Optane Memory As a Volatile Extension of DRAM and its Impact on Performance of WiredTiger

Alexandra Fedorova[1,2]

[1]MongoDB
[2]University of British Columbia

1 Introduction

Intel® Optane™ DC Persistent Memory can be used as a volatile extension of DRAM. This document compares two methods of using NVRAM as an extension of DRAM and evaluates their impact on WiredTiger's performance.

The first method we studiy is *Optane Memory Mode*. Memory Mode presents Optane NVRAM to the rest of the system as regular volatile memory, and uses DRAM transparently as its cache, with data transferred between the two in units of cache lines. Using the Memory Mode creates the illusion of a larger system memory, and this is an attractive design, because it permits using NVRAM as an extension to DRAM without requiring any code changes.

The second method we study is *NVCache* – our custom NVRAM block cache, implemented as a component inside WiredTiger. NVCache caches file blocks as they are read from or written to the file system. Like MM, NVCache does not use the persistent properties of Optane NVRAM.

2 WiredTiger NVCache

NVCache is a component of WiredTiger that caches disk blocks in NVRAM. A separate document describes its detailed architecture and motivates its features; here is provide a broad overview.

NVCache caches blocks read from or written to disk and sits underneath the WiredTiger page cache and next to the *block manager* – the code responsible for reading/writing the data from/to disk (see Fig. 1). **Read path:** If the DRAM cache cannot locate searched-for data, it issues a read to the block manager (1). The block manager checks if the block is present in NVCache (2), accessing it from NVCache if it is (3) and reading it from disk if it is not (4). It then transforms the block into a page, decrypting and decompressing it if needed, and hands it over to the DRAM cache (5). If the block is not present in NVCache, NVCache has the discretion to admit it after the block manager has read it from disk (6). NVCache stores the blocks in the same format as they are stored on disk: compressed/encrypted if those configuration options were chosen. This is a feature, as storing compressed blocks increases NVRAM effective capacity.

Write path: The write path is not symmetrical to the read path, because WiredTiger does not modify disk blocks in place. Updates are written into in-memory specific data structures, and then formatted into blocks and written back to disk during a process called *reconciliation*. Reconciliation may occur when the DRAM cache evicts pages or as part of a database checkpoint. Reconciliation always writes a new page (7), which the block manager turns into a new block. When the block manager writes a new block (8), it notifies NVCache (9); NVCache has the discretion to admit it. Obsolete blocks are eventually freed, at which time the block manager instructs NVCache to invalidate cached copies of the freed blocks (10).

When NVCache runs out of space it cannot admit new blocks. *Eviction* is needed to purge blocks less likely to be used in order to make space for new ones. We use a simple LFRU eviction policy [4]. During eviction it targets blocks that were not reused within a fixed time window and evicts the least frequently used among those. Tracking of the LRU and LFU blocks is approximated so that there is no need to maintain lists. There is an eviction thread that wakes up once a second and scans the cache for eviction candidates.

A key feature of NVCache distinguishing it from similar existing systems is its admission policy – the policy deciding when to admit blocks into cache. In our prior work we discovered that the presence of writes to Optane NVRAM disproportionately affects the throughput of reads. In other words, writing lots of data into Optane NVRAM will cause reads to be very slow. Whenever NVCache admits a new block or evicts an existing one it generates writes into the Optane memory: and producing too many of those writes will make reads, i.e., cache lookups, very slow. To prevent this situation from occurring, NVCache throttles the amount of cache admissions and the rate of eviction in accordance

