenly distributed among all entities involved in the cooperation.

With reference to Hauert[28], we assume there are M entities participating in data factor tradings in the market, including

X cooperators, Y defectors, Z loners, and W punishers (where $X + Y + Z + W = M$). From a population of M entities, N entities are randomly selected to form a game group for project implementation. δ represents the profit of loners. c represents the cost incurred by cooperators (or punishers). The value created through cooperation is r times the cost c (i.e., r denotes the rate of return). δ provides a reference baseline for cooperative benefits—specifically, it represents the market value that an entity can generate by using a given dataset independently without engaging in cooperation. The value of δ may increase with network effects and decrease accordingly.

β denotes the punishment intensity coefficient with which punishers sanction defectors, or equivalently, the punishment intensity perceived by defectors. $\alpha\beta$ denotes the punishment coefficient with which punishers sanction entities that refuse to participate in punishing defectors, or equivalently, the punishment intensity perceived by those non-participating entities. When terminals communicate via SIP signaling, the call server that provides the service typically also offers a presence service. When a punisher initiates a punishment request, the call server blocks all outgoing connection requests from the punished party, and the punished party's login status is simultaneously set to "dispute frozen." Non-participating entities can query this presence service to obtain the current status of other nodes (e.g., online, dispute frozen, etc.). γ and $\alpha\gamma$ represent the costs borne by punishers for the two aforementioned punishment behaviors, respectively. We assume $N > r > (1 + \delta/c)$ and $\beta > \gamma, 1 > \alpha > 0$. Based on the hypergeometric distribution[29],[30], the average profits corresponding to the above strategies are as follows:

$$\begin{aligned} P_x &= \frac{\binom{Z}{N-1}}{\binom{M-1}{N-1}} \delta + B(Z) - F(Z)c - \frac{W}{M-1} (N-1)G(Y)\alpha\beta \\ P_y &= \frac{\binom{Z}{N-1}}{\binom{M-1}{N-1}} \delta + B(Z) - \frac{W}{M-1} (N-1)\beta \\ P_z &= \delta \\ P_w &= \frac{\binom{Z}{N-1}}{\binom{M-1}{N-1}} \delta + B(Z) - F(Z)c - \frac{Y}{M-1} (N-1)\gamma - \frac{X}{M-1} (N-1)G(Y)\alpha\gamma \end{aligned} \quad (1)$$

where,

$$\begin{aligned} B(Z) &= rc \frac{X+W}{M-Z-1} \left(1 - \frac{1}{N(M-Z)} (M - (Z-N+1)) \frac{\binom{Z}{N-1}}{\binom{N-1}{N-1}} \right) \\ F(Z) &= 1 - \frac{r}{N} \frac{M-N}{M-Z-1} + \frac{\binom{Z}{N-1}}{\binom{M-1}{N-1}} \left(\frac{r}{N} \frac{Z+1}{M-Z-1} + \frac{r(M-Z-2)}{M-Z-1} - 1 \right) \\ G(Y) &= 1 - \frac{M-1}{M-Y-1} \frac{\binom{M-Y-1}{N-1}}{\binom{M-1}{N-1}} \end{aligned} \quad (2)$$

Apart from the scale parameters, all game-related parameters in this paper are illustrative parameters that reflect the economic behavioral characteristics of transacting subjects, rather than parameters of the information system.

3.2. Simulation of Game Behavior Among Data Transaction Entities

To secure higher profits, entities will adjust their strategies based on their opponents' strategies, environmental factors,

Table 1. Game Profits of the Four Strategies

	Cooperation(X)	Defection(Y)	Loner(Z)	Punishment(W)
Cooperation(X)	$(r-1)c$	$\frac{rc}{N} \left[1 + \frac{(X-1)(N-1)}{M-1} \right] - c$	$(r-1)c - \frac{\binom{Z}{N-1}}{\binom{Z}{N-1}} [(r-1)c - \delta]$	$(r-1)c$
Defection(Y)	$\frac{rc}{N} \left[\frac{X(N-1)c}{M-1} \right]$	$\frac{\binom{Y}{N-1}}{\binom{Y}{N-1}} \delta$	$\frac{\binom{Z}{N-1}}{\binom{Z}{N-1}} \delta$	$\frac{W}{M-1} (N-1) (\frac{rc}{N} - \beta)$
Loner(Z)	δ	δ	δ	δ
Punishment(W)	$(r-1)c$	$\frac{rc}{N} (1 + \frac{(W-1)(N-1)}{M-1}) - c - \frac{(M-W)(N-1)}{M-1} \gamma$	$(r-1)c - \frac{\binom{Z}{N-1}}{\binom{Z}{N-1}} [(r-1)c - \delta]$	$(r-1)c$

and game outcomes during repeated games. By integrating environmental influences with individual strategies, we use the Moran process to simulate the four strategy combinations. The specific game structure is presented in Table 1.

In the Moran process, the parameter $\omega \in [0, 1]$ —used to calculate fitness[27]—represents the intensity of natural selection. When ω is close to 0, the external environment in which players operate exerts a greater impact on fitness; when ω is close to 1, the interactions between players have a more significant impact on fitness. Fitness reflects the combined effect of dominance and game dynamics.

In accordance with the requirements of the Moran process, after the population is fully mixed, four types of entities—cooperators, defectors, loners, and punishers—engage in random pairwise games (self-gaming is not considered). For example, when a cooperator interacts with a defector, the profit of the cooperator is $\frac{rc}{N} \left[1 + \frac{(X-1)(N-1)}{M-1} \right] - c$, denoted as P_{XY} . The game profits between other types of entities are presented in Table 1.

Let x, y, z , and w represent the counts of the four types of entities participating in the project, where $N = x + y + z + w$. For convenience, we use the 3D array (x, y, z) to denote a combined state of the four types. Based on the definition of the Moran process, the fitness values corresponding to the four types are as follows:

$$\begin{vmatrix} f_{x,y,z} \\ g_{x,y,z} \\ h_{x,y,z} \\ l_{x,y,z} \end{vmatrix} = (1 - \omega) + \frac{\omega}{N-1} \Pi \quad (3)$$

where,

$$\Pi = \begin{vmatrix} (x-1)P_{XX} & (y)P_{XY} & (z)P_{XZ} & (N-x-y-z)P_{XW} \\ (x)P_{YX} & (y-1)P_{YY} & (z)P_{YZ} & (N-x-y-z)P_{YW} \\ (x)P_{ZX} & (y)P_{ZY} & (z-1)P_{ZZ} & (N-x-y-z)P_{ZW} \\ (x)P_{WX} & (y)P_{WY} & (z)P_{WZ} & (N-x-y-z-1)P_{WW} \end{vmatrix} \mathbf{1}_4 \quad (4)$$

Among them, $f_{x,y,z}$, $g_{x,y,z}$, $h_{x,y,z}$, and $l_{x,y,z}$ represent the fitness of cooperators, defectors, loners, and punishers, respectively. Based on the fitness of each type of entity, we can calculate the transition probability between different entity types. It is assumed that when an entity type transition occurs, only one entity type participates in the transition, and only one individual from this type is involved in the transition. The transition probability is calculated as the product of the proportion of the current entity type in the entire population and the proportion of the target entity type in the entire population under fitness weighting. $\frac{yy}{zz} T_{xx}$ represents the probability that no entity type transition occurs. $\frac{yy+1}{zz+1} T_{xx-1}$ represents the probability that one individual from entity type

x transitions to entity type y ; other similar notations follow the same interpretation. The specific calculation method is as follows:

$$\begin{vmatrix} \frac{yy}{zz} T_{xx} \\ \frac{yy+1}{zz+1} T_{xx-1} \\ \frac{yy}{zz+1} T_{xx-1} \\ \frac{yy}{zz} T_{xx-1} \\ \frac{yy-1}{zz} T_{xx+1} \\ \frac{yy-1}{zz+1} T_{xx} \\ \frac{yy-1}{zz} T_{xx} \\ \frac{yy}{zz-1} T_{xx+1} \\ \frac{yy+1}{zz-1} T_{xx} \\ \frac{yy}{zz-1} T_{xx} \\ \frac{yy}{zz} T_{xx+1} \\ \frac{yy+1}{zz} T_{xx} \\ \frac{yy}{zz+1} T_{xx} \end{vmatrix} = \mu \begin{vmatrix} x & y & z & w \\ y & 0 & 0 & 0 \\ z & 0 & 0 & 0 \\ w & 0 & 0 & 0 \\ 0 & x & 0 & 0 \\ 0 & z & 0 & 0 \\ 0 & w & 0 & 0 \\ 0 & 0 & x & 0 \\ 0 & 0 & y & 0 \\ 0 & 0 & w & 0 \\ 0 & 0 & 0 & x \\ 0 & 0 & 0 & y \\ 0 & 0 & 0 & z \end{vmatrix} \begin{vmatrix} xf_{xyz} \\ yg_{xyz} \\ zh_{xyz} \\ wl_{xyz} \end{vmatrix} \quad (5)$$

where,

$$\mu = \frac{1}{(xf_{x,y,z} + yg_{x,y,z} + zh_{x,y,z} + (N-x-y-z)l_{x,y,z})M} \quad (6)$$

Figure 5 illustrates a potential evolutionary outcome under the fourth game scenario. After multiple rounds of games, cooperators and punishers emerge as the dominant groups within the population.

3.3. Analysis of the Evolutionary Tendency of Transaction Entities

An evolutionarily stable strategy (ESS) refers to a strategy adopted by the majority of a population's members. The benefits this strategy confers on entities cannot be matched by any other strategy. In evolutionary games involving randomness in finite populations, fixation probability is a key indicator for analyzing whether a strategy can qualify as an evolutionarily stable strategy.

To calculate the fixation probability in multi-strategy games, we treat the transition matrix as a process that describes strategy transformation when an independent strategy competes against a set of other strategies. All strategies except the independent one form a "super strategy." We use the calculation method proposed in[27] to compute the fixation probability of this independent strategy. When calculating the fixation probability of other strategies, we sequentially adjust the independent strategy and its corresponding super strategy. When the individual state of the

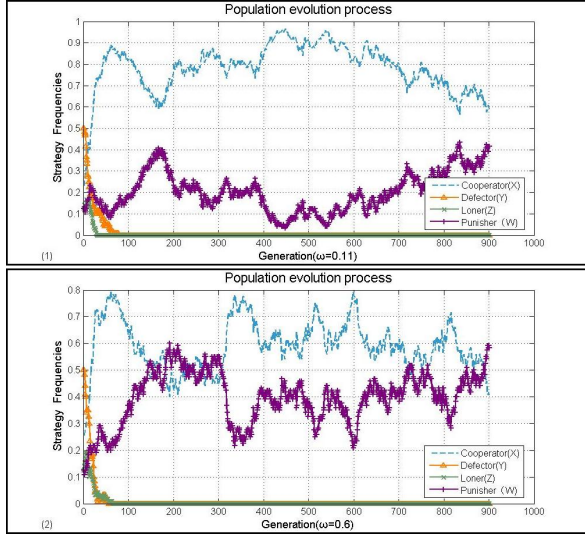


Fig. 5. Population evolution process

independent strategy is 1, it corresponds to multiple individual states of the super strategy; in such cases, we calculate the fixation probability of the independent strategy by taking the average value.

Below, we analyze the cooperation tendencies of entities across the four strategy combinations, based on the relationship between the rate of return r and fixation probability. Figure 6 illustrates the changes in fixation probability for the four combinations when the profit of loners is 0.2. Figure 7 illustrates the changes in fixation probability for the four combinations when the profit of loners is 0.5. Figure 8 illustrates the changes in fixation probability for the four combinations when the profit of loners is 1. The profit of loners serves as the baseline return that individuals in the population can obtain through independent efforts.

The gray plane in each figure represents the reference probability. For example, in the "Cooperator-Defector-Loner-Punisher" scenario, the probability of participants choosing each strategy is 0.25. We consider that when the fixation probability of a particular strategy exceeds the reference plane, this strategy exhibits a selection bias within the population. When all strategies fall below the reference plane, the population has no distinct preferred strategy, and individuals tend to adopt the strategy with a relatively higher fixation probability. The parameters used in the simulations, such as population size, punishment cost, and punishment intensity, are based on references [31][32][33].

Figure 6 to Figure 9 indicate that when individuals select a game strategy, they must comprehensively consider, at minimum, factors such as the return from cooperation, the intensity of natural selection (ω), the profit of loners (δ), and the punishment intensity exerted by punishers.

Figure 6 to Figure 8 demonstrate that, with cooperative profits held constant: as the intensity of external environmental selection (ω) increases, the fixation probability of cooperators rises significantly, while that of defectors decreases significantly; with the intensity of external environmental selection (ω) held constant, as the rate of return (r) increases, the fixation probability of cooperators rises significantly, while that of defectors decreases significantly; when the fixation probability

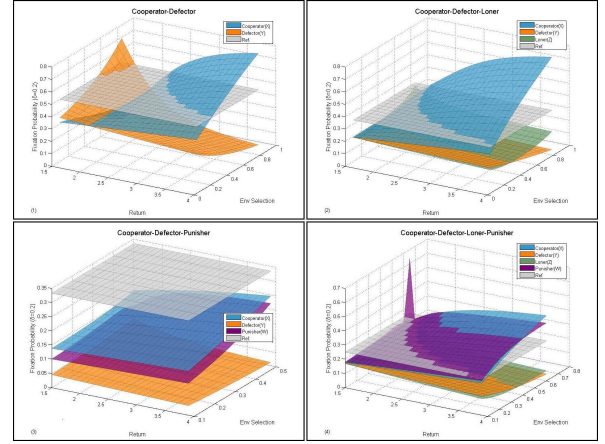


Fig. 6. Fixation Probability ($\delta = 0.2$). $M = 120$ $N = 6$ $c = 1$ $1.5 < r < 3.9$ $0.1 < \omega < 0.85$ $\alpha = 0.1$ $\beta = 1.1$ $\gamma = 0.3$ (1) $X = 60$ $Y = 60$ (2) $X = 40$ $Y = 40$ $Z = 40$ (3) $X = 40$ $Y = 40$ $W = 40$ $0.1 < \omega < 0.5$ (4) $X = 30$ $Y = 30$ $Z = 30$ $W = 30$ $0.1 < \omega < 0.65$

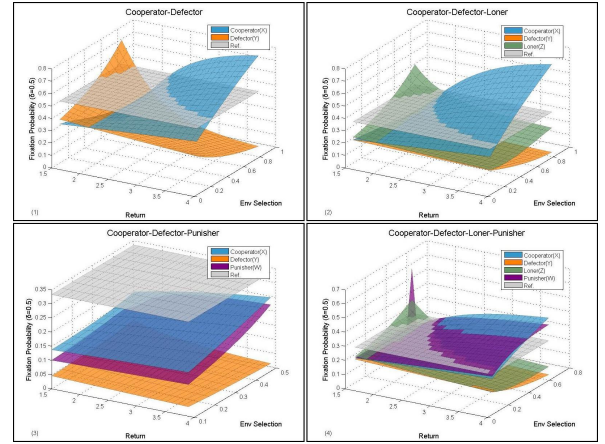


Fig. 7. Fixation Probability ($\delta = 0.5$). $M = 120$ $N = 6$ $c = 1$ $1.5 < r < 3.9$ $0.1 < \omega < 0.85$ $\alpha = 0.1$ $\beta = 1$ $\gamma = 0.3$ (1) $X = 60$ $Y = 60$ (2) $X = 40$ $Y = 40$ $Z = 40$ (3) $X = 40$ $Y = 40$ $W = 40$ $0.1 < \omega < 0.5$ (4) $X = 30$ $Y = 30$ $Z = 30$ $W = 30$ $0.1 < \omega < 0.65$

is lower than the reference probability, it indicates high uncertainty in the strategy selected by individuals.

