Energy Consumption Evaluation of Optane DC Persistent Memory for Indexing Data Structures

Manolis Katsaragakis[⋆†], Lazaros Papadopoulos[⋆], Christos Baloukas[⋆], Verena Kantere[•], Francky Catthoor[†◦], Dimitrios Soudris[⋆]

[⋆]*Microprocessors and Digital Systems Laboratory, ECE , National Technical University of Athens, Greece*

[†]*Katholieke Universiteit Leuven, Kasteelpark Arenberg 10, 3001 Heverlee, Belgium*

[•]*Knowledge and Database Systems Laboratory, ECE , National Technical University of Athens, Greece*

[◦]*IMEC, Kapeldreef 75, 3001 Heverlee, Belgium*

[⋆]{mkatsaragakis, lpapadop, cmpalouk, dsoudris}@microlab.ntua.gr

[•]vkante@dblab.ece.ntua.gr, [†]francky.catthoor@esat.kuleuven.be

***Abstract*—The Intel Optane DC Persistent Memory (DCPM) is an attractive novel technology for building storage systems for data intensive HPC applications, as it provides lower cost per byte, low standby power and larger capacities than DRAM, with comparable latency. This work provides an in-depth evaluation of the energy consumption of the Optane DCPM, using well-established indexes specifically designed to address the challenges and constraints of the persistent memories. We study the energy efficiency of the Optane DCPM for several indexing data structures and for the LevelDB key-value store, under different types of YCSB workloads. By integrating an Optane DCPM in a memory system, the energy drops by 71.2% and the throughput increases by 37.3% for the LevelDB experiments, compared to a typical SSD storage solution.**

***Index Terms*—Intel Optane DCPM, NVM, Indexes, LevelDB, Scalability, Energy Consumption, HPC**

I. Introduction

The memory system is one of the main components that limit the scalability and contribute to the energy consumption of supercomputers [1]. The integration of more DRAM modules to enable more complex simulations, analytics and effective in-memory processing has negative impact on the sustainability and maintenance costs of supercomputing centres. In particular, despite the low access latency of traditional DRAM technologies, the increased leakage and refresh power requirements limit DRAM scalability and introduce a significant challenge towards reaching exascale performance.

In order to overcome DRAM limitations, non-volatile memory (NVM) technologies have been introduced, such as the 3D-XPoint, which is a subclass of the Phase-Change Memories (PCM) [2], Spin-Transfer Torque RAM (STT-RAM) [3] and Resistive RAM (ReRAM) [4]. State-of-the-Art commercial platforms integrate Intel Optane DC Persistent Memory (DCPM) modules along with DRAM, leading to heterogeneous memory systems [5]. For instance, the upcoming Aurora exascale supercomputer employs the DAOS storage architecture, which integrates a complex memory and storage hierarchy, including Intel Optane DCPM modules [6].

This work has been partially funded by EU Horizon 2020 program under grant agreement No 101015922 AI@EDGE (https://aiatedge.eu/).

These emerging memory technologies provide higher density than DRAM, enabling increased aggregate memory capacities with fewer nodes, having positive impact on the energy consumption, resilience and sustainability. Additionally, the data persistence features of the NVM technologies can be used to provide fault tolerance support to applications. On the other hand, the Optane DCPM provides, in general, higher access latency and lower bandwidth compared to DRAM [5], [7], [8]. However, recent studies indicate that the performance of the Optane DCPM depends a lot on the workload access pattern and size [8].

The contribution of the Optane DCPM technology to enable more complex and effective HPC simulations has been investigated in several recent works [7], [1], [9], [10], [11]. They indicate that using Optane DCPM as a volatile memory and DRAM as a cache enables close to DRAM performance. However, directly replacing the DRAM with Optane DCPM, without the use of DRAM as a cache, significantly reduces performance, due to effects such as the write throttling and concurrency contention [12].

Alternatively, the Optane DCPM can be used as a persistent storage medium to enable the storage and processing of massive volumes of scientific data gathered by simulations or instruments [13], [14]. Recent studies indicate that in order to take advantage of the scalability opportunities that storage systems enabled by persistent memories can offer to applications, the indexing data structures of the storage system need to be adapted in order to address the challenges and limitations that the NVM technologies impose [15], [16], [17]. For example, a fundamental challenge when NVMs are employed for persistent storage is the data consistency. Since modern CPUs are designed for volatile DRAM architectures, they typically cache and reorder memory writes, as they target higher performance. Therefore, existing indexing structures, designed for DRAM only systems, cannot be directly deployed on heterogeneous hierarchies, as they do not take into account the persistent nature of NVMs, as well as other challenges, such as the asymmetry in terms of read and write latency and the limited write endurance of NVMs.

Towards this direction, several works exist in the literature

Final version: DOI: 10.1109/HiPC56025.2022.00022

that propose indexes specifically designed for NVMs [15], [16], [17], [18], [19]. Some recent works provide performance-based evaluation of B+ tree indexing data structures for persistent memories [20], [21], while others convert DRAM-based B+ trees, tries, radix trees, and hash table indexes into NVM-oriented and evaluate their impact in terms of metrics such as the throughput and cache utilization [22].

This work targets HPC developers who consider the use of Optane DCPM as a persistent storage for applications which access large amounts of data through indexing data structures. The existing works in the literature, either focus on the evaluation of the Optane DCPM as a volatile main memory, or provide a performance characterization of its persistence features using the Optane DCPM as a storage device. Although these works are very relevant, they still miss the important energy component in the global analysis. Therefore, to address this gap, this work focuses on the energy consumption evaluation of the Optane DCPM configured as a persistent storage. The two major contributions of this work are the following:

- A thorough evaluation of the Optane DCPM as a storage medium, in terms of energy consumption, using representative and well-established B+ tree indexing data structures, triggered by various types of YCSB workloads.
- A combined performance and energy consumption evaluation of the LevelDB key-value store deployed on an Optane DCPM and an energy and throughput comparison with the corresponding deployment on an SSD.

To the best of our knowledge, this is the first work that entirely focuses on the energy efficiency aspects of the Optane DCPM. Although there exists several works that investigate the performance behavior of this new memory architecture, as detailed in the following sections, the energy consumption characteristics of the Optane DCPM have been evaluated to a very limited extent, yet.

The rest of this paper is organized as follows: Section II discusses the related work, focusing on the main differentiations between the present and the existing works. Section III is an overview of the Optane DCPM technology. The evaluation methodology is described in Section IV. Section V presents a set of baseline scalability results and the energy consumption analysis based on B+ tree indexing data structures and on the LevelDB key-value store. Additionally, it highlights the main observations and open research directions. Finally, in Section VI we draw conclusions.

