Storage Class Memory is Dead, All Hail Managed-Retention Memory: Rethinking Memory for the AI Era

Sergey Legtchenko, Ioan Stefanovici, Richard Black, Antony Rowstron, Junyi Liu, Paolo Costa, Burcu Canakci, Dushyanth Narayanan, Xingbo Wu
Microsoft Research

ABSTRACT

AI clusters today are one of the major uses of High Bandwidth Memory (HBM). However, HBM is suboptimal for AI workloads for several reasons. Analysis shows HBM is overprovisioned on *write* performance, but underprovisioned on density and read bandwidth, and also has significant energy per bit overheads. It is also expensive, with lower yield than DRAM due to manufacturing complexity. We propose a new memory class: Managed-Retention Memory (MRM), which is more optimized to store key data structures for AI inference workloads. We believe that MRM may finally provide a path to viability for technologies that were originally proposed to support Storage Class Memory (SCM). These technologies traditionally offered long-term persistence (10+ years) but provided poor IO performance and/or endurance. MRM makes different trade-offs, and by understanding the workload IO patterns, MRM foregoes long-term data retention and write performance for better potential performance on the metrics important for these workloads.

KEYWORDS

Memory, Managed-Retention Memory, AI Inference, AI Infrastructure

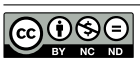
1 INTRODUCTION

To date the world has been very binary when it comes to storage: there are non-volatile and volatile storage technologies. DRAM in different forms (GDDR, HBM, LPDDR) is the dominant *volatile* memory storage technology. Data it stores is lost as soon as the energy source is removed. NAND block-oriented and NOR byte-addressable Flash are the most widely used examples of *non-volatile* memory storage. They do not need to be constantly powered to persist data. At the memory cell level, data volatility is expressed as *retention time*, which is the time that data is reliably stored without requiring a refresh. Flash cells have a retention time of 10+ years, but this comes at the cost of lower read and write throughput per memory cell than DRAM. These properties mean that DRAM is used as memory for processors, and Flash is used for secondary storage.

Several other memory technologies, like RRAM, MRAM [30, 47] and PCM [24], all have the potential to offer non-volatility. They fall into a class of memory that is often referred to as Storage Class Memory (SCM) for servers. The recently discontinued Intel Optane / 3D XPoint [16] is an iconic representative of SCM, which aimed to overcome the IO limitations of Flash while being non-volatile. The dream was to replace DRAM by offering comparable IO performance and byte addressability, while also featuring 10+ year retention. However, all attempts to date have failed to displace DRAM due to the trade-offs. They failed to offer IO performance that is comparable to DRAM at lower (or same) costs as Flash due to the challenges of density and complex manufacturing processes. For main memory, persistence of data is not as important as IO performance. For general compute workloads, nobody wants to trade primary memory IO-performance for 10+ year data retention. These technologies also struggle with *endurance*, which refers to the number of write cycles a memory cell can support before it permanently degrades [24]. Hence, SCM ended up being valuable for some use cases (e.g., embedded compute [1, 2]), but not for deployment in servers.

Ironically, we believe that the rise of Flash may have been something of a curse for memory innovation. Non-volatility is a key storage *device* property, but *at a memory cell* level it is quite misleading. For all technologies, memory cells offer simply a retention time, which is a continuum from microseconds for DRAM to many years. The technologies that underpin SCM have been forced to be non-volatile, requiring their retention time to be a decade or more. Unfortunately, achieving these high retention times requires trading off other metrics such as write and read latency, energy efficiency and endurance [13, 19, 34].

Perhaps one reason why this has been viewed historically as binary, is that even with relaxed retention times, SCM



HOTOS '25, May 14–16, 2025, Banff, AB, Canada

ACM ISBN 979-8-4007-1475-7/2025/05
https://doi.org/10.1145/3713082.3730381

technologies would not match DRAM on all metrics of importance *for general workloads*. However, *foundation models* (of which Large Language Models, or LLMs are a subset) have recently emerged as a new major workload with unique memory IO requirements [38]. The tremendous scale and growth of foundation model training and inference require novel hardware approaches. Foundation model inference has different memory IO requirements to historical workloads. For example, a large fraction of the memory is used to store model weights, for which IO performance is critical for sequential *reads*, but much less important for *writes*. Memory IO is sequential and predictable, and given the energy challenges of AI clusters, energy per bit read is also an issue. The only technology today that can match the IO performance, energy and density is HBM. However, it is no panacea, and certain key stages in foundation model inference are memory not compute bound. Further, HBM is expensive and has significant yield challenges.

We think that there is an opportunity to rethink existing "non-volatile" memory technologies for this new workload. We propose a new class of memory that we call *Managed-Retention Memory* (MRM). MRM is different from volatile DRAM as it can retain data without power and does not waste energy in frequent cell refreshes, but unlike SCM, is not aimed at long-term retention times. As most of the inference data does not need to be persisted, retention can be relaxed to days or hours. In return, MRM has better endurance and aims to outperform DRAM (and HBM) on the key metrics such as read throughput, energy efficiency and capacity.

In the rest of this paper, Section 2 first characterizes the foundation model workload characteristics and requirements. It then discusses the challenges and lack of optimality of HBM. Section 3 describes relevant emerging technologies. Finally Section 4 explores the broader systems implications of rethinking memory and introducing MRM. We are explicitly not settling on a specific technology, instead highlighting an opportunity space. This is a call for action for those working on low-level memory cell technologies, through those thinking of memory controllers, to those designing the software systems that access the memory. Hail to a cross-layer collaboration for better memory in the AI era!