By comparing (1), (2), and (3) in Figure 6: the presence of loners and punishers leads to a rapid downward trend in the fixation probability of defectors; a comparison between (2) and (3) shows that punishers can more effectively suppress individuals' defection behavior in the environment, however, under the same profit conditions, punishers also inhibit individuals' cooperative behavior; additionally, in the "Cooperator-Defector-Punisher" combination, when the intensity of external environmental selection (ω) is relatively high, punishers will penalize all participants, resulting in abnormal market operation.

A comparison between (2) and (4) shows that when the investment return (r) has increased significantly, the fixation probability of loners in the "Cooperator-Defector-Loner" combination remains relatively high—meaning there is a high probability that individuals will not engage in data transactions with one another; the trend in (4) indicates that, in the fourth strategy combination, the fixation probabilities

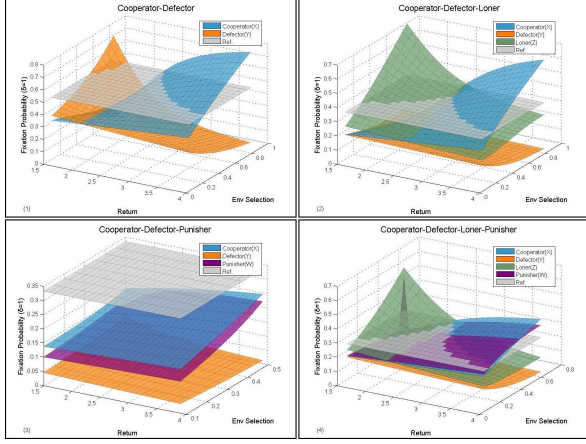


Fig. 8. Fixation Probability ($\delta = 1$) $M = 120$ $N = 6$ $c = 1$ $1.5 < r < 3.9$ $0.1 < \omega < 0.85$ $\alpha = 0.1$ $\beta = 1.1$ $\gamma = 0.3$ (1) $X = 60$ $Y = 60$ (2) $X = 40$ $Y = 40$ $Z = 40$ (3) $X = 40$ $Y = 40$ $W = 40$ $0.1 < \omega < 0.5$ (4) $X = 30$ $Y = 30$ $Z = 30$ $W = 30$ $0.1 < \omega < 0.65$

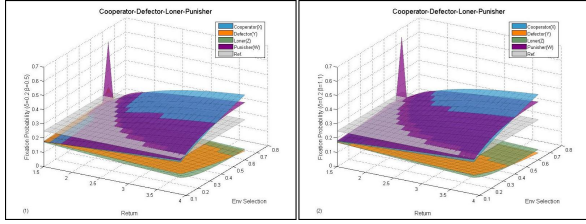


Fig. 9. Fixation Probability ($\delta = 0.2$) $M = 120$ $N = 6$ $X = 30$ $Y = 30$ $Z = 30$ $W = 30$ $0.1 < \omega < 0.65$ $c = 1$ $1.5 < r < 3.9$ $\alpha = 0.1$ $\gamma = 0.3$ $0.1 < \omega < 0.85$ $\alpha = 0.1$ $\gamma = 0.3$ (1) $\beta = 0.5$ (2) $\beta = 1.1$

of loners and defectors are significantly lower than those in (2); since punishers are a special type of cooperator, the fixation probability of the population's tendency to cooperate is significantly higher than that in (2); when the fixation probability of cooperators exceeds the reference plane, the rate of return in (4) is lower than that in (2), yet the population can obtain returns through cooperation earlier; both Figure 7 and Figure 8 exhibit this upward trend.

By comparing the fourth strategy combination across Figure 6 to Figure 8, we observe that as the profit of loners (δ) gradually increases, more efficient individual productivity delays group cooperation. Figure 9 illustrates that increasing punishment intensity can effectively suppress the tendency to defect and accelerate the fixation probability of the cooperative tendency to exceed the reference plane.

The above simulation results demonstrate that the incentive mechanism embedded in the technical scheme combining the SIP protocol with the NAT network can achieve the governance effect proposed by Common Pool Resources governance theory—namely, that the majority of individuals in the population adopt cooperative strategies. This study provides a technology-supported institutional system for non-excludable data factors. Only on the basis of traders' stable cooperation can the value of data factors—such as non-rivalry and zero replication cost—be realized.

4. Related Research

We believe that current research on data (factor) transactions can be categorized into three perspectives: value chain research centered on datasets, research on data transaction market construction schemes, and research on governance methods for data factor transaction markets. These studies reflect the process by which data evolves from specific datasets to an abstract concept.

The data value chain originated from enterprises' needs for massive data management[34] and has received widespread attention with the development of big data technologies[35]. In industry-specific research—such as in supply chains[36], finance[37], agriculture[38], and healthcare[39]—people have further recognized the new value that the data value chain brings to industries.

When data transactions need to be conducted across organizations and countries[40], constructing transaction environments has become a new research hotspot. Transaction environments have evolved from cloud-based environments[41] to distributed transaction environments[42]. Studies[42],[43],[44] have explored how to use blockchain technology to address security and credibility issues in data transactions within distributed environments. Research[45],[46] aims to promote cross-border data flow and transactions through more reasonable profit distribution mechanisms.

Research on transaction environment construction schemes has also stimulated in-depth exploration of governance mechanisms for data factor tradings. Governance mechanisms are divided into two types: governance centered on organizing units or departments, and governance based on polycentric governance theory. Governments[47],[48], internet platforms[49], and data intermediaries[50] all play organizing roles. In the "platform society"[49], a large amount of data is controlled by a few large technology platforms, making platforms a natural dominant model for data governance. By applying evolutionary game theory, the value of regulators in open data governance is elaborated[51]. Attention needs to be paid to balancing differences between individuals, and between individuals and policymakers[52].

On the other hand, data stored dispersedly among numerous interconnected entities forms a "datasphere"[53]. In environments where governments, markets, societies, and non-governmental organizations coexist, studies[54] have discussed how to use polycentric governance theory[55],[56] to promote coordination and cooperation among different entities and establish a more efficient market. While polycentric governance theory is increasingly accepted as a governance concept, relative differences between entities in the datasphere lead to divergences in their governance demands.

The abstraction process directly defines data as data factors but fails to resolve the inherent dilemmas of non-excludability and non-rivalry in data itself. Unlike existing research, while abstracting data, we have defined new connotations for data factors through an extended approach.

5. Conclusion and Future Work

The non-excludability of data is the core focus of our research. Compared with data factors, other factors of production possess excludability. The boundaries of excludable factors are easier to define, and property-rights proof can be established through registration and query, thereby achieving protection at both the physical and the property-rights levels; the boundaries of

non-excludable factors, by contrast, are difficult to define, and such factors cannot form effective proof through themselves, so they require an institutional system to constrain users' behavior in order to achieve the effect of protecting the factor. Accordingly, the governance of excludable factors can rely more on private property-rights mechanisms, whereas data factors, because they cannot define their own boundaries, rely more on Common-Pool Resource (CPR) governance mechanisms that take the constraint of users' behavior as their core.

This therefore requires that the information system be grounded in CPR governance mechanisms, so as to control the orderly flow of data factors among specific subjects and thereby create value. At the present stage, this research has identified the point of convergence between economic theory and technical solutions, linking economic and technical issues so that the next stage of development can be applied more rapidly to the construction of data trading markets. In terms of the service goal, this system is essentially different from systems that encourage information sharing.

Building upon Ostrom's CPR governance framework outlined above, we define the data factor as a six-tuple. Meanwhile, we develop a technical system embedded with the "supervision-punishment-commitment" mechanism to enable the governance of factor excludability. Relying on the "terminal-pipe-cloud" architecture, this technical system can construct an operating environment for data-factor transactions.

To verify the role of this new data factor definition in constructing transaction markets, we employed evolutionary game theory to simulate two key aspects: first, market stability under the coexistence of strategies (i.e., cooperation, defection, loner, and punishment); second, the impact of initial market institutions on participants' strategy choices. In a certain sense, when a dataset can be converted into a six-tuple format, it can become a tradable data factor.

The punishment behavior supported by the combination of the SIP protocol and the NAT network—namely, connection blocking—is essentially a pledge mechanism collateralized by the punished party's entire business cooperation value. Its effect directly impacts the punished party's ongoing and subsequent transactions, thereby effectively freezing its business relationships in the market. Compared with smart contract slashing, the blocking approach does not require defining the relationship between the collateral value and the transaction value, which simplifies transaction rules, reduces participation costs, and eliminates potential arbitrage opportunities. Reputation systems can help traders select more trustworthy counterparts; however, from a governance mechanism perspective, reputation systems only accomplish the "commitment-supervision" phase and lack an effective punishment mechanism.

The primary task for the next step is to expand the application scope of computing environments that support factor property rights protection in data trading scenarios, and to extend into data-driven cooperation and collaboration fields. To improve system reliability, we will use the SIP OPTIONS method to periodically query the status of network elements and the platform during prototype system development, and collect statistics on the effectiveness of the punishment mechanism. We will validate the prototype through a multi-party data collaboration scenario, and investigate system metrics such as the impact of fingerprint verification on model training performance (e.g., the percentage increase in sample reading time), the response time of the

punishment mechanism (connection blocking), and system throughput under concurrent transaction scenarios. We also expect to provide configuration recommendations for practical deployment by combining system performance analysis with the scenario. Secondly, improve the training efficiency of data sandboxes—for instance, by using neural networks to analyze malicious code and reducing the number of functions that need to be tracked. Thirdly, in real trading markets, transaction rules constrain the connectivity relationships among traders. Therefore, we need to investigate the impact of population topology on the equilibrium states of the game, as well as to study incentive mechanisms based on value distribution that enable more entities to benefit from data transactions. At the same time, we will treat system security and business security issues in data trading as an independent research problem.

6. Acknowledgments

This study was supported by the National Key R&D Program of China (No.2023YFB2703900).

Ethical Statement

No ethical approval was required for this study, as it did not involve human or animal subjects.

Funding

The authors declare the following interests: This work was supported by the National Key R&D Program of China (Grant No. 2023YFB2703900). The funders provided financial support for data collection and experiment implementation but had no role in the study design, data analysis, decision to publish, or preparation of the manuscript. All other authors have no competing financial or personal interests to declare.

Declaration of competing interests

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.

Data Availability Statements

Data sharing is not applicable to this article as no datasets were generated or analysed during the current study.

Credit authorship contribution statement

Chuan Fu: Conceptualization, Formal analysis, Methodology, Software Development, Writing-original draft, Writing-review & editing, Visualization, Validation.

Jian Ye: Funding acquisition, Project administration, Supervision, Review & editing, Validation.

