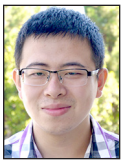
An Empirical Guide to the Behavior and Use of Scalable Persistent Memory

JIAN YANG, JUNO KIM, MORTEZA HOSEINZADEH, JOSEPH IZRAELEVITZ, AND STEVEN SWANSON



Jian Yang received his PhD in computer science at the University of California, San Diego, in 2019. He is currently working on host networking at Google. jian@ia32.me

Juno Kim is a PhD student at the University of California, San Diego, in the Non-Volatile System Laboratory. His research interests lie in the field of storage systems optimized for persistent memory. His advisor is Professor Steven Swanson. juno@eng.ucsd.edu

Morteza Hoseinzadeh is a PhD candidate at the University of California, San Diego, in the Non-Volatile System Laboratory. His research interests include software solutions for safe persistent memory programming, formal verification of persistent data structures, and system programming with a focus on non-volatile memories. He is advised by Professor Steven Swanson. mhoseinzadeh@cs.ucsd.edu

Joseph (Joe) Izraelevitz is an assistant professor at the University of Colorado, Boulder. His interests lie at the intersection of shared memory systems and non-volatile memory. joseph.izraelevitz@colorado.edu

Steven Swanson is a professor in the Department of Computer Science and Engineering at the University of California, San Diego, and the director of the Non-Volatile Systems Laboratory. His research interests include software and architecture of non-volatile, solid-state memories. swanson@eng.ucsd.edu

Researchers have been anticipating the arrival of commercially available, scalable non-volatile main memory technologies that provide "byte-addressable" storage that survives power outages. With the arrival of Intel's Optane DC Persistent Memory Module, we can start to understand the real capabilities and characteristics of these memories and start designing systems to fully leverage them. We experimented with an Intel system complete with Optane and have learned how to get the most performance out of this new technology. Our testing has helped us understand the hidden complexities of Intel's new devices.

Optane Memory Architecture

Intel's Optane DC Persistent Memory Module (which we refer to as the Optane DIMM) is the first scalable, commercially available non-volatile DIMM (NVDIMM). Compared to existing storage devices, including Optane SSDs that connect to an external interface such as PCIe, the Optane DIMM has lower latency, higher read bandwidth, and presents a memory address-based interface. Compared to DRAM, it has higher density and persistence.

Like traditional DRAM DIMMs, the Optane DIMM sits on the memory bus, and connects to the processor's integrated memory controller (iMC) (Figure 1a). Intel's Cascade Lake processors are the first CPUs to support the Optane DIMM. Each processor die has two iMCs, and each iMC supports three channels. Therefore, in total, a processor die can support six Optane DIMMs across its two iMCs.

To ensure persistence, the iMC sits within the *asynchronous DRAM refresh* (ADR) domain—Intel's ADR feature ensures that CPU stores that reach the ADR domain will survive a power failure (i.e., will be flushed to the NVDIMM within the hold-up time) [4]. The iMC maintains read and write pending queues (RPQs and WPQs) for each of the Optane DIMMs (Figure 1b), and the ADR domain includes WPQs. Once data reaches the WPQs, the ADR ensures that it will survive power loss. The ADR domain does not include the processor caches, so stores are only persistent once they reach the WPQs. Stores are pulled from the WPQ and sent to the Optane DIMM in cache-line (64-byte) granularity.

Memory accesses to the NVDIMM (Figure 1b) arrive first at the on-DIMM controller (the *Optane controller*), which coordinates access to the Optane media. Similar to SSDs, the Optane DIMM performs an internal address translation and maintains an *address indirection table* (AIT) for this translation [1].

After address translation, the actual access to storage media occurs. As the Optane physical media access granularity is 256 bytes (an *Optane block*), the Optane controller translates smaller requests into larger 256-byte accesses, causing write amplification where small stores become read-modify-write operations. The Optane controller has a small buffer (the *Optane buffer*) to merge adjacent writes.