II. Related Work

The existing works related to the Optane DCPM can be loosely classified into three categories:

- Empirical evaluation of the Optane DCPM performance through microbenchmarks and/or applications and investigation of challenges and limitations [5], [7], [1], [9], [12], [10], [22], [20], [21]
- New applications and data structures optimized for the Optane DCPM as persistent storage (mainly novel tree-based data structures) [23], [24]
- Novel workload schedulers, data placement algorithms and other runtimes for the Optane DCPM [25], [8], [26]

The present work is mostly related to first category, as it aims to thoroughly evaluate Optane DCPM using indexing data structures. However, in contrast to the existing works, it focuses on the energy consumption aspects.

Recently, the Optane DCPM was extensively evaluated using real-world scientific applications. The typical hardware configuration applied in these works was the use of the Optane DCPM as the main memory, while DRAM was used as a cache. The persistent features of the Optane DCPM are not available in this case. Recent works conclude that the performance gap between an Optane DCPM and a DRAM-only system is relatively small for this hardware configuration [12], [9]. Works that entirely replace DRAM with Optane DCPM (i.e. DRAM was entirely deactivated) highlight the fact that there is a significant performance overhead, due to the Optane DCPM bandwidth saturation for real-world multi-threaded applications [5], [1].

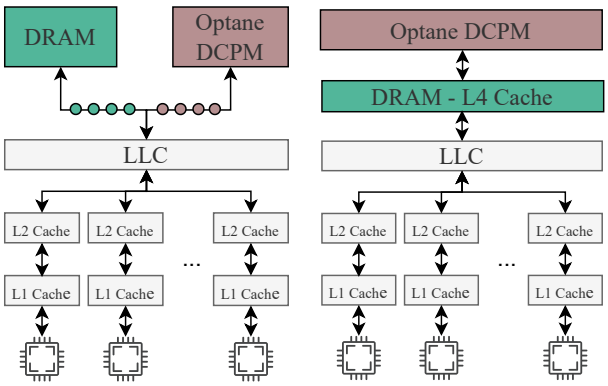
In contrast to the aforementioned works, using Optane DCPM as a persistent memory requires changes in the application source code. Additionally, the API for placing data on an Optane DCPM is available for dynamically allocated data structures only (e.g. through *libvmmalloc*) [7]. Therefore, this configuration is only partially evaluated in the existing literature using real-world scientific applications [10], [12]. The majority of works evaluate the Optane DCPM in this configuration either using microbenchmarks, such as the Stream Triad or mini-applications [7]. A more extensive evaluation, considering both the performance and energy consumption is available in the literature, but targets graph applications specifically [12]. On the other hand, there are several works which evaluate B+ tree indexing data structures on the Optane DCPM, as a persistent memory. They typically compare indexes in terms of application-level performance-related metrics, such as throughput and latency [22], [20], [21].

To summarize, in contrast with the existing studies, the present work evaluates the Optane DCPM as a persistent memory, in terms of energy consumption using B+ tree indexes and also provides a combined performance and energy consumption analysis of the LevelDB key-value store.

III. Overview of Optane DCPM

The Intel Optane DCPM is the first commercially available NVDIMM. It is based on the 3D-XPoint technology and provides both byte-addressability and persistence. From CPU perspective, the access granularity is 64 bytes. However, the physical media access granularity is 256 bytes. Therefore, write accesses smaller than 256 bytes are read-modify-write operations, which results in write amplification. The effects of write amplification have been extensively studied in the literature [10].

The Optane DCPM provides much higher capacity and lower power than DRAM, but, in general, at cost of increased latency and lower bandwidth. Fig. 1 depicts an heterogeneous memory system for the alternative operating modes of the



(a) AppDirect Mode (b) Memory Mode

Fig. 1: Overview of Intel Optane DCPM operating modes

Optane DCPM. The Optane DCPM can be configured to support either the *AppDirect* (Fig. 1a) or the *Memory mode* (Fig. 1b). In the AppDirect mode, Optane DCPM operates as a persistent storage. Access to the storage space is similar to memory mapped file operations. In *Memory Mode*, Optane DCPM operates as the main memory space without persistent properties. In this mode, the DRAM is used as an extra layer of cache on top of the Last-Level Cache.

In the *AppDirect* mode, developers can rely on the Persistent Memory Development Kit (PMDK) for persistent memory allocations [27]. Several libraries such as the *libvmmalloc* and *pmemobj* are provided that facilitate the dynamic allocation of objects that are traditionally handled by *malloc*, *free* and *memalign*. For instance, *libvmmalloc* allows a transparent runtime mapping of all dynamic allocations to the persistent memory. Thus, dynamically allocated data structures are placed into a memory mapped file, which is stored in the persistent memory. However, in *Memory mode*, all data objects are allocated to the Optane DCPM by default, transparently to developers. In this work, the Optane DCPM is configured in the *AppDirect* mode, as we aim at using it as a persistent storage for data intensive applications, instead of a volatile memory.

IV. Evaluation Methodology

In order to select representative B+ tree indexing implementations designed for persistent memories, we first identified their main design goals as described in the relevant literature. These goals are outlined below:

- **Write operations minimization**. Write operations in the Optane DCPM are expensive in terms of latency. Therefore, indexes developed for persistent memories utilize an array of methods to minimize write operations like reduced logging when performing atomic operations, or avoiding the need for CPU cache flushes that are often needed to maintain consistency of the data on the persistent memory.
- **Efficient locking mechanisms**. In order to achieve high concurrency, read operations should be done in a consistent but efficient way across multiple threads to utilize the available bandwidth as much as possible. On the other hand, write operations should make use of as fine locks as possible, constraining the critical sections to necessary code only.
- **Efficient index traversal**. Several decisions play an important role in index traversal, like sorting, the type of tree, key splitting etc. Some indexes are implemented targeting increased data locality, while others store a part of the tree on DRAM for fast traversal.

