Essay on persistent memory

Svein Gunnar Fagerheim

27th May 2019

1 Introduction

High performance computing (HPC) is being used to analyze massive amount of data. Where the DRAM has always been a bottleneck to keep in mind when developing code to analyze the data. Persistent memory is changing this by providing cheap, fast and persistent memory that can store data even after a shut down. This essay will explore where persistent memory is currently at, what are the advantages and challenges associated with persistent memory. And the essay will explore the different libraries that exist and possibly determine which one is best.

1.1 What is persistent memory

Persistent memory[8] is a non-volatile storage memory[4] that is byte-addressable and has speed close to that of DRAM. DRAM stands for direct random-access memory, this is a volatile storage system that will loose all its data when the computer is shut down or restarted. Applications and data used by the CPU are temporarily loaded into memory from a hard drive in order to reduce latency and increase bandwidth. Persistent memory is another layer between the CPU and the disk. The data the CPU has the most use for is stored in the L1-L3 cashes. When the cashe is full the data needs to be evicted the evicted data will be sent back to the memory. If the data usage of the program is so large that it exceeds the memory available on the computer then the computer will start use virtual memory on the disk which is a lot slower then DRAM. The reason virtual memory is so slow is because one must do an I/O block to read and write to disk which takes time. Persistent memory is a layer between the DRAM and the disk in which the

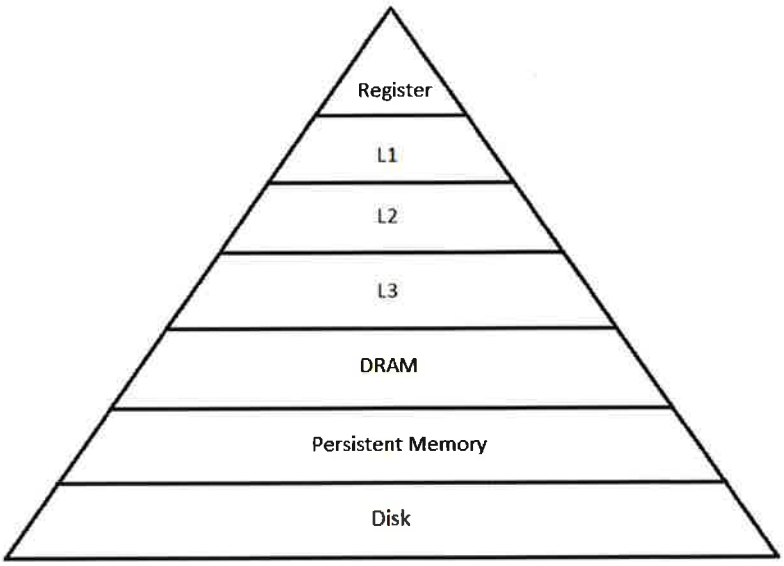

Figure 1: Persistent memory becomes a new level between DRAM and Disk[9]

CPU can access directly just like it would do a normal DRAM. Figure 1 illustrates where in the hierarchy the persistent memory is placed.

2 Challenges

2.1 Consistent updates

When writing data to storage that will persist even after a shutdown there must be a way to ensure that the data are either fully stored to the storage device or not at all. If there is an interruption taking place, for example power failure and the storage system does not have a way of ensuring that the data won't be partially written to the storage device, then that will lead to corrupted data. Not only would the unsaved data the application was working on be lost, old data loaded into the application would also be lost if the application was in the process of overwriting old data with unsaved data[3]. Traditional storage devices such as hard drive and SSD have several methods to ensure that the data won't be corrupted by interruptions such as shadow updates, soft updates, journaling, and post-reboot checking. This is done

automatically by the OS, so the programmer does not need to think about this. Since the application has direct access to the hardware, the programmers need to ensure that the application can perform consistent updates.

2.2 Security

Another challenge when it comes to persistent memory is security and privacy[1]. When an encryption key is used to unlock files, the key is stored in the memory. If it is DRAM then the key will disappear when the computer is shut down, but if it is stored in persistent memory it will persist ~~and it will~~ and remain there until it is deleted. It will be very easy for someone to extract data from persistent memory if they are able to gain access to it. The same concern also applies to personal information that is being stored in the persistent memory. When developing applications that will use persistent memory and handling sensitive information the programmer needs to remember that data he puts in persistent memory will stay there until it is deleted. It's also worth mentioning that when the data is deleted or deallocated in the memory, the OS makes the space available to another application. The data is only deleted when its overwritten.