References

1. A. Acquisti, C. Taylor, and L. Wagman, "The economics of privacy," *Journal of Economic Literature*, vol. 54, no. 2, p. 442–92, June 2016.
2. B. Martens, "An economic perspective on data and platform market power," <https://publications.jrc.ec.europa.eu/>

- repository/handle/JRC122896[Accessed on 3 April 2025], Brussels (Belgium), 2021.
3. D. A. Harper, "Property rights, entrepreneurship and coordination," *Journal of Economic Behavior & Organization*, vol. 88, pp. 62–77, 2013, Asian Institutional Economics. doi: <https://doi.org/10.1016/j.jebo.2011.10.018>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0167268111002678>
4. L. Wang *et al.*, "Bigdatabench: A big data benchmark suite from internet services," in *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*, 2014, pp. 488–499, doi: <https://doi.org/10.1109/HPCA.2014.6835958>.
5. J. Zhan, "A benchcouncil view on benchmarking emerging and future computing," *BenchCouncil Transactions on Benchmarks, Standards and Evaluations*, vol. 2, no. 2, p. 100064, 2022, doi: <https://doi.org/10.1016/j.tbench.2022.100064>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2772485922000515>
6. J. E. Raffaghelli and S. Manca, "Is there a social life in open data? the case of open data practices in educational technology research," *Publications*, vol. 7, no. 1, 2019, doi: <https://doi.org/10.3390/publications7010009>. [Online]. Available: <https://www.mdpi.com/2304-6775/7/1/9>
7. A. Conti, V. Gupta, J. Guzman, and M. P. Roche, "Incentivizing innovation in open source: Evidence from the github sponsors program," National Bureau of Economic Research, Working Paper 31668, September 2023, doi: <https://doi.org/10.3386/w31668>. [Online]. Available: <http://www.nber.org/papers/w31668>
8. C. Ihle, D. Trautwein, M. Schubotz, N. Meuschke, and B. Gipp, "Incentive mechanisms in peer-to-peer networks — a systematic literature review," *ACM Comput. Surv.*, vol. 55, no. 14s, Jul. 2023, doi: <https://doi.org/10.1145/3578581>. [Online]. Available: <https://doi.org/10.1145/3578581>
9. E. Ostrom, *Governing the Commons: The Evolution of Institutions for Collective Action*. Cambridge: Cambridge University Press, 10 2015, doi: <https://doi.org/10.1017/CBO9781316423936>.
10. H. Ren, "Governance problems of common-pool resources from the perspective of elinor ostrom's thought of self-governance: characteristics, models and difficulties," *Social Sciences in Shenzhen*, vol. 4, no. 6, pp. 60–70, 2021.
11. C. J. Ritten, C. Bastian, and O. Phillips, "The relative effectiveness of law enforcement policies aimed at reducing illegal trade: Evidence from laboratory markets," *PLOS ONE*, vol. 16, no. 11, p. e0259254, 2021.
12. C. Fu, G. Zhang, J. Yang, and X. Liu, "Study on the contract characteristics of internet architecture," *Enterprise Information Systems*, vol. 5, no. 4, pp. 495–513, 2011. [Online]. Available: <https://doi.org/10.1080/17517575.2011.570457>
13. E. Gallo, Y. E. Riyanto, N. Roy, and T.-H. Teh, "Cooperation and punishment mechanisms in uncertain and dynamic social networks," *Games and Economic Behavior*, vol. 134, pp. 75–103, 2022, doi: <https://doi.org/10.1016/j.geb.2022.03.015>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0899825622000677>
14. A. Leibbrandt, A. Ramalingam, L. Sääksvuori, and J. M. Walker, "Incomplete punishment networks in public goods games: experimental evidence," *Experimental Economics*, vol. 18, pp. 15–37, 03 2015.
15. W. Zhang, S. Zhang, and S. Guo, "A pagerank-based reputation model for personalised manufacturing service recommendation," *Enterprise Information Systems*, vol. 11, no. 5, pp. 672–693, 2017, doi: <https://doi.org/10.1080/17517575.2015.1077998>. [Online]. Available: <https://doi.org/10.1080/17517575.2015.1077998>
16. X. Yin, X. Hu, Y. Chen, X. Yuan, and B. Li, "Signed-pagerank: An efficient influence maximization framework for signed social networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 5, pp. 2208–2222, 2021, doi: <https://doi.org/10.1109/TKDE.2019.2947421>.
17. G. S. Becker, "Crime and punishment: an economic approach," *Journal of Political Economy*, vol. 76, no. 2, pp. 169–217, 1968, doi: https://doi.org/10.1007/978-1-349-62853-7_2. [Online]. Available: https://doi.org/10.1007/978-1-349-62853-7_2
18. D. Mahmudnia, M. Arashpour, and R. Yang, "Blockchain in construction management: Applications, advantages and limitations," *Automation in Construction*, vol. 140, p. 104379, 2022, doi: <https://doi.org/10.1016/j.autcon.2022.104379>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0926580522002527>
19. H. Zhao, Z. Zhu, C. Pan, Z. Yao, W. Zhu, and X. Si, "A new electronic contract system model based on blockchain," in *Blockchain and Trustworthy Systems*, H.-N. Dai, X. Liu, D. X. Luo, J. Xiao, and X. Chen, Eds. Singapore: Springer Singapore, 2021, pp. 407–417. [Online]. Available: https://doi.org/10.1007/978-981-16-7993-3_31
20. A. Venčkauskas, D. Kukta, S. Grigaliūnas, and R. Brūzgienė, "Enhancing microservices security with token-based access control method," *Sensors*, vol. 23, no. 6, 2023. [Online]. Available: <https://doi.org/10.3390/s23063363>
21. W. Xu, S. Bhatkar, and R. Sekar, "Taint-Enhanced policy enforcement: A practical approach to defeat a wide range of attacks," in *15th USENIX Security Symposium (USENIX Security 06)*. Vancouver, B.C. Canada: USENIX Association, Jul. 2006. [Online]. Available: <https://www.usenix.org/conference/15th-usenix-security-symposium/taint-enhanced-policy-enforcement-practical-approach>
22. W. Enck *et al.*, "Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones," *ACM Trans. Comput. Syst.*, vol. 32, no. 2, Jun. 2014, doi: <https://doi.org/10.1145/2619091>. [Online]. Available: <https://doi.org/10.1145/2619091>
23. S. Muttou, S. Kumar, and N. Tyagi, "A survey of computer security models," *JIMS 8i-International Journal of Information, Communication and Computing Technology (IJICCT)*, vol. II, no. 1, pp. 48–56, Jan-Jun 2014.
24. W. G. J. Halfond, A. Orso, and P. Manolios, "Using positive tainting and syntax-aware evaluation to counter sql injection attacks," in *Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ser. SIGSOFT '06/FSE-14. New York, NY, USA: Association for Computing Machinery, 2006, p. 175–185, doi: <https://doi.org/10.1145/1181775.1181797>. [Online]. Available: <https://doi.org/10.1145/1181775.1181797>
25. A. Javed, P. Lam, and A. Chan, "Change negotiation in public-private partnership projects through output specifications: an experimental approach based on game theory," *Construction Management and Economics*, vol. 32, no. 4, pp. 323–348, 05 2014. [Online]. Available:

- <https://doi.org/10.1080/01446193.2014.895846>
26. M. A. Nowak, *Evolutionary Dynamics: Exploring the Equations of Life*. Cambridge, Mass: Belknap Press of Harvard University Press, 2006. [Online]. Available: <http://www.jstor.org/stable/j.ctvjghw98>
 27. L. Hindersin, B. Wu, A. Traulsen, and J. García, “Computation and simulation of evolutionary game dynamics in finite populations,” *Scientific Reports*, vol. 9, no. 6946, 2019. [Online]. Available: 10.1038/s41598-019-43102-z
 28. C. Hauert, A. Traulsen, H. De Silva née Brandt, M. A. Nowak, and K. Sigmund, “Public goods with punishment and abstaining in finite and infinite populations,” *Biological Theory*, vol. 3, no. 2, pp. 114–122, 04 2008. [Online]. Available: <https://doi.org/10.1162/biot.2008.3.2.114>
 29. C. Hauert, S. DE MONTE, J. Hofbauer, and K. Sigmund, “Replicator dynamics for optional public good games,” *Journal of Theoretical Biology*, vol. 218, no. 2, pp. 187–194, 2002.
 30. H. Brandt, C. Hauert, and K. Sigmund, “Punishing and abstaining for public goods,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 103, no. 2, pp. 495–7, 2006.
 31. C. Taylor, D. Fudenberg, A. Sasaki, and M. A. Nowak, “Evolutionary game dynamics in finite populations,” *Bulletin of Mathematical Biology*, vol. 66, no. 6, pp. 1621–1644, 2004, doi: <https://doi.org/10.1016/j.bulm.2004.03.004>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0092824004000333>
 32. Z. Cheng, B. Wang, Y. Pan, and Y. Liu, “Strategic analysis of participants in bcf-enabled decentralized iot data sharing,” *Mathematics*, vol. 11, no. 21, pp. 1–19, November 2023. [Online]. Available: <https://ideas.repec.org/a/gam/jmathe/v11y2023i21p4520-d1272948.html>
 33. A. de Azevedo-Lopes and A. Traulsen, “Multilevel selection in multitype populations,” *PNAS Nexus*, vol. 5, no. 6, p. pgag180, 6 2026, doi: <https://doi.org/10.1093/pnasnexus/pgag180>. [Online]. Available: <https://doi.org/10.1093/pnasnexus/pgag180>
 34. H. J. Watson, C. Fuller, and T. Ariyachandra, “Data warehouse governance: best practices at blue cross and blue shield of north carolina,” *Decision support systems*, vol. 38, no. 3, pp. 435–450, 2004.
 35. A. Z. Faroukhi, I. El Alaoui, Y. Gahi, and A. Amine, “Big data monetization throughout big data value chain: a comprehensive review,” *Journal of Big Data*, vol. 7, no. 3, pp. 1–22, 2020.
 36. N. Martijn, J. Hulstijn, M. Bruijne, and Y.-H. Tan, “Determining the effects of data governance on the performance and compliance of enterprises in the logistics and retail sector,” in *Open and Big Data Management and Innovation*, M. Janssen et al., Eds. Cham: Springer International Publishing, 10 2015, pp. 454–466.
 37. G. Demarquet, “Five key reasons enterprise data governance matters to finance . . . and seven best practices to get you there,” *Journal of Corporate Accounting & Finance*, vol. 27, no. 2, pp. 47–51, 2016.
 38. L. Ge and C. A. Brewster, “Informational institutions in the agrifood sector: meta-information and meta-governance of environmental sustainability,” *Current Opinion in Environmental Sustainability*, vol. 18, no. Feb., pp. 73–81, 2016.
 39. S. Rosenbaum, “Data governance and stewardship: Designing data stewardship entities and advancing data access,” *Health services research*, vol. 45, no. 5 Pt 2, pp. 1442–55, 10 2010. [Online]. Available: <https://doi.org/10.1111/j.1475-6773.2010.01140.x>
 40. S. A. Aaronson, “Data is different, and that’s why the world needs a new approach to governing cross-border data flows,” *Digital Policy, Regulation and Governance*, vol. 21, no. 5, pp. 441–460, 2019.
 41. M. Balazinska, B. Howe, and D. Suciu, “Data markets in the cloud: an opportunity for the database community,” *Proc. VLDB Endow.*, vol. 4, no. 12, p. 1482–1485, Aug. 2011, doi: <https://doi.org/10.14778/3402755.3402801>. [Online]. Available: <https://doi.org/10.14778/3402755.3402801>
 42. A. E. Abbas, W. Agahari, M. van de Ven, A. Zuiderwijk, and M. de Reuver, “Business data sharing through data marketplaces: A systematic literature review,” *Journal of Theoretical and Applied Electronic Commerce Research*, vol. 16, no. 7, pp. 3321–3339, 2021, doi: <https://doi.org/10.3390/jtaer16070180>. [Online]. Available: <https://www.mdpi.com/0718-1876/16/7/180>
 43. F. de la Vega, J. Soriano, M. Jimenez, and D. Lizcano, “A peer-to-peer architecture for distributed data monetization in fog computing scenarios,” *Wireless Communications and Mobile Computing*, vol. 2018, no. 1, p. 5758741, 2018, doi: <https://doi.org/10.1155/2018/5758741>. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1155/2018/5758741>
 44. D. Wörner and T. von Bomhard, “When your sensor earns money: exchanging data for cash with bitcoin,” in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, ser. UbiComp ’14 Adjunct. New York, NY, USA: Association for Computing Machinery, 2014, p. 295–298, doi: <https://doi.org/10.1145/2638728.2638786>. [Online]. Available: <https://doi.org/10.1145/2638728.2638786>
 45. B. Shen, Y. Shen, and W. Ji, “Profit optimization in service-oriented data market: A stackelberg game approach,” *Future Generation Computer Systems*, vol. 95, no. JUN., pp. 17–25, 2019.
 46. B. Guo, X. Deng, Q. Guan, J. Tian, and X. Zheng, “An incentive mechanism for cross-organization data sharing based on data competitiveness,” *IEEE Access*, vol. 6, pp. 72 836–72 844, 2018.
 47. S. S. Dawes and N. Helbig, “Information strategies for open government: Challenges and prospects for deriving public value from government transparency,” in *Electronic Government, 9th IFIP WG 8.5 International Conference, EGOV 2010, Lausanne, Switzerland, August 29 - September 2, 2010. Proceedings*. Berlin: Springer, 08 2010, pp. 50–60.
 48. J. Höchtl, P. Parycek, and R. Schöllhammer, “Big data in the policy cycle: Policy decision making in the digital era,” *Journal of Organizational Computing and Electronic Commerce*, vol. 26, no. 1-2, pp. 147–169, 2016, doi: <https://doi.org/10.1080/10919392.2015.1125187>. [Online]. Available: <https://doi.org/10.1080/10919392.2015.1125187>
 49. S. Lee, L. Zhu, and R. Jeffery, “Data governance decisions for platform ecosystems,” in *Proceedings of the Annual Hawaii International Conference on System Sciences. Vol. 2019-January. IEEE Computer Society*, 2019, p. 6377–6386. [Online]. Available: <https://doi.org/10.24251/HICSS.2019.766>
 50. E. Podda, “Data governance act (dga),” in *Encyclopedia of Cryptography, Security and Privacy*. Cham: Springer

-
- Nature Switzerland, 2025, pp. 564–566.
51. Q. Li, B. Pi, M. Feng, and J. Kurths, “Open data in the digital economy: An evolutionary game theory perspective,” *IEEE Transactions on Computational Social Systems*, vol. 11, no. 3, pp. 3780–3791, 2024, doi: <https://doi.org/10.1109/TCSS.2023.3324087>.
 52. D. Castro and R. Atkinson, “Beyond internet universalism: A framework for addressing cross-border internet policy,” *SSRN Electronic Journal*, 9 2014. [Online]. Available: <https://doi.org/10.2139/ssrn.3079821>
 53. D. Reinsel, J. Gantz, and J. Rydning, “Data age 2025,” url <https://www.seagate.com/www-content/our-story/trends/files/Seagate-WP-DataAge2025-March-2017.pdf>[Accessed on 9 April 2025] , 2017.
 54. C. Aguerre, M. Campbell-Verduyn, and J. Scholte, *Global Digital Data Governance: Polycentric Perspectives*. Routledge, 01 2024, doi: <https://doi.org/10.4324/9781003388418>.
 55. E. Ostrom, “A general framework for analyzing sustainability of social-ecological systems,” *Science*, vol. 325, no. 5939, pp. 419–422, 08 2009.
 56. —, “Beyond markets and states: polycentric governance of complex economic systems,” *Transnational Corporations Review*, vol. 2, no. 2, pp. 167–209, 2010.

