Towards Memory Specialization: A Case for Long-Term and Short-Term RAM

Peijing Li
Stanford University
Stanford, CA, USA
peli@stanford.edu

Muhammad Shahir Abdurrahman
Stanford University
Stanford, CA, USA
shahir@stanford.edu

Rachel Cleaveland
Stanford University
Stanford, CA, USA

Sergey Legtchenko
Microsoft Research
Cambridge, England

Philip Levis
Stanford University
Stanford, CA, USA

Ioan Stefanovici
Microsoft Research
Cambridge, England

Thierry Tambe
Stanford University
Stanford, CA, USA

David Tennenhouse
Independent Researcher

Caroline Trippel
Stanford University
Stanford, CA, USA

H.-S. Philip Wong
Stanford University
Stanford, CA, USA

Abstract

Both SRAM and DRAM have stopped scaling: there is no technical roadmap to reduce their cost (per byte/GB). As a result, memory now dominates system cost. This paper argues for a paradigm shift from today's simple memory hierarchy toward specialized memory architectures that exploit application-specific access patterns. Rather than relying solely on traditional off-chip DRAM and on-chip SRAM, we envisage memory systems equipped with additional types of memory whose performance trade-offs benefit workloads through non-hierarchical optimization.

We propose two new memory classes deserving explicit OS support: long-term RAM (LtRAM) optimized for read-intensive data with long lifetimes, and short-term RAM (StRAM) designed for transient, frequently accessed data with short lifetimes. We explore underlying device technologies that could implement these classes, including their evolution and potential integration into current system designs given emerging workload requirements. We identify critical research challenges to realize what we believe is a necessary evolution toward more efficient and scalable computing systems capable of meeting future demands.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
DIMES '25, Seoul, Republic of Korea
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-2226-4/25/10
https://doi.org/10.1145/3764862.3768175

CCS Concepts: • **Software and its engineering** → *Operating systems*; • **Hardware** → **Memory and dense storage**; **Semiconductor memory**; *Analysis and design of emerging devices and systems.*

Keywords: Operating system, Memory architecture, Heterogeneous computing, LtRAM, StRAM

ACM Reference Format:
Peijing Li, Muhammad Shahir Abdurrahman, Rachel Cleaveland, Sergey Legtchenko, Philip Levis, Ioan Stefanovici, Thierry Tambe, David Tennenhouse, Caroline Trippel, and H.-S. Philip Wong. 2025. Towards Memory Specialization: A Case for Long-Term and Short-Term RAM. In *Workshop on Disruptive Memory Systems (DIMES '25), October 13–16, 2025, Seoul, Republic of Korea.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3764862.3768175

1 Introduction

For decades, computer system performance improvements have relied on continual memory and compute scaling through Moore's Law. For memory, that scaling has stalled. The two dominant memory technologies, SRAM and DRAM, have hit fundamental physical limitations, making memory the primary performance, power, and cost bottleneck for systems.

These scaling limitations have created a critical divergence between compute and memory capabilities. While processor and network performance continue improving through architectural advances, SRAM and DRAM densities and costs have stagnated. Memory now dominates system cost, for instance, with DRAM accounting for over 50% of server hardware costs [45]. As memory-to-compute capacity ratios fall, applications are becoming increasingly memory-bound.

Memory-intensive workloads, particularly artificial intelligence, have exacerbated this challenge. Bandwidth demands of these workloads have driven innovations such as High Bandwidth Memory (HBM), which uses advanced packaging to place DRAM interfaces close to compute elements [38]

for lower latency and better energy efficiency [42]. However, packaging improvements alone cannot overcome fundamental SRAM and DRAM scaling limitations.

Recent research has begun exploring memory technologies that trade off characteristics such as retention[1] and endurance[2] for workload-specific memory access patterns and data lifetimes.[3] For example, Managed Retention Memory (MRM) [30] proposes repurposing storage-class memories for workloads like large language model (LLM) inference by trading retention time and write performance for improved endurance and lower I/O energy. Similarly, for on-chip memory arrays, gain cell embedded DRAM [35] achieves DRAM-like density with SRAM-like integration properties through similar retention trade-offs.

Building on these insights, we propose a fundamental shift from hierarchical to specialized memory architectures. Instead of exposing numerous heterogeneous device technologies directly to system software, we propose defining new types of memory based on common access patterns. We advocate for two new abstractions of memory device technologies: long-term RAM (LtRAM) for persistent, read-heavy data, and short-term RAM (StRAM) for ephemeral, write-intensive workloads. Each one supports system optimizations and enables diverse underlying implementations tailored to match the corresponding performance tradeoffs.

This paper makes the following three contributions:

1. It proposes LtRAM and StRAM as new memory classes that define specific performance and lifetime characteristics needed by modern workloads.
2. It explores underlying device technologies that could implement LtRAM and StRAM and examines their integration into current system designs.
3. It identifies key research questions that must be resolved to realize the potential of LtRAM and StRAM in both single-node and distributed computing contexts.

2 Memory Today: SRAM and DRAM

Today's computing systems have a memory hierarchy, consisting of SRAM caches on the compute chip (L1, L2, etc.), off-chip DRAM main memory, and NAND flash or other non-volatile storage for permanent data. Table 1 shows a detailed breakdown of their tradeoffs.

Static RAM (SRAM) consists of six transistors per bit, operates at high speed, statically retains its state while powered, and integrates easily with compute logic. These properties have made SRAM the default choice for on-chip caches and scratchpad memories.

[1]Retention is the time that data is reliably stored in a memory cell without requiring a refresh.
[2]Endurance is the number of write cycles a memory cell can support before it permanently degrades.
[3]Data lifetime is the duration for which the data must remain persistent and accessible in memory hardware.

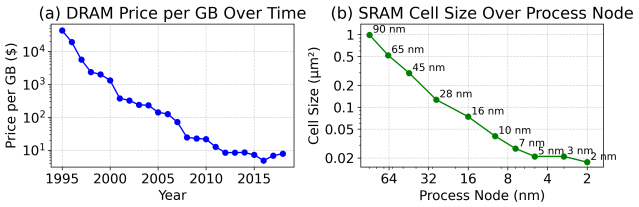

Figure 1. SRAM and DRAM scaling [4], showing (a) DRAM cost per bit stagnation and (b) SRAM cell size stagnation.

