

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.

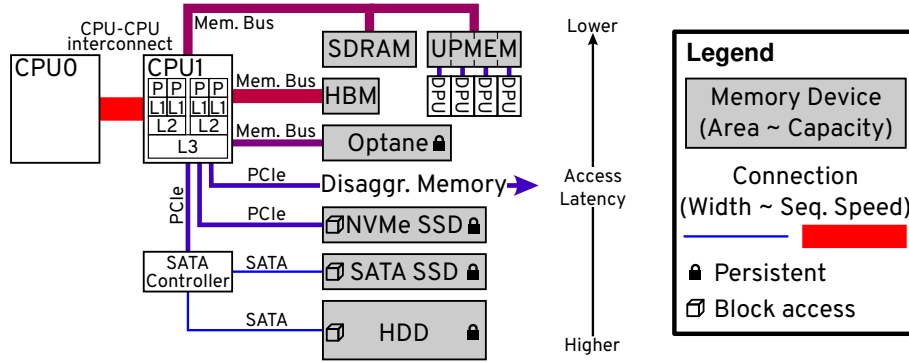


Fig. 1. Overview of different memory components in a modern computing system: The placement of the memory devices reflects their access latency, forming an arrangement inspired by the classical memory pyramid. Box size and line width (supported by color gradient) represent typical sizes and sequential 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 [4]	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/>

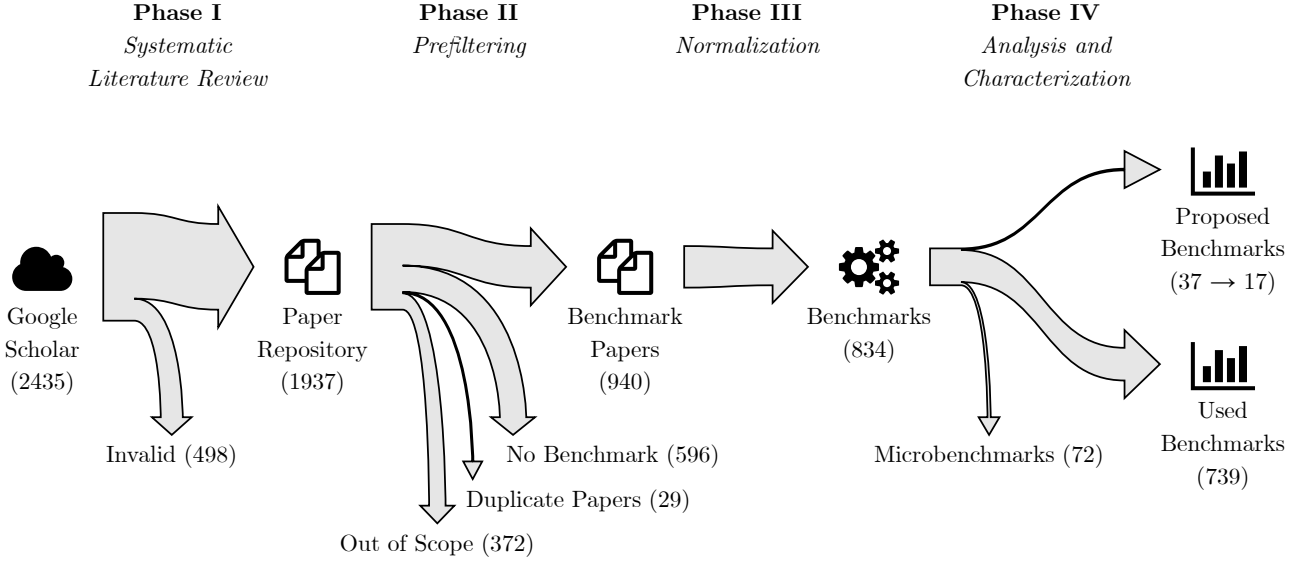


Fig. 2. High-level overview of the literature review workflow used to identify benchmarks for analysis and characterization. Arrow widths are proportional to paper/benchmark counts. Downwards arrows represent excluded papers/benchmarks. *Benchmark Papers* represent those papers that use or propose a 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	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>