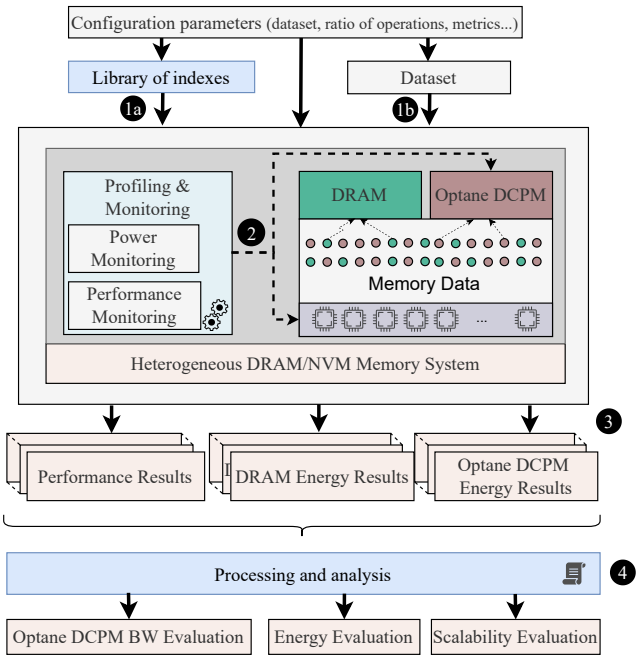


Fig. 2: Overview of the evaluation methodology

These goals are materialized differently for each index implementation as they try to offer persistence and concurrency in an efficient way. Many different implementations have been proposed including trees, hash tables, tries, radix trees etc. [22]. In this work we focus on tree implementations, as they are the most commonly used in practice.

We design a systematic methodology to enable the effective execution of four widely used indexes in the literature, which is illustrated in Fig. 2. The methodology consists of four distinct steps. The inputs of the methodology are a set of indexes 1a and a dataset 1b, which are configured by a set of parameters, such as the workload type (ratio of operations, skewness, etc.), the selected indexes for evaluation and the metrics of interest (e.g. throughput, energy consumption, scalability). The library of indexes is based on the source codes provided by [22] and includes the B+ tree implementations listed below. These indexes were selected based on the criterion of representing different ways of implementing the aforementioned design goals.

- **wBTree** is a type of main-memory B+ Tree, aiming to reduce the overhead caused by extensive NVM writes and CPU cache flush operations. To minimize the write

operations wBTree uses write atomic B+ trees that can achieve node consistency either through atomic writes in the nodes or by redo-only logging, thus reducing the necessary write operations on Optane [28]. Furthermore, the keys are maintained in a sorted order throughout the execution and the nodes employ a small indirection slot array and/or a bitmap, to quickly check for a particular key's existence. The use of the indirection slot array has the added benefit of not requiring the movement of index entries, for most insertions and deletions, thus further reducing the amount of write operations.

- **Masstree:** is a cache-efficient, highly concurrent trie-like concatenation of B+ tree nodes, and provides high performance even for long common key prefixes. We use the Masstree version converted for NVM by [22]. The Masstree combines a trie and B+ tree implementations to achieve higher performance for long keys. Particularly, long string keys are split into more than one node, so that searching is much faster, because of the faster comparisons between the common part of the keys. Masstree uses a versioning system for concurrency that improves locking performance. Furthermore, to reduce write operations, Masstree inserts new keys to leaf nodes by appending a new key-value pair to the node in an unsorted order. Internal nodes remain sorted for faster traversal.
- **Fast&Fair:** is a persistent memory B+ tree that provides lock-free reads [22]. The reads detect and tolerate inconsistencies such as duplicated elements. By making read operations tolerate transient inconsistency, Fast&Fair avoids expensive copy-on-write, logging, and even the necessity of read latches so that read transactions can be non-blocking [29]. These characteristics of Fast&Fair reduce write operations, while improving locking performance.
- **FPTree:** is a B+ Tree alternative that implements selective persistence by requiring only leaf nodes to be stored persistently. Internal nodes can be rebuilt in case of power failure. Therefore, the FPTree places all internal nodes on the DRAM for fast traversing and consequently higher performance, while the leaf nodes are placed on a persistent memory. Furthermore, the leaf nodes implement *fingerprinting*, where a one byte hash of each key contained within the node is placed at the first cacheline-sized section of the node [16]. Fingerprinting improves lookup performance by speeding up the process of finding if a key is included in a particular node, without having to traverse the node itself. For concurrency, FPTree uses Hardware Transactional Memory (HTM) [30] and fine-grained locks for the internal and leaf nodes respectively.

The *Profiling & Monitoring* component (❷) of the methodology is responsible for power and performance monitoring and it is based on Intel's Processor Counter Monitor (PCM) [31], a tool that allows energy/power sampling over DIMMs through hardware counters [32]. The sampling rate for the profiling is set to 0.01s, by default. More specifically, we use the *PCM-Power* component to monitor the energy consumed in the DRAM and Optane DCPM DIMMs. For each individual DIMM, the *PCM-Power* reports the on-DIMM energy consumed for the period of time based on our defined sampling rate. The memory accesses on the Optane DCPM are measured by the *ipmctl* utility, which is the standard for configuring and managing Optane DCPM modules, by monitoring the corresponding read and write accesses of each DIMM, respectively [33]. Finally, to calculate the energy consumption of an SSD, we use profiling information about I/O reported by the Intel VTune Amplifier regarding the page read, write and flush operations and the power values from the manufacturers' data-sheets.

TABLE I: Overview of experimental setup

Experimental Setup	
CPU, DRAM	Xeon Gold 5218R CPU, 2x20 cores @2.10GHz, with 4x32GB DDR4 DIMMs
Optane DCPM Configuration	2x256GB DIMMs, AppDirect Mode, ext4-DAX
SSD	Intel SSD S4500 Series, 480GB
Indexes	wBTree, Masstree, Fast&Fair, FPTree
Key-value store	LevelDB
YCSB Workloads	Balanced (50% reads, 50% writes), Write-heavy (99% writes), Read-heavy (100% reads), Scan-heavy (99% scan)
Workload keys	8-bytes integers
Operating system	Ubuntu 20.04.2 LTS, kernel 5.4.0-121-generic

The measurements are grouped (❸) and processed (❹) in order to provide the throughput, energy consumption and scalability results for each evaluation experiment. The generation of the aforementioned results is performed automatically, based on a set of customizable Python and shell scripts. The source codes of the tool with usage instructions and the library of indexes are publicly available [1].

V. Evaluation

A. Experimental Setup

Through the evaluation process we aim at the following:

- To evaluate the energy consumption behavior of the Optane DCPM for indexing data structures triggered by various types of workloads.
- To study the performance and the energy efficiency of the Optane DPCM for a key-value store (LevelDB), using the corresponding deployment on an SSD as a baseline.