Dynamic RAM (DRAM) consists of a single transistor and a capacitor per bit. The capacitor stores charge representing data bits, while the transistor gates cell access. To maximize density, DRAM capacitors are narrow and tall, making integration with high-performance logic impractical [49] [22]. DRAM cells have limited retention due to charge leakage, and reads are destructive in that they drain the charge from the capacitor. DRAM subsystems therefore need to "refresh" by rewriting contents periodically (e.g., every 32ms for DDR5), drawing significant power even during idle periods.

2.1 The End of Scaling

Both SRAM and DRAM face the same fundamental challenge: they have stopped scaling. Figure 1(b) shows that SRAM cell sizes have stopped shrinking significantly since the 7nm process node [63]. This stagnation stems from the fundamental physics of SRAM operation, where maintaining the delicate balance between transistor sizing and threshold voltages becomes increasingly difficult at smaller process nodes and lower operating voltages [20].

Figure 1(a) shows that DRAM cost trends have stagnated over the past 15 years. While smaller DRAM cells remain technically feasible, manufacturing difficulties from capacitor sizing constraints prevent reductions in per-cell cost [4]. Although 3D-stacking DRAM cells in HBM can extend density scaling somewhat, this density increase may stop beyond 20 dies due to packaging complexity and cost, placing an upper bound on the capacities of HBM devices [15].

2.2 Packaging Options

One current approach to address the scaling limitations of SRAM and DRAM is to innovate in packaging rather than cell technology.

SRAM is typically integrated inside compute chips but can also be 3D-stacked for greater capacity [51]. There are even more packaging options for DRAM. Historically packaged in DIMMs (Dual Inline Memory Modules) for modularity, DRAM packaging increasingly focuses on low power and/or high bandwidth options. LPDDR achieves lower power and higher bandwidth (10Gbps/pin versus 5.6Gbps/pin in conventional DDR) [52] but must be soldered to boards, sacrificing reconfigurability [32]. HBM achieves much higher bandwidth through 3D stacking of multiple dies connected

via silicon interposers [38]. This complex process requires advanced facilities and lowers manufacturing yield, making HBM substantially more expensive than DDR DRAM [45] while constraining capacity: terabyte on-package configurations are not practically possible [26].

These packaging innovations allow systems to include more memory, or provide higher bandwidth to smaller memories, but do not solve the fundamental cost per byte scaling limitations of SRAM and DRAM cell technologies.

3 Escaping the Memory Scaling Trap

To address existing memory challenges, alternative memory technologies have emerged with fundamentally different scaling roadmaps. Examples include gain cell embedded DRAM [35], ferroelectric RAM (FeRAM) [37], magnetoresistive RAM (MRAM) [12], and resistive RAM (RRAM) [53]. Each offers different trade-offs in density, endurance, retention, read/write energies, access times, and integration capabilities. None are drop-in replacements for SRAM or DRAM, as those technologies have been optimized for their specific trade-offs, making them each suitable for different applications. This section quantifies their scaling benefits, examines the constraints that prevent drop-in replacement, and demonstrates why these limitations necessitate a shift toward specialized memory systems.

3.1 Density Benefits over SRAM and DRAM

Many of the emerging memory devices are designed to be denser than SRAM or DRAM, while having the potential to maintain that advantage as process nodes shrink. One such example is gain cell embedded DRAM, which can achieve 2-3×higher density than SRAM at similar process nodes [18] while also allowing on-chip integration due to its all-transistor structure.

RRAM demonstrates lower read energy and latency than DRAM at higher densities due to its simple cell structure consisting of either one transistor and one resistor or only a single resistor [53]. Scaling trends for RRAM are illustrated in Figure 2, which compares the area densities of DRAM and HBM[4] with various RRAM configurations, from 2D planar RRAM cells [61] to 3D-stacked V-RRAM architectures with 8 and 64 layers [3, 24, 36].

Figure 2 demonstrates that while planar RRAM cells achieve up to 2x the density of DRAM, the most significant scaling benefits emerge from 3D V-RRAM architectures, which can reach up to 10x the density of state-of-the-art HBM4 at the same process node. These advantages may extend beyond current nodes: materials and device changes suggest RRAM can continue scaling to smaller process sizes [62]

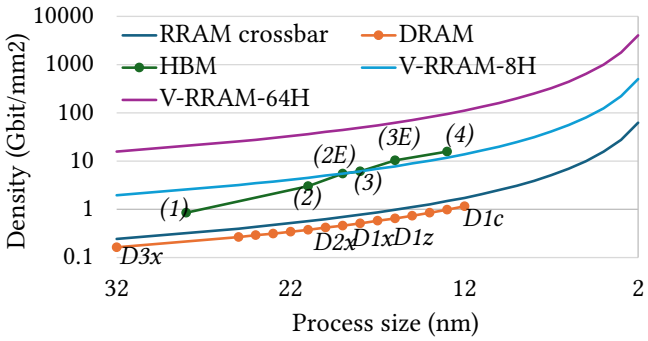

Figure 2. Scaling trends and potentials of DRAM [9] and HBM compared to RRAM.

while DRAM would struggle below 11nm, and V-RRAM architectures can stack up to 64 layers compared to HBM's 16-die limit.

3.2 Constraints of Emerging Memory Technologies

While technologies like gain cells and RRAM offer compelling density advantages, they cannot replace SRAM or DRAM due to fundamental trade-offs in cell properties. These technologies exhibit asymmetric read/write performance and explicit trade-offs between retention time and write endurance.

For instance, RRAM's limited endurance and high write energy prevent its use for frequently overwritten transient data [53], despite its superior scaling potential. Similarly, gain cell RAM requires periodic refreshes due to its reliance on intrinsic transistor capacitance rather than SRAM's feedback mechanism for data retention.

These constraints require aligning data access patterns with the characteristics of underlying cell technologies [8]. There are two key properties to consider. First, for memories with limited endurance, the write rate of the data over the expected service life of the device should be consistent with the cell endurance. Second, matching the data retention of the cells to the lifetime of the data helps reduce energy consumption. This motivates memory specialization [40], where memory systems can be built with a heterogeneous set of memory device arrays, each optimized for specific write patterns and data lifetimes, such that they can cover the full spectrum of memory accesses in a target workload.

4 A Proposal for LtRAM and StRAM

The broadening landscape of emerging memory technologies poses a significant challenge for system software. Specialized, heterogeneous memory systems that may be unique to individual devices cannot remain entirely transparent to the software stack. Instead, operating systems, middleware, and applications must adapt to different memory devices with varying properties while remaining compatible with current and future devices without constant updates.

