**Memory Hierarchy Design Analysis Using gem5 Simulation**

Chibuzo Ufomba

University of the Cumberlands

MSCS-531-B50: Computer Architecture and Design

Dr. Brandon Bass

July 20, 2025

**Memory Hierarchy Design Analysis Using gem5 Simulation**

Memory hierarchy design is a crucial element of modern computer architecture, linking high-speed processors and slower storage systems. With the advancements in computing power and performance, sufficient memory bandwidth and minimal access latency must be achieved to ensure adequate growth in processor development. The gap between processor speed performance and memory access latency has widened, creating the "memory wall challenge” (Hennessy & Patterson, 2019). This concept is illustrated by the divergent performance trajectories: processor speeds increased approximately 75% per year, while DRAM speeds improved by 7% per year during the same period (Wulf & McKee, 1995). The memory hierarchy offers a practical and effective solution to this core challenge by using the principle of locality to build a layered structure that balances speed, cost, and storage capacity (Hennessy & Patterson, 2019).

This report will examine the core technologies that form the foundation of modern memory hierarchies, focusing on the properties and trade-offs of Static Random Access Memory (SRAM) and Dynamic Random Access Memory (DRAM). Furthermore, this report will explore advanced cache optimization techniques that go beyond basic principles of locality, investigate the complexities of virtual memory systems, and address the cross-cutting design issues that influence memory hierarchy decisions in contemporary computing systems.

**Memory Technologies**

The basis of any memory hierarchy rests on comprehending the range of memory technologies and their trade-offs. Static Random Access Memory (SRAM) occupies the top of the memory hierarchy, distinguished by its high speed and minimal access latency. SRAMs can retain data without refresh cycles, which allows access times to be very close to the cycle time (Hennessy & Patterson, 2019). This makes SRAMs an excellent choice for cache applications requiring fast access; however, their high performance comes with a steep cost and higher power usage, rendering them impractical for large-capacity uses (Jacob et al., 2007).

Dynamic Random Access Memory (DRAM) represents another key technology used in the memory hierarchy for constructing caches and main memory systems. This technology occupies the intermediate level of the memory hierarchy, providing significantly greater capacity at a much lower cost than SRAM. DRAM is present in personal computers, phones, gaming consoles, and servers as the primary system memory. The key difference between DRAM and SRAM is that DRAM requires periodic refreshing to maintain stored data, making it slower but more cost-effective for large-capacity applications. To improve performance, Synchronous DRAM (SDRAM) was introduced in the mid-1990s, synchronizing operations with the system clock for predictable timing and faster access (Hennessy & Patterson, 2019). This evolution continued with Double Data Rate (DDR) DRAM in the early 2000s, further increasing bandwidth and reducing access times. The DDR standard has since evolved through multiple generations—DDR2, DDR3, DDR4, and DDR5 (Hennessy & Patterson, 2019).

At the bottom tier of the memory hierarchy, flash-based solid-state drives (SSDs) and hard disk drives (HDDs) offer large storage capacity at low cost but with much slower access times. These storage devices focus on keeping data permanently rather than providing fast access. This arrangement—SRAM for caches, DRAM for main memory, and SSDs/HDDs for storage—works well because it matches faster memory with data the processor needs most often, reducing delays and improving overall system performance.

Beyond traditional volatile memory technologies, emerging non-volatile memory solutions such as Intel Optane (3D XPoint), Resistive RAM (ReRAM), and Phase Change Memory (PCM) are reshaping the memory hierarchy landscape. These technologies ensure persistence without power, enhanced performance, scalability, and energy efficiency (Das, 2022).

**Advanced Cache Optimization**

Optimizing cache performance is crucial in minimizing memory access delays and enhancing overall system performance. Advanced strategies like prefetching, victim caches, and cache partitioning extend beyond traditional cache architectures to reduce cache miss problems.

Prefetching is a technique that fetches data from slower memory into faster cache before the processor requests it. This aims to reduce cache misses, particularly capacity and conflict misses, by anticipating future memory access patterns and improving overall execution performance. The effectiveness of prefetching has made it a standard feature in high-performance processors, including Intel Xeon and IBM POWER series (Wang & Luo, 2017). However, prefetching presents notable trade-offs: inaccurate predictions can lead to cache pollution by displacing useful data with unnecessary prefetched data, while also consuming additional memory bandwidth and potentially increasing power consumption (Wang & Luo, 2017).

Victim caches represent another advanced optimization technique designed to recover from conflict misses in direct-mapped and low-associativity caches. This approach employs a small, fully associative buffer to capture recently evicted cache lines, allowing quick retrieval if the data is needed (Jouppi, 1990). Victim caches present a significant tradeoff: hardware complexity.

Cache partitioning represents another critical technique for managing shared cache resources in multi-core systems. By dividing the cache among different applications or threads, it helps reduce interference and enhance overall performance (Suh, Devadas, & Rudolph, 2001). This technique is especially effective in mitigating inter-process conflicts that arise when multiple workloads compete for cache space.

