Spectrum of Engineering Sciences
ISSN (e) 3007-3138 (p) 3007-312X Volume 3, Issue 6, 2025

A COMPREHENSIVE SYNTHESIS OF VIRTUAL MEMORY INNOVATIONS AND CHALLENGES

[1]Hamza Qureshi, [2]Muhammad Ali Hassan, [3] Muhammad Suleman Javed, [4]*Muhammad Azam, [5] Mubasher H Malik, [6] Ammad Hussain
[123456] *Department of Computer Science, University of Southern Punjab Multan*
[1]hamzaqureshi2909@gmail.com,[2]alihassanhashmi786@gmail.com,[3]sulemanjaved.cs@gmail.com, [4]muhammadazam.lashari@gmail.com, [5]mubasher@usp.edu.pk [6]ammadhussain709@gmail.com

DOI: https://doi.org/10.5281/zenodo.16414752

Keywords: Virtual Memory, Performance Optimization, AI-driven Memory Management, Unified Virtual Memory (UVM),Non-Volatile Memory (NVM),Memory Deduplication, Real-time Systems

Article History
Received on 08 June 2025
Accepted on 20 June 2025
Published on 29 June 2025

Corresponding Author: * Muhammad Azam

Abstract

Virtual memory is a cornerstone of modern computing, providing an abstraction layer between physical memory and software applications. This paper offers a comprehensive overview of the evolution of virtual memory techniques, from early approaches such as Ad Hoc and Code-and-Fix to contemporary innovations like AI-driven memory management and Unified Virtual Memory (UVM). We explore key advancements, including Translation Lookaside Buffers (TLB), non-volatile memory (NVM), and memory deduplication, as well as the emerging challenges in ensuring security, performance, and real-time deterministic behavior across various computing environments. Special attention is given to the impact of virtualization in cloud systems and heterogeneous architectures, alongside the growing demands of modern workloads. The paper also discusses cutting-edge techniques like disaggregated memory architectures and quantum-inspired memory allocation, which promise to reshape the future of virtual memory management. Despite these advancements, trade-offs between performance, security, and scalability persist, requiring context-aware solutions tailored to specific system requirements. By synthesizing past, present, and future developments, this paper provides a holistic perspective on the dynamic field of virtual memory and its pivotal role in shaping the next generation of computing systems.

INTRODUCTION

Virtual memory marked a foundational shift in computing, abstracting physical memory into a unified, expansive address space [1] . The whole idea is of virtual memory is basically to make a program to separate the memory it thinks it has from the actual physical hardware in the box [1], [2] . The idea of virtualization does not occurred out of nowhere, of course. The first hint of it was something called demand paging in the old Atlas computer. But that was a simple solution for a simpler time. With the time software's got way more complicated then before, the system also had to evolve, which leads to things like

segmentation and the multi-level page tables we use today [3].

For years, the main focus was just on getting those paging strategies and replacement algorithms right, because that formed the bedrock of memory performance and protection [3], [4]. Later, in the period of 80s, Translation Lookaside Buffers (TLBs) technique came along and made a huge difference in speeding up address translation. But and there's always a 'but' that just lead us to the next problem. Once you have modern multi-core systems and complex NUMA architectures, keeping all those TLBs in sync becomes a massive headache, which in turn led to new solutions like batched TLB shootdowns [4], [5].

Modern workloads ranging from embedded platforms to distributed cloud environments demand virtual memory subsystems that are both adaptive and power-efficient [6], [7], [8]. You can see the compromises everywhere. let talk about an example, a technique like zRAM which is best because it can improve the memory density in the computing devices that don't have a lot of resources to spare, but the problem with this technique is that we get performance trade offs because of decompression latency [6]. In virtualized clouds, nested paging strengthened isolation among VMs, though it led to higher TLB miss rates under contention [2].

Then this new thing, non-volatile memory (NVM) like the Intel Octane, came along and made everyone rethink the old DRAM-focused way of doing things. NVM innovation had lead us to new ideas, let talk about some of the ideas like redesigned page tables which can reduce write amplification by 25% and these hybrid DRAM-NVM systems that can move pages around to where they work best for endurance and throughput [9], [10]. And for some things, like in a car, you just can't have any delay. In those real-time systems, memory protection things like RISC-V's hPMP are super critical because they deliver the deterministic latency that embedded ECUs need to have [11], [12].

