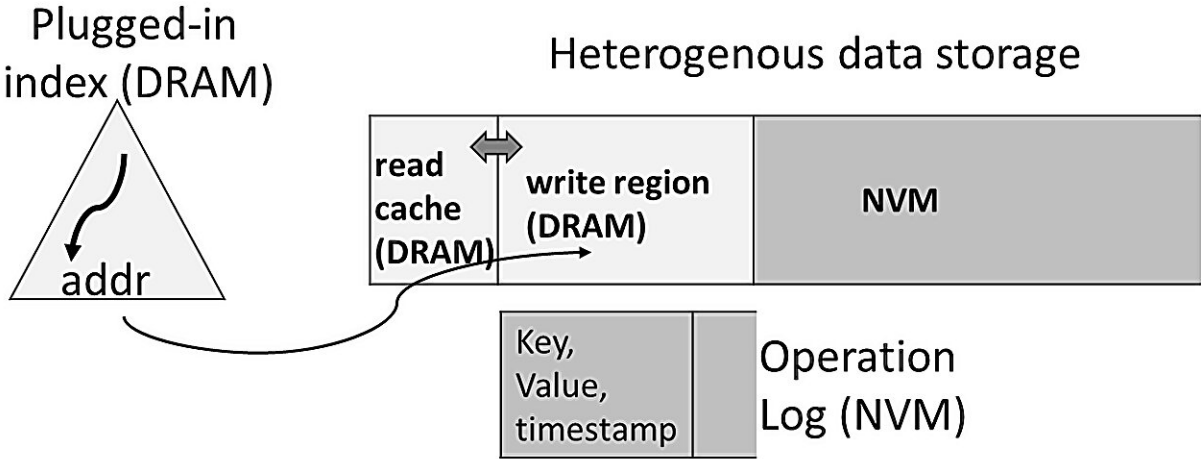# HeterMM: Applying in-DRAM index to heterogeneous memory-based key-value stores

August 30 2024

Overview of the framework. Credit: *Frontiers of Computer Science* (2024). DOI: 10.1007/s11704-024-3713-0

Emerging byte-addressable storage technologies, such as NVM, provide a more cost-effective and larger-capacity alternative to DRAM, presenting new opportunities to address the high cost, limited capacity, and volatility of in-memory key-value (KV) stores.

Numerous efforts have been dedicated to redesigning conventional structures on NVM. However, they were challenged by the substantial

engineering cost and increased complexity in order to be integrated into existing systems. Thus, a general framework to apply existing indexes to KV stores on NVM becomes necessary.

A research team led by Xuan Zhou proposes a general framework named HeterMM, for heterogeneous memory architecture consisting of DRAM and NVM. It is designed to fully leverage the superior performance of DRAM, and make the performance of the system as close to the in-DRAM one as possible. The research is published in the journal *Frontiers of Computer Science*.

Overview of the framework

The team emphasizes the importance of fully leveraging the superior performance of DRAM by holding the index and hot data in DRAM. Typically, NVM suffers from inferior performance compared to DRAM. Its specific accessing characteristics also necessitate special designs to maximize its performance.

The typical characteristics of NVM include its read-write asymmetry in terms of latency and bandwidth and poor random access performance compared to sequential accesses. In response, the research team offers a framework composed of a plugged-in in-DRAM index, a data storage mechanism on heterogeneous memory, and an operation log for failure recovery.

In particular, the index, which is the most frequently accessed and typically in a small unit and random order, is not friendly to NVM. Their data structures, which are usually optimized for DRAM, may not perform as effectively on NVM.

Newly written data in HeterMM resides in DRAM, and old data is flushed to NVM in batches. Each piece of data is allocated a logical

address upon its arrival, which remains the same unless the data is updated out-of-place.

The persistence of NVM can ensure the durability of data residing in it, while an operation log is applied to ensure the durability of data residing in DRAM. First, data in DRAM is updated in place, which could be regarded as early compaction and reduces data volume flushed to NVM. Second, data in NVM can be regarded as a checkpoint which can be used to cut off the operation log.

To optimize access to read-only data in NVM, the DRAM region is divided into a read cache and a write region, with the former holding frequently accessed data residing in NVM while the latter holding newly arrived data. They share the same space in DRAM and can be resized dynamically according to the workload.

Extensive experiments that combine HeterMM with different kinds of indexes, including CLHT, LFHT, and B+ tree, verify the efficiency of HeterMM. Specifically, HeterMM could outperform both the state-of-the-art index persist framework and state-of-the-art hybrid DRAM and NVM-based hash tables and B+ trees. This benefits from the fact that HeterMM holds the hot data in NVM which allows the read requests to be served by the DRAM without accessing NVM.

Citation: HeterMM: Applying in-DRAM index to heterogeneous memory-based key-value stores