[4]HBM uses the same process size as DRAM, and stacks 4 dies for HBM1, 8 dies for HBM2, 12 dies for HBM2E and 3, and 16 dies for HBM3E and 4

We propose taming this complexity by organizing emerging devices into just two new memory classes: Short-term RAM (StRAM) and Long-term RAM (LtRAM). Table 1 shows how these additions create a five-class memory system that preserves existing SRAM, DRAM, and NAND flash (along with other forms of non-volatile storage) while adding targeted capabilities for emerging workloads. Rather than managing numerous individual technologies, this abstraction provides operating systems with manageable abstractions accommodating diverse underlying implementations. While one could envision a finer-grained continuum of retention and access characteristics, these two classes capture the majority of emerging device and workload trade-offs without excessive segmentation.

SRAM, DRAM, and NAND flash in Table 1 are already well-established in today's memory hierarchies. We focus the remainder of this section on the two additional memory classes, StRAM and LtRAM, which complement existing technologies to service specific workloads.

4.1 Short-term RAM (StRAM)

Short-term RAM (StRAM) is a memory class optimized for transient data that are either frequently accessed or read once and discarded. It focuses on short, sub-second data lifetimes and low energy costs. It is underpinned by memory technologies that have symmetric read/write performance and very high (or unlimited) endurance, but not necessarily high retention. Key use cases of StRAM exploit data transience and include intermediate activation buffers in neural networks, pointer chasing, and temporary data structures such as FIFOs and message queues in server applications.

StRAM can be implemented using several underlying device technologies. Gain cell embedded DRAM is a prime example [35]. Another possibility is 1T1C[5] FeRAM [37] if its endurance can be brought high enough. Integration approaches range from on-die placement to near-compute configurations, depending on device characteristics and thermal constraints.

4.2 Long-term RAM (LtRAM)

Long-term RAM (LtRAM) builds on the idea of managed-retention memory [30]. LtRAM is optimized for persistent, read-heavy data with long data lifetimes on the order of minutes and longer. Unlike StRAM, LtRAM prioritizes read performance and read energy efficiency over write characteristics. LtRAM makes explicit trade-offs that accept higher write latencies and energy costs, as well as lower endurance, in exchange for superior read properties and retention characteristics. Example technologies that could underpin LtRAM include non-volatile memories such as RRAM, MRAM, FeRAM, and managed-retention DRAM variants. These offer different trade-offs with respect to read/write asymmetry, density, endurance, and integration complexity, and can be deployed on-die, in 3D-stacked packages, or off-chip modules depending on application needs.

The key insight motivating LtRAM is that long data lifetimes and read-heavy access patterns allow optimizations that are unsuitable for general-purpose memories [8]. Primary applications include model weights in ML inference, code pages, hot instruction paths, and relatively static data pages – workloads that can tolerate higher write costs in exchange for lower read energy and improved cost per bit. This specialization addresses fundamental mismatches in current systems where read-intensive data competes for the same resources as frequently modified data.

5 Specialization Opportunities

Modern workloads exhibit distinct memory access patterns and data lifetimes that align naturally with LtRAM and StRAM characteristics. Prior work on managed retention memory demonstrated this approach for LLM inference, finding that general-purpose DRAM was too slow while HBM was over-provisioned in write performance and endurance [30]. The rest of this section identifies three key application domains where memory specialization offers significant benefits by matching access patterns to underlying device characteristics.

5.1 Server Workloads

Server applications exhibit diverse memory access patterns that could benefit from specialized memory technologies.

As examples, in-memory data stores such as Redis [7], Memcached [16], and Ray Plasma [1], along with CDNs and DNS servers, exemplify read-mostly workloads with variable data lifetimes. These systems can adapt their caching algorithms to leverage different LtRAM and StRAM compositions in heterogeneous memory architectures.

Similarly, logging and telemetry systems generate write-once data with short-to-medium lifetimes, making StRAM ideal for buffering before archival to off-chip flash or other non-volatile storage. Search engines such as ElasticSearch [27] and Lucene [5] maintain large inverted indices that are read-intensive with long lifetimes, while their query processing generates short-lived data structures. Serverless platforms and database buffer pools further demonstrate this heterogeneity – serverless environments create ephemeral data that last for only a few seconds, while databases exhibit patterns where hot and cold data pages suit StRAM and LtRAM, respectively.

Code pages present another compelling case – they are read-intensive with very long lifetimes yet stored in the same DRAM as frequently modified data in current systems. LtRAM would enable more energy-efficient and cost-effective storage.

[5] 1T1C: one transistor, one capacitor

	SRAM	DRAM	NAND	StRAM	LtRAM
Strengths	Low R/W latency, long retention, low static power	Very dense	Extremely dense, low static power	Dense, low write energy, low static power	Very dense, low read energy, low static power
Weaknesses	Low density	Off-chip only, R/W energy, high static power, refresh overhead, destructive reads	Off-chip only, low bandwidth, limited endurance, expensive erases, very high latency	Short retention, refresh overhead	High write energy, high write latency, limited endurance
Uses	Fast R/W caches	Large, random access, read/write data	Storage, rarely accessed data	Fast R/W caches, write-and-read scratchpads	Read-mostly data, read-mostly caches

Table 1. Five Types of Memory. "Power" in this table refers to the continual, leakage draw from the technology, while "energy" refers to the cost of an active read or write operation of a memory cell. Static power only changes with clock frequencies while active energy scales with use.

5.2 Machine Learning Workloads

AI workloads exhibit highly predictable data flows that align with memory specialization opportunities but misalign with traditional random-access memory characteristics.

During inference, model weights are typically immutable, leading to frequent high-bandwidth, large-block reads, especially for large language models. Traditional HBM and DRAM are over-provisioned in write performance [30], so LtRAM may enable better efficiency and on-chip integration opportunities. On the other hand, activations and intermediate results are immediately discarded after computation, making them suitable for StRAM.

Training workloads combine read-heavy and write-heavy accesses to model weights with write-heavy access to gradients and optimizer states. Activation data during training exhibits distinctive temporal locality – intensive access during forward and backward passes, then immediately discarded after gradient computation. The short lifetimes of activations and gradients make them good StRAM candidates.

5.3 Memory Access Patterns within Processor Cores

Cache hierarchies reveal systematic mismatches between data lifetimes and memory provisioning. Most L1/L2/L3 cache data persists briefly: function call stacks, local temporaries, intermediate results, and thread communication buffers. SRAM's indefinite retention is over-provisioned for these short-lived objects; StRAM could provide comparable performance at higher density and lower static power.