Table I shows an overview of the experimental setup. The experiments were conducted on a single node with a 2x20 core Intel Xeon Gold 5218R CPU @2.10GHz with 4x32GB DDR4 DIMMs and 2x256GB Optane DC NVDIMMs. Intel Optane DC is configured in *App Direct* mode with ext4-DAX file system. We utilized the version 1.11 of PMDK [27] and gcc-9.4. The monitoring and profiling tools were reported earlier, in Section IV.

The YCSB is used to trigger the indexes under different types of workload: balanced, write-heavy (which corresponds to insert operations), read-heavy (i.e. lookup operations) and

[1] https://github.com/mkatsa/PENVMTool

scan-heavy (i.e. range-lookup operations) [34]. Each workload consists of 64M operations with 8-byte random integer keys.

B. Baseline Performance-oriented Results

First, we present a set of baseline performance results, which enable a comprehensive insight into the energy consumption evaluation, which follows.

Fig. 3 shows the throughput on the Optane DCPM for a variety of YCSB workloads. It demonstrates the scalability of each index under different workloads. It can be noticed that the *fptree* scales up to 32 threads for the read-heavy and scan-heavy workloads, by exceeding 10Mops/sec and 5Mops/sec, respectively. For the balanced and write-heavy workloads, the *masstree*, *fptree* and *fastfair* provide maximum throughput up to 3.3Mops/sec and 2.3Mops/sec at 16 threads, respectively. The *wbtree* does not scale, since the implementation used in this work is single-threaded and is demonstrated as a sequential baseline for the rest of the B+ tree implementations.

For comparison, we evaluate the scalability of the indexes on a DRAM-only memory system. These baseline results are shown in Fig 4. All indexes, and especially the *masstree*, scale well under the balanced workload up to 64 threads. Moreover we observe that the throughput achieved for *masstree* index is greater than the *fptree* on DRAM only, in contrast to the Optane DC execution, where *fptree* outperforms *masstree*. This is due to the fact that *fptree* places all inner nodes on DRAM and only leaf nodes on Optane, thus performing better on higher number of threads especially on read-only workloads. Similar results were observed for the rest of the workload types.

To investigate the impact of the Optane DCPM hardware configuration on the maximum throughput that can be reached, we evaluate the scalability performance of the indexes on a memory system with a single Optane DCPM DIMM, under balanced workload. By comparing the results of Fig. 3a (two DIMMs) with Fig. 5 (single DIMM), it is noticed that the use of more Optane DCPM DIMMs on the same socket enables higher parallelism, allowing improved scalability and higher throughput. Indeed, on a single DIMM, the maximum throughput is obtained at 4 threads and barely exceeds 1Mops/sec.

The above observations are inline with performance evaluation results in the existing literature. In particular, the scalability limitations of B+ tree indexes on Optane DCPM have been studied in recent works, relying on custom microbenchmarks [20]. Our findings, based on the YCSB, further confirm the observed scalability limitations.

C. Energy Consumption Results

Fig. 6 shows the energy consumed in the Optane DCPM DIMMs for each index under different types of workload, as measured through the sensors equipped in the DIMMs. In general, the energy consumption is gradually reduced up to 8 or 16 threads, indicating a direct correlation between the throughput reached and the energy consumed. The energy of *wbtree* index energy remains slightly constant. Since wbtree does not scale (Fig. 3), this leads to utilization of the memory resources for higher amount of time, therefore leading to increased energy consumption.

TABLE II: Energy consumption (KJ) of Optane DCPM DIMMs for each workload type for 16 threads.

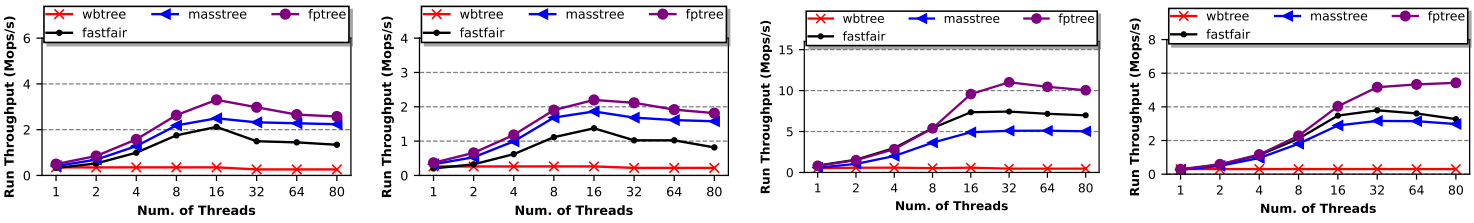
	Balanced	Write-heavy	Read-heavy	Scan
wbtree	3.539	4.069	2.966	4.372
masstree	0.615	0.701	0.524	0.645
fptree	0.554	0.627	0.465	0.562
fastfair	0.787	0.896	0.620	0.738

Table II shows the energy consumption values for 16 threads. We notice that the energy consumption for different indexes under the same workload does not vary significantly. However, the Optane DCPM consumes less energy under the read-heavy workload compared to the rest of the workload types. For example, triggered by the read-heavy, the *fptree* index yields 16% lower energy consumption compared to the corresponding balanced workload, indicating the impact that the increased throughput under a read-heavy workload has on the energy (Fig. 3).

As a baseline for the Optane DCPM energy consumption, Fig.7 shows the energy for the balanced workload in a DRAM-only system (i.e. with Optane DCPM deactivated). Similarly to Fig. 6, the results correspond to the energy consumed on the DRAM DIMMs, only. The energy is significantly reduced after 4 or 8 threads, in accordance to the throughput increase demonstrated in Fig. 4. Additionally, by comparing the energy values of Fig. 7 with Fig. 6a, we notice that the energy consumed in the Optane DCPM is about half the one in DRAM. As an example, the energy consumption when deploying the *fptree* with 16 threads on Optane DCPM is 56.3% lower compared to the corresponding DRAM experiment.

The total energy consumption, i.e. of both the CPUs and the memory system, is shown in Fig. 8. The throughput still impacts the energy consumed, as it is minimized at 8-16 threads, where the maximum throughput is reached. However, after that point, the energy consumption slightly increases, as a result of the throughput decrease after 16 threads for most of the indexes, as depicted in Fig. 3.

To further investigate the energy consumption behavior of the Optane DCPM as the number of thread increases, we monitor the number of read and write memory accesses in the persistent memory for each index, for an increasing number of threads. The profiling results are shown in Fig. 9. It is noticed that the number of read and write accesses beyond 16 threads increases significantly, due to the internal mechanisms of the indexes and the overhead of concurrency, without a positive impact on performance, as shown in Fig.3a. In particular, when the bandwidth is capped like in the case of 1 or 2 DIMMs of Optane DCPM, a significant amount of CPU time is spent by trying to acquire exclusive access to protected regions by each index, accessing internal index variables for locking and synchronization and thus increasing the overall access count. Therefore, using threads beyond that point has negative impact both on the throughput and the energy consumption.