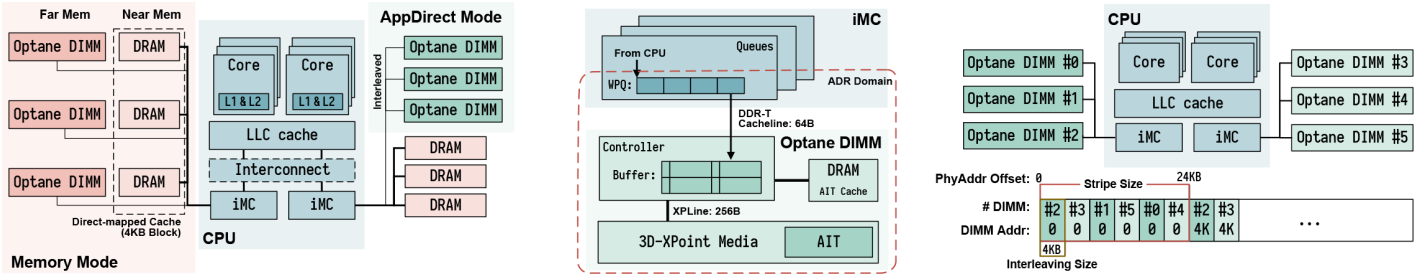


Figure 1: Overview of (a) Optane platform, (b) Optane DIMM, and (c) how Optane memories interleave. Optane DIMMs can either be a volatile far memory with a DRAM cache (Memory mode) or persistent memory (App Direct mode).

Operation Modes

Optane DIMMs can operate in two modes (Figure 1a): Memory and App Direct.

Memory mode uses Optane to expand main memory capacity without persistence. It combines an Optane DIMM with a conventional DRAM DIMM that serves as a cache for the NVDIMM. The CPU and operating system simply see the Optane DIMM as a larger (volatile) portion of main memory.

App Direct mode provides persistence and does not use a DRAM cache. The Optane DIMM appears as a separate, persistent memory device.

In both modes, Optane memory can be (optionally) interleaved across channels and DIMMs (Figure 1c). On our platform, the only supported interleaving size is 4 KB. With six DIMMs, an access larger than 24 KB will access all DIMMs.

Instruction Support

In App Direct mode, applications and file systems can access the Optane DIMMs with load and store instructions. Applications modify the Optane DIMM's content using store instructions, and those stores will eventually become persistent. The cache hierarchy, however, can reorder stores, making recovery challenging [3]. The current Intel ISA provides `clflush` and `clflushopt` instructions to flush cache lines back to memory, and `clwb` can write back (but not evict) cache lines. Alternatively, non-temporal stores (`ntstore`) bypass the caches and write directly to memory. All these instructions are non-blocking, so a program must issue an `sfence` to ensure that a previous flush, write back, or non-temporal store is complete and persistent.

Performance Characterization

We find that Optane's performance characteristics are surprising in many ways, and more complex than the common assumption that Optane behaves like slightly slower DRAM.

LATTester

Characterizing Optane memory is challenging for two reasons. First, the underlying technology has major differences from DRAM but publicly available documentation is scarce. Secondly, existing tools measure memory performance primarily as a function of locality and access size, but we have found that Optane performance also depends strongly on memory interleaving and concurrency.

Consequently, we built a microbenchmark toolkit, *LATTester*. To accurately measure the CPU cycle count and minimize the impact from the virtual memory system, LATTester runs as a dummy file system in the kernel and accesses pre-populated (i.e., no page-faults) kernel virtual addresses. LATTester also pins the kernel threads to fixed CPU cores and disables IRQ and cache prefetcher. In addition to latency and bandwidth measurements, LATTester collects a large set of hardware counters from the CPU and NVDIMM.

Our investigation of Optane memory behavior proceeded in two phases. First, we performed a broad, systematic "sweep" over Optane configuration parameters, including access patterns (random vs. sequential), operations (loads, stores, fences, etc.), access size, stride size, power budget, NUMA configuration, and interleaving. Using this data, we designed targeted experiments to investigate anomalies. Across all our tests, we collected over ten thousand data points. The program and data set are available at https://github.com/NVSL/OptaneStudy, while the analysis was published as conference proceedings [5] and a longer technical report [2].