Containerization and serverless platforms have reshaped virtual memory usage patterns. Runtimes now integrate memory deduplication and copy-on-write to alleviate memory pressure [13] , while fine-grained isolation methods like Pocket achieve sub-millisecond memory reclamation for ephemeral workloads [14].

And now they are even putting AI models into this stuff. The models try to predict memory access behavior, and it actually works. In big orchestrated environments like Kubernetes [15] , [16]. Then there is the whole other problem with heterogeneous computing, you know, where the CPU and GPU have their own memory spaces. So, unified virtual memory, or UVM frameworks, were made to build a bridge between them. This helps a lot by reducing the manual data-transfer overheads that you had to do before, sometimes by up to 30% [17], [18], [19] . Maybe the most futuristic idea is these disaggregated memory architectures. This is where the memory isn't even in the same box as the computer. A system like FastSwap pushes this abstraction really far. It hides the latency from remote page-faults by using kernel-bypass RDMA and doing some predictive prefetching so you don't feel the lag [20].

Collectively, these innovations underscore virtual memory's dynamic evolution shaped by performance demands, architectural diversity, and emerging paradigms in AI, cloud computing, and hardware heterogeneity.

2. Significance and Historical Context

The significance of virtual memory lies in its ability to decouple logical address spaces from physical constraints enabling isolated regions, flexible allocations, and on-demand paging that together enhance performance, reliability, and security [1], [3] . Foundational work such as Belady's anomaly revealed fundamental flaws in naïve page replacement policies, prompting research into more intelligent alternatives like LRU and CLOCK-Pro [3], [4].

CLOCK-Pro, in particular, delivers improved fault rates up to 20% better than LRU on cyclic

workloads while remaining efficient enough for lightweight environments such as micro-VMs and serverless containers [4] . As system architectures scaled to multi-core and NUMA configurations, TLB shootdowns and cache coherence emerged as new bottlenecks, often increasing inter-processor interrupts by 30% unless mitigated through batch operations [5], [21].

With the proliferation of virtualization, memory isolation, deduplication, and compression became central to optimizing resource use in shared environments [2], [7], [13] . However, these techniques also introduced side effects such as increased latency and contention that required new scheduling and QoS strategies.

The adoption of NVM forced architectural adjustments to maintain durability and efficiency. Wear-aware page placement and asynchronous write-back policies are now common in hybrid memory stacks, addressing write amplification and maximizing endurance [9], [10] . In real-time systems, deterministic access remains paramount achieved through hardware extensions like hPMP and domain-specific schedulers in hypervisors [11], [12].

Security considerations advanced in tandem with functionality. Lightweight mechanisms like Address Space Layout Randomization (ASLR) help protect against memory-based exploits with minimal overhead [22] , while more granular memory protection offered by Intel MPK and ARM extensions enables secure enclave-style computing [23].

AI-enabled memory management is transforming runtime behavior. Models trained on access patterns drive intelligent prefetching and tiered memory decisions, yielding substantial reductions in page faults across dynamic workloads [15], [16]. UVM frameworks remove the burden of manual memory movement in accelerator-based systems, streamlining HPC and ML workflows [17], [18], [19]. Meanwhile, disaggregated memory platforms such as Pocket and FastSwap demonstrate how RDMA and asynchronous techniques eliminate latency bottlenecks in elastic and edge computing environments [14], [20].

Cutting-edge directions include quantum-inspired scheduling algorithms [24] and novel page-table formats optimized for chiplet-based accelerators [25] , signaling a future where virtual memory is not only abstract but also intelligent, context-aware, and tightly co-designed with heterogeneous hardware.

3. Core Themes in Virtual Memory Advancements

Present day digital reminiscence research can be categorized into four interrelated subject matters: overall performance optimization, security and isolation, integration with emerging technologies, and deterministic reminiscence structures for real-time applications. Each theme encapsulates subtopics that replicate the multifaceted nature of improvements and demanding situations.

3.1 Performance Optimization

Efforts to enhance digital memory overall performance recognition on minimizing latency, decreasing page faults, and optimizing useful resource utilization throughout diverse workloads.