2 MEMORY IN THE FOUNDATION MODEL-ERA

The workload of a foundation model is quite different to traditional workloads. A foundation model is first *trained*, usually on a large cluster (e.g., 50,000+ AI accelerators), and the output is essentially a set of model weights. These weights are then deployed in production where they serve *inference* queries. Thousands or even millions of instances of the foundation models will be used but the scale of hardware per inference is much smaller (e.g., 4+ AI accelerators). It has been observed that both training and inference workloads are memory intensive [3, 57]. Training scale depends on the model size and is a one-off effort (often taking months), while the inference workload is demand-driven and served for a significant time period until the model weights are retired.

Training and inference have distinct memory access patterns and requirements, and are typically deployed on different clusters. As demand increases, we are expecting the inference infrastructure to dominate, and are hence focusing on the inference workload. More specifically, we consider foundation models that perform autoregressive token generation, i.e., new tokens are generated based on the sequence of preceding tokens. An inference query is a sequence of input tokens, in response to which the foundation model generates a sequence of output tokens. A *context* is composed of all the tokens from the user and the corresponding responses generated by the model during the interaction. Having contexts as large as possible is desirable, as it improves the model's reasonning ability via its use of the self-attention mechanism [52]. However, in deployment, contexts have limited size and range from low 1000s to a few 10,000s tokens (depending on the model), and are primarily limited by the amount of memory available. Each inference query is computationally expensive and requires distributed computation across multiple AI accelerators.

Inference relies on three main in-memory data structures: *model weights*, the *KV cache*, and *model activations*. Of these, model weights and the KV cache use up the majority of the memory capacity [22].

The model weights (a matrix) have been key to expanding the capabilities of frontier foundation models; there has been an exponential growth in the size of the model weights with each generation of foundation model. Currently, large models have (well) over 500 billion weights, representing between 250 GB and over 1 TB of data depending on the weight quantization used. The weights are effectively a non-mutable data structure. The reference model weights are persisted in storage, while a replica is distributed across the AI accelerators in every inference cluster. There are a large number of foundation models today, but in practice a small number of the most popular ones are used at scale. All inference queries made to a given foundation model version (e.g., GPT4) use a copy of the same weights.

The KV cache supports the model's self-attention mechanism. It is a sequence of self-attention vectors that encode the model's understanding of the relationship between all the tokens in a context. Every time a new token is generated in a context, a vector is appended to the end of the corresponding KV cache. Each vector is typically a few MBs, so the KV cache usually grows to a few tens of GBs until the context size limit is reached.

Lastly, model activations are the transient tensors that are created and passed between the different layers during a forward pass of the network. They are typically an order of magnitude smaller than both the weights and the KV cache, and are only stored during the forward pass computation.

The KV cache is created during the *prefill* phase, when the first set of tokens is received from a user. Subsequently, in the *decode* phase the model iteratively generates response tokens. For that, at each iteration the KV cache is read entirely and sequentially, a new token is generated, and the corresponding self-attention vector is appended to the KV cache. KV caches leverage memory to reduce computation and are soft state: they are generated by the model, and can be re-computed if needed. However, the token rate per second is usually quite low (thus expensive) so caching and using the KV cache is usually preferable to recalculation.

During inference, the entire self-attention data and weights are read for every generated token, creating substantial bandwidth demand between memory and compute. At any given time, many inference requests are multiplexed over the same cluster, but all of them are for the same model. Each AI accelerator's memory thus contains a subset of the model weights, as well as several KV caches and activations that correspond to the working set of contexts. When a new model is deployed, the cluster stops accepting new requests, services ongoing ones, then loads weights for the new model.

To summarize, foundation model inference is mostly composed of very large, predictable memory reads, while writes are smaller and mostly append only. Exact memory ranges to be read are known in advance, and large fractions of the memory are not overwritten for long periods of time. Yet, despite being read-dominated, inference still requires write rates that are very high compared to storage workloads.

2.1 The Curse of HBM

Today the majority of data used in an AI accelerator is stored on HBM, because all the data structures need to be repeatedly read at high bandwidth. Current AI accelerators can support very high main memory bandwidths, e.g., 8 TB/s for a single B200 GPU [51]. In addition, since weights and KV caches are large, AI accelerators require substantial HBM capacity. Hard engineering challenges need to be surmounted to achieve this, especially around energy usage. Signal loss over copper interconnect tracking at the required data rates means that the memory must be physically located (very) close to the compute die, typically co-packaged on the same interposer. The very wide interfaces, and high signal rates translate into more energy, and approximately a third of the energy usage for an AI accelerator is the memory. HBM is used as it enables 3D-stacking of DRAM on the same package to boost on-package memory capacity, throughput, and minimize the distance of the memory cells from the AI accelerator. Current HBM products have 8-12 layers, for an aggregate 192 GB on a B200 package [51]. Hence, HBM is used as it offers the highest throughput at the highest density with reasonable energy usage. However, even using HBM, a substantial part of every inference query is memory bound [37].

Unfortunately, there is currently no viable alternative to HBM. Non-stacked DRAM does not have the required density, while NAND and NOR Flash memory are not fast enough and have low lifetime endurance especially at higher densities where multiple bits are stored per memory cell. Both lack the energy efficiency required in package.