RESEARCH ARTICLE

ContextFidelity-Bench: A Three-Paradigm Framework for Evaluating Multi-Turn Hallucination Behavior in Language Models

Parsa Bakhtary^{1,*}¹Google, Mountain View, CA, USA

*Corresponding author. pbakhtary@gmail.com

Received on 10 June 2026; Accepted on 24 September 2026

Abstract

We introduce ContextFidelity-Bench, a reproducible benchmark and evaluation framework for studying hallucination-related behavior in multi-turn language model conversations. The framework comprises three paradigms: numeric accuracy across a 20-turn financial analysis that ends with a data correction (P1), constraint satisfaction in a 20-turn scheduling task whose requirements accumulate, are revised, and may become jointly unsatisfiable (P2), and source fidelity in a multi-document synthesis with planted contradictions, information gaps, and authority manipulations (P3). The released artifact includes scenario generators with pre-computed ground truth and verified feasibility labels, scoring scripts with a documented protocol version, prompt templates, and 550 conversations from five API-accessible models, of which 4,250 P1 answers and 800 P2 checkpoints are scored automatically and 100 P3 syntheses are judged. An initial case study illustrates the framework’s diagnostic use. Numeric accuracy and feasible-plan success rank the five models similarly (Spearman $\rho = 0.90$), whereas source-fidelity abstention ranks them differently: the model with the lowest numeric accuracy (0.880) has the highest gap abstention (0.975) and no fabrication, while the model that produces complete valid plans most often (24 of 29 feasible post-revision checkpoints) has the joint-lowest gap abstention (0.875). When requirements are jointly unsatisfiable, models differ in whether they say so and in whether they still supply a plan; when a new requirement merely appears to conflict, some models raise false alarms. These observations come from a five-model, single-run snapshot and are presented as hypothesis-generating; the framework is designed for replication and extension.

Key words: large language models, hallucination evaluation, multi-turn evaluation, benchmarks, context fidelity, source fidelity

1. Introduction

Evaluating hallucination in language models is an active area of research, yet existing benchmarks overwhelmingly operate within single-turn interactions. TruthfulQA [1], HaluEval [2], FActScore [3], and large-scale suites such as HELM [4] and BIG-Bench [5] test isolated queries, leaving a gap in our ability to evaluate how models maintain fidelity to information across sustained multi-turn conversations where earlier outputs become inputs to later reasoning. This gap matters because many practical applications of language models — financial analysis, project planning, document review — involve exactly this kind of extended, stateful interaction.

This paper introduces ContextFidelity-Bench, a reusable evaluation framework that addresses this gap. The framework comprises three paradigms, each isolating a distinct dimension of multi-turn context fidelity:

- **Paradigm 1 (P1): Numeric accuracy under a multi-turn analysis.** Can the model answer lookups, derivations, and multi-step aggregations over a financial table across 20 turns, and recompute an earlier answer after the user corrects one input value? P1 serves as a comparison condition: it shows that the models can use the conversation’s data and integrate a stated correction in a setting with exact answers.
- **Paradigm 2 (P2): Constraint satisfaction under revision.** Can the model maintain a growing set of scheduling requirements, apply an explicit replacement of one requirement mid-conversation, and respond appropriately when a later requirement makes the set jointly unsatisfiable, or only appears to? Every checkpoint carries a verified feasibility label, so planning success on satisfiable states and the handling of unsatisfiable requests are scored as separate outcomes. This extends single-turn constraint benchmarks [6, 7] to accumulation with revision.

- **Paradigm 3 (P3): Source fidelity.** Does the model abstain when evidence is absent, or fabricate? This paradigm tests epistemic calibration using multi-document scenarios with planted contradictions, information gaps, and authority manipulations.

The framework is designed for reproducibility and extension. P1 scenarios are generated by a template-based generator that produces synthetic financial data with pre-computed ground truth, enabling automatic scoring. P2 uses a reverse-generation architecture in which a satisfying solution is constructed first and requirements are derived from it; an exhaustive verifier labels every checkpoint as satisfiable or not, and the released generator version checks those labels at generation time. P3 document sets are manually constructed with annotated gaps classified into five types by what information is missing.

To demonstrate the framework’s diagnostic utility, we present an initial case study evaluating five API-accessible models at a single point in time. This application reveals several suggestive patterns:

1. **Numeric accuracy is high and similar across models.** Accuracy on phase-end questions ranges from 0.880 to 0.995, and single-step derivations are essentially solved by all models.
2. **Feasible-plan success varies widely and follows numeric accuracy.** On satisfiable post-revision checkpoints, the share of complete plans that satisfy every requirement ranges from 3 of 29 (GPT-4o, which usually leaves items unassigned) to 24 of 29 (DeepSeek-V3.2); the ranking correlates with numeric accuracy ($\rho = 0.90$).
3. **Unsatisfiable requests are handled differently.** When a new requirement makes the active set unsatisfiable, one model almost always still supplies a complete plan, two supply no plan in about half of the cases, and one supplies mostly partial plans and asks the user to decide. When a new requirement merely appears to conflict, some models raise false alarms (up to 7 of 17 feasible scenarios).
4. **Source fidelity ranks the models differently.** The model with the lowest numeric accuracy has the highest gap abstention (0.975) and no fabrication; fabrication is rare overall and concentrated in a few scenarios.

These observations are suggestive rather than definitive. The case study uses five models evaluated once each, limiting statistical power for cross-model claims. The primary contribution is the benchmark itself, together with an initial dataset and a documented scoring protocol whose corrections are disclosed in the released artifact.

Contributions. We release: (1) a three-paradigm benchmark with scenario generators, verified feasibility labels, scoring scripts, and prompt templates; (2) a dataset of 550 conversations (250 P1, 200 P2, 100 P3) with 4,250 automatically scored P1 answers, 800 scored P2 checkpoints, and 100 LLM-judged synthesis evaluations across five models; (3) cross-provider validation of the P3 LLM judge (97.2% simple agreement over 500 paired scores; AC1 = 0.971 over the 400 ordinal pairs) supplemented by an author audit; and (4) an initial case study with sensitivity analyses (extraction rules, tolerance threshold, scenario-level bootstrap, and a targeted replication) that indicate which observed patterns are robust.

2. Related Work

Hallucination taxonomy and evaluation.

Hallucination in language models has been studied extensively, with comprehensive surveys charting the landscape of intrinsic and extrinsic hallucination across NLG tasks [8]. Single-turn benchmarks have been foundational: TruthfulQA [1] targets imitative falsehoods, HaluEval [2] covers summarisation, dialogue, and question answering, and FActScore [3] decomposes long-form generations into atomic claims verified against knowledge sources. Large-scale evaluation suites such as HELM [4] and BIG-Bench [5] assess broad capabilities but typically within single-turn interactions. These benchmarks evaluate isolated queries, whereas our framework tests sustained multi-turn interactions where earlier outputs become inputs to later reasoning.

Long-context faithfulness.

Liu et al. [9] demonstrated that language models struggle to use information positioned in the middle of long contexts, establishing position-dependent retrieval as a fundamental limitation. Our work moves from static document retrieval to multi-turn conversation in which the information to be used accumulates across turns and is revised by the user.

Multi-turn and planning evaluation.

MT-Bench [10] tests conversational quality but not factual consistency across turns. DiaHalu [11] evaluates dialogue-level hallucination, and recent graph-based methods detect contextual inconsistencies in conversation [12]. In the planning domain, benchmarks such as TravelPlanner [6] and PlanBench [7] evaluate multi-constraint satisfaction, though typically in single-turn settings. Our P2 paradigm extends this to multi-turn constraint accumulation with mid-conversation conflict, testing whether models can *revise* a constraint set rather than merely satisfy a static one.

LLM-as-judge methodology.

Using LLMs as evaluators has become standard following G-Eval [13] and the Chatbot Arena framework [14]. We employ an LLM judge for our synthesis paradigm with cross-provider validation (§3.4) and supplementary author audit. Our cross-paradigm concordance uses Kendall’s W [15].

Positioning relative to recent multi-turn benchmarks.

Several recent benchmarks also evaluate sustained interaction. MultiChallenge [16] tests instruction retention, memory of user-stated information, versioned editing, and self-coherence in human-authored conversations scored against per-instance rubrics. LongMemEval [17] tests long-term memory over scalable chat histories, including knowledge updates and abstention. Laban et al. [18] show, with simulated conversations that deliver a task in underspecified shards, that performance falls sharply relative to single-turn delivery. Generative construction with pre-computed ground truth is not new in itself, and self-coherence and versioned editing are already covered by MultiChallenge. What ContextFidelity-Bench adds is a particular combination: generated numeric and scheduling tasks with exact answers, a scheduling task in which every checkpoint carries a verified label saying whether the active requirements are jointly satisfiable, inspectable deterministic scoring for those tasks, and a typed analysis of source gaps with controlled authority direction. Table 7 lists the components; the corrected scoring protocol is described in Section 3.3.

Benchmark design methodology.

Recent work treats benchmarks and datasets as maintained research artifacts that require explicit documentation, reproducibility support, and lifecycle-aware design, rather than as static scoreboards [19, 20, 21]. Our framework addresses these concerns: P1 and P2 use generative scenario construction with pre-computed ground truth, enabling unlimited instance creation; scoring pipelines are deterministic and released with the benchmark; and the P3 gap taxonomy provides structured extensibility for new scenario types. We discuss the released artifacts and their intended use in §6.

3. Benchmark Design

ContextFidelity-Bench comprises three experimental paradigms, each isolating a distinct dimension of *context fidelity* — the ability to maintain and use accurate representations of information within a conversation. P1 and P2 use 20-turn conversations and P3 uses 12-turn conversations, all conducted programmatically via each model’s API. This section describes the design of each paradigm, the scoring protocol (version 3.1, released with the artifact), and the design choices that support reproducibility and extension.

3.1. Design Principles

Three principles guided the benchmark’s construction:

Generative scenario construction. Rather than hand-crafting a fixed set of evaluation instances, P1 and P2 use generators that produce scenarios with pre-computed ground truth. The P1 generator creates synthetic company financial data with 15 question types, producing an arbitrary number of scenarios with known correct answers. The P2 generator uses a reverse-generation architecture: a feasible solution is constructed first and constraints are derived from it, guaranteeing satisfiability without requiring a constraint solver. This means the benchmark is not exhaustible — researchers can generate fresh scenarios to avoid contamination.

Automatic and inspectable scoring. P1 answers are scored automatically against ground truth with a 1% relative tolerance (or the question’s absolute tolerance). P2 plans are read only from the plan the model explicitly proposes, accepting headings, lists, and tables but never inferring assignments from recaps, reasoning, or items the model marks as unassigned; every extracted assignment records its source span. P3 uses an LLM judge with cross-provider validation (§3.4). Every scoring rule is released, and the corrections made to the protocol during review are disclosed in Section 3.3 and the artifact.

Comparison-condition architecture. P1 serves as a comparison condition. It shows whether a model can use a table of figures supplied in the conversation, perform derivations over it, and integrate a stated correction, in a setting with exact answers. Strong P1 performance is informative about that setting; it does not rule out loss or misuse of context in the different P2 and P3 tasks, and we make no causal claim about the sources of cross-task differences (Section 7).

3.2. Paradigm 1: Progressive Numerical Analysis

Each of 50 scenarios presents a fictional company’s quarterly financial data and poses 20 questions of increasing complexity across five phases: direct lookups (turns 1–5, hereafter T1–T5), single-step derivations such as margins and ratios (T6–T10), multi-step computations including cumulative sums and top- k

rankings (T11–T15), three open-ended synthesis questions (T16–T18) that are excluded from automatic scoring, and a correction phase where the user provides an updated data value and asks the model to recompute a prior answer (T19–T20).

Turns 5, 10, and 15 are phase-end questions of the phase’s own type (a lookup, a year-over-year growth rate, and a conditional aggregate); turn 20 re-asks the turn-15 question after the correction. We report accuracy on these four turns as *phase-end accuracy* (the “probe” column of Table 1) and accuracy per phase over all scored turns. Answers are scored against precomputed ground truth with a 1% relative tolerance or the question’s absolute tolerance. The scorer accepts a number that appears anywhere in the response; AI-assisted inspection of 22 selected cases in which the accepted number was not the response’s final asserted answer found seven responses whose asserted answer was outside tolerance, and their scores were corrected (case-level decisions are released). Of the 3,119 single-valued numeric answers, 97.7% are within 1% of the target, 2.1% err by between 1% and 50%, and 0.1% by more than 50%, so moderate changes of the threshold move few scores. Appendix C reports a sensitivity sweep over tolerances from 0.1% to 10%: the ordering of the models is unchanged for any tolerance between 0.5% and 5%, apart from a tie at the top.

3.3. Paradigm 2: Constraint Satisfaction Under Revision

Each of 40 scenarios presents a scheduling task (eight items, five ordered slots, stated slot capacities) in one of five domains, with requirements that accumulate over 20 turns. At checkpoint turns 6, 11, 17, and 20 the model is asked for a complete plan. At turn 12 the user explicitly replaces one earlier fixed-placement requirement with a new one. At turn 15 the user adds a requirement that the generator constructed to be violated by its own reference plan; the turn-17 prompt asks for an updated plan and asks the model to flag any conflicting requirements explicitly.

Verified feasibility.

The turn-15 requirement was designed to break the reference plan, not to be unsatisfiable. An exhaustive search over all assignments, respecting the stated slot capacities, shows that a plan satisfying every active requirement exists in 17 of the 40 scenarios at turns 15 and 17 and in 12 at turn 20 (later requirements close five more). We therefore label each post-revision checkpoint as *feasible* or *infeasible* and score the two situations separately: on feasible checkpoints the task is to produce a complete valid plan; on infeasible checkpoints the expected behavior is to say that the requirements are incompatible, and whether a plan is still supplied is reported as an output type rather than as success or failure. The released generator version verifies these labels at generation time.

Effective requirements.