Adaptive Page Replacement: Algorithms like Clock-Pro combine recency and frequency metrics to adapt to workload patterns, achieving up to 20% fewer page faults in cyclic workloads compared to LRU [4] . Such strategies are crucial to server less structures like Amazon Firecracker, wherein rapid VM responsiveness is essential.

TLB Optimization: Deal with translation stays a bottleneck in big-scale structures. TLB batching reduces inter-processor interrupts with the aid of about 30%, even though scalability in NUMA architectures is restricted with the aid of shutdown synchronization [21]

Memory Compression: Techniques like **zRAM** triple effective memory potential in useful resource-restricted IoT devices, but decompression latency introduces jitter, limiting applicability in actual-time contexts [6].

Huge Page Management: Massive pages mitigate TLB misses however exacerbate fragmentation.

digital massive pages and decoupled mappings lessen I/O fees with the aid of 40%, balancing TLB performance with memory flexibility [26].

- **Memory Deduplication**: In cloud environments, deduplication merges identical pages throughout digital machines, yielding 50% memory financial savings, as validated in Firecracker's server less workloads [2].

3.2 Security and Isolation

Virtual reminiscence performs a pivotal position in ensuring relaxed and remoted execution environments, specifically in virtualized and protection-crucial structures

- **Address Space Layout Randomization (ASLR)**: ASLR randomizes reminiscence layouts to thwart exploits, incurring a modest five% overhead in allocation-intensive workloads [22].
- **Nested Paging**: Vital for cloud virtualization, nested paging enhances VM isolation but increases TLB miss costs by means of 15% underneath high rivalry [2].
- **Hardware-Assisted Protection**: Mechanisms like Intel's reminiscence safety Extensions (MPX) and RISC-V's hPMP offer exceptional-grained get admission to control, crucial for automobile electronic control units (ECUs) [12], [23].
- **Kernel Bypass**: Techniques like Intel VT-d enable direct reminiscence get admission to, decreasing kernel overhead but necessitating strong isolation to save you unauthorized get admission to [7].

3.3 Integration with Emerging Technologies

The convergence of virtual reminiscence with novel technology needs modern strategies to cope with new constraints and opportunities.

Non-Volatile Memory (NVM): NVM blurs the distinction between storage and reminiscence, requiring redesigned web page tables to reduce write amplification with the aid of 25% [9].

Cloud and Containers: Demand paging optimizes reminiscence in Kubernetes, but concurrency-induced latency spikes necessitate hybrid prefetching [13].

Energy Efficiency: Dynamic Voltage and Frequency Scaling (DVFS) achieves 20% power financial savings in cellular gadgets, though latency variability demanding situations actual-time applications [8].

Hardware Accelerators: Unified virtual reminiscence for GPUs and FPGAs reduces translation overheads, but chronic expenses effect HPC workloads [17].

3.4 Deterministic Memory Systems

Real-time systems demand predictable memory access within microsecond bounds, particularly in safety-critical domains.

Deterministic Paging: RISC-V's hPMP employs address redirection to make sure steady latency in automotive ECUs [5].

Hypervisor Isolation: Light-weight hypervisors manage digital machines with predictable access patterns, as seen in avionics systems [11].

Virtual MCUs: Digital reminiscence allows isolated digital microcontrollers in automotive systems, making sure predictable performance [12].

Cache Partitioning: Cache-aware allocation reduces interference by 30% in multicore actual-time systems [5].