System Description

We performed our experiments on a dual-socket evaluation platform provided by Intel Corporation. The CPUs are 24-core Cascade Lake engineering samples with a similar spec as the previous-generation Xeon Platinum 8160. Each CPU has two iMCs and six memory channels (three channels per iMC). A 32-GB Micron DDR4 DIMM and a 256-GB Intel Optane DIMM

are attached to each of the memory channels. Thus the system has 384 GB (2 socket × 6 channel × 32 GB/DIMM) of DRAM, and 3 TB (2 socket × 6 channel × 256 GB/DIMM) of Optane memory. Our machine runs Fedora 27 with kernel version 4.13.0 built from source.

Experimental Configurations

As the Optane DIMM is both persistent and byte-addressable, it can fill the role of either a main memory device (i.e., replacing DRAM) or a persistent device (i.e., replacing disk). In our paper, we focus on the persistent usage.

Our baseline (referred to as *Optane*) exposes six Optane DIMMs from the same socket as a single interleaved namespace (leaving the other CPU socket idle). In our experiments, we used local accesses (i.e., from the same NUMA node) as the baseline to compare with other configurations, such as access to Optane memory on the remote socket (*Optane-Remote*) or DRAM on the local or remote socket (*DRAM* and *DRAM-Remote*). To better understand the raw performance of Optane memory without interleaving, we also create a namespace consisting of a single Optane DIMM and denote it as *Optane-NI*.

Typical Latency

Read and write latencies are key memory technology parameters. We measured read latency by timing the average latency for individual 8-byte load instructions to sequential and random memory addresses. To eliminate caching and queueing effects, we empty the CPU pipeline and issue a memory fence (`mfence`) between measurements (`mfence` serves the purpose of serialization for reading timestamps). For writes, we load the cache line into the cache and then measure the latency of one of two instruction sequences: a 64-bit store, a `clwb`, and an `mfence`; or an `ntstore` and an `mfence`.

Our results (Figure 2) show the read latency as seen by software for Optane is 2×–3× higher than DRAM. We believe most of this difference is due to Optane's longer media latency. Optane memory is also more pattern-dependent than DRAM. The random-vs-sequential gap is 20% for DRAM but 80% for Optane memory, and this gap is a consequence of the Optane buffer. For stores, the instructions commit once the data reaches the ADR at the iMC, so both DRAM and Optane show a similar latency.

Bandwidth

Detailed bandwidth measurements are useful to application designers as they provide insight into how a memory technology will impact overall system throughput. Figure 3 shows the bandwidth achieved at different thread counts for sequential accesses with 256-byte access granularity. We show loads and

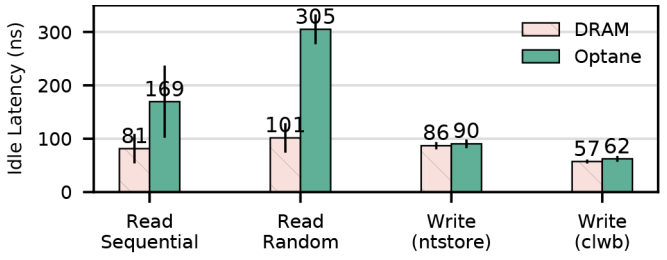


Figure 2: Typical latency. Random and sequential read latency, as well as write latency with `clwb` and `ntstore` instructions. Error bars show one standard deviation.

stores (`Write(ntstore)`), as well as cached writes with flushes (`Write(clwb)`). All experiments use AVX-512 instructions. The left-most graph plots performance for interleaved DRAM, while the center and right-most graphs plot performance for interleaved and non-interleaved Optane. In the non-interleaved measurements all accesses hit a single DIMM.

Figure 4 shows how performance varies with access size. The graphs plot aggregate bandwidth for random accesses of a given size. We use the best-performing thread count for each curve (given as "`<load` thread count`>` / `<ntstore` thread count`>` / `<store + clwb` thread count`>`" in the figure). The data shows that DRAM bandwidth is both higher than Optane and scales predictably (and monotonically) with thread count until it saturates the DRAM's bandwidth, which is mostly independent of access size.