At each checkpoint the effective requirement set consists of the active explicit requirements with the turn-12 replacement applied, one capacity condition per slot, and the turn-15 requirement when the checkpoint is feasible. Capacity is interpreted as a maximum item count per slot; the historical prompts left the unit unstated. They also state the capacity of one slot twice, in the opening table and in a later sentence (“X can hold at most k ”), and the generator drew k independently of the table: the two values agree in 17 scenarios, the sentence gives a larger value in 19, and a smaller one in 4. The primary protocol requires both statements to hold, that is, the smaller

value. Because a reader can also take the later sentence as replacing the table value, we report the main P2 outcome under that reading as well (Table 2, Appendix C); the feasibility labels are identical under both readings. A plan is *complete* if it assigns every item once, and *valid* if it is complete and satisfies every effective requirement. Fractional satisfaction is the share of effective requirements satisfied; an item-level requirement whose item is unassigned counts as unsatisfied. A response with no proposed plan scores zero on both.

Plan extraction.

Assignments are read only from the plan the model explicitly proposes (the last such plan if several appear), accepting headings, lists, and tables. Nothing is inferred from requirement recaps, verification lists, reasoning text, or items the model marks as unassigned; duplicate placements are flagged. Each response is labelled as a complete plan, a partial plan, no usable plan (a template or an all-unassigned list), or no plan, and each extracted assignment records its source span for inspection.

Incompatibility claims.

Whether a response asserts that the active requirements are jointly incompatible is labelled by a frozen written rubric applied to the visible answer with model identity and feasibility hidden from the classifier, which must quote its supporting sentence (Appendix C). A claim on an infeasible checkpoint is a correct incompatibility flag; a claim on a feasible checkpoint is a false alarm. Requests that the user resolve a conflict are recorded separately.

Corrections disclosed.

Two defects in the originally submitted scoring were found during review and corrected: the scorer did not apply the turn-12 replacement, so every reference plan scored below 1.0 after turn 12, and the plan extractor collected assignments from outside the proposed plan, including items the model had marked unassigned. All P2 results in this version use the corrected protocol; the artifact retains the original outputs and the superseded scoring code.

3.4. Paradigm 3: Multi-Document Synthesis

Each of 20 scenarios provides 3–5 documents on a shared topic spanning legal, scientific, technical, and financial domains. Each scenario contains planted contradictions, information gaps (where a plausible question has no answer in the sources), and an authority manipulation where a high-status source either supports or contradicts the evidence. In 7 scenarios the authority is correct, while in 13 it is wrong.

Responses were evaluated by an LLM judge (Claude Opus 4.6), scoring contradiction identification and resolution, gap abstention (whether the model correctly declines to answer when the sources lack sufficient evidence) versus fabrication (whether it invents unsupported claims), and authority deference (whether the model gives undue weight to high-status sources over contradicting evidence).

Cross-provider validation.

To validate reliability, all 100 entries (5 models $\times$ 20 scenarios) were independently re-judged by GPT-5.2 (OpenAI) using the identical rubric and scoring schema. This cross-provider design tests whether the rubric produces consistent scores when applied by a model from a different vendor, providing stronger validation than within-family replication. Simple agreement was 97.2% across the 500 paired scores (four ordinal dimensions and the binary fabrication label, 100 entries each). The pooled agreement

coefficients are computed over the 400 ordinal pairs: Gwet’s AC1 = 0.971 and linear-weighted κ = 0.718; the binary fabrication labels agreed on 97 of 100 entries. On the two dimensions with sufficient marginal variance for standard κ , gap abstention (κ_{lin} = 0.877) and fabrication (κ = 0.712), agreement was substantial to almost perfect. Agreement is also high on the raw, unbinned scores: exact agreement ranges from 93% to 100% by dimension, the mean absolute difference is at most 0.024, and for gap abstention the raw-score Pearson correlation is 0.947. Full reliability results, including the raw-score comparison, appear in Appendix A.

Author audit.

In addition to the cross-provider validation, the author manually inspected all fabrication cases (6 events across 3 models, including 1 from the expanded missing-evidence sample) and a stratified random sample of 15 abstention cases (3 per model). In 5 of 6 inspected fabrication cases, the LLM judge’s determination was confirmed: the model stated as fact what the source documents did not establish. In the single disagreement (P3_026, Claude Sonnet 4.5), the model explicitly flagged the answer as unknown and labeled subsequent discussion as speculation; the author judged this as adequate epistemic hedging rather than fabrication, whereas the judge scored it as fabricated on the grounds that the speculative scenarios went beyond the documents. In the abstention sample, the judge’s scores agreed with the author’s assessment in all 15 cases. Reported fabrication rates use the primary judge’s scores throughout; the audit characterizes human–judge agreement rather than re-scoring the data. This audit is not independent human annotation, but confirms that the LLM-judged scores are directionally accurate.

Gap taxonomy.

The 20 information gaps in the core P3 scenarios are classified into five types by what information is missing: *missing evidence* (sources imply supporting evidence exists without providing it), *missing detail* (a specific fact is absent), *missing number* (a quantitative value is not given), *missing reason* (a causal explanation is not provided), and *implied not stated* (a conclusion is suggested but never explicitly made). This taxonomy enables gap-type-specific analysis of fabrication triggers and can be extended with additional types in future work.

3.5. Case Study Configuration

To demonstrate the framework, five API-accessible models were evaluated at a single point in time (model version strings: `gpt-4o-2024-11-20`, `claude-sonnet-4-5-20250929`, `deepseek-reasoner`, `gemini-2.5-pro`, `MiniMax-M2.5`), each completing all 50 P1, 40 P2, and 20 P3 scenarios, yielding 550 conversations (10,200 generated assistant turns), of which 4,250 P1 answers and 800 P2 checkpoints are scored automatically and 100 P3 syntheses are judged. All models were run at temperature 0.0, standard for factual and reasoning evaluations [1, 4] where sampling noise is a confound; the targeted replication (Appendix B) partially addresses variance concerns. Maximum output tokens were 2,048 (16,384 for DeepSeek-V3.2, 8,192 for Gemini 2.5 Pro) to accommodate model-specific verbosity. Each paradigm used a task-appropriate system prompt (full prompts in the released materials). Each model-scenario combination was run once.

We refer to the models by short names (GPT-4o, Sonnet 4.5, DeepSeek-V3.2, Gemini 2.5 Pro, MiniMax M2.5) for readability. During data collection in February 2026 we queried DeepSeek’s

Table 1. P1 numeric accuracy. “Probe”: accuracy on the four phase-end questions (turns 5, 10, 15, 20). Phase columns: accuracy over all scored turns of that phase. Seven scores were corrected after AI-assisted inspection of extraction cases (Section 3.2).

Model	Probe Acc.	Lookup (T1–T5)	Single (T6–T10)	Multi (T11–T15)	Corr. (T19–T20)
Sonnet 4.5	0.985	0.962	1.000	0.971	0.970
DeepSeek-V3.2	0.995	0.974	0.992	0.968	0.983
Gemini 2.5 Pro	0.965	0.980	1.000	0.968	0.962
GPT-4o	0.955	0.908	1.000	0.887	0.880
MiniMax M2.5	0.880	0.952	0.996	0.875	0.867

API using the model alias `deepseek-reasoner`. The provider’s change log¹ identifies that alias as DeepSeek-V3.2 in thinking mode during this period; we therefore correct the DeepSeek-R1 label used in the original submission. This attribution follows the documented endpoint mapping rather than a pinned checkpoint identifier, and relabelling does not alter the recorded responses or their scores. The remaining four identifiers are pinned or stable releases. All five are API snapshots, and specific rankings may shift with model updates or repeated runs. The case study results should be interpreted as a demonstration of the framework’s diagnostic capabilities rather than a durable ranking of current models.

4. Case Study Results

We present results from the initial application of ContextFidelity-Bench to five models, organised by paradigm. Section 4.1 reports numeric accuracy (Table 1); Section 4.2 reports feasible-plan success and the handling of unsatisfiable requests (Tables 2 and 3, Figure 1); Section 4.3 reports source fidelity (Tables 4 and 5); Section 4.4 collects the primary metric of each paradigm in one table (Table 6, Figure 2) and asks how stable the cross-paradigm pattern is.

4.1. P1: Numeric Accuracy (Comparison Condition)

Across 50 scenarios per model, phase-end accuracy (turns 5, 10, 15, and 20) ranges from 0.880 (MiniMax M2.5) to 0.995 (DeepSeek-V3.2), and accuracy over all 17 scored turns per conversation ranges from 0.926 to 0.980. The scored questions include lookups, single-step derivations, multi-step aggregations, and the recomputation of an earlier answer after a data correction.

Table 1 shows high performance across all question phases. Single-step derivations (margins, ratios, growth rates) are answered at 0.992–1.000 by all models. Recomputation after the correction is at or above 0.867 for all models. Of the 4,250 scored answers, 73 are wholly wrong and a further 261 receive partial credit on multi-part questions (top- k lists and paired quarter-and-value answers); the wholly wrong answers cluster in conditional aggregations requiring multiple filtering steps, in the post-correction recomputation, and in count-above queries where models miscount by one.

P1 is the comparison condition. It shows that all five models use the supplied table accurately, perform multi-step derivations, and integrate a stated correction in a setting with exact answers. It says nothing about whether the same models maintain or use context in the different P2 and P3 tasks; those results are examined on their own terms below.

4.2. P2: Feasible-Plan Success and Unsatisfiable Requests

P2 separates two questions: when a complete valid plan exists, does the model produce one, and when the active requirements are jointly unsatisfiable, does the model say so? Table 2 reports the first on the feasible checkpoints (all 40 scenarios at turns 6 and 11; the 17 feasible scenarios at turn 17; the 12 at turn 20), and Table 3 reports the second on the infeasible checkpoints.

Feasible checkpoints.

DeepSeek-V3.2 produces a complete plan at all but one of its 160 checkpoints and a complete valid plan on 24 of the 29 feasible post-revision checkpoints. Sonnet 4.5 and Gemini 2.5 Pro produce valid plans on roughly half of them, MiniMax M2.5 on 10, and GPT-4o on 3. GPT-4o’s low rate reflects a consistent behavior rather than requirement violations: it leaves items unassigned in 126 of its 160 checkpoint responses (mean coverage 0.56 of the eight items; Table 10), so its fractional satisfaction is moderate while its complete-plan rate is near zero. Capacity is the most common failure under the primary rule: among responses that assign at least one item, 12% (Gemini 2.5 Pro) to 25% (MiniMax M2.5) place more items in a slot than the effective capacity allows. Much of this traces to the historical prompts. When the later capacity sentence is read as replacing the table value (Section 3.3), capacity violations fall to 2 of 160 responses for DeepSeek-V3.2 and 2 of 137 for Gemini 2.5 Pro, DeepSeek-V3.2 produces a valid plan at every feasible checkpoint (29 of 29 post-revision), and every model’s valid-plan count rises (Table 2, “alt. cap.”); there are no rank reversals; Sonnet 4.5 and Gemini 2.5 Pro tie under the alternative reading. All six of Sonnet 4.5’s no-plan responses at turns 6 and 11 occur in scenarios where the two capacity statements differ: five ask the user which value applies, and one reads capacity as a developer count. The prompts also give capacities without a unit, and some responses read capacity as staff or hours; the released generator states the unit and gives each slot a single capacity value. A scenario-level bootstrap (Section 4.4) shows that only DeepSeek-V3.2’s first place (98% of resamples) and GPT-4o’s last place (98%) are stable on the post-revision valid-plan rate; the middle three are not separable.

Infeasible checkpoints.

On the 23 scenarios where the turn-15 requirement makes the active set unsatisfiable, the models behave differently at turn 17. DeepSeek-V3.2 supplies a complete plan in 22 of 23 cases and asks the user to choose in 6. Sonnet 4.5 and Gemini 2.5 Pro decline to supply any plan in 11 and 12 of 23 cases; Gemini 2.5 Pro supplies a partial plan in a further 10. Sonnet 4.5 asks the user how to resolve the conflict in 14 cases. MiniMax M2.5 supplies a complete plan in 17 cases and asks the user in 16. GPT-4o supplies a partial plan in 19 cases and asks the user in 17. By turn 20 most models supply complete plans again. As a named diagnostic we check each complete

¹ <https://api-docs.deepseek.com/updates/>

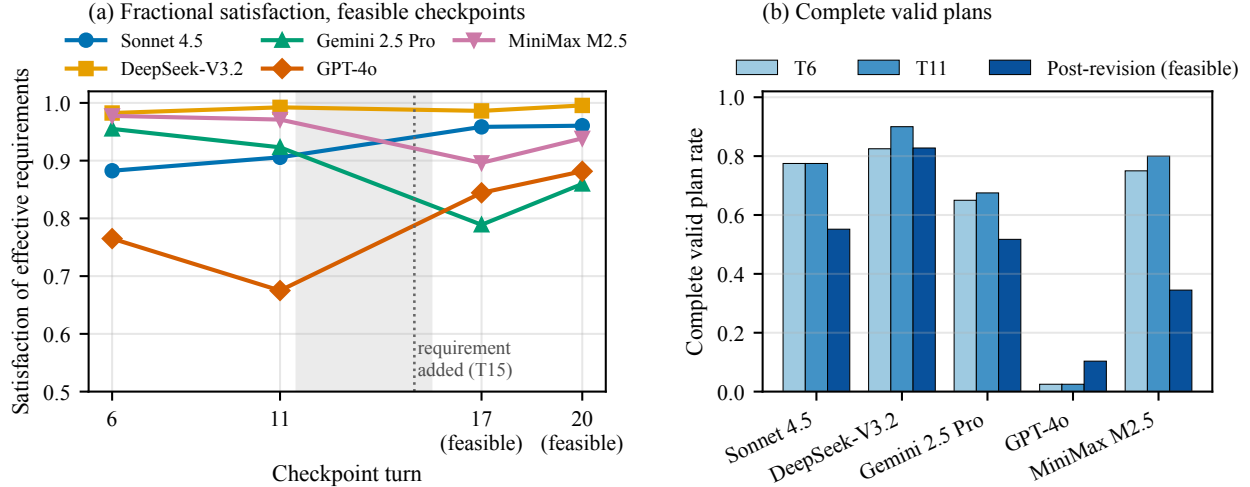


Fig. 1. P2 on feasible checkpoints. (a) Fractional satisfaction of the effective requirements at each checkpoint (turns 17 and 20 restricted to feasible scenarios); the dotted line marks the added requirement at turn 15. (b) Complete valid plan rate at turns 6 and 11 (all scenarios) and on the feasible post-revision checkpoints. Scenario-bootstrap 95% intervals appear in Table 2.