6 Systems Design Challenges

LtRAM and StRAM disrupt traditional memory system design, creating research challenges spanning the entire system stack. We identify several critical research questions that must be addressed to realize the potential of specialized memory systems.

6.1 Abstractions for Post-Hierarchical Memories

Traditional memory systems expose uniform abstractions in flat address spaces that hide device complexity, treating memory as homogeneous. While this provides simple interfaces and programming models, it cannot exploit specialized characteristics of emerging memory technologies or application-specific access patterns that motivated LtRAM and StRAM.

Traditional hierarchies assume proximity correlates with performance: SRAM is closest, fastest, and most expensive; DRAM and NAND flash are progressively farther, slower, and cheaper. Our proposed specialized memory classes break these assumptions, necessitating non-hierarchical optimizations for data placement and access. For example, heterogeneous SRAM and StRAM may both be on-chip scratchpads, with placement determined by application requirements [64] rather than hierarchical positioning. Similarly, LtRAM may be off-chip yet directly accessed for read-heavy, long-lived data despite its lower bandwidth [65].

Successfully navigating the landscape of "post-hierarchical memories" requires exposing individual memory class characteristics to applications and system software. Recent systems demonstrate this trend: NAND flash increasingly exposes internal organization for efficient access [2], while GPU architectures provide explicit SRAM instructions for performance-critical libraries [17]. These examples demonstrate that providing low-level hardware control can yield considerable performance benefits, albeit with increased code complexity.

Thus, the critical research challenge becomes determining how operating systems should expose the retention characteristics, endurance limitations, and read/write asymmetries of LtRAM and StRAM while maintaining programmability. New memory system abstractions must balance performance optimization opportunities with software complexity.

6.2 Data Placement Policies

State-of-the-art memory management policies tend to assume hierarchical relationships that do not apply to specialized memory systems. Linux's NUMA support places memory blocks with similar performance in "nodes," each associated with a group of cores denoting local memory. This supports multi-socket systems and memory locality, creating hierarchies from local DRAM to remote DRAM on interconnects like CXL [46]. These hierarchical assumptions prompted proposals for page placement across memory tiers based on access frequency [28] [29] [47], latency [59], or memory-level parallelism [33].

Systems with post-hierarchical memories violate the aforementioned hierarchical assumptions, especially for LtRAM with large read-write performance differences. The differences between reads and writes are complicated by the fact that reads load into the cache synchronously, while writes, in contrast, write back to main memory asynchronously (e.g., on cache line eviction). Operating systems will require intelligent policies that determine *when* and *where* to place data without relying on hierarchical assumptions.

The timing and authority for data placement decisions vary significantly across applications. Specialized, highly engineered software such as databases and ML frameworks possess domain knowledge to make informed placement decisions. Databases, for example, manage their own buffer pools, while ML frameworks often know their dataflow pipelines. General-purpose or less performance-critical applications, however, can rely on automatic placement by the OS (e.g., hot-cold policies [28, 29], placing pages mapped read-only in LtRAM, or inference based on program behavior [21]). This spectrum necessitates operating systems that transparently manage memory, support application hints, or enable explicit control depending on needs. Prior work on Hotpads showed substantial benefits from changing hardware-software memory management boundaries [56].

Current memory management operates at coarse page-level granularities, which are unable to capture nuanced patterns benefiting from specialization [34]. Access frequency (hot-cold policies) and memory bandwidth tracking [47] [59] [33] insufficiently express lifetime, endurance, caching, or read-write trade-offs. Future systems need fine-grained profiling for these features [31]. Developing lightweight profiling without significant overhead represents a critical challenge. Additionally, propagating the insights from profiling remains an open question. Compiler annotations or special instructions may provide useful hints about application memory access patterns and requirements, similar to the Expressive Memory (XMem) interface [58].

6.3 Memory Consistency and Coherence

Specialized memory classes introduce new challenges for shared memory systems, which rely on memory consistency models (MCMs) to define the legal return values of loads [41]. MCMs vary among instruction set architectures, but generally require that all software threads agree on a total order for accesses to the same memory location, a requirement typically enforced by a hardware cache coherence protocol. Today's MCMs and coherence protocols assume that processors are homogeneous [43] [10] and threads access the same memories – which are volatile and feature high retention times, low write latency and energy, and high endurance – for all data. LtRAM and StRAM each violate some of these assumptions, creating challenges for each class.

```
// Thread 1              // Thread 2
Write data = 1;          // data corrupted to 42
                         Read data = 42;
```

Figure 3. Out-of-thin-air data due to StRAM retention limit.

```
// Thread 1              // Thread 2
Write data = 1;          Read data = 1;
                         // Data corrupted to 0
                         Read data = 0;
```

Figure 4. Coherence violation due to StRAM retention limit

StRAM's limited retention time introduces both MCM and coherence complexities due to the potential for data corruptions. First, corruptions can cause data to appear "out-of-thin-air" (Figure 3),[6] where a read returns a data value that was never written, violating most modern MCMs [6] [44] [50]. Second, data corruptions can cause threads to disagree on the order of memory accesses to the same location (Figure 4), violating coherence. Cache controllers can prevent these behaviors by refreshing cache lines before retention limits expire or proactively evicting them. Software-initiated invalidations, with or without writebacks depending on dataflows, could reduce hardware complexity while maintaining correctness. Notably, these solutions must also address coherence metadata, which faces identical corruption risks.

LtRAM's high write costs may make pervasive invalidation-based coherence protocols less attractive than other alternatives. For example, update-based protocols [10] could benefit applications with read-mostly data and producer-consumer access patterns with few producers and many consumers by avoiding many writes to cache line metadata due to invalidations. Hardware designers may also address this issue by separately storing cache line metadata in write-optimized StRAM and data in read-optimized LtRAM.

Moreover, cache levels featuring multiple memory technologies – e.g., distinct SRAM, StRAM, and LtRAM components within a single L2 cache [31] – create MCM implementation considerations analogous to multibanked memories, since a single software thread can, in principle, access data

[6] All data values in the following code examples are initialized to 0.

```
// Thread 1          // Thread 2
Write data = 1;      Read flag = 1;
Write flag = 1;      Read data = 0; // stale data
```

Figure 5. MCM violation due to LtRAM R-W asymmetry

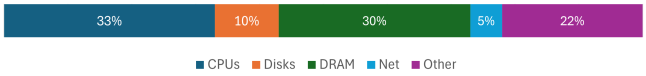

Figure 6. Peak power usage breakdown by components of a warehouse-scale computer [13]

from them concurrently. Such a scenario can cause Thread 1 in Figure 5 to read stale data despite reading an updated flag. This behavior is forbidden by some MCMs (e.g., sequential consistency, total store order), but allowed by others (e.g., ARMv8, RISC-V's weak memory order), which provide fence instructions to explicitly prevent it when required by software [55].