(a) Balanced workload (b) Write-heavy workload (c) Read-heavy workload (d) Scan-heavy workload

Fig. 3: Scalability evaluation on Intel Optane DCPM for different types of workloads.

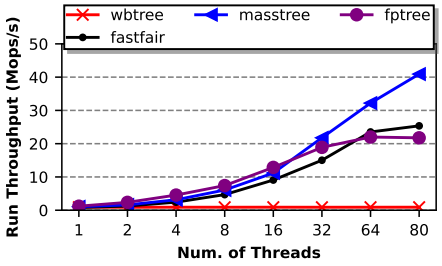


Fig. 4: Scalability evaluation on DRAM-only for balanced workload.

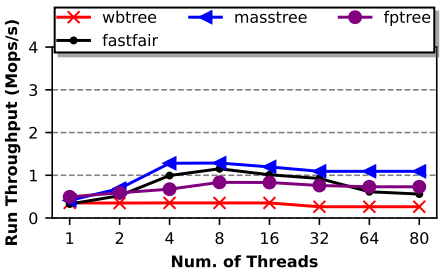


Fig. 5: Throughput for balanced workload using a single Optane DCPM DIMM.

Additionally, the increase in the memory access count is expected to have impact the endurance of the persistent memory hardware [35]. Interestingly, the writing hardware endurance of the Optane DCPM has not been evaluated in the existing literature, yet.

Finally, to investigate the impact of the available persistent memory bandwidth on the energy consumption, Fig. 10 shows the energy for the balanced workload when integrating a single Optane DCPM DIMM. An interesting observation is that although when integrating 2 DIMMs the throughput increases almost 2x (as shown by Fig.3a and Fig.5), the energy is reduced by 5x (e.g. for the *masstree* as shown by comparing the results of Fig.6a and Fig. 10). Therefore, it is reasonable to expect that increasing the Optane DCPM bandwidth by integrating even more DIMMs will further decrease the energy consumption and increase the throughput.

D. LevelDB key-value store Performance-Energy Exploration

In this section, we perform a combined performance-energy consumption analysis of the Optane DCPM using the LevelDB key-value store[2].

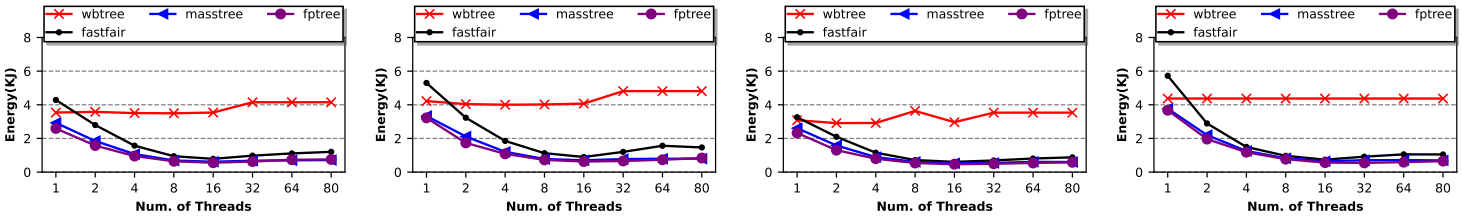
The LevelDB is a fast key-value storage library developed by Google, where keys and values are arbitrary byte arrays, allowing any type of data to be sorted by a key. LevelDB is not an SQL database and it does not provide an interface for SQL queries making it more inline with the other indexes described in this work. The only difference is that the LevelDB keeps data stored on disk, while maintaining a configurable buffer in the memory for performance. The inherent support for data persistence on HDD/SSD and data caching in DRAM make it a suitable candidate for demonstrating the benefits of storing the whole database in an Optane DCPM instead of an SSD. The LevelDB uses a log-structured merge-tree (also known as LSM tree) as a data structure for storing data. The LSM tree comprises two parts: one tree in the memory serving as a buffer and another on the disk. The advantage and particular characteristic of the LSM trees is that each part can be configured differently for the platform they reside.

After integrating the YCSB in the LevelDB, we evaluate the two memory hierarchy configurations depicted on Fig. 11, in terms of throughput and energy consumption, under different workloads. In the first experiment, the LevelDB is stored on the SSD, while in the second it is allocated in the Optane DCPM. The tools employed for profiling and monitoring are described earlier, in Section IV. For brevity, we present results for the balanced and read-heavy workloads only, but the observations apply to the write-heavy and scan workloads, as well.

The throughput results are presented in Fig. 12a and Fig. 12b, for balanced and read-heavy workloads, respectively. We notice the following:

- The maximum throughput for the Optane DCPM is reached at 16 threads for the balanced workload and at 32 for the read-heavy. This result is inline with the observations based on the indexing data structure experiments on Optane DCPM (Fig. 3a and Fig. 3c). For the LevelDB, increasing the number of threads beyond these points, results in performance degradation on the Optane DCPM, as noticed in most of the corresponding experiments using B+ tree indexes (Fig. 3).
- Increasing the number of threads beyond the point of the maximum throughput, results in an increasing memory access count on the Optane DCPM. Indeed, for the

[2]LevelDB: https://github.com/google/leveldb



(a) Balanced workload (b) Write-heavy workload (c) Read-heavy workload (d) Scan-heavy workload

Fig. 6: Energy consumption evaluation on Intel Optane DCPM for different types of workloads.

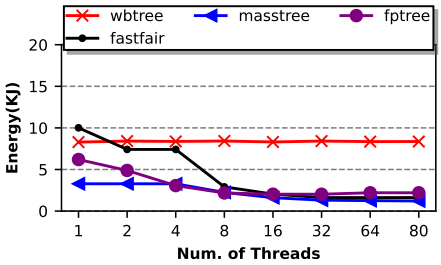


Fig. 7: Energy consumption on DRAM-only for balanced workload.