It should be noted that HBM comes with several fundamental challenges. First, memory vendors are struggling to continue to scale the density. The per-layer scaling is struggling with challenges inherited from DRAM [40]. So, the next generation of HBM (HBM4) is only expected to increase capacity per layer by 30% compared to current HBM3e. Secondly, the 3D-stacking of DRAM both significantly reduces the yield of the manufacturing process and also leads to heat dissipation challenges, especially when tightly packaged with an AI accelerator die. Currently, the industry does not expect it to scale beyond 16 layers in the foreseeable future [50] as 3D-stacking is extremely complex. Finally, the power density of the infrastructure is very high and continues to grow, increasing the need for every Watt to be spent on useful work. Due to cell-level capacitor leakage, HBM fundamentally requires frequent refreshing (~ every tens to hundreds of miliseconds), consuming power even when the memory is idle.

These factors, combined with high demand, fueled by exponential growth of cloud infrastructure for foundation models, means that HBM accounts for a substantial fraction of an AI cluster's cost. This is unlikely to change in the foreseeable future, and AI clusters will remain dependent on HBM.

2.2 A New Hope?

Foundation model inference is very different from the general-purpose main memory workload for which DRAM was designed. First, it is extremely **read-intensive**. For example, *each token* generated during decode requires reading *all* the weights, and the entire KV cache [37], for one self-attention vector write. Self-attention vector size is usually at most a few MBs [4, 44], while weights and KV caches are typically 10s of GBs, which imply read:write ratios of over 1000:1.

There are efforts to reduce the amount of data read during inference. For example, batching allows weight reuse across requests [3]. However, batching is limited by latency requirements [3]. Reuse of the KV cache across requests [54] and KV cache compression [27] are also used, but each has

its limitations and even together they do not fundamentally change the heavily read-dominated nature of the workload.

Second, memory accesses are **sequential** and **predictable**. There are no in-place updates for weights or KV caches, and the same weights and KV cache are read iteratively for every foundation model response. Memory virtualization mechanisms have been proposed to address memory fragmentation [22], but even in that case, pages are read in the same order. Each page is typically over 10 vectors (typically several MBs to 10s of MBs) and is read sequentially [22]. Furthermore, the mapping between virtual pages and physical addresses is typically static.

These properties suggest that most of the HBM capacity is used for data that has little use for the general-purpose properties HBM inherits from DRAM (random access, byte-addressability, comparable read and write performance). HBM is, in a sense, overprovisioned for the requirements of this foundation model inference workload. This overprovisioning leads to suboptimal cost and energy overheads.

It also raises the tantalizing question: if we correctly provision the memory to the workload, can we address this suboptimal cost and energy challenges for memory in inference clusters?

3 THE MEMORY OPPORTUNITY

We posit that the combination of (i) the importance and scale of foundation model infrastructure, (ii) the large difference between the workload patterns of conventional server CPUs and that of AI accelerators, and (iii) the poor match of HBM to the workload, opens a field of computer architecture research in better memory for this application.

We now motivate that this opportunity is best addressed by a new type of memory, as opposed to DRAM, HBM or Flash. Flash cannot be used because it does not have enough endurance, even with Single Level Cells (SLC) [7], and cannot satisfy the high throughput and energy efficiency requirements [14, 36]. The non-volatility of Flash is also unnecessary: the data is either persisted elsewhere (weights) or is soft state (KV caches, activations).

On the other hand, some workload properties are close to ones typically exhibited by storage workloads. For example, byte addressability is not required, because IO is large and sequential. Similar to storage infrastructure, storage capacity and total cost of ownership (TCO)/TB are key metrics, on which HBM is underperforming. Combining HBM and lower-cost, lower-throughput LPDDR for cooler data would reduce the overall hardware cost but also reduce the bandwidth at which the data is available to the GPU, and fundamentally not improve the HBM's read energy efficiency.

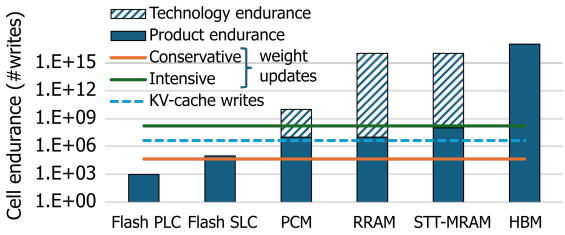


Figure 1: Endurance requirements for KV cache and model weights vs. endurance of memory technologies.

Finally, as power efficiency is perhaps the most important metric, housekeeping operations internal to the memory device need to be minimized. Many housekeeping overheads in existing technologies result from a mismatch between cell retention and data lifetime. DRAM's retention is too short, requiring frequent refreshes. Flash retention is too long, which is achieved at the expense of endurance, requiring FTL mechanisms (wear levelling, garbage collection). In both cases, housekeeping leverages the write path, and is typically energy-intensive. In contrast, matching retention to the lifetime of the data makes refresh, deletion, or wear-leveling unnecessary. In effect, instead of a data persistence management mechanism, retention becomes a cornerstone of device power management.