6.4 Power-/Thermal-Aware Integration

Cloud computing and AI growth have significantly increased data center energy consumption and power densities. While typical CPU-based servers require 20 kW [25], current AI racks draw up to 120 kW [19], and next-generation AI racks (late-2027) will draw 600 kW [39]. Operators cannot run full AI racks in existing installations due to power delivery constraints, and are increasingly turning to on-site generation [48], even nuclear power [54], while developing power delivery for over 1 MW per rack [23]. As such, reducing power consumption is becoming a first-order objective for new systems.

While compute typically receives the most attention, memory also comprises a large fraction of overall power draw in modern systems, as Figure 6 shows. Memory contributes significantly to overall system power consumption through static leakage, refresh operations, and data movement costs [11] of both on-chip and off-chip memory arrays.

Memory specialization offers substantial power optimization opportunities through better matching of cell characteristics to workloads. Matching data lifetimes to memory cell retention characteristics can substantially reduce static power by minimizing unnecessary SRAM leakage and DRAM refresh operations. Data movement energy, scaling superlinearly with interconnect distance, typically dominates total memory power. Memory specialization must therefore consider interconnect and packaging energy consumption. Co-optimization of memory cells, interconnects, packaging, and data assignment is essential.

Increasing power densities and tight integration of dense memory arrays require careful attention to cooling to ensure effective and reliable operation of the hardware. This may involve inventing and integrating new technologies (e.g., microfluidics [57]) to achieve effective cooling.

7 Future Memories and Integration

This paper focuses on memory technologies available today with known integration options and performance trade-offs. Beyond current technologies, emerging memory innovations may enable alternative heterogeneous implementations, though their commercial viabilities particularly in terms of cost remain uncertain.

One emerging possibility to increase DRAM density is monolithic 3D DRAM [60], with multilayer DRAM cells fabricated on a single silicon wafer, contrasting with HBM's post-fabrication die stacking. 3D DRAM requires at least 64 layers to match 2D density but could provide considerable benefits from stacking additional layers despite manufacturing complexity and costs.

More significantly, LtRAM and StRAM offer superior integration flexibility compared to conventional memories. Unlike DRAM and NAND requiring off-chip placement due to process incompatibilities, StRAM and LtRAM support the full integration spectrum from on-chip to 2.5D chiplets to 3D stacking [14]. This flexibility enables co-optimization of memory cells, interconnects, and data placement—critical for realizing memory specialization benefits.

8 Conclusion

Memory scaling has stalled, making memory the primary bottleneck in cost, performance, and power for modern systems. We propose a paradigm shift from traditional hierarchical memory systems to specialized architectures organized around two classes: LtRAM for read-intensive, persistent data and StRAM for transient, frequently accessed data.

This specialization approach provides manageable abstractions that capture essential trade-offs of diverse emerging technologies without overwhelming software complexity. It enables workload-specific optimizations matching application patterns to device characteristics while leveraging the superior scaling potential of emerging technologies.

Realizing this vision requires coordinated research: fine-grained access pattern profiling, lifetime-aware data placement policies, memory system abstractions for heterogeneous devices, and coherence protocols for specialized memories. Success depends on cross-disciplinary collaboration spanning materials science, device physics, computer architecture, and systems design to address post-hierarchical memory system challenges.

Acknowledgments

We would like to thank Shridhar Mukund from EMD, who first suggested LtRAM and StRAM as a useful decomposition in the context of ML inference. We would also like to thank EMD, Google, and ASML for their support of this work.

References

[1] Robin Abrahamse, Ákos Hadnagy, and Zaid Al-Ars. 2022. Memory-Disaggregated In-Memory Object Store Framework for Big Data Applications. In *2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, Lyon, France, 01–07. doi:10.1109/IPDPSW55747.2022.00211

[2] Michael Allison, Arun George, Javier Gonzalez, Dan Helmick, Vikash Kumar, Roshan R. Nair, and Vivek Shah. 2025. Towards Efficient Flash Caches with Emerging NVMe Flexible Data Placement SSDs. In *Proceedings of the Twentieth European Conference on Computer Systems (EuroSys '25)*. Association for Computing Machinery, Rotterdam, Netherlands, 1142–1160. doi:10.1145/3689031.3696091

[3] Yue Bai, Huaqiang Wu, Riga Wu, Ye Zhang, Ning Deng, Zhiping Yu, and He Qian. 2014. Study of Multi-level Characteristics for 3D Vertical Resistive Switching Memory. *Scientific Reports* 4, 1 (July 2014), 5780. doi:10.1038/srep05780

[4] Asya Bergal. 2020. *Trends in DRAM price per gigabyte*. https://aiimpacts.org/trends-in-dram-price-per-gigabyte/

[5] Andrzej Białecki, Robert Muir, and Grant Ingersoll. 2012. Apache Lucene 4. In *Proceedings of the SIGIR 2012 Workshop on Open Source Information Retrieval*. Association for Computing Machinery, Portland, OR, USA, 17–24. http://opensearchlab.otago.ac.nz/paper_10.pdf

[6] Hans-J. Boehm and Brian Demsky. 2014. Outlawing ghosts: avoiding out-of-thin-air results. In *Proceedings of the workshop on Memory Systems Performance and Correctness (MSPC '14)*. Association for Computing Machinery, New York, NY, USA, 1–6. doi:10.1145/2618128.2618134

[7] Josiah L. Carlson. 2013. *Redis in Action*. Manning Publications Co., Shelter Island, NY, USA.