The results for Optane are wildly different. First, for a single DIMM, the maximal read bandwidth is 2.9× the maximal write bandwidth (6.6 GB/s and 2.3 GB/s, respectively), where DRAM has a smaller gap (1.3×) between read and write bandwidth. Second, with the exception of interleaved reads, Optane performance is non-monotonic with increasing thread count. For the non-interleaved (i.e., single-DIMM) cases, performance peaks at between one and four threads and then tails off. Interleaving pushes the peak to 12 threads for `store + clwb`. Third, Optane bandwidth for random accesses under 256 bytes is poor.

Interleaving (which spreads accesses across all six local DIMMs) adds further complexity: Figure 3 (center) and Figure 4 (center) measure bandwidth across six interleaved NVDIMMs. Interleaving improves peak read and write bandwidth by 5.8× and 5.6×, respectively. These speedups match the number of DIMMs in the system and highlight the per-DIMM bandwidth limitations of Optane. The most striking feature of the graph is a dip in performance at 4 KB—this dip is an emergent effect caused by contention at the iMC, and it is maximized when threads perform random accesses close to the interleaving size. We return to this phenomenon later.

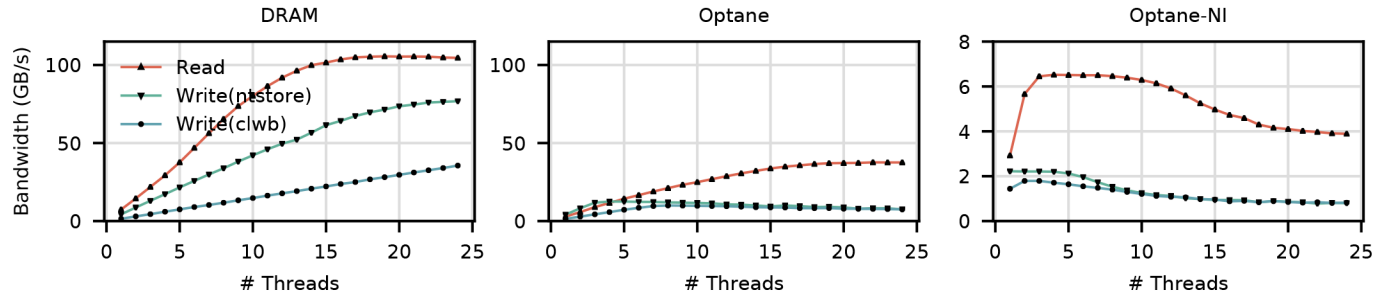


Figure 3: Bandwidth vs. thread count. Maximal bandwidth as thread count increases on local DRAM, non-interleaved, and interleaved Optane memory. All threads use a 256-byte access size.

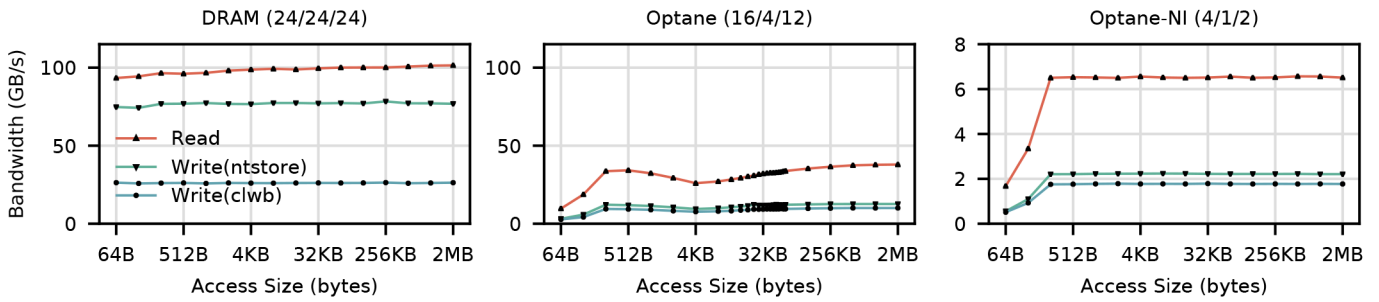


Figure 4: Bandwidth over access size. Maximal bandwidth over different access sizes on local DRAM, interleaved, and non-interleaved Optane memory. Graph titles include the number of threads used in each experiment (Read/Write (`ntstore`) / Write (`clwb`)).