Table 2. P2 feasible-plan success. Fractional satisfaction (scenario-bootstrap 95% intervals, 2,000 resamples) and complete valid plan rate on feasible checkpoints. “Valid, post-rev.” pools the 17 feasible turn-17 and 12 feasible turn-20 checkpoints; “Valid, alt. cap.” is the same count when the later capacity sentence is read as replacing the table value (Section 3.3).

Model	T6 sat.	T11 sat.	Feasible T17 sat.	Feasible T20 sat.	Valid, post-rev.	Valid, alt. cap.
DeepSeek-V3.2	.982 [.970, .993]	.992 [.985, .998]	.986 [.972, .997]	.996 [.987, 1.00]	24/29	29/29
Sonnet 4.5	.882 [.782, .963]	.906 [.821, .971]	.958 [.934, .983]	.961 [.921, .991]	16/29	19/29
Gemini 2.5 Pro	.955 [.927, .980]	.923 [.871, .965]	.789 [.640, .920]	.860 [.737, .965]	15/29	19/29
MiniMax M2.5	.978 [.953, .995]	.971 [.946, .990]	.896 [.824, .952]	.939 [.873, .982]	10/29	13/29
GPT-4o	.765 [.740, .790]	.675 [.631, .719]	.844 [.796, .889]	.882 [.833, .925]	3/29	4/29

plan against every requirement except the added one: DeepSeek-V3.2’s plan passes in 13 of 23 turn-17 cases, MiniMax M2.5’s in 2, and no other model’s in more than 1. We do not score these outputs as success or failure: declining to plan when the requirements cannot all be met is a defensible response to the historical prompt, which asked for a plan and for conflicts to be flagged without stating a priority rule. What the paradigm measures here is whether the incompatibility is stated and what the model does about it; the released generator adds an explicit priority instruction for future collections.

Incompatibility claims and false alarms.

The last two columns of Table 3 give the rubric classification of the turn-15 and turn-17 responses. On the infeasible scenarios, the number of responses stating that the active requirements cannot all be satisfied ranges from 9/22 (GPT-4o) to 23/23 (Gemini 2.5 Pro) at turn 15, when the requirement is introduced, and from 12/23 (GPT-4o) to 23/23 (Gemini 2.5 Pro) at turn 17. On the 17 feasible scenarios the same statement is a false alarm; these range from 0/16 (GPT-4o) to 6/17 (Gemini 2.5 Pro) at turn 15 and from 0/17 (DeepSeek-V3.2) to 7/17 (GPT-4o) at turn 17, and DeepSeek-V3.2 makes 1 in its 34 feasible responses. Denominators exclude 7 of the 400 responses: three that the classifier labelled unclear and four whose supporting quote could not be verified. In at least 9 of the 33 false alarms the quoted sentence attributes the conflict to slot capacity (developer counts, hours, or story points read as the capacity unit), where the historical prompts are ambiguous (Section 3.3); these are not

simple misreadings of the requirements. The turn-20 responses were not classified.

4.3. P3: Source Fidelity and Epistemic Calibration

The P3 paradigm probes a different competence from P1 and P2: whether models appropriately abstain when evidence is absent or instead fabricate unsupported claims.

Table 4 presents the P3 results. All five models achieved near-perfect contradiction identification (1.000) and resolution (0.973–1.000). Gap abstention ranged from 0.875 (DeepSeek-V3.2 and GPT-4o) to 0.975 (MiniMax M2.5). Fabrication rates were low but non-zero for three models: Sonnet 4.5 at 5%, DeepSeek-V3.2 and GPT-4o at 10%, while Gemini 2.5 Pro and MiniMax M2.5 produced zero fabrications. On authority bias, across 13 scenarios where the high-authority source was wrong, *every model scored zero inappropriate deference*. No model was misled by status alone. In the 7 scenarios where authority was correct, all models deferred appropriately at 100%, except Sonnet 4.5 and GPT-4o at approximately 86%, which is a mild over-skepticism rather than a failure of reasoning.

Fabrication by gap type.

We classified the information gaps in the P3 scenarios into five types by what is missing. In the 20 core scenarios, fabrication appeared mainly at “missing evidence” gaps, where sources imply that supporting evidence exists without providing it. To examine this with a larger sample we constructed eight additional missing-evidence scenarios (included in the released dataset) spanning scientific, legal, and technical domains and

Table 3. P2 handling of unsatisfiable requests. Output types are counts of responses on infeasible checkpoints (23 scenarios at turn 17, 28 at turn 20; turn 15 requests no plan). “Asks user”: the response explicitly asks the user how to resolve the conflict (phrase patterns released with the artifact; generic closing offers are not counted). “States incompatibility”: responses whose visible answer asserts that the active requirements cannot all be satisfied, among the 23 infeasible scenarios; “False alarm”: the same assertion among the 17 feasible scenarios (rubric classification, Section 3.3; responses treated as unclear are excluded; turn 20 was not classified).

Model	Turn	No plan	Partial plan	Complete plan	Asks user	States incompatibility	False alarm
DeepSeek-V3.2	T15	—	—	—	—	20/22	1/17
	T17	0	1	22	6	20/21	0/17
	T20	0	0	28	1	—	—
Sonnet 4.5	T15	—	—	—	—	20/23	2/17
	T17	11	5	7	14	22/23	4/17
	T20	4	4	20	0	—	—
Gemini 2.5 Pro	T15	—	—	—	—	23/23	6/17
	T17	12	10	1	5	23/23	6/17
	T20	10	12	6	1	—	—
MiniMax M2.5	T15	—	—	—	—	18/22	2/17
	T17	4	2	17	16	19/23	5/16
	T20	1	2	25	3	—	—
GPT-4o	T15	—	—	—	—	9/22	0/16
	T17	0	19	4	17	12/23	7/17
	T20	0	11	17	16	—	—

Table 4. P3 multi-document synthesis results. All models achieve zero inappropriate authority deference. MiniMax M2.5, the weakest numeric tracker in P1, leads or ties for the lead on every source-fidelity metric in this case study.

Model	Contr. Res.	Gap Abst.	Fab. Rate	Appr. Def.	Inappr. Def.
Sonnet 4.5	0.980	0.885	0.050	86%	0%
DeepSeek-V3.2	1.000	0.875	0.100	100%	0%
Gemini 2.5 Pro	1.000	0.930	0.000	100%	0%
GPT-4o	0.973	0.875	0.100	86%	0%
MiniMax M2.5	1.000	0.975	0.000	100%	0%

ran all five models on them, which expands the missing-evidence sample from 5 to 13 scenarios and leaves the other gap types unchanged.

Table 5 summarizes the result. Because the five models share each scenario, the unit of inference is the scenario, and the primary endpoint is whether any of the five models fabricated on it; model-scenario pair counts are reported as descriptive breakdowns only. Three of the 13 missing-evidence scenarios produced at least one fabrication (five fabricating responses in total, three of them on one scenario, P3_019), one of the six missing-detail scenarios produced one, and the nine scenarios of the remaining three types produced none. Fisher’s exact test at scenario level, missing evidence versus all other types, gives $p = 0.31$. The pattern is therefore suggestive rather than statistically established, and the two types with only two scenarios provide little evidence either way. We use the term *implication-to-certainty collapse* to describe the observed cases: when a gap sits at the boundary between what is stated and what is implied, the fabricating responses convert a qualified implication into a stated certainty. Whether this is a property of the gap type is the question the framework’s taxonomy is designed to let future collections answer.

Contemporary supplement.

Because the two least-sampled gap types had two scenarios each, we collected a supplement in September 2026 under a plan fixed before collection (released with the artifact): four new scenarios of each of those types, plus the eight supplementary missing-evidence scenarios re-collected in the same batch for comparison. The batches are never pooled. Four of the five

models were collected. The endpoint that served DeepSeek-V3.2 in February no longer serves that model, and the planned third-party-hosted session had not been run at the time of writing, so the prespecified scenario-level endpoint, which requires all five models, is not reported. Descriptively, the judge labelled a fabrication in 0 of 16 missing-reason pairs, 0 of 16 implied-not-stated pairs, and 1 of 32 missing-evidence comparison pairs (GPT-4o, on a scenario that was flagged for Sonnet 4.5 in February; confirmed in the author’s unblinded audit). On the comparison scenarios, per-model fabrication labels agreed between the two collections in 30 of 32 cases. These counts are too small to confirm or refute a gap-type effect.

As a concrete example, one scenario presented clinical trial documents where an internal memo stated that human trials “excluded drinkers,” and a gap probe asked whether any human participants consumed alcohol. Three of five models responded with a definitive “no,” converting an enrollment exclusion criterion (who was *eligible* to participate) into a behavioral claim about what participants *did* (whether anyone consumed alcohol during the study). The response states as certainty what the documents only imply.

A second example comes from the legal domain: a cease-and-desist letter states “we have evidence you created [the algorithm] while employed by us” without describing that evidence, and one model named a specific causal link between two documents that no source establishes.

Table 5. Fabrication by gap type across the core and expanded P3 scenarios (February 2026 batch). Primary endpoint: scenarios on which any of the five models fabricated. Pair-level counts (model $\times$ scenario) are descriptive. Scenario-level Fisher’s exact test, missing evidence versus all other types: $p = 0.31$.

Gap type	Scenarios	Scenarios with any fabrication	Pairs	Fabricating responses
Missing evidence	13	3	65	5
Missing detail	6	1	30	1
Missing number	5	0	25	0
Missing reason	2	0	10	0
Implied not stated	2	0	10	0

Table 6. Cross-paradigm summary. Ranks in parentheses, ties receiving the average of the tied positions. P2 primary metric: complete valid plans on the 29 feasible post-revision checkpoints; P2 satisfaction on the same checkpoints is shown for reference.

Model	P1 phase-end acc.	P2 valid plans (feasible)	P2 sat. (feasible)	P3 gap abstention
DeepSeek-V3.2	.995 (1)	24/29 = .83 (1)	.990	.875 (4.5)
Sonnet 4.5	.985 (2)	16/29 = .55 (2)	.959	.885 (3)
Gemini 2.5 Pro	.965 (3)	15/29 = .52 (3)	.818	.930 (2)
GPT-4o	.955 (4)	3/29 = .10 (5)	.860	.875 (4.5)
MiniMax M2.5	.880 (5)	10/29 = .34 (4)	.914	.975 (1)

4.4. Cross-Paradigm Patterns

Table 6 collects the primary metric of each paradigm: P1 phase-end accuracy, the P2 complete valid plan rate on feasible post-revision checkpoints (with the fractional satisfaction on the same checkpoints for reference), and P3 gap abstention. Figure 2 shows the same values and the corresponding failure rates.

Two patterns stand out. First, numeric accuracy and feasible-plan success rank the models similarly: the Spearman correlation between the P1 and P2 columns is $+0.90$ (exact permutation $p = 0.08$), with DeepSeek-V3.2 first on both and GPT-4o and MiniMax M2.5 at the bottom of both. Second, source fidelity ranks them differently: MiniMax M2.5 is last on P1 and first on P3 gap abstention, and Gemini 2.5 Pro is second on P3 while third on P1 and P2. The P1–P3 correlation is -0.56 ($p = 0.40$) and the P2–P3 correlation -0.21 ($p = 0.73$); with five models we report exact two-sided permutation p -values, because the usual t approximation is unreliable at this sample size. Kendall’s W across the three rankings, with tie correction, is 0.367 ($\chi^2 = 4.41$, $df = 4$, $p = 0.35$ under the chi-square approximation; exact permutation $p = 0.41$). With five models these correlations are descriptive; they are reported to characterize this snapshot, not to establish population relationships.

Because the five-model statistics are weak, we ask which parts of the pattern survive resampling of the scenarios that produce the scores. With 2,000 scenario-level bootstrap resamples (the same draw applied to all five models; for P2 a resampled scenario carries all of its feasible post-revision checkpoints), MiniMax M2.5 is last on P1 and first on P3 in 93% of resamples, DeepSeek-V3.2 is first on the P2 valid-plan rate in 98%, and GPT-4o is last on it in 98%; the ordering of the three middle models on P2 is not stable. These intervals describe uncertainty over scenarios for these fixed runs; they say nothing about run-to-run variability (Appendix B) or about models outside the sample.

The original submission reported low concordance and rank inversions between all three paradigms and interpreted them as evidence of dissociable competencies. Under the corrected P2 protocol, the P1–P2 inversion disappears; what remains is the divergence between numeric accuracy and feasible planning on one side and source-fidelity abstention on the other, together with the separate behaviors on unsatisfiable requests. Confirming or refuting that divergence with more models is a primary direction for future use of the benchmark.

5. Discussion

What the case study shows.

In this five-model snapshot, numeric accuracy over a supplied table is high for all models, and the models that are most accurate there also most often produce complete valid plans when one exists. Source-fidelity abstention does not follow that order. The framework’s value in the case study is not a ranking but the separation of outcomes that an aggregate score would merge: whether a model completes a plan, whether it respects stated capacities, whether it says that requirements cannot all be met, whether it raises that claim when they can, and whether it declines to answer a question the sources cannot answer.

Unsatisfiable requests.

The models’ responses to jointly unsatisfiable requirements differ in kind. DeepSeek-V3.2 usually supplies a complete plan, and in over half of the cases that plan satisfies everything except the added requirement. Sonnet 4.5 and Gemini 2.5 Pro supply no plan in about half of the cases. GPT-4o supplies partial plans and asks the user to resolve the conflict. MiniMax M2.5 usually supplies a complete plan that violates further requirements. We describe these as output types rather than as success or failure because the historical prompt asked for a plan and for conflicts to be flagged without stating which requirement should yield. Which behavior is preferable depends on the application; the framework makes the behaviors visible. Any explanation in terms of training or alignment would be speculation, and we offer none.

Task-dependent failure profiles.

The divergence between numeric and planning accuracy on one side and source-fidelity abstention on the other means that a model chosen for one kind of task may be a poor choice for another. In this snapshot, MiniMax M2.5 is the most conservative on unanswerable questions and the least accurate on numeric recomputation; GPT-4o’s numeric accuracy is high (0.955) but it rarely commits to a complete plan. Whether such profiles are stable across models and versions is an empirical question the benchmark is designed to let others test.

