Memory Management

Advanced Operating System week6

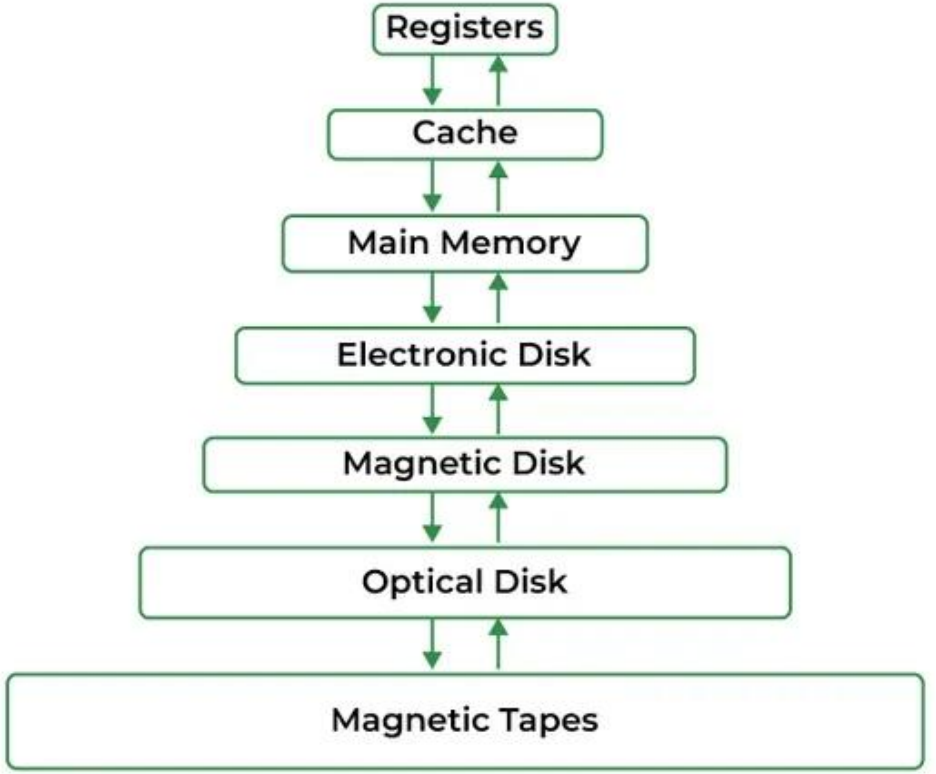
Hunter College, CUNY

Faye

czhang2@gradcenter.cuny.edu

Today's agenda

- **Review:**
- Main memory
- Virtual Memory
- **What's new today**



Review Reference: https://www.cs.hunter.cuny.edu/~sweiss and https://os-book.com/OS10/slide-dir/index.html

Important Due

1. Project Idea Submission:

- via email, czhang2@gradcenter.cuny.edu and cc others
- Please make the subject clear: AOS-idea-your name
- Due: March 9, 2026

2. Proposal Submission:

- Your final proposal should be written in a structured proposal format and submitted as a PDF file.
- Due: March 18, 2026

3. If you are not interested in doing a project

- Please send an email with the subject line: AOS-Final Exam Request
- Due: March 9, 2026

About the Proposal

- The proposal should clearly describe your project idea.
- It should explain the problem you want to study and why it is worth exploring.
- It should present your main goal and your planned approach.
- It should include the related work or background literature you have reviewed.
- It should also list any relevant engineering resources, implementations, or GitHub repositories (if you have).
- In addition, it should briefly describe the expected outcome and the overall scope of the project.
- Please make sure the proposal is realistic and feasible within the course timeline.
- A reference list should be included at the end.

Part 1 Review

- Main memory
- Reference： part 4 chapter9
- https://os-book.com/OS10/slide-dir/index.html

Part 2 Review

- Virtual Memory
- Reference: part 4 chap10
- https://os-book.com/OS10/slide-dir/index.html

What’s new:

- I. Breaking the "Memory Wall": High Bandwidth Memory (HBM) and the Reshaping of Compute-in-Memory Architecture
- II. Tiered Memory Driven by CXL Protocol and OS Transparent Page Scheduling
- III. Data Center-Scale Memory Disaggregation and Hardware Offloading of Cloud-Native Hypervisors
- IV. Physical Characteristics of Non-Volatile Memory (NVM) and the Blurring of OS Storage Boundaries
- V. Rebuilding Kernel Security from Language Design: Memory Safe Languages (MSL) and Rust's System-Level Integration
- VI. Hardware-Assisted Dynamic Memory Safety Defense: ARM MTE and Instruction-Set Level Verification
- VII. Concurrent Garbage Collection and Virtual Thread Scheduling in Modern Managed Runtimes
- VIII. From Engineering Patches to Mathematical Theorems: Formal Verification of Memory Allocators and Microkernels

