A Comparison of Modern Memory Management Schemes in HPC

Sina Karimi
Boston University
Boston, MA, USA

Kurt Keville
MIT
Cambridge, MA, USA

Extended Abstract

Data-intensive applications are gaining more attention, and the requirements for main memory are increasing at least linearly. Unfortunately, DRAMs scaling is at its limit. To solve this problem, researchers have suggested new types of memory, such as high-bandwidth memories (HBM), nonvolatile memories (NVM), compute express link-based memory (CXL), and other emerging technologies. Each type of memory has its own distinct characteristics and advantages. Therefore, future computer systems are anticipated to have multiple types of memory.

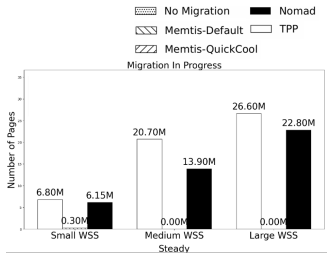
Typically, these systems consist of a fast-tier memory and a capacity-tier memory. Fast-tier memory exhibits better performance in terms of latency and bandwidth but has limited capacity, and programs cannot store all their data in it. Running programs on capacity-tier memory can cause performance degradation compared to fast-tier memory. Therefore, managing how data is allocated and migrating between these memory hierarchies becomes an important problem.

Optimizing page management within the operating system is crucial to exploit tiered memory systems. Policies for optimal page allocation and placement and page migration between different memories are necessary to guarantee minimum performance degradation while using them. To do so, we must make sure that pages that are being accessed frequently (called hot pages) reside in fast-tier memory and pages that are rarely being accessed (called cold pages) are in capacity-tier memory.

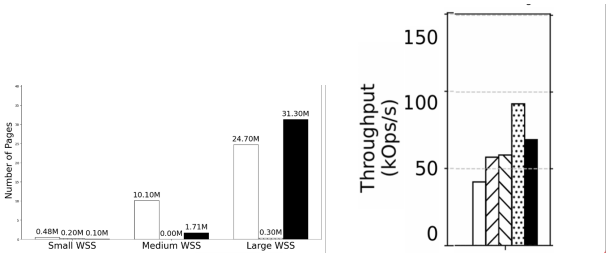
Many researchers have proposed different page migration mechanisms and policies, such as TPP, Nimble, Memtis, and NOMAD. These methods usually consist of a mechanism to detect the temperature of memory pages and decide whether to migrate the page to another memory tier. Typically, there are a few ways to detect the temperature of pages: One is causing a page fault in each access so we can track the page during the fault handling. Or we could use hardware-assisted sampling approaches such as PEBS for x86 architectures. These techniques can cause a large overhead when applied to a large memory region. Also, the proposed solutions for page migration assume that the size of hot data is always smaller than the fast tier memory. When the size of hot data exceeds the capacity of fast-tier memory, these proposed methods perform worse than not migrating data due to memory thrashing. NOMAD has addressed this issue using non-exclusive memory tiering and transactional page migration. However, they still suffer from the same problems as previous approaches. We propose a different approach for page temperature detection.

We propose finding regions of interest using the probability distribution of access frequency for pages of memory. Our proposed method profiles the initial distribution of access frequency in an application for ground truth. After that, we sample pages from memory to detect the temperature of pages and update the distribution model using the Bayesian inference technique. By doing this, we can find regions of interest in memory that make us more likely to find the information that we want. As an example, for thermal management problems, we can find hot regions in memory space. As for tiered memory systems, we can find regions that are more likely to contain hot pages that recently became cold. By doing this, we decrease the search space for temperature detection techniques. In so doing, we can reduce memory thrashing and migrate data that is not being accessed frequently to the capacity-tier memory. We are planning to test our method with the same microbenchmarks that are used in NOMAD. The microbenchmark tests the system with different sizes of hot data. By increasing the size of hot data over the size of fast-tier memory, we can test whether our proposed method is able to find regions of interest in the memory. Also, we are planning on doing a head-to-head comparison with other works such as TPP, Memtis, and NOMAD over graph benchmarks and machine learning inference applications.

Although many applications can benefit from migration policies, there are cases that migration can cause extra overhead for the system. In the adjacent figures, a benchmark with different Working Set Sizes (WSS, hat data) is running on the system.



Observations indicate that when the size of hat data exceeds the capacity of the fast memory tier, migration policies cause memory thrashing and degrade performance.



Memory thrashing can also occur in workloads with mostly uniform memory access. The figure to the right shows the performance of a Redis database with a workload generated by YCSB. Since there are no appropriate migration policies, no migration policy outperforms other proposed solutions.