SPAST Open Access Research

Next generation storage: Non-Volatile Memory

Olzhas Kaiyrakhmet[1]

email: olzhabay.i@gmail.com

Abstract

The next generation memory storage revolution is triggered with the advent of the Non-Volatile Memory (NVRAM), also known as Persistent Memory (PM). Previous revolution had occurred when flash memory was introduced. Likewise, NVRAM is going to deliver new possibilities for applications. However, the new memory can hardly find usage for regular consumers now. Most of the research is currently focused around the applications pertaining to fast operations on large data, such as SQL/NoSQL databases, data analytics, and storage systems. Unless the foundations of a good understanding and practices are not laid down, this technology would not reach its full potential of applications. In this paper we introduce this new memory technology, the current state of research with it, and the possible future developments.

Keywords: NVRAM; Memory; Storage, Software Design.

Introduction

For over a decade, there existed a large performance void between a random-access memory and a solid-state drive in the storage hierarchy. The performance difference between them in terms of speed was about 100x-1000x [1-2]. However, the differences do not stop there, as DRAM is volatile to power cycles, and data stored in it can be easily lost [2-4]. Consequently, applications that need to persist their data must be designed to store in non-volatile storage media. Due to storage, the media are having architecture limitations- direct data allocation is not possible, and all I/O instructions must go through the page cache, which reside in the DRAM. Nevertheless, applications and operating systems have advanced to allow the user to manipulate these data operations in an extremely efficient way.

Over the past decade, researchers have been forecasting a new storage technology arrival to fill this empty performance gap [4-5]. Listed in Figure 1, the researchers foresee it to have the properties, such as scalability, byte-addressability, persistence, and low latency. With this vision, researchers did not sit idly until such technology to come and have been proposing new programming models, file systems, libraries, and software architecture designs. Only last year, Intel has made the world's first commercially available non-volatile memory, Intel Optane DIMM [6], making all the wait worth it. Now we can directly develop

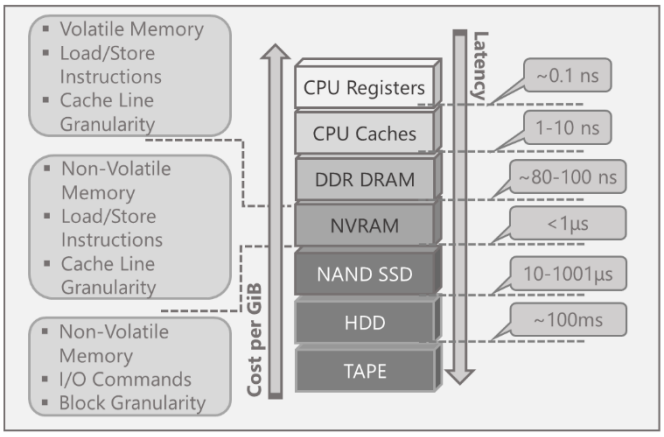

Figure.1. Memory-Storage Hierarchy

and test all the applications, showing all the real performance numbers.

One of the challenges is the development of software for NVRAM, which is not as straightforward as it seems. Although, it does provide faster data allocation with persistence. However, the design of software must be carefully optimized, else a significant performance loss would be the undoing of superior memory-storage technology. This especially holds true when dealing with the consistency of data operations and the persistence of allocated data.

Received: 1st January 2021 Accepted: 12th January 2021

Intel Optane DC Persistent Memory

Intel Optane DC Persistent Memory (Optane PM) [6,7] is becoming a widely available memory product. It is based on 3D XPoint device technology [8]. Compared to other devices with a similar name, Optane NVMe, which connects to the PCIe interface and is a block-based storage media. Optane PM does connect to the memory interface DIMM and has direct memory-addressability. Compared to DRAM, it has a much high data density and data persistence. On the current market, Optane PM is available in 128, 256, and 512 GB memory stick modules, thus it can scale up to 6TB of memory per system.

At present, Optane PM is supported only on Intel's new server-grade Xeon processors, Cascade Lake or newer. It works on both Linux and Windows operating systems, but Linux has better device support due to being a popular OS for servers.

TABLE 1
LATENCY AND BANDWIDTH PERFORMANCE COMPARISON OF DRAM AND NVRAM (OPTANE PM) PER ONE DEVICE

	Read Latency	Write Latency	Read Bandwidth	Write Bandwidth
DRAM	~90ns	~71ns	~18.7GB/s	~8.9GB/s
NVRAM	~237ns	~76ns	~8.0GB/s	~1.5GB/s

Optane PM uses a new DDR-T interface, which has the same mechanical and electrical interface as DDR4. Since it is based on the same cache-line size (64-byte) granularity, but it uses a different underlying communication protocol. Every memory instruction is sent to the NVRAM, first arrives at an internal **on-DIMM** controller, more precisely XPController, that organizes all the accesses to the memory. This is used to internally manage address translation for reducing wear-leveling, and bad-block management.