Can MRM match AI cluster requirements? PCM, RRAM, and STT-MRAM have *read* performance and energy on par or better than DRAM or even SRAM [28]. They also have potential for higher density and/or lower TCO/TB [17]. STT-MRAM and RRAM cells have already demonstrated potential for multi-level encoding [10], high endurance [25], and can be organized into high-density, transistor-less crossbar layouts [56]. They are also typically easier to stack on the same die, because resistive cells do not use tall capacitors [40]. Reducing retention allows lower voltage writes, unlocking advanced scaling processes, at 7 nm or beyond [58]. These technologies thus demonstrate a plausible roadmap towards lower read energy, higher read throughput and capacity than DRAM. Further, they are already deployed in real products. PCM was shipped at scale in Intel Optane devices, while RRAM and STT-MRAM have matured over the past few years, and are used for automotive, wearable and IoT applications [1, 2, 6].

These technologies have lower endurance than DRAM, and we now estimate the approximate endurance requirements for weight and KV cache writes. Weight updates are infrequent, bulk overwrites when the model is replaced. The update frequency is currently typically low (hours+), but could evolve as models diversify. We estimate the endurance required over 5 years for a conservative *hourly* update and an intensive *once per second* update. KV cache writes occur

both during prefill and decode, one self-attention vector per context token. Prefill is typically higher throughput than decode, and we use the throughputs and median context lengths reported for the Llama2-70B model in Splitwise [37]. For an expected lifetime of five years, we compute the number of KV cache writes, and infer the average number of writes per cell.

Figure 1 shows a comparison between endurance of existing memory/storage technologies and the workload endurance requirements. When applicable, we differentiate endurance observed in existing products from the potential demonstrated by the technology. We use technology endurance from [30, 47], while product endurance is taken from device specifications and benchmarks (Intel Optane PCM [5], Weebit RRAM [32] and Everspin STT-MRAM [39]). We observe that 1) HBM is vastly overprovisioned on endurance, and 2) existing SCM devices do not meet the endurance requirements but the underlying technologies have the potential to do so. We believe this is partly due to current devices being designed for non-volatility, which is achieved by trading off other important metrics such as write latency, energy efficiency or endurance [19, 34]. We see this as an opportunity to rethink existing memory technologies, currently used for SCM, specifically for AI workloads, by trading off non-volatility for other key metrics.

4 SOFTWARE STACK IMPLICATIONS

In this section, we motivate why MRM is of interest to the computer systems community. Foundation models are becoming pervasive which leads to a diversification of the requirements: some use cases have tight latency SLAs (e.g., user-in-the-loop conversation), some are throughput hungry and heavily use batching, others are background best-effort jobs (e.g., meeting recap). The workload is becoming more complex, with vastly different input:output token ratios, expert models tailored for specific use cases, and dependencies on advanced augmentation mechanisms (e.g. RAG [59]). In addition to that, the resource-heavy nature of the workload and the cost of the hardware require hollistic and efficient orchestration. This is addressed by leveraging key OS mechanisms (e.g., virtual memory [22], power-aware scheduling [46] or speculative execution [31]), effectively building up towards a rack-scale OS for foundation model inference. In that context, the emergence of MRM brings a set of exciting challenges and opportunities to explore.

Retention-aware data placement and scheduling. MRM is unlikely to be a one-size-fits-all solution, and will co-exist with other types of memory, such as HBM for write-heavy data structures (e.g., activations), and LPDDR as a slower tier. Fine-grained understanding of lifetime and access patterns of the data will be required to lay out the data. The scheduler will need to track the data expiration times, and decide whether to refresh it or move it to another tier based on the state of the requests that depend on that data.

Lightweight memory controllers. There is potential to make the MRM controller extremely simple and energy efficient. The lack of random access requirements opens up a unique prospect of a block-level access *memory* controller, with implications on the software stack. Much of the functionality that is typically handled on the device, such as refresh and wear-levelling can be left up to a software control plane higher up in the stack, which is best-placed to make these decisions while satisfying global application requirements. This approach is akin to zoned storage interfaces for Flash [60].

Dynamically Configurable Memory (DCM). Since the control plane has cluster-level visibility over both applications and user workloads, it is also best-placed to *dynamically* decide the retention period needed for each data when it is written, effectively right provisioning the MRM to the workload. This is a fully-flexible instantiation of MRM. At the hardware level, the memory controller would support writing at different durations and energies, allowing retention time to be programmed at runtime. The foundation model OS could then orchestrate optimal data refresh, wear-leveling, and garbage collection *at the cluster level*.

Retention-aware error correction. MRM's relaxed retention requirements also raise an interesting question: how do we think about data integrity? Much of the data stored in MRM will either be durably stored elsewhere (e.g., weights), or be soft state (e.g., KV cache). As such, the requirements for persistence are not as stringent as for traditional storage systems. Nonetheless, the system still needs to enforce integrity in order to guarantee correctness of computation involving the data, and avoid frequent re-computation of soft state. Leveraging existing state-of-the-art error correction techniques for memory[55] is a good start, however a large block-based MRM interface means that there is scope for considering error correction techniques that operate on larger code words and have less overhead [8]. Designing efficient error correction for MRM that meets the strigent latency and throughput requirements will be a fruitful area for open research.

5 RELATED WORK

The trade-offs between retention, endurance and write energy efficiency have been well studied both for STT-MRAM [18, 43, 48] and RRAM [15, 23, 34, 41]. Leveraging this mechanism has been proposed to improve the energy efficiency of hybrid on-die CPU caches [18, 41, 43, 48]. In contrast to our

work, this strand of work focuses on general-purpose multi-core CPUs, and is hence addressing a different optimization problem. AI clusters have rack-scale energy and cooling requirements, and have a more complex set of memory tiers and interconnects, but more predictable workloads.