2.3 Durability

Another challenge for persistent memory is durability. While persistent memory behaves more like DRAM it still has a considerably shorter lifespan than DRAM[1]. This is because persistent memory can only write data to a certain amount of time to a region before the region can no longer hold any data reliably. While the storage capacity of persistent memory has increased, so has the bit error rate increased even more. The solution to this is to either have the hardware mask all the regions with bit error from the software or have the hardware expose them to the software and let the software handle the rest. There is also the possibility of letting the hardware and software work together in order to mask regions with bit errors. This will expand the lifespan of persistent memory, but the challenge for hardware producers is come up with new technologies that can increase the durability of the persistent memory.

to be concerned about interruption either.

References

[1] Anirudh Badam. 'How Persistent Memory Will Change Software Systems'. In: *Computer* 46 (2013), pp. 45–51.

[2] Gerhard Wellein Georg Hager. *Introduction to High Performance Computing for Scientists and Engineers*. CRC Press, 2010. ISBN: 9781439811924.

[3] Andres Jaan Tack Haris Volos and Michael M. Swift. 'Mnemosyne: Lightweight Persistent Memory'. In: *ACM* 39 (2011), pp. 91–104.

[4] Mahmut Kandemir. 'Exploring the future of out-of-core computing with compute-local non-volatile memory'. In: *Scientific Programming* 22 (2014), pp. 125–139.

[5] Gaku Nakagawa and Shuichi Oikawa. 'NVM/DRAM Hybrid Memory Management with Language Runtime Support via MRW Queue'. In: *IEEE* (2015).

[6] Dushyanth Narayanan and Orion Hodson. 'Whole-System Persistence'. In: *ACM* (2012), pp. 401–410.

[7] Andy Rudoff. 'Persistent Memory Porgramming'. In: *Login* 42 (2017), pp. 125–139.

[8] Andy Rudoff. 'Persistent Memory: The Value to HPC and the Challenges'. In: *Workshop on Memory Centric Programming for Hpc* (2017), pp. 7–10.

[9] Andy m. Rudoff. *Intel Optane DC Persistent Memory: A Major Advance in Memory and Storage Architecture*. URL: `https://software.intel.com/en-us/blogs/2018/10/30/intel-optane-dc-persistent-memory-a-major-advance-in-memory-and-storage-architecture` (visited on 25/05/2019).

[10] Lisa Spelman. *Reimagining the Data Center Memory and Storage Hierarchy*. URL: `https://newsroom.intel.com/editorials/re-architecting-data-center-memory-storage-hierarchy/#gs.d0rr1r` (visited on 22/05/2019).

[11] Karin Strauss and Doug Burger. 'What the Future Holds for Solid-State Memory'. In: *Computer* (2014), pp. 24–31.

order of authors

publication source not entirely correct

missing authors!

publication source info missing

Order of authors

[12] Ada Gavrilovska Sudarsun Kannan Dejan Milojicic and Karsten Schwan. 'Using Active NVRAM for I/O Staging'. In: *ACM* (2011), pp. 15–22.

[13] Steven Swanson. 'NV-Heaps: Making Persistent Objects Fast and Safe with Next-Generation, Non-Volatile Memories'. In: *ACM* (2011), pp. 105–118.

[14] *The libmemking library*. URL: `https://github.com/memkind/memkind` (visited on 25/05/2019).

[15] *The libpmem library*. URL: `http://pmem.io/pmdk/libpmem/` (visited on 25/05/2019).

[16] *The libpmemblk library*. URL: `http://pmem.io/pmdk/libpmemblk/` (visited on 25/05/2019).

[17] *The libpmemlog library*. URL: `http://pmem.io/pmdk/libpmemlog/` (visited on 25/05/2019).

[18] *The libpmemobj library*. URL: `http://pmem.io/pmdk/libpmemobj/` (visited on 25/05/2019).

[19] *The librpmem library*. URL: `http://pmem.io/pmdk/librpmem/` (visited on 25/05/2019).

[20] *The libvmem library*. URL: `http://pmem.io/pmdk/libvmem/` (visited on 25/05/2019).

[21] *The libvmmalloc library*. URL: `http://pmem.io/pmdk/libvmmalloc/` (visited on 25/05/2019).

missing authors

incomplete info