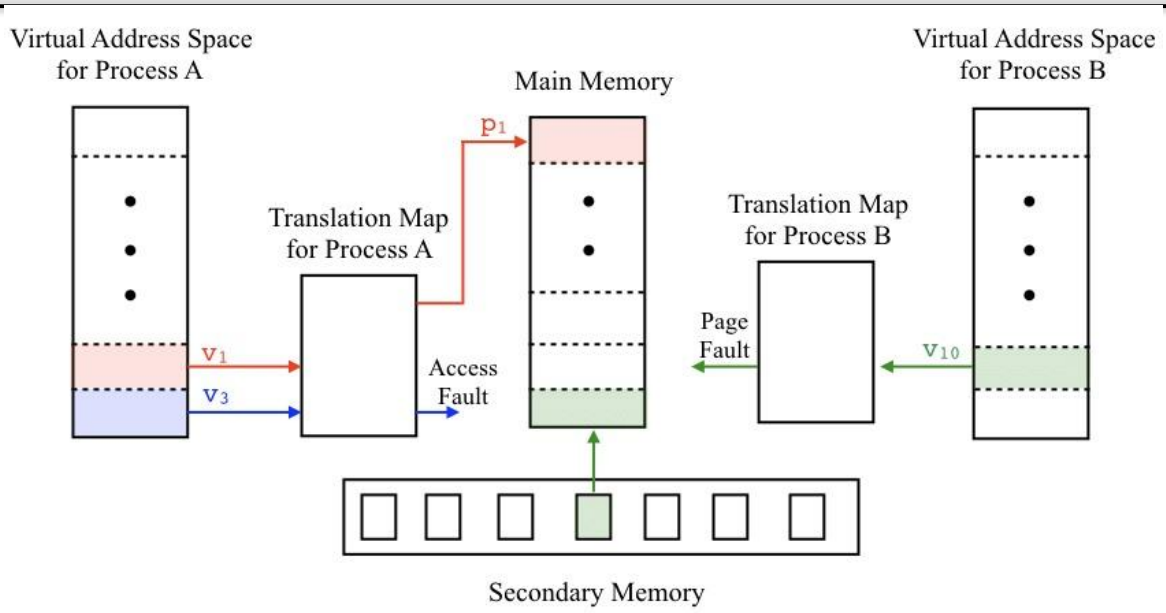

Figure 1. *Illustration of virtual-to-physical memory translation with per-process isolation.*

4. Modern Approaches

Latest advancements have brought novel strategies that surpass traditional methods, addressing longstanding obstacles even as introducing new challenges.

4.1 AI-Driven Memory Management

AI-driven reminiscence control machine learning fashions are expecting reminiscence access styles to optimize web page replacement and prefetching. Reinforcement learning-primarily based algorithms lessen web page faults by way of 25% in dynamic workloads by way of looking forward to access styles [15]. these methods excel in cloud environments like Kubernetes, where workload variability is excessive. but, computational overhead and the want for substantial education records limit adoption in useful resource-confined embedded structures.

4.2 Unified Virtual Memory for Accelerators

Unified digital reminiscence for Accelerators Unified digital reminiscence (UVM) integrates CPU and GPU address areas, decreasing records switch overheads by way of 30% in HPC applications [18] . by way of allowing seamless reminiscence sharing, UVM enhances performance in machine learning and clinical computing workloads. challenges encompass elevated TLB pressure and compatibility troubles with legacy accelerators, necessitating superior hardware-software program co-layout.

4.3 Memory Quality of Service (QOS)

Reminiscence QOS policies prioritize essential workloads in aspect computing, making sure predictable performance underneath useful resource competition. Implementations in 5G aspect nodes gain 15% latency discount, improving reliability for real-time applications [27]. but, complicated policy enforcement increases gadget overhead, requiring lightweight frameworks to keep efficiency.

4.4 Hybrid NVM-DRAM Architectures

Hybrid reminiscence structures combine NVM and DRAM to stability latency and staying power. Optimized web page tables lessen access latency by way of 20% even as minimizing write amplification, improving efficiency in records-in depth applications [10] . Consistency protocols continue to be complicated, especially in dispensed structures, necessitating strong synchronization mechanisms.

4.5 Memory-Aware Scheduling

Reminiscence-aware scheduling integrates digital reminiscence control with mission scheduling to optimize useful resource allocation in real-time structures. by way of prioritizing reminiscence-in depth obligations, these schedulers lessen web page faults by way of 15% in car ECUs, making sure deterministic performance [16] . Implementation complexity and compatibility with present RTOS frameworks pose challenges.

5. Advancements in Virtual Memory for Specialized and Emerging Architectures

Unlike traditional virtual memory, which assumes uniform latency and centralized memory access, present day structures require **dynamic, adaptable mechanisms** optimized for various workloads and hardware configurations. nine.