Stanford has recently started a 5-year project to address the anticipated upcoming increase in tiering and heterogeneity of main memory [45]. We share the same observation that the memory wall [40] is a major challenge for key workloads, and is likely to lead to more memory heterogeneity due to lack of one-size-fits-all technology. While novel in data centers, this trend is common place in other applications. For example, the embedded world historically used ROM (Read Only Memory) [9, 33] which was a write once read many technology, EPROM (Erasable Programmable Read Only Memory) [29] write few read many which was used to store programs and could be erased using UV light, and of course RAM. ROM and EPROM offered non-volatile storage, and careful design choices had to be made to best leverage the upsides of the different technologies.

There is ongoing effort to overcome the memory wall by tighly integrating memory and compute. This is done by either adding more memory onto the compute die [26, 42], or with in-memory computing (IMC) [53]. Similar to our work, IMC is often aimed at AI workloads with either analog [11] or digital [20, 21] computation, and can be MRAM [12] or RRAM [12] based. Our work is orthogonal, because it aims to optimize the mainstream memory/compute model, instead of exploring a new paradigm.

Finally, there is substantial work on leveraging the heterogeneous memory access patterns in AI clusters. For example, it has been proposed to use CPU main memory for offloading idle KV caches [49**?**]. The latest Nvidia's GB200 superchip has an integrated LPDDR5 controller for a higher capacity, slower memory tier [35]. This suggests that memory heterogeneity is going to be common place in AI clusters. Our work proposes to leverage more aspects of data access heterogeneity to maximize tokens generated per dollar.

6 CONCLUSION

The emergence of AI workloads and their dependence on HBM memory has highlighted the limitations of HBM. AI inference workloads demand high read throughput, density, and energy efficiency, which HBM struggles to provide cost-effectively. We propose a new class of memory that can co-exist with HBM, Managed-Retention Memory (MRM), which enables the use of memory technologies originally proposed for SCM, but trades retention and other metrics like write throughput for improved performance metrics crucial for these AI workloads. By relaxing retention time requirements, MRM can potentially enable existing proposed SCM technologies to offer better read throughput, energy efficiency, and density. We hope this paper really opens new thinking about innovation in memory cell technologies and memory chip design, tailored specifically to the needs of AI inference clusters.

7 ACKNOWLEDGEMENTS

We would like to thank the HotOS reviewers for their feedback and comments. We would also like to thank Ed Nightingale, Alvin Lebeck, and Jacob Nelson for fruitful discussions about Managed-Retention Memory and AI infrastructure more broadly.

REFERENCES

[1] 2025. Next-generation memory for computers. https://www.intrinsicsemi.com/.
[2] 2025. The ReRAM Market Opportunity. https://www.weebit-nano.com/market/market-overview/.
[3] Amey Agrawal, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav S. Gulavani, and Ramachandran Ramjee. 2023. SARATHI: Efficient LLM Inference by Piggybacking Decodes with Chunked Prefills. arXiv:2308.16369 [cs.LG] https://arxiv.org/abs/2308.16369
[4] Artificial-Fintelligence 2023. Transformer inference tricks. https://www.artfintel.com/p/transformer-inference-tricks.
[5] blocksandfiles.com 2019. Is Optane DIMM endurance good enough? Quick answer…Yes, Intel has delivered. https://blocksandfiles.com/2019/04/04/enduring-optane-dimm-question-is-its-endurance-good-enough-yes-intel-has-delivered/.
[6] blocksandfiles.com 2022. CrossBar tries to secure embedded ReRAM IoT market. https://blocksandfiles.com/2022/04/21/no-sniffing-crossbar-tries-to-secure-its-embedded-reram-iot-market-niche/.
[7] Yuan-Hao Chang, Jen-Wei Hsieh, and Tei-Wei Kuo. 2007. Endurance enhancement of flash-memory storage systems: An efficient static wear leveling design. In *Proceedings of the 44th annual Design Automation Conference*. 212–217.
[8] S. Dolinar, Dariush Divsalar, and F. Pollara. 1998. Code Performance as a Function of Block Size. *Telecommunications and Mission Operations Progress Report* (01 1998).
[9] Marcello Duhalde, Alain Greiner, and Frederic Petrot. 1995. A high performance modular embedded ROM architecture. In *1995 IEEE International Symposium on Circuits and Systems (ISCAS)*, Vol. 2. IEEE, 1057–1060.
[10] Keming Fan, Wei-Chen Chen, Sumukh Pinge, H. S. Philip Wong, and Tajana Rosing. 2024. Efficient Open Modification Spectral Library Searching in High-Dimensional Space with Multi-Level-Cell Memory. arXiv:2405.02756 [cs.AR] https://arxiv.org/abs/2405.02756
[11] D. Fick. 2022. Analog Compute-in-Memory For AI Edge Inference. In *2022 International Electron Devices Meeting (IEDM)*. 21.8.1–21.8.4. https://doi.org/10.1109/IEDM45625.2022.10019367
[12] Yasmin Halawani, Baker Mohammad, and Hani Saleh. 2021. Design Exploration of ReRAM-Based Crossbar for AI Inference. *IEEE Access* 9 (2021), 70430–70442. https://doi.org/10.1109/ACCESS.2021.3076445
[13] Andrew Hay, Karin Strauss, Timothy Sherwood, Gabriel H. Loh, and Doug Burger. 2011. Preventing PCM banks from seizing too much power. In *2011 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 186–195.
[14] W. T. Huang, C. T. Chen, C. H. Chen, and C. C. Cheng. 2008. Energy-Efficient Buffer Architecture for Flash Memory. In *2008 International*

