Track B: Research & Design Projects

Track B is open-ended, design-oriented, and research-driven: the goal is not to finish a product but to grow as an independent researcher by combining at least two of architectural modeling, performance analysis, algorithm design, and systems software implementation. You will be successful if you show serious effort, meaningful progress with credible preliminary results (even partial or negative), and sound methodology with honest analysis. Keep your work reproducible and documented; by semester's end, publish a GitHub repository (code, run/plot scripts, README) and a report covering motivation, design or model, methodology, preliminary results, analysis, and lessons learned.

Suggested baselines and metrics (non-binding): Topic 1: CPU thread vs. simulated IOPS core (cycles/op, IOPS); Topic 2: DRAM-only ANN vs. tiered ANN (recall@k, qps, I/O amplification); Topic 3: single-tier vs. tiered DS (ops/s, p95 latency, migration overhead).

Below is the general grading rubric:

Category	Points	Description
Literature review	20	Comprehensive review of related work
Design clarity & innovation	45	Clear problem statement and solution description
Implementation / simulation rigor	45	Simulation/evaluation/analysis results
Quantitative analysis & insight	45	Correct metrics; throughput/latency interpretation
Presentation & report quality	45	Organization, figures, references, and clarity.

Topic 1: Microarchitectural Modeling and OS Interface of an IOPS-Optimized Domain-Specific Core

Driven by NVIDIA, the industry is developing a new breed of NVMe SSD, Storage-Next SSD, targeting tens of millions of random IOPS (≈10× today's NVMe SSDs). These devices are designed for TB/PB-scale AI workloads dominated by random access (e.g., vector search, retrieval-augmented generation, large-scale feature stores, low-latency data staging). As Storage-Next SSDs arrive, the primary bottleneck increasingly shifts to host-side software overheads (e.g., queue submission/completion handling, checksums, and small metadata transforms). To mitigate host-path overheads, NVIDIA is developing mechanisms that let GPUs interface directly and efficiently with Storage-Next NVMe SSDs.

This project explores a domain-specific "IOPS core" on the host (CPU/GPU side), a lightweight, latency-deterministic engine modeled at the microarchitectural level and paired with a minimal OS/driver interface, to reduce per-I/O software overhead. The goal is to quantify how microarchitecture + OS-interface co-design (command queues, batching, context management) improves IOPS and p95/p99 latency for AI-centric I/O patterns vs. a general-purpose CPU thread or an unaccelerated firmware path. Through simulation and analytical modeling, students investigate how instruction simplicity, pipeline width, and queue depth interact with software submission paths, and how those choices propagate to end-to-end I/O throughput and tail latency in AI workloads.

Focus on modeling and software; RTL design and kernel development are non-goals for this project.

Topic 2: Nearest-Neighbor Search for Tiered DRAM + Storage-Next SSDs

Large-scale machine-learning inference, recommendation, and semantic search increasingly rely on Approximate Nearest-Neighbor (ANN) search over billions of high-dimensional vectors. Historically, high-recall ANN systems are DRAM-resident; attempts to make them SSD-resident typically incur large latency penalties or sacrifice search quality (e.g., aggressive pruning/quantization) to meet I/O constraints. Meanwhile, next-generation SSDs can deliver millions of random reads per second with microsecond-scale latency, blurring the boundary between "memory" and "storage." To exploit high-IOPS SSDs, ANN systems must treat I/O as a first-class design dimension.

Students in this project will rethink ANN algorithms under the new assumption that SSD latency is not infinite and that the index may be partially resident on a high-IOPS device. They will quantify the trade-offs between recall, latency, and cost when memory is no longer the only performance limiter. This topic sits at the frontier of data-center system design where storage devices participate directly in analytics and AI pipelines.

Target a working tier-aware prototype or simulator; end-to-end production quality or perfect recall is not required.

Topic 3 — Concurrent Data Structures for Heterogeneous Memory Hierarchy

As DRAM scaling plateaus, system memory is evolving into multi-tier fabrics: DDR DRAM for hot data, CXL-attached compressed DRAM + NAND for warm data, and SSDs for cold persistence. Software must now reason about which bytes live in which medium and how concurrency interacts with variable latency.
Traditional concurrent indexes (e.g., hash maps, skip lists, and B-trees) assume uniform memory access and low-latency synchronization. In a heterogeneous hierarchy, these assumptions fail: cache-coherence overheads rise, locking delays depend on tier placement, and naïve memory allocation can bottleneck both capacity and throughput.

This project rethinks concurrent data structures for multi-tier memory with a focus on CXL main-memory compression as the intermediate tier. You will study placement policies that account for compression ratio and (de)compression latency, tier-aware locking to reduce cross-tier contention, and data migration strategies that bound pauses and preserve consistency. Beyond throughput, evaluate correctness under concurrency, tail latency, and cost-capacity trade-offs when DDR DRAM, compressed CXL memory, and SSD interact. The work brings together cache coherence/NUMA, memory compression, and advanced indexing & synchronization into a single research problem.

Emulate tiers and latencies in user space; OS/driver changes are out of scope.