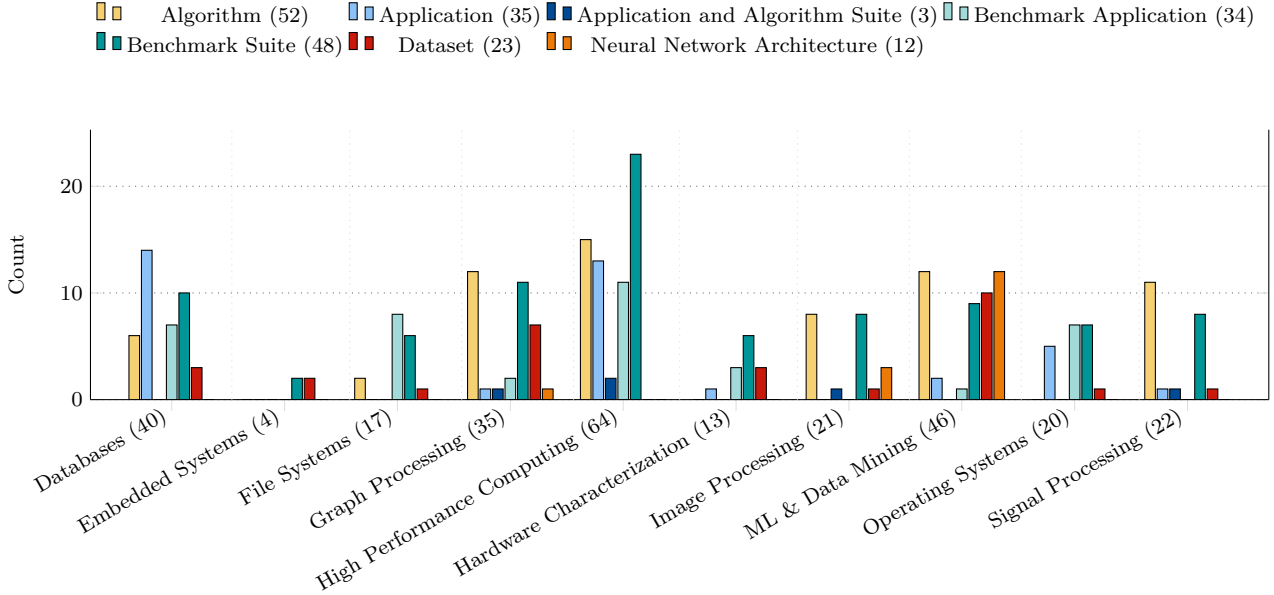


Fig. 3. Histogram showing for which domain which types of benchmarks have been found. Numbers in parentheses reflect the total group counts for each domain or type. Notably, each benchmark only has a single type but can belong to several domains (especially suites), which leads to a higher sum of benchmarks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

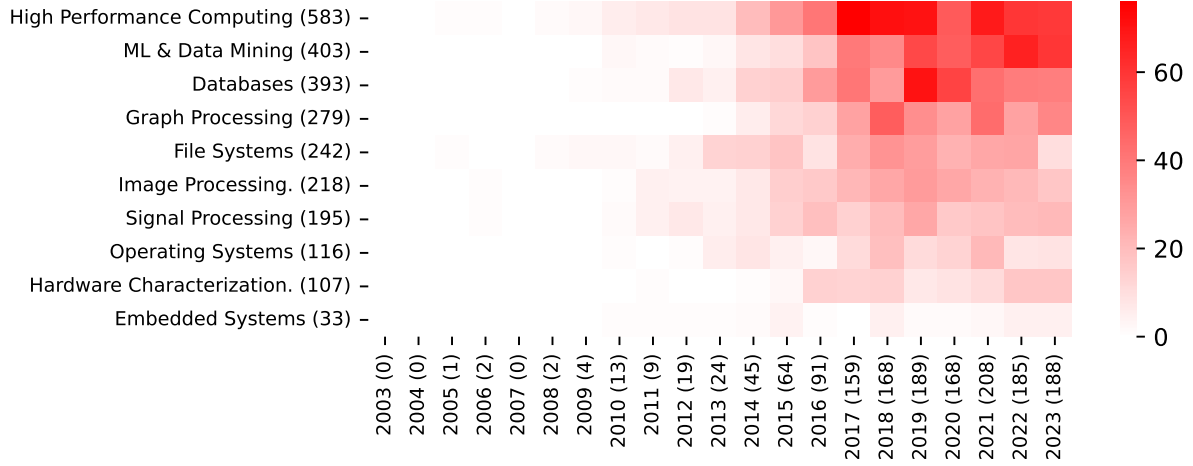
As a first indicator of how extensively the community benchmarks their systems, we investigate the number of domains that are used

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.

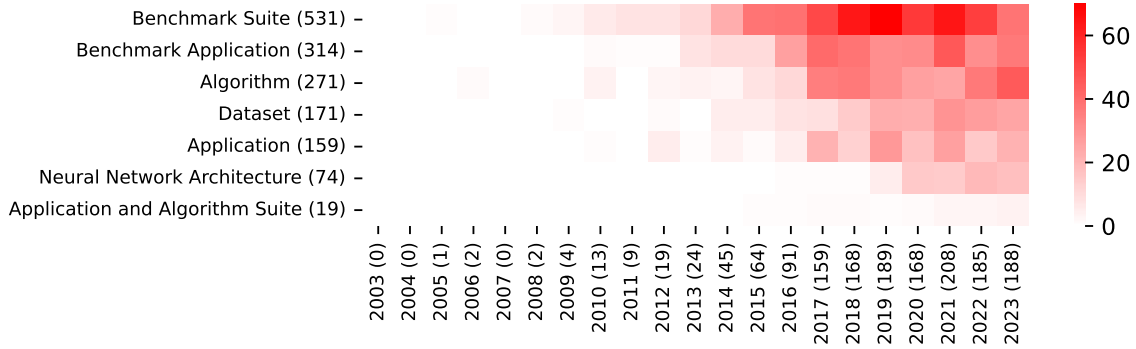
6.3.2. $AQ_{3.2}$ – Using Different Benchmarking Types

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.