[8] Hsiang-Yun Cheng, Chun-Feng wu, Christian Hakert, Kuan-Hsun Chen, Yuan-Hao Chang, Jian-Jia Chen, Chia-Lin Yang, and Tei-Wei Kuo. 2021. Future Computing Platform Design: A Cross-Layer Design Approach. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 312–317. doi:10.23919/DATE51398.2021.9474229

[9] Jeongdong Choe. 2022. *DRAM Scaling Trend and Beyond*. https://www.techinsights.com/blog/dram-scaling-trend-and-beyond

[10] Rachel Cleaveland and Caroline Trippel. 2024. Memory Consistency Model-Aware Cache Coherence for Heterogeneous Hardware. In *2024 Formal Methods in Computer-Aided Design (FMCAD)*. 1–12. doi:10.34727/2024/isbn.978-3-85448-065-5_22

[11] William Dally. 2022. On the model of computation: point. *Commun. ACM* 65, 9 (Aug. 2022), 30–32. doi:10.1145/3548783

[12] James M. Daughton and Jack S. T. Huang. 1988. Magnetoresistive memory including thin film storage cells having tapered ends. https://patents.google.com/patent/US4731757A/en

[13] Miyuru Dayarathna, Yonggang Wen, and Rui Fan. 2016. Data Center Energy Consumption Modeling: A Survey. *IEEE Communications Surveys & Tutorials* 18, 1 (2016), 732–794. doi:10.1109/COMST.2015.2481183

[14] Samuel Dayo, Shuhan Liu, Peijing Li, Philip Levis, Subhasish Mitra, Thierry Tamb, and David Tennenhouseand H.-S. Philip Wong. 2025. The Future of Memory: Limits and Opportunities. In *Proceedings of the First International Workshop on Big Memory (BigMem)*.

[15] Jiawei Deng, Long Zheng, Mingsheng Luo, Guangchao Lyu, Can Zhao, and Xinping Zhang. 2024. Simulation Studies of Hygro-Thermal-Vapor Induced Stresses and Interface Delamination in High Bandwidth Memory (HBM) Package With Different Numbers of Stacked Chips During Reflow Process. In *2024 25th International Conference on Electronic Packaging Technology (ICEPT)*. IEEE, Tianjin, China, 1–6. doi:10.1109/ICEPT63120.2024.10668584

[16] Brad Fitzpatrick. 2025. *Memcached: A distributed memory object caching system*. https://github.com/memcached/memcached

[17] Mark Gebhart, Stephen W. Keckler, Brucek Khailany, Ronny Krashinsky, and William J. Dally. 2012. Unifying Primary Cache, Scratch, and Register File Memories in a Throughput Processor. In *2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE, Vancouver, BC, Canada, 96–106. doi:10.1109/MICRO.2012.18

[18] Robert Giterman, Amir Shalom, Andreas Burg, Alexander Fish, and Adam Teman. 2020. A 1-Mbit Fully Logic-Compatible 3T Gain-Cell Embedded DRAM in 16-nm FinFET. *IEEE Solid-State Circuits Letters* 3 (2020), 110–113. doi:10.1109/LSSC.2020.3006496

[19] Ivan Goldwasser, Martin Hsu, and Harry Petty. 2025. *Building the Modular Foundation for AI Factories with NVIDIA MGX*. https://developer.nvidia.com/blog/building-the-modular-foundation-for-ai-factories-with-nvidia-mgx/

[20] Waqas Gul, Maitham Shams, and Dhamin Al-Khalili. 2022. SRAM Cell Design Challenges in Modern Deep Sub-Micron Technologies: An Overview. *Micromachines* 13, 8 (Aug. 2022), 1332. doi:10.3390/mi13081332

[21] Zhiyuan Guo, Zijian He, and Yiying Zhang. 2023. Mira: A Program-Behavior-Guided Far Memory System. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP '23)*. Association for Computing Machinery, New York, NY, USA, 692–708. doi:10.1145/3600006.3613157

[22] J.W. Han, S.H. Park, M.Y. Jeong, K.S. Lee, K.N. Kim, H.J. Kim, J.C. Shin, S.M. Park, S.H. Shin, S.W. Park, K.S. Lee, J.H. Lee, S.H. Kim, B.C Kim, M.H. Jung, I.Y. Yoon, H. Kim, S.U. Jang, K.J. Park, Y.K. Kim, I.G. Kim, J.H Oh, S.Y. Han, B.S. Kim, B.J. Kuh, and J.M. Park. 2023. Ongoing Evolution of DRAM Scaling via Third Dimension -Vertically Stacked DRAM -. In *2023 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*. IEEE, Kyoto, Japan, 1–2. doi:10.23919/VLSITechnologyandCir57934.2023.10185290

[23] Madhusudan Iyengar and Amber Huffman. 2025. *AI infrastructure is hot. New power distribution and liquid cooling infrastructure can help*. https://cloud.google.com/blog/topics/systems/enabling-1-mw-it-racks-and-liquid-cooling-at-ocp-emea-summit

[24] Zizhen Jiang, Shengjun Qin, Haitong Li, Shosuke Fujii, Dongjin Lee, Simon Wong, and H.-S. Philip Wong. 2019. Next-Generation Ultrahigh-Density 3-D Vertical Resistive Switching Memory (VRSM)—Part II: Design Guidelines for Device, Array, and Architecture. *IEEE Transactions on Electron Devices* 66, 12 (Dec. 2019), 5147–5154. doi:10.1109/TED.2019.2950595

[25] Rani Kahn, Saurabh Borkar, and Zaid Dighe. 2024. *Accelerating industry-wide innovations in datacenter infrastructure and security*. https://azure.microsoft.com/en-us/blog/accelerating-industry-wide-innovations-in-datacenter-infrastructure-and-security/

[26] Taesoo Kim, Jiwon Yoon, Seonguk Choi, Haeyeon Kim, Haeseok Suh, Hyunjun An, Jungmin Ahn, Hyunah Park, and Joungho Kim. 2024. Design and Analysis of Twin Tower High Bandwidth Memory (HBM) Architecture for Large Memory Capacity and High Bandwidth System. In *2024 IEEE Electrical Design of Advanced Packaging and Systems (EDAPS)*. IEEE, Bengaluru, Karnataka, India, 1–3. doi:10.1109/EDAPS64431.2024.10988462

[27] Oleksii Kononenko, Olga Baysal, Reid Holmes, and Michael W. Godfrey. 2014. Mining modern repositories with elasticsearch. In *Proceedings of the 11th Working Conference on Mining Software Repositories (MSR 2014)*. Association for Computing Machinery, Hyderabad, Telangana, India, 328–331. doi:10.1145/2597073.2597091

[28] Andres Lagar-Cavilla, Junwhan Ahn, Suleiman Souhlal, Neha Agarwal, Radoslaw Burny, Shakeel Butt, Jichuan Chang, Ashwin Chaugule, Nan Deng, Junaid Shahid, Greg Thelen, Kamil Adam Yurtsever, Yu Zhao, and Parthasarathy Ranganathan. 2019. Software-Defined Far Memory in Warehouse-Scale Computers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems* (Providence, RI, USA) *(ASPLOS '19)*. Association for Computing Machinery, New York, NY, USA, 317–330. doi:10.1145/3297858.3304053