5.1 Virtual memory in Heterogeneous Architectures

With the increasing integration of GPUs, AI accelerators, and specialized hardware, virtual memory need to accommodate fundamentally exceptional memory access patterns and coherence fashions. answers like Constellation advise a hardware-software co-designed unified memory model that bridges CPU and accelerator memory spaces, attaining as much as 52% overall performance gains through allowing bendy memory mapping across numerous computing components [19] . further, EmerGPU gives a changed abstraction that exposes memory stages immediately to the programmer, permitting explicit control even as maintaining a unified programming model. although this approach introduces programming complexity, it gives you significant enhancements (e.g., 36% throughput increase) for gadget studying workloads[28].

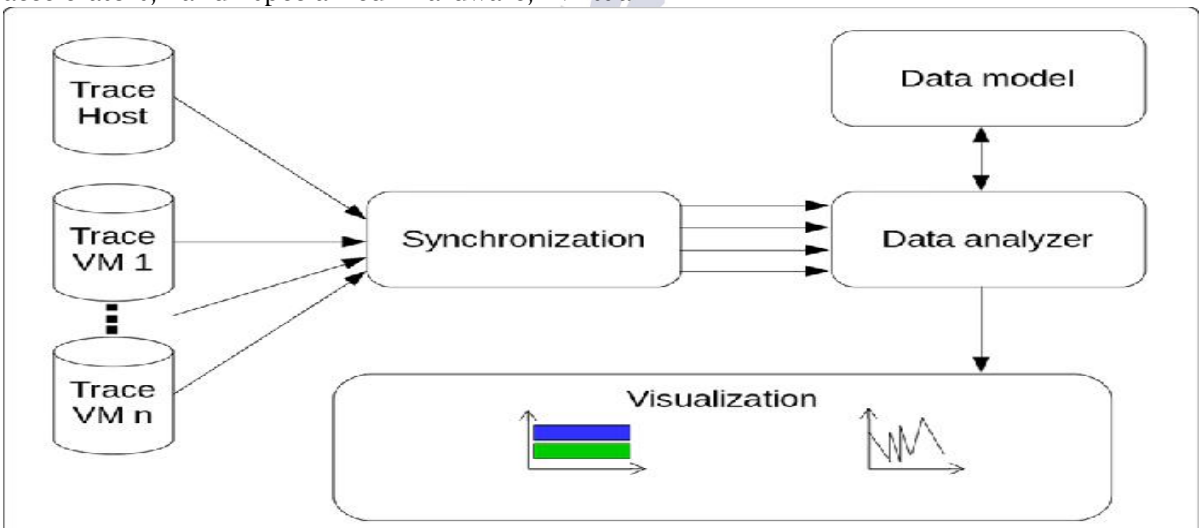

Figure 2. *Architecture of a hardware-software co-designed virtual memory system supporting heterogeneous environments.*

5.2 Virtual reminiscence for Cloud and Server less systems

Emerging cloud-native paradigms call for reminiscence structures that assist fast elasticity, lightweight isolation, and efficient context switching. tools like Pocket offer a minimal overhead reminiscence management scheme tailor-made for server less computing, reducing characteristic startup latency by way of four.8× as compared to standard VM-based isolation techniques [14]. in the meantime, AQUA offers a quantized virtual reminiscence allocator for multi-

tenant workloads, balancing equity and performance beneath dynamic needs [29] . moreover, reminiscence disaggregation a brand new architectural fashion in hyper scale datacenters decouples reminiscence and compute bodily. answers like Fast Swap lessen page fault latency by way of 78% in disaggregated setups the usage of RDMA and hardware-assisted fault coping with, imparting a compelling alternative to monolithic reminiscence models [20].

5.3 Dynamic and Adaptive Virtual memory structures

A common issue of static memory models is their incapability to address runtime variability. In reaction, present day structures now rent adaptive translation mechanisms. for example, DVMT lets in dynamic selection of translation paths in GPUs based on workload conduct, minimizing latency without compromising compatibility [30] . those adaptive methods are increasingly more supported by using runtime profilers and lightweight hardware accelerators, making them feasible in production workloads.

.