Conference on Multimedia and Ubiquitous Engineering (mue 2008). 543–546. https://doi.org/10.1109/MUE.2008.61

[15] D. Ielmini, F. Nardi, C. Cagli, and A. L. Lacaita. 2010. Trade-off between data retention and reset in NiO RRAMS. In *2010 IEEE International Reliability Physics Symposium*. 620–626. https://doi.org/10.1109/IRPS.2010.5488761

[16] Intel 2019. Intel Optane Memory - Responsive Memory, Accelerated Performance. https://www.intel.com/content/www/us/en/products/details/memory-storage/optane-memory.html.

[17] Engin Ipek, Jeremy Condit, Edmund B Nightingale, Doug Burger, and Thomas Moscibroda. 2010. Dynamically Replicated Memory: Building Reliable Systems from Nanoscale Resistive Memories. In *ASPLOS 2010: 15th International Conference on Architectural Support for Programming Languages and Operating Systems, Pittsburgh, PA*. ACM. https://www.microsoft.com/en-us/research/publication/dynamically-replicated-memory-building-resilient-systems-from-unreliable-nanoscale-memories/ ASPLOS Best Paper Award.

[18] Adwait Jog, Asit K. Mishra, Cong Xu, Yuan Xie, Vijaykrishnan Narayanan, Ravishankar Iyer, and Chita R. Das. 2012. Cache revive: Architecting volatile STT-RAM caches for enhanced performance in CMPs. In *DAC Design Automation Conference 2012*. 243–252. https://doi.org/10.1145/2228360.2228406

[19] Myoungsoo Jung, Youngbin Jin, and Mustafa Shihab. 2014. Area, Power and Latency Considerations of STT-MRAM to Substitute for Main Memory.

[20] Byeongho Kim, Sanghoon Cha, Sangsoo Park, Jieun Lee, Sukhan Lee, Shin-haeng Kang, Jinin So, Kyungsoo Kim, Jin Jung, Jong-Geon Lee, Sunjung Lee, Yoonah Paik, Hyeonsu Kim, Jin-Seong Kim, Won-Jo Lee, Yuhwan Ro, YeonGon Cho, Jin Hyun Kim, JoonHo Song, Jaehoon Yu, Seungwon Lee, Jeonghyeon Cho, and Kyomin Sohn. 2024. The Breakthrough Memory Solutions for Improved Performance on LLM Inference. *IEEE Micro* 44, 3 (2024), 40–48. https://doi.org/10.1109/MM.2024.3375352

[21] Jin Hyun Kim, Yuhwan Ro, Jinin So, Sukhan Lee, Shin-haeng Kang, YeonGon Cho, Hyeonsu Kim, Byeongho Kim, Kyungsoo Kim, Sangsoo Park, Jin-Seong Kim, Sanghoon Cha, Won-Jo Lee, Jin Jung, Jong-Geon Lee, Jieun Lee, JoonHo Song, Seungwon Lee, Jeonghyeon Cho, Jaehoon Yu, and Kyomin Sohn. 2023. Samsung PIM/PNM for Transformer Based AI : Energy Efficiency on PIM/PNM Cluster. In *2023 IEEE Hot Chips 35 Symposium (HCS)*. 1–31. https://doi.org/10.1109/HCS59251.2023.10254711

[22] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles* (Koblenz, Germany) *(SOSP '23)*. Association for Computing Machinery, New York, NY, USA, 611–626. https://doi.org/10.1145/3600006.3613165

[23] Corey Lammie, Mostafa Rahimi Azghadi, and Daniele Ielmini. 2021. Empirical metal-oxide RRAM device endurance and retention model for deep learning simulations. *Semiconductor Science and Technology* 36, 6 (apr 2021), 065003. https://doi.org/10.1088/1361-6641/abf29d

[24] Benjamin C. Lee, Engin Ipek, Onur Mutlu, and Doug Burger. 2009. Architecting phase change memory as a scalable dram alternative. *SIGARCH Comput. Archit. News* 37, 3 (June 2009), 2–13. https://doi.org/10.1145/1555815.1555758

[25] H. Y. Lee, Y. S. Chen, P. S. Chen, P. Y. Gu, Y. Y. Hsu, S. M. Wang, W. H. Liu, C. H. Tsai, S. S. Sheu, P. C. Chiang, W. P. Lin, C. H. Lin, W. S. Chen, F. T. Chen, C. H. Lien, and M.-J. Tsai. 2010. Evidence and solution of over-RESET problem for HfOX based resistive memory with sub-ns switching speed and high endurance. In *2010 International Electron Devices Meeting*. 19.7.1–19.7.4. https://doi.org/10.1109/IEDM.2010.5703395

[26] Shuhan Liu, Shengjun Qin, Koustav Jana, Jian Chen, Kasidit Toprasertpong, and H.-S. Philip Wong. 2024. First Experimental Demonstration of Hybrid Gain Cell Memory with Si PMOS and ITO FET for High-speed On-chip Memory. In *2024 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*. 1–2. https://doi.org/10.1109/VLSITechnologyandCir46783.2024.10631344