As Optane PM is a memory device, it is always compared to DRAM in terms of sheer performance [4, 5]. From Table 1, one can see that NVRAM has slower access latency for read instructions, around 2.65 times, and almost the same latency for write instruction. In terms of bandwidth performance, Optane memory is slower 2.4x and 6x times for read and write operations, respectively. Please note, that these measurements are from single instruction tests and do not directly represent the performance of the device under the real workload. Some reports show that Optane PM can be underperformed by up to 12x times in mixed read and write workloads. Moreover, NVRAM poorly scales on multithreaded workloads as well. Thus, proper software design must be in place when programming applications for this new memory, and to achieve it, a new programming model for persistent memory will be of help.

Programming Model

Before the real hardware, Intel Optane DC Persistent Memory came out, researchers and engineers already have been foreseeing a programming model for NVRAM. SNIA NVM Programming Technical Work Group (TWG) [9, 10] has started model specification, which consists of members from Intel, NetApp, Samsung, Microsoft, Cisco, Dell, VMWare, and many others, back in 2012. The aim was to have a well-sophisticated programming standard when the real hardware will come up.

From the start, the vision was to be able to have consistent access to persisted data on NVRAM. It might not seem easy to directly access the same region of memory after the power cycle, and thus standard method with filesystems was admitted to being most efficient with some additional updates. For it to work efficiently on non-volatile memory, filesystems needed to have DAX [1] support. DAX is the mechanism that allows making operations on data stored in persistent memory arrays without the need for a page cache. That will allow avoiding double operations on memory.

Additionally, the programming model had implicitly described calling CPU instructions from user space, such as CLFLUSH or CLWB, to flush the CPU cache line to ensure the persistence of the data written to NVRAM [9]. However, as the cache line granularity is 64-byte, there must be proper consideration of operations order in software to ensure a flawless transaction state. On top of that, to provide atomic operations accessing the same data in a multithreaded environment does pose quite a challenge. Thus, all these aforementioned factors add up to the complexity of the development of the software for non-volatile memories.

Now, with the guidance of the SNIA NVM Programming Model, there is a collection of libraries directly supported by Intel, the Persistent Memory Development Kit (PMDK) [2]. As it is followed by the programming model, the PMDK is vendor-neutral, and thus will be able to work with other upcoming non-volatile memory devices. It has a collection library for C and C++ programming languages to work with NVRAM, as well as tools to create, check, and sanitize memory pools.

Even there are libraries for writing applications to work directly with non-volatile memory, Intel's new memory has a mode, which does not require any software modification. It is called Memory Mode, Figure 2A, and in this case, Intel's new memory works as an extension to DRAM. A software does not recognize where data goes, NVRAM or DRAM, as