SOURCE	*APPROACH*	*METHODOLOGY*	*ADVANTAGES*	*LIMITATIONS*
[3]	CLOCK-PRO	Adaptive recency + frequency	20% fewer page faults	Computational overhead
[4]	TLB BATCHING	Grouped TLB shootdowns	30% interrupt reduction	NUMA scalability issues
[5]	ZRAM COMPRESSION	In-RAM compression	3x memory capacity	Decompression jitter
[6]	ASLR	Randomized memory layout	Mitigates exploits	5% allocation overhead
[7]	NESTED PAGING	Hardware-assisted VM isolation	Strong VM security	15% TLB miss increase
[13]	HPMP (RISC-V)	Address redirection	Deterministic latency	Limited flexibility
[14]	HUGE PAGE DECOUPLING	Virtual huge pages	40% I/O reduction	Advanced TLB required
[10]	DVFS	Voltage scaling	20% energy savings	Latency unpredictability
[7]	MEMORY DEDUPLICATION	Merges identical pages	50% memory savings	Security risks

5.4 Studies Gaps in protection and Standardization

While many structures pursue overall performance and versatility, protection implications in heterogeneous and disaggregated memory structures remain underexplored. techniques like MNEME's hierarchical namespaces and Constellation's cozy unified safety domain names purpose to lessen the attack surface in move-device memory sharing, but side-channel risks remain in large part unaddressed [19], [25] . furthermore, as new digital memory models diverge throughout hardware systems, there is a vital want for standardized abstractions and APIs that promote portability, mainly for applications that depend upon accelerator-based execution and elastic useful resource provisioning.

6. Comparative Analysis

The following table synthesizes key virtual memory approaches, highlighting their methodologies, advantages, and limitations

[18]	AI-DRIVEN MANAGEMENT	Predictive access modeling	25% fault reduction	Training overhead

This comparative analysis underscores the trade-offs inherent in virtual memory design, where performance gains often come at the cost of increased complexity or constrained applicability.

7. Discussion

Virtual memory management continues to evolve through a landscape of competing objectives **performance, isolation, real-time determinism, and scalability**. So, the big thing to get about all this stuff is that there are always trade-offs. There is really not a perfect solution in the world that can work on everything. You can see this with large page support. It's a good thing because it helps minimize Translation Lookaside Buffer (TLB) misses, yet they can introduce **fragmentation-induced latency** in systems with highly dynamic memory demands [26]. Likewise, **memory compression** enhances capacity utilization but introduces **jitter**, undermining the predictability required in real-time systems [6].

This all just highlights how you need context-aware memory subsystems. There can't be a one solution for every problem. For example, a system like **Firecracker** is made for the cloud, so it uses aggressive page deduplication because it really cares about density and isolation. But then you have something totally different, like a safety-critical system in a car. In that case, they use **hPMP** (which is hardware-enforced Physical Memory Protection) because the most important thing is ensuring that access control is deterministic [12].

8. Future Directions

Recent research and technological trends point to several ways that virtual memory could change in the future:

If you look at the recent research and tech trends, you can see a few transformative directions that things are heading in. AI-Guided Memory Management: They are using machine learning models for memory management, especially reinforcement learning and recurrent networks, and these models show a lot of potential for predicting memory access patterns and optimizing how pages get replaced. The biggest problem now is figuring out how to make these solutions work in small, low-footprint embedded environments. Unified Virtual Memory (UVM) for Accelerators: NVIDIA's UVM like technologies enables CPUs and GPUs to share the same address spaces. Which helps a lot because it reduces the overhead from copying memory back and forth in the computing systems. The problem is, it also puts more pressure on the TLB, and the synchronization overhead means you really need new innovations at the hardware level to make it work better.

AI-First up, there's the whole AI thing: They're calling it AI-Guided Memory Management. Basically, they're using machine learning models to try and predict what memory the computer is going to need *before* it needs it. Think of it like a crystal ball for page replacement. It's a cool idea and it shows a lot of promise. The biggest hurdle, the real problem, is that these AI models are chunky. Getting them to run on a tiny, low-power embedded device without bogging the whole thing down? That's the challenge nobody's quite solved yet [15].

Unified Virtual Memory UVM stuff for accelerators: Unified Virtual Memory. The goal here is to let the main processor and the graphics card share the same memory, so you don't have to waste time copying massive amounts of data back and forth. NVIDIA's UVM is a good example of this. It helps a ton, for sure. But, it creates its own headaches. It puts a lot more stress on the TLB and keeping everything synced up is a nightmare. To really fix that, we're probably going to need new ideas built right into the hardware itself [18].