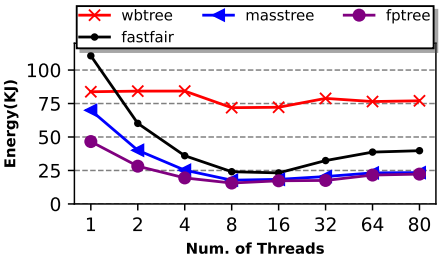


Fig. 8: Energy consumption of CPUs and memory system.

balanced workload, the number of accesses increases by 27.4% for 32 threads and by 39.1% for 64, compared to the number of accesses on 16 threads. Similarly, for the read-heavy workload it increases by 13.6% for 64 threads compared to 32. These results correspond to the ones demonstrated in Fig. 9 for the indexing data structures experiments. The increase in the memory access count is attributed to the overhead of the synchronization mechanisms and is expected to have negative impact on the Optane DCPM hardware endurance.

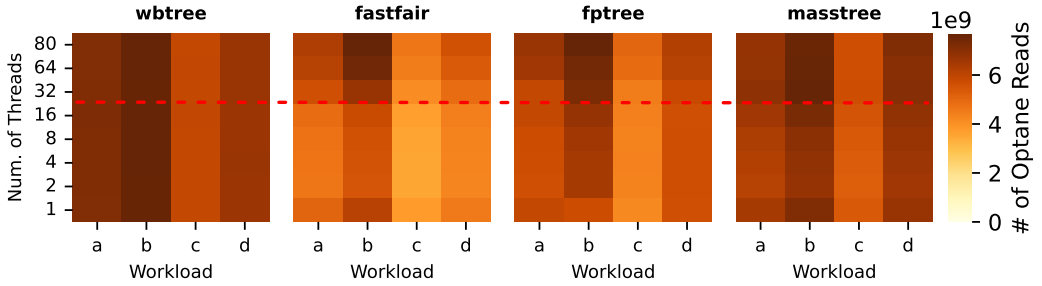
- Finally, Fig. 12a and Fig. 12b demonstrate the performance gains which can be obtained for the LevelDB when stored in an Optane DCPM instead of an SSD. Performance gains in terms of throughput reach 37.3% for the balanced workload (at 16 threads) and 34.6% for the read-heavy at 32 threads.

The energy consumption results are presented in Fig.13a and Fig.13b for the balanced and the read-heavy workloads, respectively. Detailed measurements on (i) CPUs, DRAM DIMMs and SSD, (ii) Optane DIMMs only, (iii) CPUs, DRAM and Optane DIMMs are presented. We observe the following:

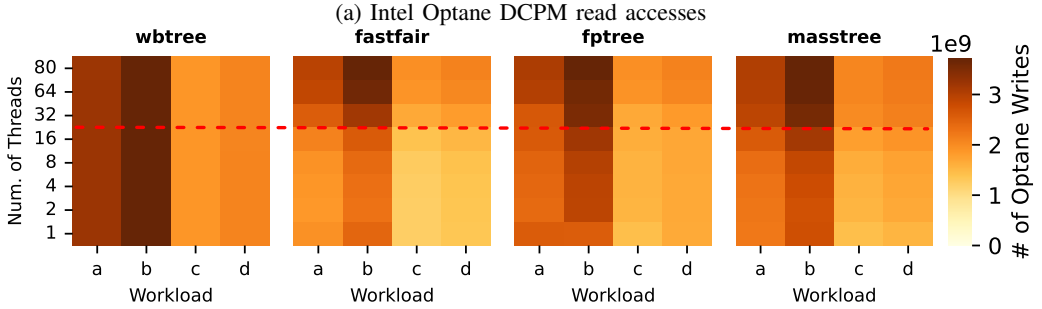
- As the number of threads increases, the total energy consumption for the balanced workload for the Optane DCPM experiment (i.e. CPU, DRAM, Optane DCPM) slightly drops reaching a minimum value of 2.6KJ at 8 threads, which is 16% lower than the energy for a single thread. The energy consumption on the Optane DCPM DIMMs is also in direct correlation with the throughput (Fig.12a). After that point, the energy increases following the decrease in throughput. In contrast with the Optane DCPM experiments, the energy consumption when the LevelDB is stored in the SSD is relatively independent of the number of threads for the balanced workload, slightly decreasing at 16 threads, by 2.1% compared to the single thread experiment.
- For the read-heavy workload, the total energy consumption for the Optane DCPM experiment (i.e. CPU, DRAM, Optane DCPM) drops to 0.7KJ at 32 threads, where the throughput is maximized. Compared to the single thread experiment, the energy is decreased by 31%. The energy consumed by the Optane DIMMs only, is also minimized at 32 threads. Additionally, the energy when storing the LevelDB to the SSD drops up to 32 threads and increases after that point. Another interesting observation is the fact that the energy consumed by the Optane DCPM is about the same between the experiments of different workload types (i.e. about 0.5KJ for both the balanced and read-heavy workloads). For the read-heavy workload the energy consumed by the CPU and DRAM DIMMs is between 0.4-0.6KJ, depending on the number of threads, as shown in Fig. 13b. However, for the balanced workload, a much larger portion of the total energy is consumed by the CPU and the DRAM (about 2.5KJ, which is almost 6x higher than the energy consumed by the Optane DCPM DIMMs only, as shown in Fig. 13a).
- Significant energy gains can be obtained by storing the LevelDB to an Optane DCPM, instead of an SSD. For example, for the read-heavy workload, the energy consumption is reduced up to 71.2%. This demonstrates the efficiency of Optane DCPM as a storage medium, it terms of energy.

E. Discussion

In this subsection we highlight the main key findings based on the analysis of the experimental results and we indicate



(a) Intel Optane DCPM read accesses



(b) Intel Optane DCPM write accesses

Fig. 9: Number of Intel Optane DCPM read and write memory accesses. Workload a: balanced, b: write-heavy, c: read-heavy, d: scan-heavy.

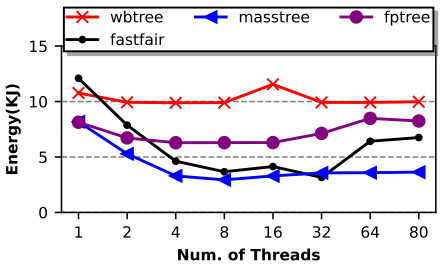
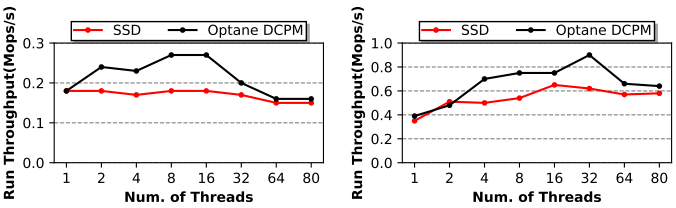


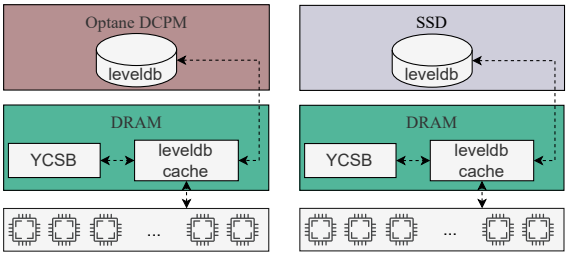
Fig. 10: Energy consumption on a single Optane DCPM DIMM for balanced workload.