Best Practices for Optane DIMMs

There are many differences between Optane and conventional storage and memory. These differences mean that existing intuitions about optimizing software do not apply directly to Optane. We distill our experiments into a set of four principles for building Optane-based systems.

1. Avoid random accesses smaller than 256 bytes.
2. Use non-temporal stores when possible for large transfers, and control cache evictions.
3. Limit the number of concurrent threads accessing an Optane DIMM.
4. Avoid NUMA accesses (especially read-modify-write sequences).

Avoid Small Random Accesses

Internally, Optane DIMMs update Optane contents at a 256-byte granularity. This granularity means that smaller updates are inefficient since they incur write amplification. The less locality the accesses exhibit, the more severe the performance impact.

To characterize the impact of small stores, we performed two experiments. First, we quantify the inefficiency of small stores using a metric we have found useful in our study of Optane DIMMs. The *Effective Write Ratio (EWR)* is the ratio of bytes issued by the iMC divided by the number of bytes actually written to the Optane media (as measured by the DIMM's hardware counters). EWR is the inverse of write amplification. EWR values below one indicate the Optane DIMM is operating inefficiently since it is writing more data internally than the application requested. Figure 5 plots the strong correlation between EWR and device bandwidth for a single DIMM for all measurements in our sweep of Optane performance. Maximizing EWR is a good way to maximize bandwidth.

Notably, 256-byte updates are EWR efficient, even though the iMC breaks them into 64 byte (cache-line sized) accesses to the DIMM—the Optane buffer is responsible for buffering and combining 64-byte accesses into 256-byte internal writes. As a consequence, Optane DIMMs can efficiently handle small stores, *if they exhibit sufficient locality*. To understand how much locality is sufficient, we crafted an experiment to measure the size of the Optane buffer. First, we allocate a contiguous region of N Optane blocks. During each "round" of the experiment, we first update the first half (128 bytes) of each Optane block. Then we update the second half of each Optane block. We measured the EWR for each round. Figure 6 shows the results. Below $N = 64$ (a region size of 16 KB), the EWR is near unity, suggesting the accesses to the second halves are hitting in the Optane buffer. Above 16 KB, write amplification jumps, indicating a sharp rise in the miss rate, implying the Optane buffer is approximately 16 KB in size. Together these results provide specific guidance for maximizing Optane store efficiency: avoid small stores or, alternatively, limit the working set to 16 KB per Optane DIMM.

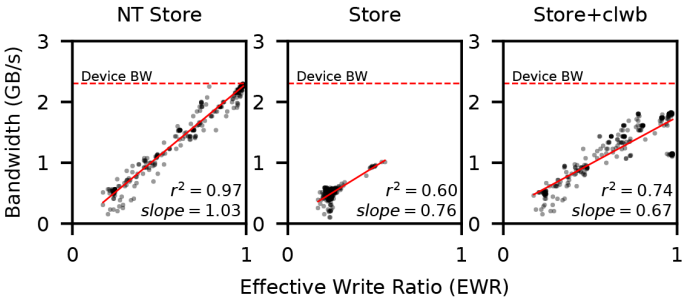


Figure 5: Relationship between EWR and throughput on a single DIMM. Each dot represents an experiment with different access size, thread count, and power budget configurations. Note the correlation between the metrics.