INTRODUCTION

Modern Operating System Memory Management

Solving Legacy OS Bottlenecks with Cutting-Edge Evolutions

The Original Memory Crisis

Physical & Structural Limits

The Problem: For decades, OS theory relied on a static hierarchy (Cache -> DRAM -> Disk). With the explosion of AI workloads, CPU computing power vastly outpaced data movement capabilities (The "Memory Wall").
Furthermore, motherboards rigidly bind physical slots to specific CPUs, resulting in strict capacity limits and massive "stranded memory" waste in distributed data centers.

Software & Concurrency Flaws

The Problem: Traditional 4KB page granularity is too coarse, failing to catch internal Use-After-Free (UAF) and buffer overflow vulnerabilities inside legal memory allocations.
Additionally, rigid 1:1 kernel thread mappings consume excessive memory and trap execution contexts during blocking I/O, forming hard ceilings on system throughput.

HARDWARE EVOLUTION

1. Breaking the Memory Wall

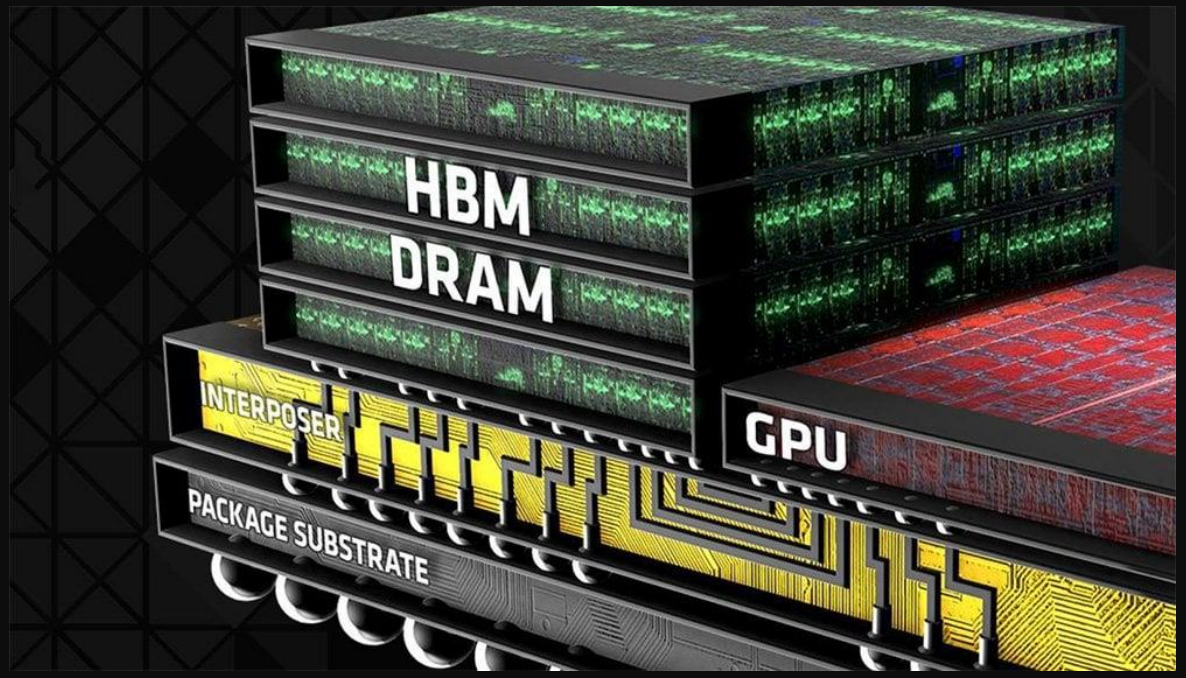
High Bandwidth Memory (HBM) & Compute-in-Memory (CIM)

HARDWARE EVOLUTION

High Bandwidth Memory

Original Problem: Moving data from physically separated DRAM to processor cores creates immense latency and starvation for data-intensive AI workloads.

The Solution: HBM uses 3D stacking and silicon interposers to package memory chips directly next to compute units.
This provides physical bandwidth that far exceeds traditional DDR memory (up to 819 GB/s), feeding massive neural networks with extreme efficiency.



HARDWARE EVOLUTION

New OS Challenges from HBM

Capacity Limits

Problem: 3D stacking restricts single-stack capacity to 16-32 GB, falling short of AI demands.

Fix: The OS must now manage extremely asymmetric memory topologies.