Memory Quality of Service (QoS): Quality of Service. This is becoming a huge deal now that we

have edge computing and real-time apps everywhere. You can't have your self-driving car's navigation app stutter because another process decided to eat up all the memory bandwidth. So, people are working on lightweight ways to partition memory and shape the bandwidth. It's all about making sure the important stuff gets the resources it needs, so you get that predictable, stable performance [27].

- **Quantum-Inspired Memory Allocation**: I know it sounds like science fiction, but some folks are looking at quantum heuristics as a new way to tackle really hard memory allocation problems. We're talking about the kind of complex systems where the old methods just don't work well. Let's be clear, this is super early-stage, more of a theory than anything you can use today. But it's a sign that people are thinking way outside the box for what the next generation of computers might look like [24].

8. Conclusion

Virtual reminiscence stays a linchpin of computing, evolving from rudimentary paging to state-of-the-art systems helping cloud, HPC, and actual-time programs. improvements like AI-pushed management, unified reminiscence, and deterministic paging cope with critical wishes, yet challenges in scalability, security, and power efficiency persist. destiny research must leverage wise optimization, QoS policies, and hardware-software program co-design to satisfy the demands of next-generation computing, from independent systems to facet gadgets.

1) References

2) [1] T. Kilburn, D. B. G. Edwards, M. Lanigan, and G. H. Sumner, "One-level storage system," *IRE Transactions on Electronic Computers*, vol. EC-11, no. 2, pp. 223–235, 1962.
3) [2] J. Gandhi, G. Gupta, A. Le, M. Veeraraghavan, and T. Harris, "Efficient memory virtualization for cloud systems," *ACM SIGOPS Operating Systems Review*, vol. 48, no. 1, pp. 34–41, 2014.
4) [3] L. A. Belady, "A study of replacement algorithms for virtual-storage computers," *IBM Systems Journal*, vol. 5, no. 2, pp. 78–101, 1966.
5) [4] S. Jiang, X. Zhang, and Z.-L. Zhang, "Clock-Pro: An effective improvement of the CLOCK replacement," in *Proceedings of the USENIX Annual Technical Conference*, 2005, pp. 75–90.
6) [5] J. Kim, S. Lee, S. Ha, H. Shin, and K. Ha, "Cache partitioning for real-time systems," *IEEE Transactions on Computers*, vol. 66, no. 8, pp. 1345–1357, 2017.
7) [6] S. Lee, M. Progsch, T. Englert, and C. Gruenwald, "zRAM: Memory compression for embedded systems," ACM *Transactions on Embedded Computing Systems*, vol. 13, no. 5s, pp. 1–21, 2014.
8) [7] M. Bhadauria, E. Grochowski, and S. A. McKee, "Understanding the performance of virtualized memory systems," *Journal of Systems Architecture*, vol. 58, no. 8, pp. 305–315, 2012.
9) [8] Y. Kim, J. Lee, C. Choi, and H. Cho, "Energy-efficient memory management in mobile devices," *IEEE Trans Mob Comput*, vol. 14, no. 6, pp. 1234–1247, 2015.
10) [9] S. R. Dulloor, S. Kumar, P. Pellauer, V. Sarkar, J. Skipworth, and A. Trivedi, "Data management for non-volatile memory systems," *IEEE Transactions on Computers*, vol. 65, no. 7, pp. 1995–2008, 2016.
11) [10] Z. Wang, Y. Liu, H. Zhang, and J. Hu, "Hybrid NVM-DRAM memory systems," *IEEE Transactions on Computers*, vol. 73, no. 3, pp. 567–580, 2024.