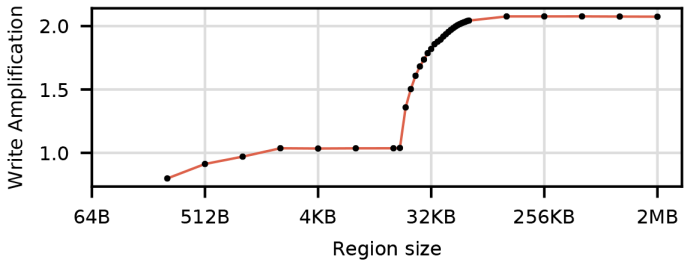


Figure 6: Optane buffer capacity. The Optane DIMM can use the Optane buffer to coalesce writes spread across 16 KB.

Use Non-Temporal Stores for Large Writes

When writing to persistent memory, programmers have several options, each with performance implications. After a regular `store`, programmers can either evict (`clflush`, `clflushopt`) or write back (`clwb`) the cache line. Alternatively, an `ntstore` writes directly to memory, bypassing the cache hierarchy. For all these instructions, a subsequent `sfence` ensures their effects are persistent.

In Figure 7, we compare bandwidth (left) and latency (right) for sequential accesses using AVX-512 stores with three different instruction sequences: `ntstore`, `store + clwb`, and `store` all followed by a `sfence`. Our bandwidth test used six threads since it gives good results for all instructions. The data show that flushing after each 64-byte store improves the bandwidth for accesses larger than 64 bytes. Letting the cache naturally evict cache lines adds nondeterminism to the stream that reaches the Optane DIMM, whereas proactively cleaning the cache ensures that accesses remain sequential. The EWR correlates: adding flushes increases EWR from 0.26 to 0.98.

The data also shows that non-temporal stores have lower latency for accesses over 512 bytes, and the highest bandwidth for accesses over 256 bytes. Here, the performance is due to the fact that a store must load the cache line into the CPU's local cache before execution, thereby using up some of the Optane DIMMs bandwidth. As ntstores bypass the cache, they avoid this extraneous read.

Limit the Number of Concurrent Threads Accessing an Optane DIMM

Systems should minimize the number of threads targeting a single DIMM simultaneously. We have identified two distinct mechanisms that contribute to this effect.

Contention in the Optane Buffer

Contention among threads for space in the Optane buffer will lead to increased evictions, driving down EWR. For example, using eight threads issuing sequential non-temporal stores achieves an EWR of 0.62 and 69% bandwidth compared to a single thread, which has an EWR of 0.98. Figure 3 (right) shows this contention effect in action.

Contention in the iMC

The limited queue capacity in the iMC also hurts performance when multiple cores target a single DIMM. On our platform, the WPQ buffer queues up to 256-byte data issued from a single thread. Since Optane DIMMs are slow, they drain the WPQ slowly, which leads to head-of-line blocking effects.

Figure 4 (center) shows an example of this phenomenon: Optane bandwidth falls drastically when doing random 4 KB accesses across interleaved Optane DIMMs. Due to the random access pattern, periodically all threads will end up colliding on a single DIMM, starving some threads. Thread starvation occurs more often as the access size grows, reaching maximum degradation at the interleaving size (4 KB). For accesses larger than the interleaving size, each core starts spreading their accesses across multiple DIMMs, evening out the load. The write data also show small peaks at 24 KB and 48 KB where accesses are perfectly distributed across the six DIMMs. This degradation effect will occur whenever 4 KB accesses are distributed nonuniformly across the DIMMs.

Avoid Mixed or Multithreaded Accesses to Remote NUMA Nodes

NUMA effects for Optane are much larger than for DRAM, so designers should avoid cross-socket traffic. The cost is especially steep for accesses that mix loads and stores or include multiple threads. Between local and remote Optane memory, the read latency difference is 1.79× (sequential) and 1.20× (random). For writes, remote Optane's latency is 2.53× (`ntstore`) and 1.68× higher compared to local. For bandwidth, remote Optane can achieve 59.2% and 61.7% of local read and write bandwidth at optimal thread count (16 for local read, 10 for remote read, and 4 for local and remote write).