(a) Balanced workload

(b) Read-heavy workload

Fig. 12: LevelDB throughput for different memory configurations under balanced and read-heavy workload.



(a) DRAM/SSD configuration.

(b) DRAM/Optane DCPM configuration.

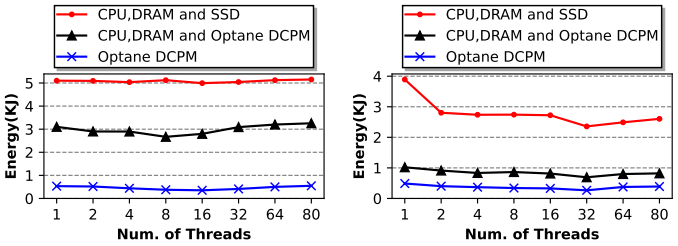
Fig. 11: System configurations for the LevelDB experiments: DRAM/SSD and DRAM/Optane DCPM.

some major open research directions.

The Optane DCPM is a relevant persistent memory architecture for data intensive HPC applications, providing advantages in terms of both throughput and energy consumption compared to the typical SSD storage. However, the bandwidth limitations of the Optane DCPM need to be taken into consideration. The evaluation results indicate that **fine-tuning the deployment of the indexing data structures on an heterogeneous memory system** is needed, in order to provide significant gains in terms of combined energy consumption and performance.

The hardware configuration is critical for the scalability and the energy consumption of applications deployed on the Optane DCPM. Based on the experiments of Fig. 5 and Fig. 3a, we can argue that by integrating more DIMMs into the memory system, higher throughput will be reached due to improved parallelism. It is reasonable to expect that improvements in throughput will result in lower energy consumption. However, it will be interesting to investigate the existence of a saturation point, after which integrating more DIMMs does not contribute to further energy efficiency.

The write indexing operations are costly in terms of energy consumption on the Optane DCPM compared to the read-only (e.g. lookup operations). However, the increased amount of energy is not consumed by the Optane DIMMs,



(a) Balanced workload (b) Read-heavy workload

Fig. 13: LevelDB energy consumption for different memory configurations under balanced and read-heavy workload.

as demonstrated in the LevelDB experiments of Fig. 13 (but also in Fig. 6). Instead, it is consumed by the CPUs, due to the synchronization and consistency overhead, requiring cache flushes and similar instructions to maintain the data coherence in the persistent device. This energy consumption overhead can be observed both in the B+ tree and LevelDB experiments. Hardware support for data consistency targeting NVMs is expected to benefit the energy efficiency of the whole system significantly.

Although the analysis of the experimental results was based on experiments using 8-byte integer keys, **similar observations can be obtained by the use of other types of keys**. As an indicative example, for 24-byte string keys on Optane DCPM, using the same experimental setup, we noticed the energy consumed by the Optane DCPM DIMMs to follow a similar pattern with the corresponding integer-key experiments. Particularly, it is reduce up to 8 and 16 threads for the balanced and read-heavy workloads respectively, and reaches a plateau after that point.

During the profiling analysis of Section V, we have evaluated the scalability of the dynamic allocations using the *libvmmalloc* library of the PMDK [27]. We noticed that the ***libvmmalloc* does not scale with the number of threads**. This is inline with observations in the existing literature [20]. This limitation of the *libvmmalloc* will affect the scalability and the energy consumption of the indexing data structures on the Optane DCPM, as well as the performance of dynamic applications which make heavy use of the heap memory and are deployed to Optane DCPM through the PMDK libraries. Custom and scalable dynamic memory allocators for persistent memories are expected to enable higher performance and lower energy consumption for such applications.

VI. Conclusion

This work provides new insights about the Optane DCPM as a persistent storage device. Based on a comprehensive evaluation of indexing data structures triggered by various types of workloads, we investigate the energy consumption characteristics of the Optane DCPM. We show that significant performance and energy consumption gains can be obtained by the effective use of Optane DCPM integrated in a memory system, compared to a typical SSD. However, proper configuration is required, considering the bandwidth limitations of the Optane DCPM, in order to benefit from the advantages it offers. The outcomes of this work can be exploited by developers who target the Optane DCPM as a persistent storage medium for HPC applications.

References

[1] O. Patil, L. Ionkov, J. Lee, F. Mueller, and M. Lang, "Performance characterization of a dram-nvm hybrid memory architecture for hpc applications using intel optane dc persistent memory modules," in *Proceedings of the International Symposium on Memory Systems*, 2019, pp. 288–303.

[2] H.-S. P. Wong, S. Raoux, S. Kim, J. Liang, J. P. Reifenberg, B. Rajendran, M. Asheghi, and K. E. Goodson, "Phase change memory," *Proceedings of the IEEE*, vol. 98, no. 12, pp. 2201–2227, 2010.

[3] E. Kültürsay, M. Kandemir, A. Sivasubramaniam, and O. Mutlu, "Evaluating stt-ram as an energy-efficient main memory alternative," in *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 2013, pp. 256–267.

[4] S.-S. Sheu, K.-H. Cheng, M.-F. Chang, P.-C. Chiang, W.-P. Lin, H.-Y. Lee, P.-S. Chen, Y.-S. Chen, T.-Y. Wu, F. T. Chen *et al.*, "Fast-write resistive ram (rram) for embedded applications," *IEEE Design & Test of Computers*, vol. 28, no. 1, pp. 64–71, 2010.

[5] J. Izraelevitz, J. Yang, L. Zhang, J. Kim, X. Liu, A. Memaripour, Y. J. Soh, Z. Wang, Y. Xu, S. R. Dulloor *et al.*, "Basic performance measurements of the intel optane dc persistent memory module," *arXiv preprint arXiv:1903.05714*, 2019.

[6] Daos architecture. [Online]. Available: https://www.alcf.anl.gov/events/daos-next-generation-data-management-exascale

[7] M. Weiland, H. Brunst, T. Quintino, N. Johnson, O. Iffrig, S. Smart, C. Herold, A. Bonanni, A. Jackson, and M. Parsons, "An early evaluation of intel's optane dc persistent memory module and its impact on high-performance scientific applications," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2019, pp. 1–19.