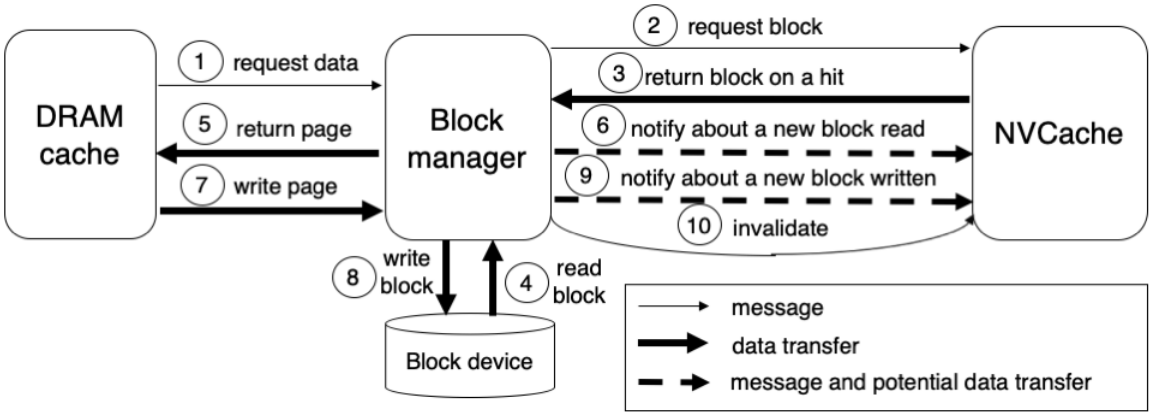


Figure 1: Interaction of NVCache with the rest of the storage engine.

aligning them with the frequency of lookup. Detailed design of this policy, experiments and motivating data are provided in our paper describing the NVCache.

2.1 NVCache vs. MM

NVCache and Optane Memory Mode differ in the following ways:

1. NVCache is a software component that is part of WiredTiger. Using MM is transparent and does not require any additional source code.

2. MM provides a transparent extension of system volatile memory, and it can be used to enable a larger WiredTiger page cache or to provide additional space for the kernel buffer cache expansion. On the other hand, NVCache caches disk blocks only, and currently cannot be used to host WiredTiger page cache.

3 Experimental methodology

3.0.1 Experimental system

System: Our system is a Lenovo ThinkSystem SR360 built with two Intel Xeon Gold 5218 processors, each having 16 hyper-threaded cores.

Memory: There are two Optane NVRAM modules, 126GB each, for a total capacity of 252GB. The modules are placed in separate sockets as per manufacturer recommendation.

Memory Mode can be enabled only in specific hardware configurations ([1], Table 17). We were able to successfully configure MM such that each NVDIMM was "paired" with a DRAM DIMM, meaning that it must be placed in the unused slot of the same channel of the same iMC (integrated memory controller) as the NVDIMM. Using additional DRAM DIMMs that were not paired with NVDIMMs produced configuration errors on our system, so we could only use the configuration with two NVDIMMs and two DRAM DIMMs. Our DRAM DIMMs were 16GB in size, so that restricted us to a configuration with 32GB of DRAM. Fortunately, MM could be configured to use all or part of the NVRAM, so we were able to vary the amount of NVRAM in the experiments.

Disk: We use Intel Optane P4800X SSD, built with the same physical media as NVRAM DIMMs, but packaged as an SSD on the PCIe bus. This SSD provides up to 2.5GB/s sequential read bandwidth and up to 2.2GB/s sequential write bandwidth.

3.1 Workoads

We evaluate NVCache using the YCSB benchmarks [2, 3], whose parameters are shown in Table 1[1].

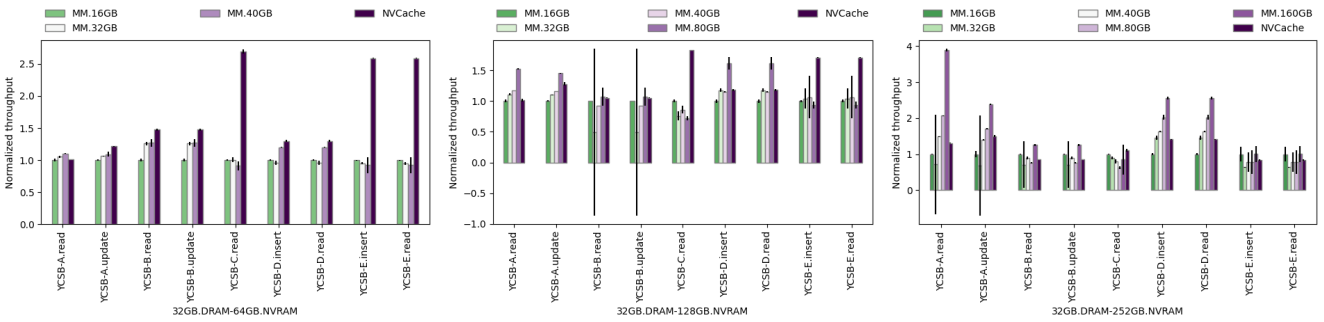
Workload	Op mix, threads	Dataset
YCSB-A	50% read, 50% update, 20	130GB
YCSB-B	50% read, 50% update, 20	194GB
YCSB-C	100% read, 20	259GB
YCSB-D	95% read, 5% insert, 100	219GB
YCSB-E	95% scan, 5% insert, 20	210GB