[27] Yuhan Liu, Hanchen Li, Yihua Cheng, Siddhant Ray, Yuyang Huang, Qizheng Zhang, Kuntai Du, Jiayi Yao, Shan Lu, Ganesh Ananthanarayanan, Michael Maire, Henry Hoffmann, Ari Holtzman, and Junchen Jiang. 2024. CacheGen: KV Cache Compression and Streaming for Fast Large Language Model Serving. In *Proceedings of the ACM SIGCOMM 2024 Conference* (Sydney, NSW, Australia) *(ACM SIGCOMM '24)*. Association for Computing Machinery, New York, NY, USA, 38–56. https://doi.org/10.1145/3651890.3672274

[28] Tommaso Marinelli, José Ignacio Gómez Pérez, Christian Tenllado, Manu Komalan, Mohit Gupta, and Francky Catthoor. 2022. Microarchitectural Exploration of STT-MRAM Last-level Cache Parameters for Energy-efficient Devices. *ACM Trans. Embed. Comput. Syst.* 21, 1, Article 3 (Jan. 2022), 20 pages. https://doi.org/10.1145/3490391

[29] Fujio Masuoka, Masaki Momodomi, Yoshihisa Iwata, and Riichiro Shirota. 1987. New ultra high density EPROM and flash EEPROM with NAND structure cell. In *1987 International Electron Devices Meeting*. IEEE, 552–555.

[30] Jagan Singh Meena, Simon M. Sze, Umesh Chand, and Tseung Yuen Tseng. 2014. Overview of emerging nonvolatile memory technologies. *Nanoscale Research Letters* 9 (2014), 526 – 526. https://api.semanticscholar.org/CorpusID:3932089

[31] Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, et al. 2023. SpecInfer: Accelerating Generative Large Language Model Serving with Tree-based Speculative Inference and Verification. *arXiv preprint arXiv:2305.09781* (2023).

[32] Gabriel Molas, Giuseppe Piccolboni, Alessandro Bricalli, Anthonin Verdy, I. Naot, Y. Cohen, Amir Regev, Ishai Naveh, Damien Deleruyelle, Quentin Rafhay, Niccolo Castellani, Lucas Reganaz, Alain Persico, R. Segaud, Jean-François Nodin, Valentina Meli, Shelia A. Martin, François Andrieu, and Laurent Grenouillet. 2022. High temperature stability embedded ReRAM for 2x nm node and beyond. *2022 IEEE International Memory Workshop (IMW)* (2022), 1–4. https://api.semanticscholar.org/CorpusID:249049282

[33] Carlo Montangero. 1974. An approach to the optimal specification of read-only memories in microprogrammed digital computers. *IEEE Trans. Comput.* 100, 4 (1974), 375–389.

[34] C. Nail, G. Molas, Philippe Blaise, Giuseppe Piccolboni, Benoit Sklénard, Carlo Cagli, M. Bernard, Anne Roule, Muhamad Azzaz, E. Vianello, C. Carabasse, R. Berthier, David Cooper, C. Pelissier, T. Magis, Gerard Ghibaudo, Christophe Vallée, D. Bedeau, O. Mosendz, and L. Perniola. 2016. Understanding RRAM endurance, retention and window margin trade-off using experimental results and simulations. 4.5.1–4.5.4. https://doi.org/10.1109/IEDM.2016.7838346

[35] nvidia.com 2024. The NVIDIA Blackwell Architecture. https://resources.nvidia.com/en-us-blackwell-architecture?ncid=no-ncid.

[36] Veera Papirla and Chaitali Chakrabarti. 2009. Energy-aware error control coding for flash memories. In *Proceedings of the 46th Annual Design Automation Conference*. 658–663.

[37] Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. 2024. Splitwise: Efficient generative LLM inference using phase splitting. In *ISCA*. https://www.microsoft.com/en-us/research/publication/splitwise-efficient-generative-llm-inference-using-phase-splitting/

[38] Reuters.com 2025. Microsoft plans to invest $80 billion on AI-enabled data centers in fiscal 2025. https://www.reuters.com/technology/artificial-intelligence/microsoft-plans-spend-80-bln-ai-enabled-data-centers-fiscal-2025-cnbc-reports-2025-01-03/.

[39] D. Shum, D. Houssameddine, S. T. Woo, Y. S. You, J. Wong, K. W. Wong, C. C. Wang, K. H. Lee, K. Yamane, V. B. Naik, C. S. Seet, T. Tahmasebi, C. Hai, H. W. Yang, N. Thiyagarajah, R. Chao, J. W. Ting, N. L. Chung, T. Ling, T. H. Chan, S. Y. Siah, R. Nair, S. Deshpande, R. Whig, K. Nagel, S. Aggarwal, M. DeHerrera, J. Janesky, M. Lin, H.-J. Chia, M. Hossain, H. Lu, S. Ikegawa, F. B. Mancoff, G. Shimon, J. M. Slaughter, J. J. Sun, M. Tran, S. M. Alam, and T. Andre. 2017. CMOS-embedded STT-MRAM arrays in 2x nm nodes for GP-MCU applications. In *2017 Symposium on VLSI Technology*. T208–T209. https://doi.org/10.23919/VLSIT.2017.7998174

[40] SiliconMatter 2024. The Memory Wall and Its Implications. https://siliconmatter.substack.com/p/the-memory-wall-and-its-implications.