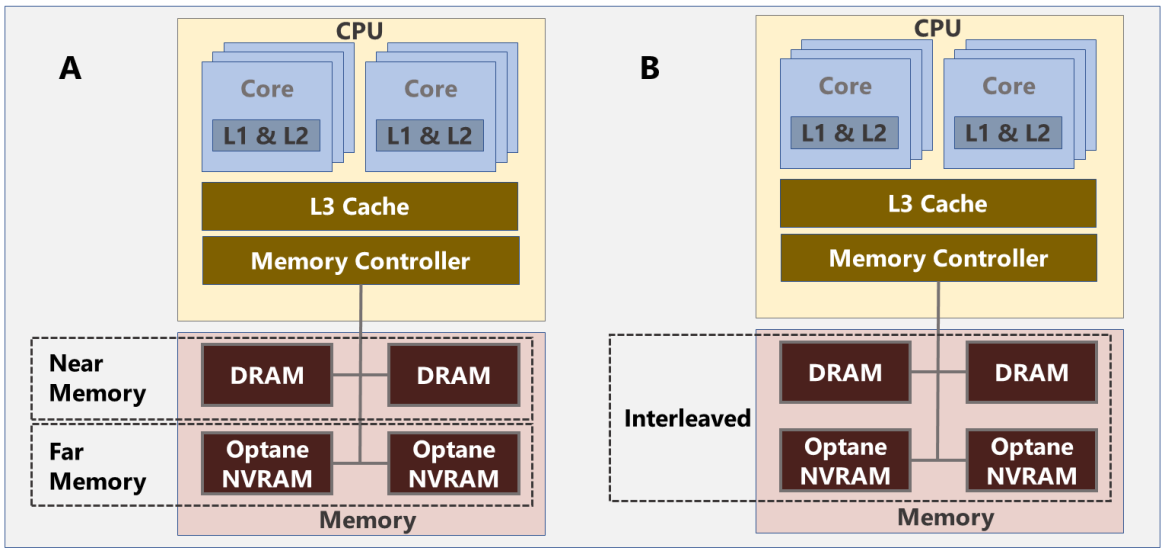

Figure.2. Memory (A) Memory Mode in Optane DC PM (B) App Direct Mode in Optane DC PM.

it is all controlled internally by the host memory controller. Essentially, this is a cache layer system, where data accesses go first to DRAM, and if missed, it goes to Optane Memory. Also, even the data will be stored in NVRAM, it is not recognized as persistent data. This mode gives a large volatile memory pool for a much cheaper price, and almost no performance degradation.

In order to distinctively access and write data to DRAM and NVRAM separately, the system needs to be in App Direct mode as noted in Figure 2B. In this mode, as expected, data written to NVRAM stays persistent in power cycles. With the use of PMDK [2], or other libraries for NVM, such as go-pmem [20] for Go language, or pcj [3] for Java, software development is made easy for the new hardware and take advantage of fast persistent data access.

Current application developments

In this section, we will discuss some of the current application developments done for non-volatile memory.

Data Structures

Data structures are the foundations of any software that has ever been written. They make the organization of data efficient and easily manageable. Each data structure has its use case, ofcourse it depends on what kind of access pattern is needed. Thus, data structures are the first entities to be researched for the development of a sturdy non-volatile memory.

PMDK[2] and pcj[3] do have a library with standard data structure collections, such as array, stack, queue, etc. Simple data structures are quite easy to port to NVRAM providing atomicity, consistency, and persistence. However, more advanced data structures are complicated and thus need a more careful approach.

One of the areas where an in-depth research is ongoing is the area of data structures is B-tree, these are used in SQL databases. It is primarily used as an index structure, which gives a fast point and range access. One of the recent works FAST and FAIR B-tree [11] provides fast point access, and B^3-tree [12] does have superiority in fast range queries and concurrent accesses.

There are also other advanced data-structures other than B-tree, that are not simple to port to be implemented on NVRAM and provide atomicity, consistency, and persistence. Recipe[13] did convert concurrent data-structures used for indexing and provide remarkable performance. The modification list consists of B-tree, radix tree, tries, and hash-table. Their results showed up to 5.2x outperformance in comparison to the same PM data structure counterpart.

Databases

Most of the popular databases are based on LSM-tree and B-tree. There are several approaches when designing these databases, e.g. NVRAM based only, or hybrid (DRAM, NVRAM, and/or SSD) based. Many state-of-the-art databases, such as SQLite, MySql, Redis, RocksDB, MongoDB, and many others, were ported to use the only NVRAM as a persistent storage media [5]. As a result, it showed up to a 20x performance increase compared to SSD based databases. However, increased performance comes at a larger cost, as non-volatile memory becomes more expensive to implement. Therefore, hybrid-based databases are being deeply researched, as we speak, to give both lower cost and better performance. MatrixKV[14], SplitKV[15], NoveLSM[16], and SLM-DB[17] do use different techniques to challenge storing data in both NVRAM and SSD.

File Systems

Memory only filesystems existed before NVRAM existed on paper and were designed for DRAM. As such, they were fast but volatile, which will have data loss after a system crash. With the early introduction of the persistent programming model [9] famous filesystems, ext4 and xfs started to be developed for NVRAM, simulating their behaviour on DRAM. As result, the DAX [1] mechanism came to be. Also, file systems can be distributed and multi-node, making storage larger and faster. Examples of successful ones are Assise [18] and Orion [19].

Possible future of the technology

This is might seem like mere speculation; however, we should not exclude the possible development of the technology. In an essence, that is what scientists do, trying to predict future possible technologies, visualizing it in their imagination, and work on it with capable resources. After many trials and errors, something new might come up.

For now, this technology is capable of working only on server hardware machines, and thus making NVRAM applicable only for commercial use applications, as it was described above. It has to be seen if Intel, or some other vendor, will be able to develop this novel technology further, so any machine and device will be able to install it without hardware restriction. Imagine having smartphones that launch applications in an instant and doing OS restart in a few seconds, or IoT devices that have almost zero downtime due to instant reboot; or personal computers that will be able to handle photo and video editing of large scales without any latency. All of this might be possible if we think deeply about how this new memory technology can potentially change how the software works.