[29] Taehyung Lee, Sumit Kumar Monga, Changwoo Min, and Young Ik Eom. 2023. MEMTIS: Efficient Memory Tiering with Dynamic Page

Classification and Page Size Determination. In *Proceedings of the 29th Symposium on Operating Systems Principles* (Koblenz, Germany) *(SOSP '23)*. Association for Computing Machinery, New York, NY, USA, 17–34. doi:10.1145/3600006.3613167

[30] Sergey Legtchenko, Ioan Stefanovici, Richard Black, Antony Rowstron, Junyi Liu, Paolo Costa, Burcu Canakci, Dushyanth Narayanan, and Xingbo Wu. 2025. Storage Class Memory is Dead, All Hail Managed-Retention Memory: Rethinking Memory for the AI Era. In *Proceedings of the 2025 Workshop on Hot Topics in Operating Systems (HotOS '25)*. Association for Computing Machinery, Banff, AB, Canada, 111–118. doi:10.1145/3713082.3730381

[31] Peijing Li, Matthew Hung, Yiming Tan, Konstantin Hoßfeld, Jake Jiajun Cheng, Shuhan Liu, Lixian Yan, Xinxin Wang, H.-S. Philip Wong, and Thierry Tambe. 2025. GainSight: Application-Guided Profiling for Composing Heterogeneous On-Chip Memories in AI Hardware Accelerators. doi:10.48550/arXiv.2504.14866 arXiv:2504.14866

[32] Yuxuan Li, Bei Pan, Zhenting Ge, Pengpeng Chen, Bo Bi, Xin Yi, Chaochao Wu, and Ce Wang. 2025. Soldering and Bonding in Contemporary Electronic Device Packaging. *Materials* 18, 9 (April 2025), 2015. doi:10.3390/ma18092015

[33] Jinshu Liu, Hamid Hadian, Hanchen Xu, and Huaicheng Li. 2025. Tiered Memory Management Beyond Hotness. In *19th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2025, Boston, MA, USA, July 7-9, 2025*, Lidong Zhou and Yuanyuan Zhou (Eds.). USENIX Association, 731–747. https://www.usenix.org/conference/osdi25/presentation/liu

[34] Jinshu Liu, Hamid Hadian, Hanchen Xu, and Huaicheng Li. 2025. Tiered Memory Management Beyond Hotness. In *Proceedings of the 19th USENIX Symposium on Operating Systems Design and Implementation (OSDI 25)*. USENIX Association, Boston, MA, USA, 731–747. https://www.usenix.org/conference/osdi25/presentation/liu

[35] Shuhan Liu, Koustav Jana, Kasidit Toprasertpong, Jian Chen, Zheng Liang, Qi Jiang, Sumaiya Wahid, Shengjun Qin, Wei-Chen Chen, Eric Pop, and H.-S. Philip Wong. 2024. Design Guidelines for Oxide Semiconductor Gain Cell Memory on a Logic Platform. *IEEE Transactions on Electron Devices* 71, 5 (May 2024), 3329–3335. doi:10.1109/TED.2024.3372938

[36] Brett Lowe. 2024. *Developing ReRAM As Next Generation On-Chip Memory For Machine Learning, Image Processing And Other Advanced CPU Applications.* https://newsroom.lamresearch.com/reram-on-chip-memory

[37] Anni Lu, Junmo Lee, Tae-Hyeon Kim, Muhammed Ahosan Ul Karim, Rebecca Sejung Park, Harsono Simka, and Shimeng Yu. 2024. High-speed emerging memories for AI hardware accelerators. *Nature Reviews Electrical Engineering* 1, 1 (Jan. 2024), 24–34. doi:10.1038/s44287-023-00002-9

[38] Hongyu Miao, Myeongjae Jeon, Gennady Pekhimenko, Kathryn S. McKinley, and Felix Xiaozhu Lin. 2019. StreamBox-HBM: Stream Analytics on High Bandwidth Hybrid Memory. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '19)*. Association for Computing Machinery, Providence, RI, USA, 167–181. doi:10.1145/3297858.3304031

[39] Sebastian Moss. 2025. *Nvidia's Rubin Ultra NVL576 rack expected to be 600kW, coming second half of 2027.* https://www.datacenterdynamics.com/en/news/nvidias-rubin-ultra-nvl576-rack-expected-to-be-600kw-coming-second-half-of-2027/

[40] O. Mutlu. 2016. Rethinking Memory System Design. In *2016 Mobile System Technologies Workshop (MST)*. 1–3. doi:10.1109/MST.2016.12

[41] Vijay Nagarajan, Daniel J. Sorin, Mark D. Hill, David A. Wood, and Natalie Enright Jerger. 2020. *A Primer on Memory Consistency and Cache Coherence* (2nd ed.). Morgan & Claypool Publishers.

[42] Dimin Niu, Shuangchen Li, Yuhao Wang, Wei Han, Zhe Zhang, Yijin Guan, Tianchan Guan, Fei Sun, Fei Xue, Lide Duan, Yuanwei Fang, Hongzhong Zheng, Xiping Jiang, Song Wang, Fengguo Zuo, Yubing Wang, Bing Yu, Qiwei Ren, and Yuan Xie. 2022. 184QPS/W 64Mb/mm23D Logic-to-DRAM Hybrid Bonding with Process-Near-Memory Engine for Recommendation System. In *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, Vol. 65. IEEE, San Francisco, CA, USA, 1–3. doi:10.1109/ISSCC42614.2022.9731694

[43] Nicolai Oswald, Vijay Nagarajan, Daniel J. Sorin, Vasilis Gavrielatos, Theo Olausson, and Reece Carr. 2022. HeteroGen: Automatic Synthesis of Heterogeneous Cache Coherence Protocols. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. 756–771. doi:10.1109/HPCA53966.2022.00061

[44] Peizhao Ou and Brian Demsky. 2018. Towards understanding the costs of avoiding out-of-thin-air results. *Proc. ACM Program. Lang.* 2, OOPSLA (Oct. 2018), 136:1–136:29. doi:10.1145/3276506

[45] Neel Patel, Amin Mamandipoor, Derrick Quinn, and Mohammad Alian. 2023. XFM: Accelerated Software-Defined Far Memory. In *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO '23)*. Association for Computing Machinery, Toronto, ON, Canada, 769–783. doi:10.1145/3613424.3623776