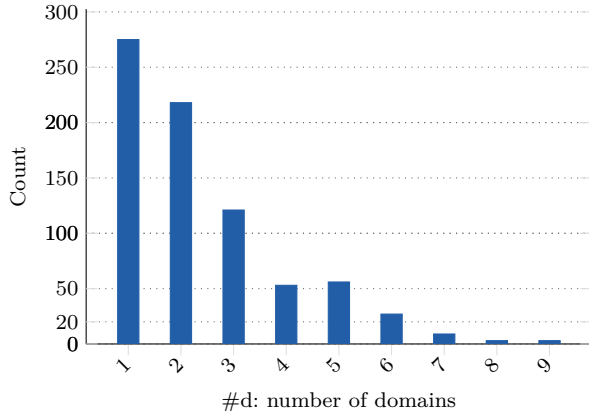


Fig. 4. Distribution of papers based on the number of domains used for evaluation. The figure illustrates the count of papers utilizing benchmarks from #d different domains.

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

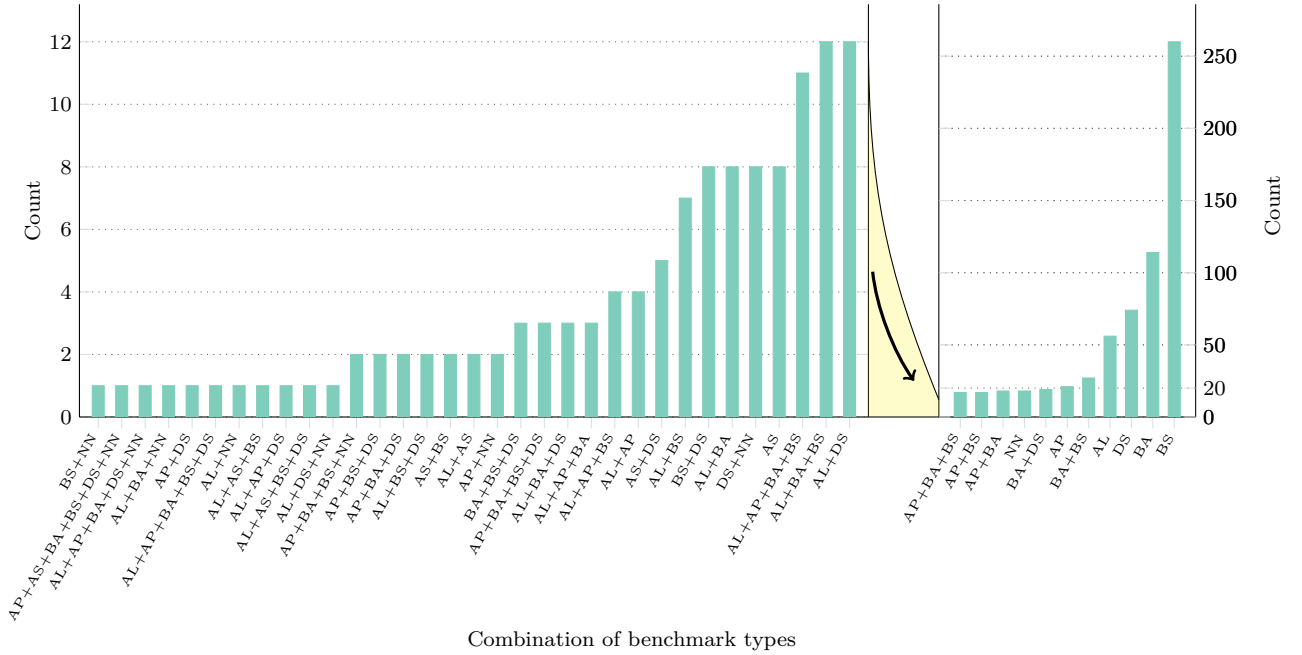


Fig. 5. Distribution of benchmark type combination across papers. The figure shows the number of benchmarks that utilize specific sets of benchmark types. Used abbreviations in the figure: **AL**: *Algorithm*, **AP**: *Application*, **AS**: *Application & Algorithm Suite*, **BA**: *Benchmark Application*,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