Authority bias.

The zero inappropriate authority deference observed across all five models is reassuring but conditional. The authority

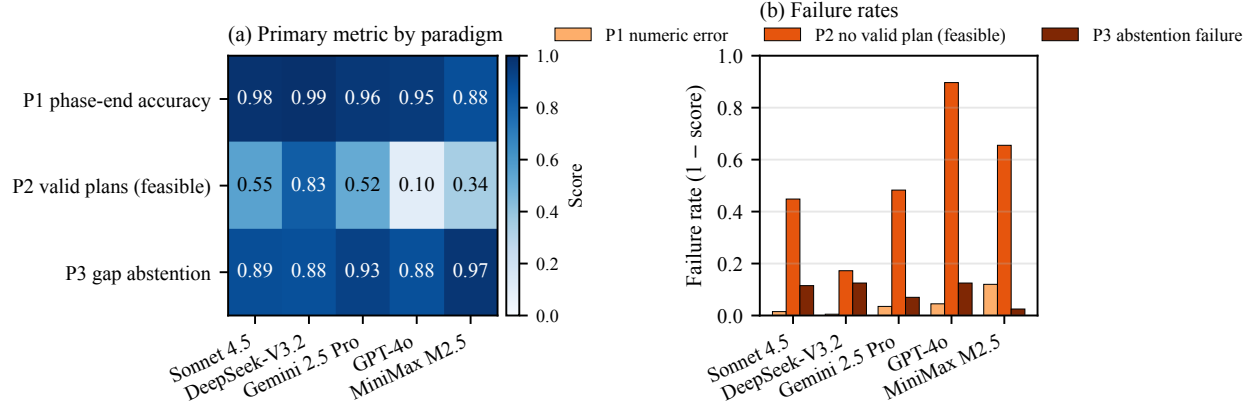


Fig. 2. Cross-paradigm summary. (a) Primary metric of each paradigm: P1 phase-end accuracy, P2 complete valid plan rate on feasible post-revision checkpoints, and P3 gap abstention. (b) The corresponding failure rates (one minus each score).

Table 7. ContextFidelity-Bench artifacts prepared for release. The repository/archive includes both generated evaluation data and executable tooling for extension.

Component	Contents	Reuse purpose
Scenario generators	P1 financial-data generator; P2 reverse-generated scheduling tasks; P3 document templates and gap annotations	Generate fresh benchmark instances and reduce contamination risk
Prompt templates	System and user prompts for all paradigms and checkpoints	Replicate the evaluation protocol across model providers
Scoring scripts	Numeric tolerance scorer with adjudication records; P2 protocol v3.1 (plan extractor, feasibility verifier, effective-requirement scorer, incompatibility rubric); P3 judging rubric and reliability scripts	Reproduce scores and audit individual model outputs
Model outputs	550 multi-turn conversations (10,200 assistant turns), 4,250 scored P1 answers, 800 scored P2 checkpoints, and 100 synthesis evaluations	Reanalyze the case study without re-querying APIs
Analysis and replication scripts	Scoring, analysis, figure and table scripts; collection protocols and API clients	Reproduce reported analyses and guide additional collections

direction is mixed (13 scenarios in which the high-status source is wrong, 7 in which it is right), so a policy of always distrusting the higher-status source is contradicted by the 86–100% appropriate deference on the seven authority-correct scenarios. This comparison rules out that one heuristic; it does not show that the models weighed the evidence rather than responding to other cues, and it does not test subtler authority signals. Matched authority reversals and document-order controls would test the broader question.

Implications for evaluation practice.

Current leaderboards typically report aggregate scores. The outcomes separated here suggest reporting, for multi-turn settings, at least the following: accuracy on tasks with exact answers, complete-valid-plan rates on verified-feasible planning states, the handling of verified-unsatisfiable states, and abstention on unanswerable questions, each with its unit of analysis and denominator stated.

6. Released Artifacts and Reproducibility

The primary contribution of this work is the released evaluation framework, designed to support replication and extension. The released artifacts comprise the following components:

Scenario generators. The P1 generator produces synthetic company data with 15 question types and pre-computed ground truth. The P2 generator constructs a satisfying schedule and derives requirements from it. The released generator version verifies at every checkpoint whether the active requirements are jointly satisfiable, gives each slot a single capacity value with a stated unit, and adds an explicit priority instruction for the added requirement; the historical scenarios are preserved unchanged with their corrected feasibility annotations.

Scoring scripts. P1 uses a numeric comparison with 1% tolerance and the released adjudication records. P2 uses protocol version 3.1: block-scoped plan extraction with source spans, effective requirements including capacities, verified feasibility labels, and the frozen incompatibility rubric with its classification outputs and validation. The superseded P2 scoring code is retained for comparison. For P3 we release the judge rubric, the scoring schema, and the cross-provider validation scripts.

Replication and extension. The released scripts reproduce the case-study tables and the analyses of Appendices A–C. The framework accommodates additional models, and the P3 gap taxonomy gives a structured way to add scenario types.

7. Limitations and Future Work

Several limitations of the current case study warrant acknowledgment, alongside opportunities for extending the framework.

Model sample size. The cross-paradigm concordance statistics are computed over five models, limiting statistical power. The primary support for paradigm-specific variation comes from within-paradigm results computed over hundreds of scored answers and checkpoints and from the scenario-level stability of the divergence between numeric accuracy and source-fidelity abstention, rather than from the five-model concordance statistics. Evaluating additional and more diverse models — particularly open-weight models where architecture and training pipelines can be inspected, and non-autoregressive architectures such as diffusion language models — is the most direct way to strengthen or refute the cross-paradigm findings.

Single-run design. Each model-scenario combination was run once. A targeted replication of 20 selected scenarios (Appendix B) found that P1 answers were stable but that P2 outputs on the hardest scenarios vary materially between runs, including a 0.18 change in one model’s mean satisfaction on the replicated feasible checkpoints. Point estimates for closely clustered models should be read as approximate, and the scenario-bootstrap intervals do not capture this run-to-run variation.

LLM-as-judge evaluation. P3 evaluation used an LLM judge from the same model family as one subject. The cross-provider validation (97.2% binned agreement, AC1 = 0.971, raw-score exact agreement 93–100%) and author audit provide confidence that the rubric produces consistent and directionally accurate scores, but independent human annotation of all P3 outputs would provide stronger validation. We plan to add blinded human annotation in a future version of the benchmark.

Synthetic scenarios. All scenarios are synthetic, with planted contradictions, explicit gaps, and clearly attributed sources. In naturalistic settings, contradictions are implicit, gaps are unmarked, and authority signals are subtler. The direction of this bias is likely conservative: implicit contradictions should be harder to detect, making our near-perfect detection rates an upper bound. A small validation slice using public real-world documents would strengthen the framework’s external validity and is a priority for future releases.

Gap-type sample sizes. The unit of inference for the gap-type analysis is the scenario, and two gap types have only two scenarios each; the scenario-level comparison between missing-evidence and other gaps is not significant ($p = 0.31$). The contemporary supplement (Section 4.3) adds four scenarios to each of those types but covers four of the five models, so it is reported descriptively; completing that panel and expanding every gap type with contemporaneously collected comparison cases are the natural next steps.

Endpoint aliasing. One evaluated model was accessed through a provider alias (`deepseek-reasoner`) that has since been repointed to a newer model. We document the model the alias resolved to during our runs (DeepSeek-V3.2, thinking mode), but exact re-runs now require third-party hosting of the open weights. Pinned model identifiers should be preferred wherever a provider offers them.

Scoring corrections and their limits. The P2 results in this version follow a corrected protocol (Section 3.3); the corrections were made after the original submission and are disclosed there and in the artifact. The extraction rules were validated on selected cases (a 40-case spot-check and

23 previously refusal-labelled responses); a later corpus-wide consistency check found and corrected 25 of the 800 core extractions (Appendix C), and undetected errors may remain. The incompatibility-claim classification was compared with two samples labelled in a separate AI-assisted review, not with human annotation of the corpus. Capacity is scored as a maximum item count although the historical prompts left the unit unstated, and the prompts state one slot’s capacity twice with different values in 23 of 40 scenarios; absolute valid-plan counts depend on how that is read (DeepSeek-V3.2: 24 or 29 of 29), with no rank reversals (Sonnet 4.5 and Gemini 2.5 Pro tie under the alternative reading). P1 answer extraction accepts a number anywhere in the response; a selected inspection corrected seven cases, but corpus-wide extraction accuracy was not estimated. The P1 model order is stable for relative tolerances between 0.1% and 5%, apart from a tie for first place at 5% (Appendix C).

Pairwise cross-model correlations. The negative P1–P3 association ($\rho = -0.56$) observed in this case study is descriptive only. It cannot be treated as a reliable finding at $n = 5$ and would require confirmation with additional models.

8. Conclusion

We have introduced ContextFidelity-Bench, a three-paradigm evaluation framework for studying hallucination-related behavior in multi-turn conversations. The framework separates numeric accuracy over supplied data, constraint satisfaction under revision with verified feasibility labels, and source fidelity with a typed gap taxonomy, and provides scenario generators, a documented scoring protocol, prompt templates, and a dataset of 550 conversations to support replication and extension.

An initial case study of five API-accessible models illustrates the framework’s diagnostic use. Numeric accuracy is high for all models (0.880–0.995 on phase-end questions) and feasible-plan success follows a similar order, from 24 of 29 complete valid plans to 3 of 29. Responses to unsatisfiable requests differ in kind, and source-fidelity abstention ranks the models differently from the other two paradigms. These observations come from a single run of five models, and the P2 results follow a scoring protocol corrected during review; they are offered as hypotheses for the framework’s future users rather than as findings about the models. We release the full experimental framework, generators, scored results with their corrections, and replication scripts to support extension with additional models, scenarios, and evaluation dimensions.

Ethical Statement

No ethical approval was required for this study, as it did not involve human or animal subjects.

This study evaluates computational systems on synthetic financial and scheduling scenarios and manually constructed document sets. It does not involve human participants, human-derived materials, animal subjects, or personally identifying data. The evaluated AI systems and AI-assisted judging tools are described in Section 3.4 and Section 3.5.

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Declaration of competing interests

The author is employed by Google. One of the five models evaluated in the case study (Gemini 2.5 Pro) is a Google product. The research was conducted independently and does not represent the views or endorsement of Google. The author’s employer had no role in the design, execution, analysis, or reporting of this study.

Declaration of generative AI use

During this work the author used AI coding assistants (Anthropic Claude and OpenAI Codex) to help write the analysis and scoring code. The author takes full responsibility for the content of this publication.

Data Availability Statements

The scenario generators, prompt templates, data-collection and scoring scripts, model outputs, scored results, figure-generation scripts, and replication instructions are publicly available at <https://github.com/ilpenzo/hallucination-fragility> and archived at <https://doi.org/10.5281/zenodo.23074039> under the MIT License. The archive contains all generated data needed to reproduce the results reported in the paper, except for access credentials required to re-query commercial model APIs.

CRediT authorship contribution statement

The author was responsible for Conceptualization, Methodology, Software, Data Curation, Formal Analysis, Investigation, Validation, Visualization, Project Administration, Writing - Original Draft, and Writing - Review & Editing.

A. Inter-Rater Reliability

To validate the P3 LLM judge, all 100 entries (5 models $\times$ 20 scenarios) were independently re-judged by GPT-5.2 (OpenAI, reasoning effort: medium) using the identical rubric and scoring schema applied by the primary judge (Claude Opus 4.6).

Scores were binned into ordinal categories (low: [0, 0.25), mid: [0.25, 0.75), high: [0.75, 1.0]). Because three of five dimensions exhibit extreme marginal skew (nearly all entries scored “high” by both judges), Cohen’s κ is subject to the Feinstein–Cicchetti prevalence paradox [22], producing near-zero or negative values despite $\geq 95\%$ agreement. We therefore report Gwet’s AC1 [23] as the primary agreement statistic, which remains well-behaved under prevalence skew, alongside raw agreement and κ where the latter is interpretable.

Table 8

Inter-rater reliability: Claude Opus 4.6 vs. GPT-5.2 on all 100 P3 entries. AC1 is Gwet’s prevalence-robust agreement coefficient. κ_{lin} : Cohen’s κ with linear weighting (reported where marginal variance permits interpretation). The overall agreement percentage covers all 500 pairs; the overall AC1 and κ_{lin} cover the 400 ordinal pairs.

Dimension	Agree	AC1	κ_{lin}	Note
Contr. identified	100%	1.000	undef.	Prevalence
Contr. resolved	95%	0.949	−0.013	Prev. paradox
Reasoning quality	98%	0.980	−0.010	Prev. paradox
Gap abstention	96%	0.955	0.877	Almost perfect
Fabrication	97%	0.967	0.712	Substantial
Overall	97.2%	0.971	0.718	

Table 8 presents the full results. Overall binned agreement is 97.2% across 500 paired dimension scores (5 dimensions $\times$ 100 entries); the overall AC1 (0.971) and κ_{lin} (0.718) are computed over the 400 ordinal pairs, and fabrication is a separate binary comparison. Gap abstention and fabrication, the two dimensions with sufficient marginal variance for standard κ , yield $\kappa_{\text{lin}} = 0.877$ (almost perfect) and $\kappa = 0.712$ (substantial), respectively. Bootstrap 95% confidence intervals (2,000 replicates) for gap abstention are AC1: [0.907, 0.989], κ_{lin} : [0.735, 0.973].

The negative κ values for contradiction resolution and reasoning quality reflect the prevalence paradox, not disagreement: both judges assign nearly all entries to the “high” bin. PABAK (prevalence-adjusted κ ; [22]) for these dimensions is 0.925 and 0.970 respectively, consistent with the AC1 values.

Table 9

Raw-score (unbinned) agreement between the two judges on the same 100 entries. “Within 0.1” is the share of entries whose scores differ by at most 0.1; the Pearson correlation is undefined where one dimension has no variance.