[8] L. Xiang, X. Zhao, J. Rao, S. Jiang, and H. Jiang, "Characterizing the performance of intel optane persistent memory: a close look at its on-dimm buffering," in *Proceedings of the Seventeenth European Conference on Computer Systems*, 2022, pp. 488–505.

[9] V. Mironov, I. Chernykh, I. Kulikov, A. Moskovsky, E. Epifanovsky, and A. Kudryavtsev, "Performance evaluation of the intel optane dc memory with scientific benchmarks," in *2019 IEEE/ACM Workshop on Memory Centric High Performance Computing (MCHPC)*. IEEE, 2019, pp. 1–6.

[10] I. Peng, K. Wu, J. Ren, D. Li, and M. Gokhale, "Demystifying the performance of hpc scientific applications on nvm-based memory systems," in *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2020, pp. 916–925.

[11] R. S. Venkatesh, T. Mason, P. Fernando, G. Eisenhauer, and A. Gavrilovska, "Scheduling hpc workflows with intel optane persistent memory," in *2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, 2021, pp. 56–65.

[12] I. B. Peng, M. B. Gokhale, and E. W. Green, "System evaluation of the intel optane byte-addressable nvm," in *Proceedings of the International Symposium on Memory Systems*, 2019, pp. 304–315.

[13] S. D. Smart, T. Quintino, and B. Raoult, "A high-performance distributed object-store for exascale numerical weather prediction and climate," in *Proceedings of the Platform for Advanced Scientific Computing Conference*, 2019, pp. 1–11.

[14] J. Ejarque, R. M. Badia, L. Albertin, G. Aloisio, E. Baglione, Y. Becerra, S. Boschert, J. R. Berlin, A. D'Anca, D. Elia *et al.*, "Enabling dynamic and intelligent workflows for hpc, data analytics, and ai convergence," *Future Generation Computer Systems*, vol. 134, pp. 414–429, 2022.

[15] J. J. Levandoski, D. B. Lomet, and S. Sengupta, "The bw-tree: A b-tree for new hardware platforms," in *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, 2013, pp. 302–313.

[16] I. Oukid, J. Lasperas, A. Nica, T. Willhalm, and W. Lehner, "Fptree: A hybrid scm-dram persistent and concurrent b-tree for storage class memory," in *Proceedings of the 2016 International Conference on Management of Data*, 2016, pp. 371–386.

[17] S. Chen and Q. Jin, "Persistent b+-trees in non-volatile main memory," *Proc. VLDB Endow.*, vol. 8, no. 7, p. 786–797, feb 2015. [Online]. Available: https://doi.org/10.14778/2752939.2752947

[18] D. Hwang, W.-H. Kim, Y. Won, and B. Nam, "Endurable transient inconsistency in Byte-Addressable persistent B+-Tree," in *16th USENIX Conference on File and Storage Technologies (FAST 18)*. Oakland, CA: USENIX Association, Feb. 2018, pp. 187–200. [Online]. Available: https://www.usenix.org/conference/fast18/presentation/hwang

[19] Y. Mao, E. Kohler, and R. T. Morris, "Cache craftiness for fast multicore key-value storage," in *Proceedings of the 7th ACM European Conference on Computer Systems*, ser. EuroSys '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 183–196. [Online]. Available: https://doi.org/10.1145/2168836.2168855

[20] L. Lersch, X. Hao, I. Oukid, T. Wang, and T. Willhalm, "Evaluating persistent memory range indexes," *Proceedings of the VLDB Endowment*, vol. 13, no. 4, pp. 574–587, 2019.

[21] Y. He, D. Lu, K. Huang, and T. Wang, "Evaluating persistent memory range indexes: Part two," *arXiv preprint arXiv:2201.13047*, 2022.

[22] S. K. Lee, J. Mohan, S. Kashyap, T. Kim, and V. Chidambaram, "Recipe: Converting concurrent dram indexes to persistent-memory indexes," in *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, 2019, pp. 462–477.

[23] X. Zhou, L. Shou, K. Chen, W. Hu, and G. Chen, "Dptree: differential indexing for persistent memory," *Proceedings of the VLDB Endowment*, vol. 13, no. 4, pp. 421–434, 2019.

[24] Y. Chen, Y. Lu, K. Fang, Q. Wang, and J. Shu, "utree: a persistent b+-tree with low tail latency," *Proceedings of the VLDB Endowment*, vol. 13, no. 12, pp. 2634–2648, 2020.

[25] M. Weiland and B. Homoelle, "Usage scenarios for byte-addressable persistent memory in high-performance and data intensive computing," *Journal of Computer Science and Technology*, vol. 36, no. 1, pp. 110–122, 2021.

[26] M. Katsaragakis, L. Papadopoulos, C. Baloukas, and D. Soudris, "Memory management methodology for application data structure refinement and placement on heterogeneous dram/nvm systems," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2022, pp. 748–753.

[27] Persistent memory development kit. [Online]. Available: https://pmem.io/pmdk/

[28] S. Chen and Q. Jin, "Persistent b+-trees in non-volatile main memory," *Proceedings of the VLDB Endowment*, vol. 8, no. 7, pp. 786–797, 2015.

[29] D. Hwang, W.-H. Kim, Y. Won, and B. Nam, "Endurable transient inconsistency in {Byte-Addressable} persistent {B+-Tree}," in *16th USENIX Conference on File and Storage Technologies (FAST 18)*, 2018, pp. 187–200.

[30] R. M. Yoo, C. J. Hughes, K. Lai, and R. Rajwar, "Performance evaluation of intel® transactional synchronization extensions for high-performance computing," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, ser. SC '13. New York, NY, USA: Association for Computing Machinery, 2013. [Online]. Available: https://doi.org/10.1145/2503210.2503232

[31] Processor counter monitor. [Online]. Available: https://github.com/opcm/pcm

[32] D. Masouros, S. Xydis, and D. Soudris, "Rusty: Runtime interference-aware predictive monitoring for modern multi-tenant systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 1, pp. 184–198, 2020.

[33] Pmdk ipmctl. [Online]. Available: https://docs.pmem.io/ipmctl-user-guide/

[34] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with ycsb," in *Proceedings of the 1st ACM symposium on Cloud computing*, 2010, pp. 143–154.

[35] J. Boukhobza, S. Rubini, R. Chen, and Z. Shao, "Emerging nvm: A survey on architectural integration and research challenges," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 23, no. 2, pp. 1–32, 2017.