[41] Devesh Singh and Donald Yeung. 2024. MORSE: Memory Overwrite Time Guided Soft Writes to Improve ReRAM Energy and Endurance. In *Proceedings of the 2024 International Conference on Parallel Architectures and Compilation Techniques* (Long Beach, CA, USA) *(PACT '24)*. Association for Computing Machinery, New York, NY, USA, 26–39. https://doi.org/10.1145/3656019.3676890

[42] 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)*. 876–889. https://doi.org/10.1109/ISCA59077.2024.00068

[43] Clinton W. Smullen, Vidyabhushan Mohan, Anurag Nigam, Sudhanva Gurumurthi, and Mircea R. Stan. 2011. Relaxing non-volatility for fast and energy-efficient STT-RAM caches. In *2011 IEEE 17th International Symposium on High Performance Computer Architecture*. 50–61. https://doi.org/10.1109/HPCA.2011.5749716

[44] Spheron 2024. How Much GPU Memory is Required to Run a Large Language Model? Find Out Here! https://blog.spheron.network/how-much-gpu-memory-is-required-to-run-a-large-language-model-find-out-here.

[45] Stanford 2024. DAM: Differentiated Access Memory Systems and Applications. https://dam.stanford.edu/assets/Stanford_DAM_2_Pages_2024.pdfl.

[46] Jovan Stojkovic, Chaojie Zhang, Íñigo Goiri, Esha Choukse, Haoran Qiu, Rodrigo Fonseca, Josep Torrellas, and Ricardo Bianchini. 2025. TAPAS: Thermal- and Power-Aware Scheduling for LLM Inference in Cloud Platforms. arXiv:2501.02600 [cs.DC] https://arxiv.org/abs/2501.02600

[47] Guangyu Sun. 2013. *Exploring Memory Hierarchy Design with Emerging Memory Technologies*. Springer Publishing Company, Incorporated.

[48] Zhenyu Sun, Xiuyuan Bi, Hai Li, Weng-Fai Wong, Zhong-Liang Ong, Xiaochun Zhu, and Wenqing Wu. 2011. Multi retention level STT-RAM cache designs with a dynamic refresh scheme. In *2011 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 329–338.

[49] Yupeng Tang, Runxiang Cheng, Ping Zhou, Tongping Liu, Fei Liu, Wei Tang, Kyoungryun Bae, Jianjun Chen, Wu Xiang, and Rui Shi. 2024. Exploring CXL-based KV Cache Storage for LLM Serving. *Workshop on ML for Systems at NeurIPS 2024* (2024). https://mlforsystems.org/assets/papers/neurips2024/paper17.pdf

[50] TomsHardware 2023. Micron Plans HBM4E in 2028. https://www.tomshardware.com/pc-components/ddr5/micron-plans-hbm4e-in-2028-256gb-ddr5-12800-ram-sticks-in-2026.

[51] TomsHardware.com 2024. Nvidia's next-gen AI GPU is 4X faster than Hopper: Blackwell B200 GPU delivers up to 20 petaflops of compute and other massive improvements. https://www.tomshardware.com/pc-components/gpus/nvidias-next-gen-ai-gpu-revealed-blackwell-b200-gpu-delivers-up-to-20-petaflops-of-compute-and-massive-improvements-over-hopper-h100.

[52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[53] Naveen Verma, Hongyang Jia, Hossein Valavi, Yinqi Tang, Murat Ozatay, Lung-Yen Chen, Bonan Zhang, and Peter Deaville. 2019. In-memory computing: Advances and prospects. *IEEE Solid-State Circuits Magazine* 11, 3 (2019), 43–55.

[54] vLLM.ai 2024. Automatic Prefix Caching. https://docs.vllm.ai/en/latest/features/automatic_prefix_caching.html.

[55] Run-Jin Wu, Feng Chen, Cheng-Jer Yang, Feng Xu, OneGyun Na, and Ying-Qi Yang. 2022. A Fully Parallel On-Die ECC Architecture with High Area Reduction and RAS Enhancement for HBM3. In *2022 IEEE 16th International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*. 1–3. https://doi.org/10.1109/ICSICT55466.2022.9963306

[56] 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)*. 476–488. https://doi.org/10.1109/HPCA.2015.7056056

[57] Hanmei Yang, Jin Zhou, Yao Fu, Xiaoqun Wang, Ramine Roane, Hui Guan, and Tongping Liu. 2024. ProTrain: Efficient LLM Training via Memory-Aware Techniques. *arXiv preprint arXiv:2406.08334* (2024).

[58] Shimeng Yu, Wonbo Shim, Xiaochen Peng, and Yandong Luo. 2021. RRAM for Compute-in-Memory: From Inference to Training. *IEEE Transactions on Circuits and Systems I: Regular Papers* 68, 7 (2021), 2753–2765. https://doi.org/10.1109/TCSI.2021.3072200

[59] Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, Jie Jiang, and Bin Cui. 2024. Retrieval-Augmented Generation for AI-Generated Content: A Survey. arXiv:2402.19473 [cs.CV] https://arxiv.org/abs/2402.19473

[60] zonedstorage.io 2019. SSDs with NVMe Zoned Namespace (ZNS) Support. https://zonedstorage.io/docs/introduction/zns.