**Virtual Memory and Virtual Machines**

Virtual memory is a key abstraction that enables modern computing systems to manage memory resources with improved efficiency and security. It creates the illusion of a large, contiguous memory space for each process, regardless of the actual physical memory configuration or availability. At its core, virtual memory uses page tables to establish the mapping between virtual addresses used by applications and physical addresses in RAM (Hornyack et al., 2013). The Memory Management Unit (MMU) performs this address translation in hardware. However, since page table lookups can be costly, the Translation Lookaside Buffer (TLB) serves as a high-speed cache that stores recently used virtual-to-physical address mappings, dramatically reducing translation overhead (Hornyack et al., 2013). When the operating system attempts to access data that is not currently loaded in physical memory, a page fault will occur. Page fault handling involves the operating system employing page replacement algorithms such as Least Recently Used (LRU) and First-In First-Out (FIFO) to determine which page to displace from memory (Zhao & Jin, 2014).

Virtual machines (VMs) build upon virtual memory principles by emulating complete hardware environments to run multiple operating systems on a single physical host. The hypervisor, also known as a Virtual Machine Monitor (VMM), serves as the software layer responsible for managing these virtual machines, allocating physical resources, and maintaining isolation between different VM instances (Hennessy & Patterson, 2019). Virtual machines directly affect memory hierarchy performance by altering cache utilization and memory access patterns.

**Cross-Cutting Issues**

Creating efficient memory hierarchies requires carefully balancing trade-offs between cost, power use, complexity, and performance for various workload types. Cost constraints fundamentally limit the amount of fast memory that can be economically used, requiring careful allocation of costly SRAM resources. While SRAM provides exceptional speed, its high cost and power consumption limit its scalability. DRAM is more affordable but requires periodic refreshes, which add latency and increase energy use. Storage devices such as SSDs and HDDs are cost-effective for large capacities but have higher access latencies, making them less ideal for latency-sensitive applications.

Several emerging trends are transforming the future of memory hierarchy design. Non-volatile memory technologies—such as Intel Optane, Resistive RAM, and Phase Change Memory (PCM)—bridge the gap between traditional storage and main memory by providing persistent storage with high performance (Das, 2022). However, these technologies introduce new design constraints, such as longer write latencies and write energy consumption (Sun et al., 2018).

**Conclusion**

Memory hierarchy design is crucial for optimizing the balance between speed, cost, performance, and memory capacity in computing systems. Memory technologies such as SRAM, DRAM, and flash devices play big roles in memory storage and access. Advanced optimization techniques help to minimize memory access delays and enhance overall system performance. As technologies continue to evolve, memory hierarchy design principles must be carefully considered.

**References**

Das, R. (2022). Emerging memory technologies: A systematic literature review with focus on resistive RAM (RRAM) and phase-change memory (PCM). Retrieved from <https://www.researchgate.net/publication/382694048>

Hennessy, J. L., & Patterson, D. A. (2019). Computer Architecture: A Quantitative Approach (6th ed.). Morgan Kaufmann.

Hornyack, P., Ceze, L., Gribble, S., Ports, D., & Levy, H. (2013). A study of virtual memory usage and implications for large memory. University of Washington Technical Report. Retrieved from <https://homes.cs.washington.edu/~luisceze/publications/vmstudy-uwtr2013.pdf>

Jacob, B., Ng, S., & Wang, D. (2007). Memory Systems: Cache, DRAM, Disk. Morgan Kaufmann.

**Jouppi, N. P.** (1990). Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers. In Proceedings of the 17th Annual International Symposium on Computer Architecture (pp. 364–373). IEEE. <https://doi.org/10.1109/ISCA.1990.134547>

**Suh, G. E., Devadas, S., & Rudolph, L.** (2001). Analytical cache models with applications to cache partitioning. Proceedings of the 15th International Conference on Supercomputing (ICS '01). <https://people.csail.mit.edu/suh/papers/ics01.pdf>

Sun, G., Zhao, J., Poremba, M., Xu, C., & Xie, Y. (2018). Memory that never forgets: Emerging nonvolatile memory and the implication for architecture design. National Science Review, 5(4), 577–592. <https://doi.org/10.1093/nsr/nwx082>

**Wang, H., & Luo, Z.** (2017). Data Cache Prefetching with Perceptron Learning. arXiv. https://arxiv.org/abs/1712.00905

Wulf, W. A., & McKee, S. A. (1995). Reflections on the memory wall. ACM SIGARCH Computer Architecture News, 23(1), 134–135. <https://doi.org/10.1145/216585.216588>

**Zhao, Y., & Jin, H.** (2014). Virtual memory streaming: Virtualization of memory resources for cloud computing. In 2014 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM) (pp. 1–6). IEEE. https://doi.org/10.1109/CCEM.2014.6726496