Dimension	Exact	≤ 0.1	MAD	Max	r
Contr. identified	100%	100%	0.000	0.00	undef.
Contr. resolved	93%	94%	0.024	0.80	0.47
Reasoning quality	96%	98%	0.009	0.50	0.25
Gap abstention	94%	94%	0.020	0.50	0.95
Fabrication (binary)	97%	—	—	—	—

Table 9 reports agreement on the raw scores before binning. The two dimensions with near-zero variance (contradiction resolution and reasoning quality) have low raw correlations for the same prevalence reason that depresses κ , but exact agreement of 93–96% and mean absolute differences below 0.025; gap abstention, where scores vary, correlates at 0.95.

Only three fabrication judgments (across two scenarios) disagreed, and per-model binned agreement ranged from 95% to 100% on every dimension.

B. Targeted Replication

To assess sensitivity to the single-run design, we replicated 20 scenarios selected for high cross-model variance or high stakes (5 P1, 10 P2, 5 P3; 100 conversations, 5 models, temperature 0, collected in late February 2026). Variation at temperature 0 reflects implementation nondeterminism (for example floating-point and batching differences) rather than sampling; we did not test its cause.

For P1, scored with the same final extraction rule as the main results, the four phase-end answers agreed between runs on 95 of 100 comparisons; per-model phase-end accuracy on the five replicated scenarios changed by at most 0.10 (MiniMax M2.5, 0.80 to 0.90).

For P2, scored with protocol v3.1, the ten replicated scenarios were chosen as the hardest post-conflict cases and all but one are infeasible after turn 15, so the replication speaks mainly to the feasible checkpoints at turns 6 and 11 (21 matched feasible

checkpoints per model). Mean satisfaction of the effective requirements on those checkpoints changed between runs by +0.18 for Sonnet 4.5 (0.74 to 0.92), +0.08 for GPT-4o, +0.06 for Gemini 2.5 Pro, and by less than 0.01 for MiniMax M2.5 and DeepSeek-V3.2. Sonnet 4.5’s change comes from checkpoints where it declined to plan in the original run (asking which capacity value applies, or asserting a conflict) and supplied a plan in the replication, a full-scale change for that checkpoint. Complete valid plans on the 21 checkpoints changed from 14 to 15 (Sonnet 4.5), 20 to 17 (DeepSeek-V3.2), 16 to 18 (Gemini 2.5 Pro), 0 to 1 (GPT-4o), and 15 to 12 (MiniMax M2.5). DeepSeek-V3.2 had the highest satisfaction in both runs; on valid plans it was first in the original run and second to Gemini 2.5 Pro in the replication. We report this variation without attributing it to a cause, and Table 2’s scenario-bootstrap intervals do not include it.

For P3, fabrication judgments agreed on 23 of 25 scenario-model pairs (92%). Of the five original fabrication-labeled pairs in the replicated subset, three replicated on the same scenario-model pairs and two shifted from fabrication to partial abstention. All 20 originally non-fabricating pairs remained clean.

C. Sensitivity Analyses

Plan extraction.

Table 10 classifies the 160 checkpoint responses per model by the plan the model explicitly proposed, under the block-scoped extraction rules of Section 3.3. The originally submitted extractor read assignments from outside the proposed plan (in 118 of GPT-4o’s and 53 of Gemini 2.5 Pro’s checkpoints it assigned items the model had listed as unassigned), which inflated those models’ satisfaction; the rigid parser of the first scoring pass counted unparsed plans as failures. Neither is used in this version. A corpus-wide comparison of the block-scoped extractor against a permissive scan then exposed five kinds of false negative, listed in the artifact (for example code fences counted as blank lines, and the abbreviation “HR”). The corrected rules changed 25 of 800 core and 9 of 200 replication extractions, mostly for MiniMax M2.5; each change is listed with its source text in the artifact and was checked in AI-assisted review, not by human annotation. The phrase patterns behind the “asks user” counts were corrected in the same pass (question forms added, generic closing offers removed).

Table 10

Plan output types over all 160 checkpoint responses per model (block-scoped extraction). “No usable plan”: a placeholder template or an all-unassigned list. Cov.: mean share of the eight items assigned.

Model	Complete	Partial	No usable	No plan	Cov.
DeepSeek-V3.2	159	1	0	0	1.00
Sonnet 4.5	121	18	1	20	0.80
MiniMax M2.5	140	15	3	2	0.93
Gemini 2.5 Pro	90	47	6	17	0.68
GPT-4o	34	126	0	0	0.56

Capacity reading.

Table 11 compares the primary capacity rule (both capacity statements must hold) with the alternative reading (the later sentence replaces the table value) on the feasible checkpoints. The verified feasibility labels are identical under both. The

Spearman correlation between P1 and the P2 valid-plan rate is +0.90 under the primary rule and +0.87 under the alternative.

Table 11

Complete valid plans and capacity violations under the two capacity readings (primary / alternative). Valid plans: of 40 checkpoints at turns 6 and 11 and of 29 feasible post-revision checkpoints. Violations: responses placing more items in a slot than allowed, among responses that assign at least one item (n).

Model	T6	T11	Post-rev.	Violations	n
DeepSeek-V3.2	33 / 40	36 / 40	24 / 29	24 / 2	160
Sonnet 4.5	31 / 33	31 / 34	16 / 19	24 / 11	139
Gemini 2.5 Pro	26 / 32	27 / 32	15 / 19	17 / 2	137
MiniMax M2.5	30 / 31	32 / 34	10 / 13	39 / 27	155
GPT-4o	1 / 1	1 / 1	3 / 4	37 / 31	160

Numeric tolerance.

Table 12 recomputes P1 phase-end accuracy from the raw responses at relative tolerances from 0.1% to 10%, retaining each question’s absolute tolerance and using the final asserted answer for the seven adjudicated responses, so its 1% row reproduces Table 1. The model order at the 1% rule is also obtained at 0.1%, 0.5% and 2%; Sonnet 4.5 ties DeepSeek-V3.2 for first at 5% and 10%, and GPT-4o passes Gemini 2.5 Pro only at 10%.

Table 12

P1 phase-end accuracy by relative tolerance (seven adjudicated corrections included).

Tol.	Sonnet	DeepSeek	Gemini	GPT-4o	MiniMax
0.1%	0.985	0.995	0.965	0.915	0.875
0.5%	0.985	0.995	0.965	0.950	0.875
1% (used)	0.985	0.995	0.965	0.955	0.880
2%	0.990	0.995	0.965	0.955	0.890
5%	0.995	0.995	0.965	0.960	0.915
10%	0.995	0.995	0.965	0.980	0.955

Incompatibility-claim classification.

The visible answer of each turn-15 and turn-17 response (400 responses) was labelled YES, NO, or UNCLEAR against a frozen written rubric by an LLM classifier (Claude Opus 4.6) with model identity, scenario, and feasibility hidden; each label must carry a verbatim quote. Of the 400 quotes, 396 were verified against the response; the four responses whose quote could not be verified are treated as unclear, together with three that the classifier itself labelled unclear. Two samples (60 selected cases, then 30 further cases) had been labelled beforehand in a separate AI-assisted review, not by human annotators. The classifier agreed with that review on 55 of 60 and on 28 of 29 decided cases (1 labelled unclear); all labels, quotes, disagreements, and the rubric are released.

References

1. S. Lin, J. Hilton, and O. Evans, “TruthfulQA: Measuring how models mimic human falsehoods,” in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics, 2022, pp. 3214–3252, doi: <https://doi.org/10.18653/v1/2022.acl-long.229>. [Online]. Available: <https://aclanthology.org/2022.acl-long.229/>
2. J. Li, X. Cheng, W. X. Zhao, J.-Y. Nie, and J.-R. Wen, “HaluEval: A large-scale hallucination evaluation

- benchmark for large language models,” in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2023, pp. 6449–6464, doi: <https://doi.org/10.18653/v1/2023.emnlp-main.397>. [Online]. Available: <https://aclanthology.org/2023.emnlp-main.397/>
3. S. Min *et al.*, “FactScore: Fine-grained atomic evaluation of factual precision in long form text generation,” in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2023, pp. 12 076–12 100, doi: <https://doi.org/10.18653/v1/2023.emnlp-main.741>. [Online]. Available: <https://aclanthology.org/2023.emnlp-main.741/>
 4. P. Liang *et al.*, “Holistic evaluation of language models,” *Transactions on Machine Learning Research*, 2023. [Online]. Available: <https://openreview.net/forum?id=iO4LZibEqW>
 5. A. Srivastava *et al.*, “Beyond the imitation game: Quantifying and extrapolating the capabilities of language models,” *Transactions on Machine Learning Research*, 2023. [Online]. Available: <https://openreview.net/forum?id=uyTL5Bvosj>
 6. J. Xie *et al.*, “TravelPlanner: A benchmark for real-world planning with language agents,” in *Proceedings of the 41st International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, vol. 235, 2024, pp. 54 590–54 613. [Online]. Available: <https://proceedings.mlr.press/v235/xie24j.html>
 7. K. Valmeekam, M. Marquez, A. Olmo, S. Sreedharan, and S. Kambhampati, “PlanBench: An extensible benchmark for evaluating large language models on planning and reasoning about change,” in *Advances in Neural Information Processing Systems 36 (NeurIPS)*, 2023, pp. 38 975–38 987. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/hash/7a92bcdede88c7afd108072faf5485c8-Abstract-Datasets_and_Benchmarks.html
 8. Z. Ji *et al.*, “Survey of hallucination in natural language generation,” *ACM Computing Surveys*, vol. 55, no. 12, pp. 1–38, 2023, doi: <https://doi.org/10.1145/3571730>.
 9. N. F. Liu *et al.*, “Lost in the middle: How language models use long contexts,” *Transactions of the Association for Computational Linguistics*, vol. 12, pp. 157–173, 2024, doi: <https://doi.org/10.1162/tacl.a.00638>. [Online]. Available: <https://aclanthology.org/2024.tacl-1.9/>
 10. L. Zheng *et al.*, “Judging LLM-as-a-judge with MT-Bench and Chatbot Arena,” in *Advances in Neural Information Processing Systems 36 (NeurIPS)*, 2023. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets_and_Benchmarks.html
 11. K. Chen, Q. Chen, J. Zhou, Y. He, and L. He, “DiaHalu: A dialogue-level hallucination evaluation benchmark for large language models,” in *Findings of the Association for Computational Linguistics: EMNLP 2024*. Association for Computational Linguistics, 2024, pp. 9057–9079, doi: <https://doi.org/10.18653/v1/2024.findings-emnlp.529>. [Online]. Available: <https://aclanthology.org/2024.findings-emnlp.529/>
 12. V. Rathore, S. Aneesh, and H. Singh, “Temporal graph network: Hallucination detection in multi-turn conversation,” *arXiv preprint*, 2026, arXiv:2601.03051. doi: <https://doi.org/10.48550/arXiv.2601.03051>. [Online]. Available: <https://arxiv.org/abs/2601.03051>
 13. Y. Liu, D. Iter, Y. Xu, S. Wang, R. Xu, and C. Zhu, “G-Eval: NLG evaluation using GPT-4 with better human alignment,” in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2023, pp. 2511–2522, doi: <https://doi.org/10.18653/v1/2023.emnlp-main.153>. [Online]. Available: <https://aclanthology.org/2023.emnlp-main.153/>
 14. W.-L. Chiang *et al.*, “Chatbot Arena: An open platform for evaluating LLMs by human preference,” in *Proceedings of the 41st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 235, 2024, pp. 8359–8388. [Online]. Available: <https://proceedings.mlr.press/v235/chiang24b.html>
 15. M. G. Kendall and B. Babington Smith, “The problem of m rankings,” *The Annals of Mathematical Statistics*, vol. 10, no. 3, pp. 275–287, 1939, doi: <https://doi.org/10.1214/aoms/1177732186>.
 16. K. Deshpande *et al.*, “MultiChallenge: A realistic multi-turn conversation evaluation benchmark challenging to frontier LLMs,” in *Findings of the Association for Computational Linguistics: ACL 2025*. Association for Computational Linguistics, 2025, pp. 18 632–18 702, doi: <https://doi.org/10.18653/v1/2025.findings-acl.958>. [Online]. Available: <https://aclanthology.org/2025.findings-acl.958/>
 17. D. Wu, H. Wang, W. Yu, Y. Zhang, K.-W. Chang, and D. Yu, “LongMemEval: Benchmarking chat assistants on long-term interactive memory,” in *Proceedings of the Thirteenth International Conference on Learning Representations (ICLR)*, 2025. [Online]. Available: <https://arxiv.org/abs/2410.10813>
 18. P. Laban, H. Hayashi, Y. Zhou, and J. Neville, “LLMs get lost in multi-turn conversation,” *arXiv preprint*, 2025, arXiv:2505.06120. [Online]. Available: <https://arxiv.org/abs/2505.06120>
 19. T. Gebru *et al.*, “Datasheds for datasets,” *Communications of the ACM*, vol. 64, no. 12, pp. 86–92, 2021, doi: <https://doi.org/10.1145/3458723>.
 20. A. Paullada, I. D. Raji, E. M. Bender, E. Denton, and A. Hanna, “Data and its (dis)contents: A survey of dataset development and use in machine learning research,” *Patterns*, vol. 2, no. 11, p. 100336, 2021, doi: <https://doi.org/10.1016/j.patter.2021.100336>.
 21. D. Kiela *et al.*, “Dynabench: Rethinking benchmarking in NLP,” in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, 2021, pp. 4110–4124, doi: <https://doi.org/10.18653/v1/2021.naacl-main.324>. [Online]. Available: <https://aclanthology.org/2021.naacl-main.324/>
 22. T. Byrt, J. Bishop, and J. B. Carlin, “Bias, prevalence and kappa,” *Journal of Clinical Epidemiology*, vol. 46, no. 5, pp. 423–429, 1993, doi: [https://doi.org/10.1016/0895-4356\(93\)90018-V](https://doi.org/10.1016/0895-4356(93)90018-V).
 23. K. L. Gwet, “Computing inter-rater reliability and its variance in the presence of high agreement,” *British Journal of Mathematical and Statistical Psychology*, vol. 61, no. 1, pp. 29–48, 2008, doi: <https://doi.org/10.1348/000711006X126600>.