[46] Vinicius Petrucci, Eishan Mirakhur, Nikesh Agarwal, Su Wei Lim, Vishal Tanna, Rita Gupta, and Mahesh Wagh. 2023. *CXL Memory Expansion: A Closer Look on Actual Platform.* White Paper. Micron Technology, Boise, ID, USA. https://assets.micron.com/adobe/assets/urn:aaid:aem:17ee2655-7d57-4aed-b7b2-70bc9715db8d/original/as/cxl-memory-expansion-a-close-look-on-actual-platform.pdf

[47] Amanda Raybuck, Tim Stamler, Wei Zhang, Mattan Erez, and Simon Peter. 2021. HeMem: Scalable Tiered Memory Management for Big Data Applications and Real NVM. In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles* (Virtual Event, Germany) *(SOSP '21)*. Association for Computing Machinery, New York, NY, USA, 392–407. doi:10.1145/3477132.3483550

[48] Drew Robb. 2025. *Data Centers Bypassing the Grid to Obtain the Power They Need.* https://www.datacenterknowledge.com/energy-power-supply/data-centers-bypassing-the-grid-to-obtain-the-power-they-need

[49] Shigeru Shiratake. 2020. Scaling and Performance Challenges of Future DRAM. In *2020 IEEE International Memory Workshop (IMW)*. IEEE, Dresden, Germany, 1–3. doi:10.1109/IMW48823.2020.9108122

[50] Matthew D. Sinclair, Johnathan Alsop, and Sarita V. Adve. 2017. Chasing Away RAts: Semantics and Evaluation for Relaxed Atomics on Heterogeneous Systems. In *Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA '17)*. Association for Computing Machinery, New York, NY, USA, 161–174. doi:10.1145/3079856.3080206

[51] Alan Smith, Gabriel H. Loh, Michael J. Schulte, Mike Ignatowski, Samuel Naffziger, Mike Mantor, Mark Fowler Nathan Kalyanasundharam, Vamsi Alla, Nicholas Malaya, Joseph L. Greathouse, Eric Chapman, and Raja Swaminathan. 2024. Realizing the AMD Exascale Heterogeneous Processor Vision : Industry Product. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA '24)*. IEEE, Buenos Aires, Argentina, 876–889. doi:10.1109/ISCA59077.2024.00068

[52] Lukas Steiner, Matthias Jung, Michael Huonker, and Norbert Wehn. 2023. Unveiling the Real Performance of LPDDR5 Memories. In *Proceedings of the 2022 International Symposium on Memory Systems (MEMSYS '22)*. Association for Computing Machinery, Washington, DC, USA, 1–3. doi:10.1145/3565053.3565062

[53] Zainab Swaidan, Rouwaida Kanj, Johnny El Hajj, Edward Saad, and Fadi Kurdahi. 2019. RRAM Endurance and Retention: Challenges, Opportunities and Implications on Reliable Design. In *2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. IEEE, Genova, Italy, 402–405. doi:10.1109/ICECS46596.2019.8964707

[54] Michael Terrell. 2024. *New nuclear clean energy agreement with Kairos Power.* https://blog.google/outreach-initiatives/sustainability/google-

kairos-power-nuclear-energy-agreement/

[55] Caroline Trippel. 2019. *Concurrency and Security Verification in Heterogeneous Parallel Systems.* Academic dissertations (Ph.D.). Princeton University, Princeton, NJ, USA. https://dataspace.princeton.edu/handle/88435/dsp01gt54kq90n

[56] Po-An Tsai, Yee Ling Gan, and Daniel Sanchez. 2018. Rethinking the Memory Hierarchy for Modern Languages. In *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO).* 203–216. doi:10.1109/MICRO.2018.00025

[57] Remco Van Erp, Reza Soleimanzadeh, Luca Nela, Georgios Kampitsis, and Elison Matioli. 2020. Co-designing electronics with microfluidics for more sustainable cooling. *Nature* 585, 7824 (2020), 211–216.

[58] Nandita Vijaykumar, Abhilasha Jain, Diptesh Majumdar, Kevin Hsieh, Gennady Pekhimenko, Eiman Ebrahimi, Nastaran Hajinazar, Phillip B. Gibbons, and Onur Mutlu. 2018. A Case for Richer Cross-Layer Abstractions: Bridging the Semantic Gap with Expressive Memory. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA '18).* 207–220. doi:10.1109/ISCA.2018.00027

[59] Midhul Vuppalapati and Rachit Agarwal. 2024. Tiered Memory Management: Access Latency is the Key!. In *Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles* (Austin, TX, USA) *(SOSP '24).* Association for Computing Machinery, New York, NY, USA, 79–94. doi:10.1145/3694715.3695968

[60] Xiangjin Wu, Luke R. Upton, Jian Chen, Po-Kai Hsu, Shimeng Yu, and H.-S. Philip Wong. 2025. Signal Margin, Density, and Scalability of 3-D DRAM: A Comparative Study of Two Bitline Architectures. *IEEE Transactions on Electron Devices* 72, 2 (2025), 671–677.

[61] Cong Xu, Dimin Niu, Naveen Muralimanohar, Rajeev Balasubramonian, Tao Zhang, Shimeng Yu, and Yuan Xie. 2015. Overcoming the challenges of crossbar resistive memory architectures. In *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA).* IEEE, Burlingame, CA, USA, 476–488. doi:10.1109/HPCA.2015.7056056

[62] Xiang Yang and I-Wei Chen. 2012. Dynamic-Load-Enabled Ultra-low Power Multiple-State RRAM Devices. *Scientific reports* 2 (10 2012), 744. doi:10.1038/srep00744

[63] Shimeng Yu and Tae-Hyeon Kim. 2024. Semiconductor Memory Technologies: State-of-the-Art and Future Trends. *Computer* 57, 4 (April 2024), 150–154. doi:10.1109/MC.2024.3363269

[64] Sai Qian Zhang, Thierry Tambe, Nestor Cuevas, Gu-Yeon Wei, and David Brooks. 2024. CAMEL: Co-Designing AI Models and eDRAMs for Efficient On-Device Learning. In *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA).* IEEE, Edinburgh, Scotland, UK, 861–875. doi:10.1109/HPCA57654.2024.00071

[65] Xin Zhao, Zhicheng Hu, Zilong Guo, Haodong Fan, Xi Yang, Jing Zhou, and Liang Chang. 2024. A RRAM-based High Energy-efficient Accelerator Supporting Multimodal Tasks for Virtual Reality Wearable Devices. In *Proceedings of the 61st ACM/IEEE Design Automation Conference* (San Francisco, CA, USA) *(DAC '24).* Association for Computing Machinery, San Francisco, CA, USA, Article 37, 6 pages. doi:10.1145/3649329.3658474