Thermal Walls

Problem: Proximity to the CPU makes HBM hyper-sensitive to heat, rendering DRAM states unpredictable.

Fix: Kernels must implement "Thermal-aware" page scheduling mechanisms.

Silent Corruption

Problem: Scaling increases physical error characteristics.

Fix: High-availability fault-tolerance mechanisms at the OS level using strict SEC-DED checks.

HARDWARE EVOLUTION

Compute-in-Memory (CIM)

The Data Movement Flaw

Original Problem: In traditional SIMD GPU architectures, a massive proportion of energy (e.g., ~700W TDP on high-end hardware) is entirely wasted merely moving data back and forth across buses.

The CIM Revolution

The Solution: CIM executes Multiply-Accumulate (MAC) operations directly inside the memory array. The OS no longer treats memory as simple storage, but maps computational instructions directly to it.

2. Tiered Memory Architecture

Breaking Physical Server Boundaries with the CXL Protocol

TIERED MEMORY & CXL

The New Memory Hierarchy

Architecture	Access Latency	Advantages	OS Strategy
Local DRAM	Extremely low (~70-100ns)	Balanced latency & bandwidth	Default hot data residency
CXL Expansion	Higher (~150-200ns)	Breaks motherboard slot limits	Target for cold data demotion
Stacked HBM	Extremely low (Direct packaging)	Extreme bandwidth & low energy	Deeply bound to explicit drivers

OS vs. Hardware Tiering

Hardware 2LM (Inclusive)

The Flaw: Directly mapping local DRAM as a hardware cache for slower CXL memory requires no OS intervention. However, total available capacity equals only the CXL capacity, completely wasting the premium local DRAM capacity.

OS-Level zNUMA (Software)

The Solution: The OS treats CXL memory as an independent "CPU-less" node. Capacities are linearly superimposed to maximize cloud resource efficiency, forcing the OS to manage complex page migrations.

Meta's TPP Mechanism

17%

Throughput Boost

Transparent Page Placement

Original Problem: When fast memory exhausts, processes stall entirely waiting for the kernel to scan and reclaim pages.

The Solution: TPP proactively demotes "cold" pages to CXL in the background, keeping local headroom open. If pages become hot, a 64-bit Bitmap system rapidly promotes them back, outperforming traditional AutoNUMA balancing.

3. Memory Disaggregation

Resolving Cloud Resource Fragmentation

CLOUD-NATIVE DISAGGREGATION

Memory Pooling (Azure Pond)

- **The Problem:** About 50% of Virtual Machines use less than half their allocated memory. Due to physical boundaries, this "Stranded Memory" is entirely wasted.
- **The Solution:** Using CXL switches to build high-speed shared memory pools across server racks.
- Machine learning models predict and dynamically allocate exactly how much local DRAM vs pooled CXL memory a VM requires, radically reducing hardware costs.

CLOUD-NATIVE DISAGGREGATION

Virtualization Offloading

Original Problem: Traditional Hypervisors permanently reside in host memory and consume CPU caches, resulting in a continuous "Hypervisor Tax" that restricts guest OS potential.

The Solution: Systems like AWS Nitro strip Hypervisor and SDN functions away from the main CPU, offloading them entirely to custom ASIC cards, returning near 100% of host resources.

Security Bonus: This decouplement leverages OS "Hot-unplug" features to rigidly partition physical memory, fundamentally neutralizing side-channel attacks targeting shared caches.

4. Non-Volatile Memory (NVM)

Blurring the Line Between RAM and Storage

NON-VOLATILE MEMORY

Bypassing Legacy Storage

The Traditional Storage Chain

Original Problem: Saving data enforces a serialized, slow chain: Read from disk -> Load into volatile DRAM -> Modify -> Serialize data -> Copy through kernel space -> Write back to disk.

App Direct Persistence

The Solution: NVM is both byte-addressable and persistent. In App Direct Mode, the OS completely bypasses the Page Cache, allowing standard C/C++ pointers to directly persist massive data structures instantly.

5. Rebuilding Kernel Security

Solving 50 Years of Legacy Memory Flaws

KERNEL SECURITY

The Rust Revolution

The CVE Crisis

Problem: Up to 70% of modern OS vulnerabilities originate from memory bugs (UAF, dangling pointers) inherent in C/C++'s manual memory management.

Ownership Rules

Fix: Rust's compiler guarantees every value has a single owner. When out of scope, the compiler automatically inserts release code, ensuring safety.