Conclusions

In conclusion, NVRAM, or PM, is a very promising technology. Now, it is not broadly used, but only in certain large-scale applications. Even though, new memory has revealed a ton of possibilities and many more will come. If this technology will develop even further, might come one day that every device might have dense and performant non-volatile memory. Then thus almost every software will be able to access any data in an instant and have zero downtime.

References

1. Direct access for files, kernel.org https://www.kernel.org/doc/Documentation/filesystems/dax.txt
2. PMDK https://pmem.io/pmdk/
3. Persistent Collections for Java https://github.com/pmem/pcj
4. Yang, Jian, et al. "An empirical guide to the behavior and use of scalable persistent memory." *18th {USENIX} Conference on File and Storage Technologies ({FAST} 20)*. 2020. https://www.usenix.org/conference/fast20/presentation/yang
5. Izraelevitz, Joseph, et al. "Basic performance measurements of the intel optane DC persistent memory module." *arXiv preprint arXiv:1903.05714* (2019). https://arxiv.org/abs/1903.05714
6. Intel Optane™ DC Persistent Memory https://www.intel.com/content/www/us/en/architecture-and-technology/optane-dc-persistent-memory.html
7. Intel Persistent Memory Programming. https://pmem. io/pmdk/
8. Intel and Micron's 3D XPointTM Technology. https://www.micron.com/about/ our-innovation/3d-xpoint-technology
9. Rudoff, Andy. "Persistent memory programming." *Login: The Usenix Magazine* 42.2 (2017): 34-40. https://www.usenix.org/system/files/login/articles/login_summer17_07_rudoff.pdf/
10. NVM Programming Model https://www.snia.org/tech_activities/standards/curr_standards/npm
11. Kim, Wook-Hee, et al. "FAST and FAIR B+-Tree for Byte-Addressable Persistent Memory." http://nvmw.ucsd.edu/nvmw2019-program/unzip/current/nvmw2019-final51.pdf
12. Cha, Hokeun, et al. "B3-Tree: Byte-Addressable Binary B-Tree for Persistent Memory." *ACM Transactions on Storage (TOS)* 16.3 (2020): 1-27. https://dl.acm.org/doi/abs/10.1145/3394025
13. Lee, Se Kwon, et al. "Recipe: converting concurrent DRAM indexes to persistent-memory indexes." *Proceedings of the 27th ACM Symposium on Operating Systems Principles*. 2019. https://dl.acm.org/doi/pdf/10.1145/3341301.3359635
14. Yao, Ting, et al. "MatrixKV: Reducing Write Stalls and Write Amplification in LSM-tree Based {KV} Stores with Matrix Container in {NVM}." *2020 {USENIX} Annual Technical Conference ({USENIX}{ATC} 20)*. 2020. https://www.usenix.org/conference/atc20/presentation/yao
15. Han, Shukai, Dejun Jiang, and Jin Xiong. "SplitKV: Splitting {IO} Paths for Different Sized Key-Value Items with Advanced Storage Devices." *12th {USENIX} Workshop on Hot Topics in Storage and File Systems (HotStorage 20)*. 2020. https://www.usenix.org/conference/hotstorage20/presentation/han
16. Kannan, Sudarsun, et al. "Redesigning LSMs for nonvolatile memory with NoveLSM." *2018 {USENIX} Annual Technical Conference ({USENIX}{ATC} 18)*. 2018. https://www.usenix.org/conference/atc18/presentation/kannan
17. Kaiyrakhmet, Olzhas, et al. "SLM-DB: single-level key-value store with persistent memory." *17th {USENIX} Conference on File and Storage Technologies ({FAST} 19)*. 2019. https://www.usenix.org/conference/fast19/presentation/kaiyrakhmet
18. Anderson, Thomas E., et al. "Assise: Performance and Availability via Client-local {NVM} in a Distributed File System." *14th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 20)*. 2020. https://www.usenix.org/conference/osdi20/presentation/anderson
19. Yang, Jian, Joseph Izraelevitz, and Steven Swanson. "Orion: A distributed file system for non-volatile main memory and RDMA-capable networks." *17th {USENIX} Conference on File and Storage Technologies ({FAST} 19)*. 2019. https://www.usenix.org/conference/fast19/presentation/yang
20. George, Jerrin Shaji, et al. "go-pmem: Native support for programming persistent memory in go." *2020 {USENIX} Annual Technical Conference ({USENIX}{ATC} 20)*. 2020. https://www.usenix.org/conference/atc20/presentation/george