KV Cache Granularity and Prefix Sharing on CPU

CMU 15-779 independent study project specification
AI-assisted artifact; learner name / Andrew ID not supplied

Problem. Paged K/V storage can avoid maximum-length reservation and share prompt prefixes, but page granularity trades tail waste against page-table size and partial-page copying. Which benefits remain when a prefix ends just before a page boundary? How do retained tensor slots differ from a preallocated physical pool and from full-process peak memory?

Research questions. RQ1: How do block size and shared-prefix alignment change retained K/V slots? RQ2: How many old token slots must be copied by dynamic concatenation versus partial-page copy on write? RQ3: What CPU overhead does this teaching implementation incur, and which GPU performance conclusions are unsupported?

Related work. PagedAttention (Kwon et al., SOSP 2023) provides paged attention and block sharing. SGLang (Zheng et al., NeurIPS 2024) adds structured programs and RadixAttention. This project adopts their published concepts but implements its own CPU allocator and benchmark harness; it does not claim a novel paging algorithm, radix tree, or reproduction of their GPU results.

Implementation. Use Python/PyTorch CPU float64 tensors, a physical K/V pool, per-sequence block tables, reference counts, prefix forks, copy on write and release. Compare uniform maximum-length static reservation, dynamically concatenated K/V, independent paged K/V and prefix-shared paged K/V. A common dense attention expression checks every decode output. No pretrained model, external dataset or GPU is required.

Evaluation. Eight requests per workload; five synthetic workloads with 0, 3, 31, 32 or 63 common-prefix tokens, ragged prompt suffixes, and 1–8 decode steps. Sweep blocks of 2, 4, 8, 16, 32 and 64 tokens, three fixed seeds and seven shuffled timed repetitions after warmup. Save environment, source hashes, raw timings, output errors, active slot bytes, pool bytes, copied old slots and page-table entries. Check random fork/append/release histories, atomic out-of-memory handling and analytical sharing/copy formulas. Timing includes cache setup, append, gather and attention; input generation and correctness assertions are excluded.

Deliverables and limits. Preserve code/data, a stage update, an MLSys 2025-format report with no more than eight body pages, and an 8–10 minute narrated demonstration. This cache study runs on CPU; subsequently authorized CUDA kernel checks are recorded separately. Retained slot counts are not RSS/peak memory; CPU gather is not a paged GPU kernel. Synthetic trace results do not establish real LLM throughput or quality.

Record status. This specification was prepared after an initial CPU prototype and is not represented as a past proposal submission. Identity, personal understanding, classroom participation and any formal submission remain unverified.