Ribbon: High Performance Cache Line Flushing for Persistent Memory

Kai Wu, Ivy Peng[+], Jie Ren and Dong Li

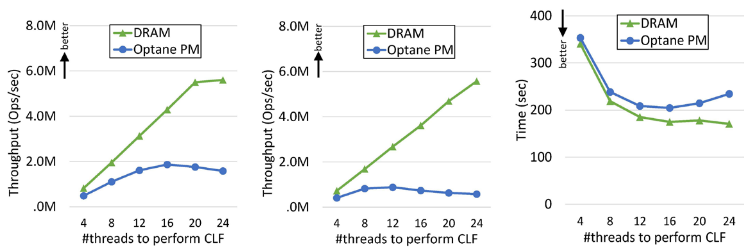
University of California Merced   Lawrence Livermore National Laboratory[+]

Motivation & Introduction

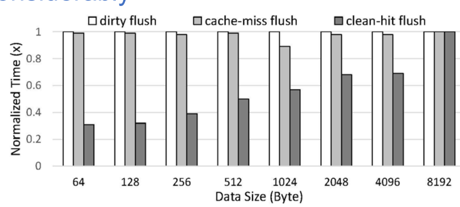
- Cache line flushing (CLF) is a fundamental building block for programming persistent memory (PM) to ensure crash consistency
  - CLF can reduce system throughput by 62% for database applications like Redis
- Most existing works focus on optimizing persistency semantics (e.g., skipping CLF or relaxing constraints on persist barriers)
  - Use different fault models or recovery mechanisms that are designed for specific application characteristics
- Unlike the previous works, we design a runtime system (Ribbon) to accelerate the CLF mechanism in PM-aware applications without impacting program correctness and crash recovery

Performance Analysis of CLF on Intel Optane DC PMM

- ✓ Concurrent CLF can create resource contention on the hardware buffer inside PM devices and memory controllers, which causes performance loss



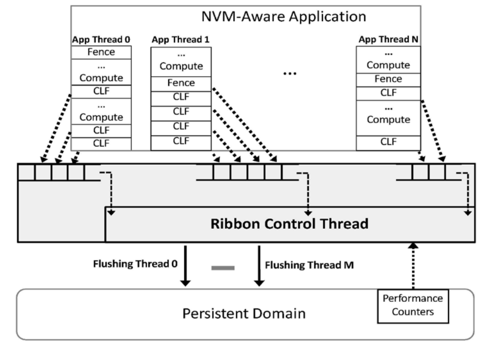
- ✓ The status of a cache line can impact the performance of CLF considerably



- ✓ Flushed cache lines may have low dirtiness, wasting memory bandwidth and decreasing the efficiency of CLF
  - Dirtiness = the fraction of dirty bytes in the cache line
  - Our evaluation of Redis with YCSB (loads and A-F) and TPC-C workloads shows that the average dirtiness of flushed cache lines is only 47%

Ribbon Design

Ribbon optimizes CLF mechanisms through decoupled concurrency control, proactive CLF, and cache line coalescing

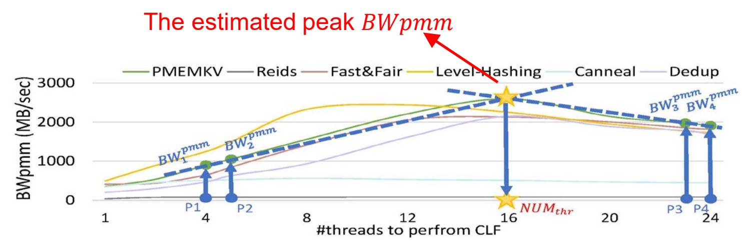


Decoupled Concurrency Control of CLF

- Decouple CLF from the application and adjusts the level of CLF concurrency (the number of threads performing CLF, $NUMthr$) adaptively. Ribbon throttles CLF concurrency if contention on PM devices is detected. Conversely, it ramps up CLF concurrency when PM bandwidth is underutilized

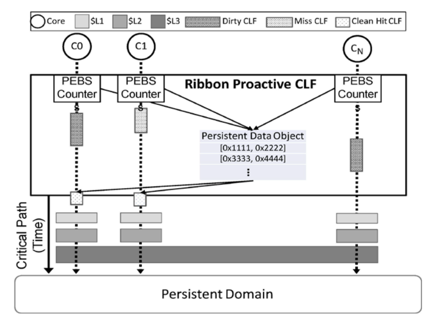
➢ Determine $NUMthr$

Run a helper thread to samples the write bandwidth to PM DIMMs ($BWpmm$) at four concurrency points (e.g., P1-P4) to estimate $NUMthr$ for achieving the peak $BWpmm$ periodically



Proactive Cache Line Flushing

- Leverage the precise address sampling capability in hardware performance counters to collect the virtual memory addresses of store instructions
- Use a background thread to proactively flush cache lines to transform cache lines to clean state
  - ✓ Reduce the time spent on the critical path of the application executing the CLF



Coalescing Cache Line Flushing

- Identify two reasons that account for the low dirtiness of flushed cache lines: unaligned CLF and uncoordinated CLF (Find more details in our paper)
- Addresse them through a new memory allocation that considers the semantic correlation between memory allocations

Evaluation on Intel Optane DC PMM

- Workloads: PMEMKV, Redis, Level-hashing, Fast&Fair (B+-tree), and Parsec
- Ribbon achieves up to 17.6% (9.3% on average) and up to 49.8% (20.2% on average) improvement of the overall application performance at four application threads (low thread-level papalism) and 24 application threads (high thread-level papalism), respectively

The full paper is published in the 29th International Conference on Parallel Architectures and Compilation Techniques (PACT'20)