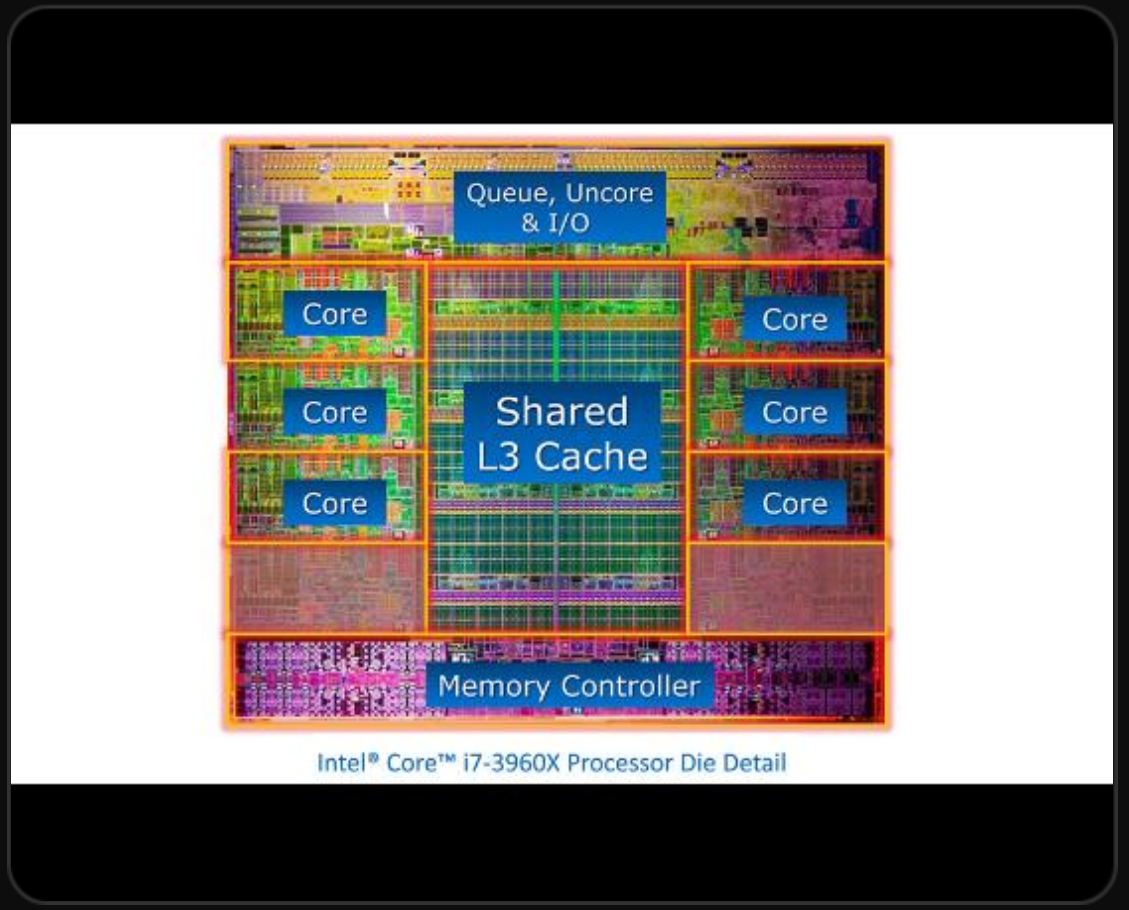
The Borrow Checker

Fix: "Shared immutable, mutable unshared". Static analysis mathematically prevents data races at compile time with zero runtime overhead.

HARDWARE-ASSISTED SECURITY

ARM Memory Tagging (MTE)

- **Original Problem:** OS page protections (4KB granularity) are too coarse and cannot detect out-of-bounds access between variables on the same page.
- **The Solution:** MTE partitions physical memory into 16-byte granules, each secretly assigned a random 4-bit hardware color tag.
- When the CPU executes access instructions, it high-frequency checks if the pointer's tag matches the granule's tag, intercepting unauthorized access instantly.



CONCURRENCY & RUNTIMES

Virtual Threads & Concurrent GC

Original Problem: 1:1 OS thread mapping traps execution during blocking I/O; older Garbage Collectors enforce massive "Stop-The-World" pauses on large heaps.

The Solution: Virtual threads use MN: multiplexing, decoupling from OS kernel limitations to support concurrency in the millions.

Concurrent GCs execute alongside application threads, capping pause times at sub-1ms, drastically improving tail latencies.

FORMAL VERIFICATION

The Ultimate Standard

"Memory management security no longer relies on billions of Fuzzing heuristics, but is instead guaranteed by impregnable mathematical theorems."

— Moving beyond defensive programming to mathematically proven microkernels (seL4) and allocators (StarMalloc)