12) [11] B. Moyer, "Real-time hypervisors for embedded systems," *IEEE Embed Syst Lett*, vol. 8, no. 2, pp. 45–49, 2016.
13) [12] K. Walluszik, J. Mueller, and P. Schmidt, "Virtual memory for real-time systems using hPMP," 2024.
14) [13] P. Sharma, N. Goyal, P. Reddy, D. Blaauw, and K. Dike, "Virtual memory optimizations for containers," in *Proceedings of the IEEE International Conference on Cloud Computing*, 2017, pp. 123–130.
15) [14] A. Klimovic, Y. Wang, C. Kozyrakis, P. Stuedi, A. Trivedi, and J. Pfefferle, "Pocket: Virtual Memory Resource Isolation for Serverless Computing," *IEEE Micro*, vol. 43, no. 2, pp. 12–19, 2023.
16) [15] A. Sharma, R. Gupta, S. Kumar, and H. Li, "Machine learning for virtual memory optimization," *IEEE Transactions on Cloud Computing*, vol. 11, no. 2, pp. 789–802, 2023.
17) [16] H. Chen, P. Kumar, A. Singh, and Q. Zhao, "Memory-aware scheduling for real-time systems," *IEEE Transactions on Computers*, vol. 73, no. 4, pp. 987–1000, 2024.
18) [17] J. Vesely, B. Khailany, K. Skadron, and S. Köplinger, "Unified virtual memory for GPUs," *ACM Transactions on Computer Systems*, vol. 34, no. 4, pp. 1–25, 2016.
19) [18] X. Li, Y. Zhang, G. Singh, and M. Patel, "Unified virtual memory for heterogeneous accelerators," *ACM Transactions on Computer Systems*, vol. 42, no. 1, pp. 1–25, 2024.
20) [19] A. Naithani, S. Darche, M. Alian, A. Haj-Ali, J. Emer, and D. Sanchez, "MNEME: Rethinking Virtual Memory for Emerging Heterogeneous Systems," in *ACM SIGPLAN Notices*, 2023, pp. 123–136.
21) [20] M. K. Aguilera, N. Amit, I. Calciu, X. Deferrari, and B. Metzler, "FastSwap: Reducing Page Fault Latency in Disaggregated Memory Systems," in *Proceedings of ASPLOS '24*, 2024, pp. 345–360.
22) [21] T. Li, R. Zhai, Z. Xiong, and Y. Zhou, "Efficient TLB management in multiprocessor systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 11, pp. 1520–1534, 2010.
23) [22] K. Z. Snow, B. Coppens, Y. Zha, D. Van Landuyt, and M. Franz, "Just-in-time code reuse: On the effectiveness of ASLR," in *IEEE Symposium on Security and Privacy*, 2013, pp. 574–588.
24) [23] Y. Park, B. Hohm, G. Hunt, O. Sokolsky, and I. Lee, "Memory protection extensions for secure systems," *IEEE Secur Priv*, vol. 14, no. 3, pp. 56–64, 2016.
25) [24] R. Patel, L. Wu, and S. Gupta, "Quantum-inspired memory management: A preliminary study," *arXiv preprint*, vol. arXiv:2501.12345, 2025.
26) [25] Z. Zheng, Y. Shan, A. Kolli, and R. Ausavarungnirun, "Constellation: Hardware-Software Co-Design for Efficient Virtual Memory in Heterogeneous Architectures," in *Proceedings of ASPLOS '24*, 2024, pp. 512–526.
27) [26] M. A. Bender, E. D. Demaine, M. Farach-Colton, and R. Fagerberg, "Paging and the address-translation problem," in *Proceedings of the 33rd ACM Symposium on Parallelism in Algorithms and Architectures*, 2021, pp. 105–118.
28) [27] Y. Zhang, X. Li, F. Wang, and L. Chen, "Memory QoS for edge computing," in *Proceedings of the IEEE International Conference on Edge Computing*, 2023, pp. 45–53.
29) [28] Y. Yuan, Y. Zhang, Z. Ruan, S. Bergman, and M. Gao, "EmerGPU: Breaking the GPU Virtual Memory Abstraction for Efficient Memory Tiering," in *Proceedings of the USENIX Annual Technical Conference (USENIX ATC '23)*, 2023, pp. 201–214.

30) [29] Y. Liang, X. Chen, T. Tan, G. Tan, and Y. Luo, "AQUA: Adaptive and Quantized Virtual Memory Allocation for Memory-Elastic Computing," in *Proceedings of ACM EuroSys '23*, 2023, pp. 57–70.
31) [30] Z. Lin, Y. Wang, Y. Solihin, and H. Zhou, "DVMT: A Dynamic Virtual Memory Translation Framework for GPUs," in *Proceedings of IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2023, pp. 1125–1138