Table 1: YCSB characteristics

3.1.1 Experimental goals

Memory mode provides a transparent extension to system memory. WiredTiger can benefit from additional system memory in two ways: (1) By using a larger page cache, (2) by keep-

[1]We did not include YCSB-F: it is modify-heavy, and modify operations in our storage engine were designed to trade performance for smaller cache footprint and smaller log records. Therefore, the overall throughput in modify operations was very low and insensitive to memory configurations.



(a) 64GB NVRAM (b) 128GB NVRAM (c) 252GB NVRAM

Figure 2: YCSB throughput in memory mode and NVCache. Each bar labeled *MM*-N*GB* shows a configuration in memory mode with varying sizes of the engine's page cache. The NVCache configuration uses the 16GB DRAM page cache and the NVRAM block cache size of 64GB, 128GB and 180GB respectively.

ing the page cache of a modest size so that the kernel buffer cache, which caches the engine's file blocks, can expand into a larger system memory. To that end, we vary two experimental parameters: the size of the available NVRAM and the size of the WiredTiger's page cache. For the configuration using NVCache, we only vary the size of the NVRAM block cache, but keep the WiredTiger's page cache fixed, because the amount of DRAM used in the NVCache configuration is also fixed.

4 Experimental results

Figure 2 shows the throughput of YCSB with 32GB of DRAM and 64GB, 128GB and 252GB of NVRAM. The NVCache configuration uses a 16GB WiredTiger page cache, because the amount of system RAM is fixed to 32GB. Memory Mode allows us to expand the page cache, because the amount of system memory grows with the available NVRAM, so we use multiple page cache sizes in MM configurations. The configurations using Memory Mode and a 16GB page cache are similar to NVCache: we keep the WiredTiger page cache the same size as in the NVCache configuration and let the kernel buffer cache occupy the available system memory.

We make the following conclusions: when the amount of available NVRAM is smaller than the size of the dataset (e.g., Figure 2(a)), NVCache provides superior performance to MM. When the amount of NVRAM approaches or exceeds the dataset size, MM provides similar or superior performance to NVCache. Unfortunately, as the amount of NVRAM grows, we also observe extremely high standard deviation in Memory Mode configurations, while the standard deviation with NVCache always remains low. We have run hundreds of experiments with NVCache and never observed a high variability in performance. We hypothesize that performance variability in MM occurs when there is a high burst of writes sent to the NVRAM. NVCache controls the rate of writes, so this issue does not occur. We emphasize that this is a hypothesis, and we don't (yet) have any data to back it up.

5 Conclusion

In this document we compared the performance of two systems that use Intel® Optane™ DC Persistent Memory as a volatile extension of DRAM: Memory Mode – a transparent hardware mechanism, and NVCache – a software component of WiredTiger that uses Optane memory for caching disk blocks. We conclude that while Memory Mode is an attractive solution requiring no software modifications, it performs worse than NVCache when the dataset size substantially exceeds the amount of available NVRAM. Although it performs comparably to or better than NVCache when the dataset fits into NVRAM, it also produces high performance variability, whose causes are not yet known.

References

[1] Intel® Server Board S2600WF Product Family. Technical Product Specification. https://www.intel.com/content/dam/support/us/en/documents/server-products/server-boards/S2600WF_TPS.pdf, 2021.

[2] Yahoo! Cloud Serving Benchmark, Git Repo. https://github.com/brianfrankcooper/YCSB, 2021.

[3] Brian F. Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears. benchmarking cloud serving systems with ycsb.

[4] Donghee Lee, Jongmoo Choi, Jong-Hun Kim, Sam H. Noh, Sang Lyul Min, Yookun Cho, and Chong Sang Kim.

On the Existence of a Spectrum of Policies That Subsumes the Least Recently Used (LRU) and Least Frequently Used (LFU) Policies. In *Proceedings of the 1999 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '99, page 134–143, New York, NY, USA, 1999. Association for Computing Machinery.