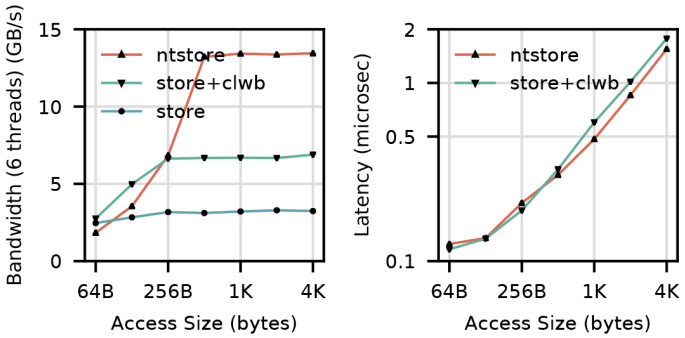


Figure 7: Persistence instruction performance. Flush instructions have lower latency for small accesses, but `ntstore` has better latency for larger accesses. Using `ntstore` avoids an additional read from memory, resulting in higher bandwidth.

The performance degradation ratio above is similar to remote DRAM to local DRAM. However, the bandwidth of Optane memory is drastically degraded when either the thread count increases or the workload is read/write mixed. Based on the results from our systematic sweep, the bandwidth gap between local and remote Optane memory for the same workload can be over 30×, while the gap between local and remote DRAM is, at max, only 3.3×.

Conclusion

Our guidelines provide a starting point for building and tuning Optane-based systems. By necessity, they reflect the idiosyncrasies of a particular implementation of a particular persistent memory technology, and it is natural to question how applicable the guidelines will be both to other memory technologies and to future versions of Intel's Optane memory. Ultimately, it is unclear how persistent memory will evolve. Several of our guidelines are the direct product of architectural characteristics of the current Optane incarnation. The size of the Optane buffer and iMC's WPQ might change in future implementations, which would limit the importance of minimizing concurrent threads and reduce the importance of the write granularity. However, expanding these structures would increase the energy reserves required to drain the ADR during a power failure.

The broadest contribution of our analysis and guidelines is that they provide a road map to potential performance problems that might arise in future persistent memories and the systems that use them. Our analysis shows how and why issues like interleaving, buffering, instruction choice, concurrency, and cross-core interference can affect performance. If future technologies are not subject to precisely the same performance pathologies as Optane, they may be subject to similar ones.

Acknowledgments

We thank Subramanya R. Dulloor, Sanjay K. Kumar, and Karthik B. Sukumar from Intel for their support in accessing the test platform. We would also like to thank Jiawei Gao, Xiao Liu, Amirsaman Memaripour, Yun Joon Soh, Zixuan Wang, Yi Xu, Lu Zhang, and Jishen Zhao for technical suggestions. This work was supported in part by CRISP, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.

References

[1] B. Beeler, "Intel Optane DC Persistent Memory Module (PMM)": https://www.storagereview.com/intel_optane_dc_persistent_memory_module_pmm.

[2] J. Izraelevitz, J. Yang, L. Zhang, J. Kim, X. Liu, A. Memaripour, Y. Joon Soh, Z. Wang, Y. Xu, S. R. Dulloor, J. Zhao, and S. Swanson, "Basic Performance Measurements of the Intel Optane DC Persistent Memory Module," arXiv, August 9, 2019: http://pages.cs.wisc.edu/~yxy/cs839-s20/papers/optane_measurement.pdf.

[3] S. Pelley, P. M. Chen, and T. F. Wenisch, "Memory Persistency," in *Proceedings of the International Symposium on Computer Architecture (ISCA 2014)*, pp. 265–276: http://web.eecs.umich.edu/~twenisch/papers/isca14.pdf.

[4] A. Rudoff, "Deprecating the PCOMMIT Instruction," Intel, September 12, 2016: https://software.intel.com/en-us/blogs/2016/09/12/deprecate-pcommit-instruction.

[5] J. Yang, J. Kim, M. Hoseinzadeh, J. Izraelevitz, and S. Swanson, "An Empirical Guide to the Behavior and Use of Scalable Persistent Memory," in *Proceedings of the18th USENIX Conference on File and Storage Technologies (FAST '20)*, pp. 169–182: https://arxiv.org/pdf/1908.03583.pdf.